Mathematische morfologie in de beeldverwerking

Mathematical Morphology in Image Processing

Alessandro Ledda

Promotor: prof. dr. ir. W. Philips
Proefschrift ingediend tot het behalen van de graad van
Doctor in de Ingenieurswetenschappen

Vakgroep Telecommunicatie en Informatieverwerking
Voorzitter: prof. dr. ir. H. Bruneel
Faculteit Ingenieurswetenschappen
Academiejaar 2006 - 2007

UNIVERSITEIT
GENT

ISBN 978-90-8578-127-1
NUR 983, 965
Wettelijk depot: D/2007/10.500/1

Aan mijn ouders, voor alles

Dankwoord

Het voorwoord is een van de leukere dingen om te lezen in een thesis. Daarna wacht de lezer een grote brok informatie die waarschijnlijk iets minder vlot te verteren valt. Ik heb in elk geval getracht de zaken zo duidelijk mogelijk te formuleren. Maar vooraleer ik aan schrijven toe kwam, moest ik eerst nog een bergachtig parcours afleggen. In al die jaren zijn er hoogtes en laagtes, depressies en eureka-momenten geweest. In het begin van een doctoraat is het nog erg zoeken welke richting je uit wil en hoe je de zaken zal aanpakken. Tussendoor beland je wel eens in een dipje wanneer het onderzoek niet zo vlot loopt als gewenst. Op andere momenten blijkt dan weer alles mee te zitten en lukt het bijna vanzelf.

Het schrijven van dit werk kan in elk geval een unieke ervaring genoemd worden. Nu weet ik bijvoorbeeld hoe een chronisch tekort aan slaap en een teveel aan stress aanvoelen. Het idee dat 80-uren werkweken niets voor mij zijn, heb ik in de praktijk getoetst en kan ik bij deze bevestigen. Het leukste van al was waarschijnlijk de hittegolf van de afgelopen zomer. Een mens hoopt telkens op een mooie, warme dag, maar ik vermoed dat ik de enige Belg ben die dolblij was dat augustus zo'n rotmaand was.

De eerste persoon die ik in de spreekwoordelijke bloemetjes wens te zetten, is mijn promotor prof. Wilfried Philips. Hij heeft me gedurende meer dan 4 jaar met inzet en professioneel enthousiasme begeleid. Hij heeft al die jaren de moed niet opgegeven. Chapeau.

Een woord van dank gaat ook uit naar mijn examencommissie. Deze mensen — dr. Steve De Backer, prof. Joris Degrieck, dr. Mike Nachtegael, dr. Aleksandra Pižurica, prof. Kees Slump, prof. Heidi Steendam, dr. Peter Veelaert en voorzitter prof. Paul Kiekens — hebben (hopelijk) de moeite genomen mijn thesis te doorbladeren en op een kritische wijze te bestuderen.

Het feit dat je dit boek momenteel in handen hebt, wil zeggen dat mijn digitale gegevens niet verloren zijn gegaan. Dit is te danken aan de systeembeheerders van TELIN*, Philippe Serbruyns en Davy Moreels, de* Batman en Robin *van de vakgroep. Zij staan steeds klaar om ons te redden van alle computerproblemen. Zonder hen zou het op* ICT*-gebied in het honderd lopen.*

De tweede peiler waar onze vakgroep op steunt, is het secretariaat. Dankzij Annette Nevejans, Patrick Schaillée en Alice Verheylesonne kunnen wij zonder

zorgen ons ding doen, zonder ons al te veel te moeten bekommeren om de administratieve rompslomp die men daarrond altijd weet te creëren.

Een thesis schrijf je uiteindelijk zelf, maar gedurende die ganse periode van onderzoek kom je in contact met mensen die op een of andere manier bijdragen aan dit boek. Zo zijn er Jan Quintelier en Pieter Samyn van Labo Soete. Dit zijn de mannen van de destructieve sector (i.e., tribologie) die mij foto's van slijtagedeeltjes hebben bezorgd en tevens wegwijs hebben gemaakt in de wondere wereld van polymeren en composieten. Met Valérie De Witte van de Vaaglogische groep heb ik verschillende discussies gehad over kleuren, morfologie en interpolatie.

Ik heb ook erg veel gehad aan verscheidene collega's van TELIN. *Veel interessants heb ik onder andere geleerd van Hiêp in verband met interpolatietechnieken. Ook Ewout heeft mij veel bijgebracht, met name de wiskundigere kant van een aantal zaken. Stefaan wil ik danken voor het ter beschikking stellen van zijn* PHP/MySQL*-code voor het psychovisuele experiment.*

Onder het motto "de boog kan niet altijd gespannen staan" wil ik mijn fijnste collega's/vrienden Linda en Marleen bedanken voor alle leuke momenten binnen en buiten TELIN. *Tevens op professioneel vlak hebben ze mij bijgestaan met hun kennis en nuttige tips. Er zitten echter nog andere sympathieke mensen in de vakgroep. Ik ben ze niet vergeten, maar ze allemaal opsommen wordt misschien wat te langdradig. Ik wil hen dan ook allemaal bedanken voor de toffe sfeer en de leuke en nuttige momenten.*

Er is echter ook nog een leven buiten de universiteit. Vrienden die mij gesteund, opgepept en ontspannen hebben zijn Werner, Ans, Sofie, Evelien, Tine, Chris, An en Evi. Uiteraard mag ik Marianne hier niet vergeten te vermelden.

Last, but allesbehalve least, wil ik ook een dikke merci geven aan mijn ouders, die me al die jaren gesteund, gekoesterd en gesoigneerd hebben. Zonder hen zou ik dit hier niet hebben kunnen neerschrijven. Ook mijn zusje mag hier niet ontbreken, gewoon omdat ze een toffe is. Tot slot een speciale vermelding voor Tessa, waarmee ik gedurende 15 jaar lief en leed deelde. Zij was er steeds om me goedgezind te maken en me de mooie dingen in het leven te doen blijven zien.

Alessandro
Oktober 2006

Samenvatting

Mathematische morfologie (MM) is een theoretisch raamwerk voor de analyse van (vormen in) beelden en gebaseerd op settheorie. In het begin was het dan ook enkel bruikbaar voor de verwerking van binaire beelden, die als verzamelingen van zwarte pixels tegen een witte achtergrond kunnen worden beschouwd. Later werden uitbreidingen ontwikkeld voor grijswaarden- en kleurbeelden. Morfologie kan vanuit een erg theoretisch standpunt bekeken worden, maar in deze thesis kijken we naar de bruikbaarheid van de gereedschappen die deze theorie ons biedt.

De theorie van mathematische morfologie steunt op twee basisbeeldverwerkingsoperatoren: de *dilatatie* en de *erosie*. De dilatatie zet een beeldobject als het ware uit, terwijl een erosie het object doet inkrimpen. Kenmerkend aan deze operatoren is dat zij de vorm van de objecten grotendeels behouden. Morfologie is dan ook een theorie waarbij de vorm en grootte van objecten een grote rol spelen.

Een belangrijke parameter van de morfologische operatoren is het *structuurelement*. Dit is een set, meestal veel kleiner dan de set die het beeld voorstelt, die over het beeld geschoven wordt. De vorm van dit structuurelement is bepalend voor het resultaat van de morfologische bewerking. Met behulp van de basisoperatoren kunnen veel complexere operatoren geconstrueerd worden. Het combineren van verschillende operaties maakt het mogelijk morfologie te gebruiken voor verschillende toepassingen, zoals segmentatie, hoekdetectie, ruisfiltering, ...

Voor de uitbreiding van de theorie naar grijswaarden kan gebruik gemaakt worden van de *schijfjesbenadering* (Engels: threshold approach) of de *umbrabenadering*. Beide methoden hebben hun voor- en nadelen. Het grote verschil met de binaire theorie is het vervangen van de setoperaties unie en doorsnede door maximum en minimum. De beelden worden dan ook niet meer aanzien als sets, maar als functies.

Een uitbreiding naar kleurbeelden is minder vanzelfsprekend. De kleur van een pixel wordt voorgesteld door een vector, typisch bestaande uit een rode, groene en blauwe kleurcomponent (de RGB-kleurruimte). Kleuren kunnen in principe niet op een ondubbelzinnige en natuurlijke wijze totaal geordend worden, hetgeen nodig is om de morfologische operatoren van de grijswaardentheorie uit te breiden naar kleurbeelden. Soms wordt elke kleurcomponent afzonderlijk be-

handeld. Een andere mogelijkheid is om een artificiële lexicografische ordening toe te passen. Zo krijgt bijvoorbeeld de luminantiewaarde een grotere prioriteit in de bepaling van de kleurvolgorde dan de saturatie- en tintwaarde.
In dit proefschrift ontwikkelen we een aantal nieuwe beeldverwerkingstechnieken gebaseerd op mathematische morfologie. Bovendien onderzoeken we de bruikbaarheid van een specifieke techniek, het patroonspectrum, in materiaalkundig onderzoek.
Als een eerste bijdrage van dit proefschrift stellen we het *meerderheidsordeningsschema* voor, getiteld MSS. De kleuren worden gerangschikt naargelang de frequentie dat ze voorkomen in het beeld. Voor een aantal typen beelden is dit interessant. Op het geordende beeld, de MSS-map, worden de morfologische bewerkingen uitgevoerd. Deze methode is nuttig voor de behandeling van beelden waarop een andere kleurordening niet goed werkt. Verscheidene problemen worden aangekaart en oplossingen worden aangereikt, alsook de eigenschappen van de MSS worden besproken. Zo is er het probleem dat verschillende kleuren dezelfde rangorde kunnen hebben in de MSS-map (en dus beschouwd en behandeld worden als dezelfde kleur).
Een ander onderzoek heeft betrekking op het morfologische *patroonspectrum* (PS) en aanverwante spectra die veel minder rekenintensief zijn. Eerst doen we een vergelijkende studie van de verschillende technieken. Vervolgens passen we deze technieken toe in het onderzoek — via beeldverwerking — van de slijtage van polymeren en composieten die gebruikt worden als glijlagers. Tribologisch onderzoek onthult informatie over het gedrag van deze materialen onder verscheidene temperaturen en belasting. Wij onderzoeken de brokstukken, afkomstig van de slijtage-experimenten, en correleren de karakteristieken van deze deeltjes, bekomen via het patroonspectrum, aan de experimentele parameters. We bespreken de bekomen resultaten en gaan na hoe bruikbaar de spectra zijn.
Tot slot passen we morfologie toe in het domein van de beeldinterpolatie. In een rasterbeeld wordt elke beeldcoördinaat voorgesteld door een pixel, in dewelke een kleur- of grijswaarde opgeslagen zit. Wensen we in te zoomen op het beeld of de resolutie aan te passen, dan moeten we gebruik maken van interpolatietechnieken. Afhankelijk van de gebruikte interpolatietechniek kunnen artefacten optreden. Bij pixelreplicatie bijvoorbeeld zien de objectranden in het uitvergrote beeld er gekarteld uitzien; bij bilineaire interpolatie wordt het beeld dan weer wazig. Andere effecten zijn ringvorming, segmentatie- of veegeffecten.
Wij stellen een nieuwe adaptieve interpolatietechniek voor die gebaseerd is op mathematische morfologie, mmINT. Deze techniek vertrekt van een pixelgerepliceerd binair beeld dat vervolgens verbeterd wordt. Dit gebeurt door de hoeken van de gekartelde randen “af te ronden”. Hiervoor moeten we de hoeken eerst detecteren, hetgeen mogelijk is met de morfologische hit-miss transformatie. De hoeken van de gekartelde randen worden afgevlakt door de pixelwaarden van deze hoeken om te wisselen. In mmINT gebeurt dit in een iteratief proces. De iteratieve werking van mmINT maakt de methode traag. Daarom hebben we ook mmINTone ontwikkeld, een techniek die erg gelijkaardig is wat betreft

de filosofie en het interpolatieresultaat ten opzichte van mmINT, maar die de interpolatie uitvoert in een enkele cyclus. Het verschil in rekentijd kan dan tot enkele grootte-ordes zijn.
We hebben ook een uitbreiding naar grijswaarden geïmplementeerd, mmINTg. Voor de hoekdetectie worden de grijswaarden eerst lokaal gebinarizeerd, waarbij de drempelwaarde afhangt van de minimum- en maximumwaarde in het lokale venster. Ons voornoemde voorstel van de meerderheidsordening gebruiken we om een grijswaarde lokaal te classificeren als voorgrond of achtergrond. De stap waar de hoekpixels van waarde veranderen is ook gewijzigd, aangezien we nu niet enkel over de kleuren wit en zwart beschikken. De gemiddelde waarde van naburige pixels bepaalt de nieuwe pixelwaarde.
Onze op morfologie gebaseerde interpolatiemethoden blijken beter te werken dan bestaande algoritmes. Dit blijkt uit visuele vergelijkingen, kwantitatieve experimenten en testen met een testpubliek.

Summary

Mathematical morphology (MM) is a theoretical framework for the analysis of (the shapes in) images, based on set theory. Initially, it was only applicable to binary images, which can be considered as sets of black pixels against a white background. Later on, extensions to greyscale images and colour images have been developed. In this thesis we concentrate on the usability of several morphological tools, rather than on theoretical development.
The theory of mathematical morphology is built on two basic image processing operators: the *dilation* and the *erosion*. Simply put, the dilation enlarges the objects in an image, while the erosion lets them shrink. A feature of these operators is the fact that they preserve the objects' shapes for the most part. Morphology is thus a theory where size and shape of objects play an important role.
An important parameter of the morphological operators is the *structuring element*. This is a small set, mostly much smaller than the image set, that scans the image. The pixels covered by this structuring element determine the output value of the currently covered pixel. Using the basic operators, much more complex operators can be constructed. These new operators can be used in different applications, such as image segmentation, corner detection, noise filtering, ...
For the extension of the theory to greyscale images we can use the *threshold approach* or the *umbra approach*. Both methods have their advantages and disadvantages. The main difference between the greyscale and binary case is that the union and intersection are now replaced by a maximum and minimum operation. The images are therefore no longer treated as sets, but as functions.
An extension to colour images is less straightforward. We represent a pixel colour by a vector, typically consisting of a red, green and blue colour component (the RGB colour space). Colours cannot be totally ordered in a unique sensible way. If the colours are ordered, then we can apply greyscale morphology on colour images. Sometimes, each colour component is treated separately. Another option is to use an artificial lexicographical ordering scheme. For example, a pixel with higher luminance is ranked higher, while saturation and hue are less important for the ordering.
In this dissertation, we develop a number of image processing techniques that are based on mathematical morphology. We also investigate the usability of a

specific technique, the pattern spectrum, in the field of material science.
Our first contribution is the proposal of the *majority ordering*, a.k.a. the *majority sorting scheme* (MSS). The image colours are ranked in accordance to their frequency of occurrence in the image. This ordering is interesting for a certain type of images. On this ordered image, the MSS-map, we can apply morphological operations. This method is useful if operations using other colour orderings do not produce a satisfying result. We discuss several possible problems and solutions, as well as the properties of the MSS. One of the problems we can encounter is that different colours can be given the same ranking in the MSS-map (and therefore be considered and treated as the same colour).
Another part of our research concerns the investigation of the morphological *pattern spectrum* (PS) and related (and much faster) spectra. First, we compare the different techniques. Afterwards, we use these techniques for the analysis — using image processing — of polymers and composite materials that are used as sliding bearings. Tribological research reveals information about the behaviour of these materials under certain temperatures and loads. We analyse the debris particles from the wear experiments, and correlate the characteristics of these particles, obtained from the pattern spectrum, with the experimental parameters. We discuss the obtained results and investigate the usefulness of the spectra.
Finally, we concentrate on image interpolation. A bitmap image consists of pixels, where each pixel contains a grey value or colour value. To zoom in on the image or to change the spatial resolution, we must make use of interpolation techniques. Artefacts can occur during the interpolation process, and they depend on the technique used. For example, pixel replication introduces jagged edges ("jaggies"), while a bilinear interpolation makes the image blurry. Other effects are ringing artefacts, segmentation effects or painting effects.
We propose a new and adaptive interpolation technique, mmINT, based on mathematical morphology. This technique improves a pixel-replicated image. This is done by smoothing the jagged edges in the magnified image. We detect the corners of such edges with the morphological hit-miss transform. The corners of the jagged edges are smoothened by swapping the pixel values. In mmINT this is done in an iterative process.
The iterative nature of mmINT makes the method slow. Therefore, we developed mmINTone, a technique very similar to mmINT, both in result and in philosophy. The big difference is that only one iteration is needed, making mmINTone much faster. The speed improvement can be up to several orders of magnitude.
We also present an extension to greyscale images, mmINTg. The corner detection step is updated by introducing a local binarization step. The threshold value depends on the minimum and maximum value in the local window. We also use our aforementioned proposal of the majority ordering to locally classify a grey value as object pixel or background. The pixel swapping step is also replaced, since we now can have other colours besides black and white. The average grey value of neighbouring pixels defines the new pixel value.

Our morphology-based interpolation methods perform better than existing techniques. We come to this conclusion after comparing the results visually, performing quantitative experiments, and by scoring of the interpolation results by a test panel.

Contents

Symbols and acronyms used in this thesis

Symbols

a	Scalar value
$\mathbf{a}$	Vector
A	Set (sometimes function)
f	Function
$A \oplus B$	Dilation of A by B
$A \ominus B$	Erosion of A by B
$A \bullet B$	Closing of A by B
$A \circ B$	Opening of A by B
$A \otimes (B, C)$	Hit-miss transform of A by B and C
A^c	Complement of A
$\breve{A}$	Reflection of A
$T_{\mathbf{r}}(A)$	Translation of A over vector $\mathbf{r}$
$T_v(A)$	Greyscale translation of A with value v
$H_\lambda(A)$	Scaling of A with factor λ
$A_\times(t)$	Cross-section of A at level t
$T[A]$	Top surface of set A
$U[f]$	Umbra of function f
$\Omega(f)$	Support of f
$\sharp[A]$	Cardinality of set A
$\sharp[f]$	Total sum of grey values of function f
$s(x, y)$	Continuous signal
$s(m, n)$	Discrete signal
$s_s(x, y)$	Sampled signal

$S(u, v)$	Fourier transform
$\mathbb{N}$	Set of all positive integers
$\mathbb{N}_0$	Set of all strict positive integers
$\mathbb{Z}$	Set of all integers
$\mathbb{Z}_0$	Set of all integers, except 0
$\mathbb{R}$	Set of all real numbers
$\lfloor a \rfloor$	Floor operation
$\lceil a \rceil$	Ceiling operation
$n \bmod m$	Modulo of n to m
f^*	Complex conjugate of f
$F(u, v)$	Fourier transform of $f(x, y)$
$\mathcal{F}(f)$	Fourier transform of f
$\mathcal{F}^{-1}(F)$	Inverse Fourier transform of F
$\mathcal{R}$	Real part of F
$\mathcal{I}$	Imaginary part of F

Acronyms

nD	n-Dimensional
AF	Alternating Filter
AMD	Advanced Micro Devices
APS	Area Pattern Spectrum
AQua	Adaptively QUAdratic interpolation
ASF	Alternating Sequential Filter
BT	Broadcasting service (Television)
CC I	Complementary Corner of type I
CC II	Complementary Corner of type II
CIE	Commission Internationale de l'Eclairage
CMC	Ceramic Matrix Composite
CMY	Cyan-Magenta-Yellow
CMYK	Cyan-Magenta-Yellow-BlacK
CPS	Colour Pattern Spectrum
CPU	Central Processing Unit
CRT	Cathode Ray Tube
DFT	Discrete Fourier Transform
DPI	Dots Per Inch

DST	Discrete Size Transform
EDI	Edge-Directed Interpolation
EPS	Encapsulated PostScript
	Erosion Pattern Spectrum
FFT	Fast Fourier Transform
FPS	Fourier Pattern Spectrum
FT	Fourier Transform
GIF	Graphics Interchange Format
GLA	Generalized Lloyd Algorithm
HA	High Alloy
HDTV	High Definition TeleVision
HQ	High Quality interpolator
HSI	Hue Saturation Intensity
HSL	Hue Saturation Luminance
HSL_c	Hue Saturation Luminance (Cylindrical)
HSL_{dc}	Hue Saturation Luminance (Double-Cone)
HSV	Hue Saturation Value
ICC	International Color Consortium
IR	InfraRed
ITU	International Telecommunication Union
ITU-R	ITU, Radiocommunication sector
JPEG	Joint Photographic Experts Group
LCD	Liquid Crystal Display
LUT	LookUp Table
MDS	Multi-Dimensional Scaling
ML	Maximum-Likelihood
MM	Mathematical Morphology
MMC	Metal Matrix Composite
mmINT	Mathematical Morphological Interpolation
mmINTg	mmINT for Greyscale images
mmINTone	mmINT in ONE step
MSE	Mean Squared Error
MSS	Majority Sorting Scheme
NEDI	New Edge-Directed Interpolation
NTSC	National Television System Committee
OPS	Oriented Pattern Spectrum

OT	Opening Tree
PAL	Phase Alternating Line
PDE	Partial Differential Equation
PDF	Portable Document Format
	Probability Density Function
PGF	Peer Group Filtering
PMC	Polymer Matrix Composite
PNG	Portable Network Graphics (Unofficial: PNG's Not GIF)
POMH	PolyOxyMethylene Homopolymer
PPI	Pixels Per Inch
PS	Pattern Spectrum
	PostScript
PSNR	Peak Signal-to-Noise Ratio
PVL	PeriVentricular Leukomalacia
RAM	Random Access Memory
RGB	Red-Green-Blue
RMSE	Root Mean Squared Error
ROI	Region Of Interest
SNR	Signal-to-Noise Ratio
SP	Sintered Polyimide
SVG	Scalable Vector Graphics
SWF	ShockWave Flash
TP	Thermoplastic Polyimide
US	UltraSound
UV	UltraViolet
VLBW	Very Low Birth Weight
W3C	World Wide Web Consortium
WMD	White Matter Damage
WMF	Windows Meta File

Definitions

Translation by vector $\mathbf{r}$:

$$T_{\mathbf{r}}(B) \quad = \quad \{\mathbf{b} \mid \mathbf{b} - \mathbf{r} \in B\} \ .$$

Binary morphology

Basic operators:

$$\begin{aligned}
\textbf{Dilation}: &\quad A \oplus B = \textstyle\bigcup_{\mathbf{b}\in B} T_{\mathbf{b}}(A)\ , \\
\textbf{Erosion}: &\quad A \ominus B = \textstyle\bigcap_{\mathbf{b}\in B} T_{-\mathbf{b}}(A)\ , \\
\textbf{Closing}: &\quad A \bullet B = (A \oplus B) \ominus B\ , \\
\textbf{Opening}: &\quad A \circ B = (A \ominus B) \oplus B\ .
\end{aligned}$$

Mutual relationship, when $\mathbf{0} \in B$:

$$A \ominus B \subseteq A \circ B \subseteq A \subseteq A \bullet B \subseteq A \oplus B\ .$$

Scaling of a discrete structuring element:

$$nB \quad = \quad \underbrace{B \oplus B \oplus \cdots \oplus B}_{n \text{ times}}\ .$$

When $n = 0$, the scaled result is the origin $\mathbf{0}$.
Binary hit-miss transform:

$$A \otimes (B, C) \quad = \quad (A \ominus B) \cap (A^c \ominus C),\ \text{with } B \cap C \neq \emptyset\ .$$

Greyscale morphology (t-approach)

Basic operators:

$$\begin{aligned}
\textbf{Dilation}: &\quad (A \oplus B)(\mathbf{a}) = \max\{A(\mathbf{r}) \mid \mathbf{a} - \mathbf{r} \in B\}\ , \\
\textbf{Erosion}: &\quad (A \ominus B)(\mathbf{a}) = \min\{A(\mathbf{r}) \mid \mathbf{r} - \mathbf{a} \in B\}\ , \\
\textbf{Closing}: &\quad A \bullet B = (A \oplus B) \ominus B\ , \\
\textbf{Opening}: &\quad A \circ B = (A \ominus B) \oplus B\ .
\end{aligned}$$

Morphological pattern spectrum:

$$PS(A; B)(n) \quad = \quad \sharp[A \circ nB - A \circ (n+1)B],\ n \geq 0\ .$$

Oriented pattern spectrum:

$$OPS(A; B_\theta)(n) \quad = \quad \sharp[\max_\theta\{A \circ nB_\theta(x, y)\} - \max_\theta\{A \circ (n+1)B_\theta(x, y)\}]\ ,$$

with $n \geq 0$.

Pattern spectrum parameters

Maximal size

$$\begin{aligned} N_{max}(A;B) &= \min\{n \mid \forall n' > n : PS(A;B)(n') = 0\} \\ &= \max\{n \mid \exists \mathbf{a} \in \Omega(A) : \mathbf{a} \in \Omega(A \circ nB)\} \ . \end{aligned}$$

Average size

$$S(A;B) = \frac{\sum_{n=0}^{N_{max}} nPS(A;B)(n)}{\sharp[A]} \ .$$

Average roughness (entropy)

$$E(A;B) = -\frac{\sum_{n=0}^{N_{max}} PS(A;B)(n) \log_2 \left(\frac{PS(A;B)(n)}{\sharp[A]} \right)}{\sharp[A]} \ ,$$

where $P \log_2(P) = 0$ if $P = 0$.

Normalized average roughness

$$E_N(A;B) = \frac{E(A;B)}{\log_2(N_{max}+1)} \ .$$

B-shapiness

$$BS(A;B) = \frac{PS(A;B)(N_{max})}{\sharp[A]} \ .$$

Fourier transform

Continuous Fourier transform (2D):

$$\begin{aligned} F(u,v) &= \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} f(x,y) e^{-i2\pi(ux+vy)} dxdy \ , \\ f(x,y) &= \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} F(u,v) e^{i2\pi(ux+vy)} dudv \ . \end{aligned}$$

Discrete Fourier transform (2D):

$$F(u,v) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) e^{-i2\pi(ux/M+vy/N)} \ ,$$

$$f(x,y) \quad = \quad \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u,v) e^{i2\pi(ux/M+vy/N)} \ .$$

Fourier spectrum (2D):

$$|F(u,v)| \quad = \quad \sqrt{\mathcal{R}^2(u,v) + \mathcal{I}^2(u,v)} \ .$$

Mathematical Morphology in Image Processing

Chapter 1

Introduction

Mathematical morphology is a theory that dates back to 1964 when Georges Matheron studied the geometry of porous media in relation to their permeabilities. At the same time, Jean Serra had to quantify the petrography (i.e., the macroscopic and microscopic study of rocks) of iron ores, in order to predict their milling properties [Serra, 1982]. Both men initiated the development of the theory of mathematical morphology for binary images. Over the years, the theory has been extended to greyscale images and with some limitations to colour images. Even fuzzy logic is used in morphology [Kerre and Nachtegael, 2000, Nachtegael, 2002].

Mathematical morphology is used for the analysis of spatial structures in images. The keyword for morphology is *shape*: unlike, e.g., an averaging filter which averages grey values, a morphological operator does not look at the grey values in the first place, but uses geometrical features of objects in an image.

In theory, mathematical morphology can be applied in any field of image processing where shape plays some role. This can be object extraction, noise filtering, edge enhancement, segmentation, texture analysis, classification, shape description, and so on.

This thesis considers a few of the possible applications, namely *colour morphology*, *tribology* and *image interpolation*.

1.1 Colour mathematical morphology

A binary image can be represented as a set, i.e., the set of all pixels with value 1. The background pixels with value 0 do not belong to that set. The morphological theory uses the Minkowski set addition and subtraction, or alternatively the union and intersection operation, to define morphological image operators.

The morphological theory can be extended to grey values by representing an image as a function and by essentially replacing the set operations union and

intersection by respectively the maximum and minimum of functions representing the grey values of the image. The image function A has a grey value $A(\mathbf{a})$ at location $\mathbf{a}$.

The theory can be readily extended to other data than grey values, provided the maximum and minimum operation can be defined on the data. In the case of colour, this is not trivial. The use of a maximum or minimum operation requires a total ordering of the pixel values. A 3D colour space, however, is not totally ordered in a natural way. It can be ordered, but not in a unique way that makes sense in image processing. There exist several approaches to deal with this issue.

The most straightforward method is to separately totally order each colour component. Each component can be seen as a greyscale image. They are treated independently and their results are combined. A better approach is the use of a lexicographical ordering: one component is considered most important, another one is considered least important. The question that rises here is which component should be considered more important. A logical approach is to separate the luminance information from the chrominance information and order first by luminance, since most interesting differences, like edges, are obtained from the luminance component. Depending on the application and goal, one could also order first by saturation or hue.

In this thesis, we suggest a colour ordering that allows us to perform greyscale mathematical morphology on colour images. Our technique, the *majority ordering*, is based on the assumption that the frequency of the colours in the image is related to the importance of these colours. This majority ordering is therefore an image-dependent ordering.

We investigate how morphological operations perform when we use this ordering scheme. There are some problems to be tackled, which we discuss extensively. Amongst them, the number of colours present in the image has an influence, and the technique assumes a few conditions to be met concerning the image content. We mention them and look at the properties of the majority ordering.

1.2 Granulometries used in material science

Tribology is a research domain of material science where the wear of materials is investigated. This is very important research as equipment manufacturing companies want the cheapest materials with the best performance and properties. For some applications, the reliability and/or durability is extremely important. Therefore, the wear properties of materials such as polymers and composites must be tested. We want to contribute to the tribological research by investigating the usefulness of mathematical morphology in this field.

A morphological tool that we use in this domain is the *pattern spectrum*, a size distribution histogram. Such spectrum gives us quantitative information about

the shapes and sizes of the objects in the image. Next to the regular pattern spectrum, we discuss several other types of pattern spectra, each suited for computing a histogram in function of a certain object feature (such as length, orientation, size, area and/or shape).

We investigate how the algorithm for each type of pattern spectrum performs. The computation of the regular morphological pattern spectrum can be very costly. The other techniques are much faster and less dependent on the image content. An increase in speed of several orders of magnitude can be achieved.

We use these spectra for the analysis of sliding bearing materials. Such materials can be made of polymer or composite polymer. Bearings made of (composite) polymer are very useful in different areas, and the understanding of the underlying processes helps us to choose the right material for the job. Our part of this research concerns the investigation of the debris particles obtained from wear experiments and their relation to the experimental parameters, such as load and temperature. This is done using image processing techniques, namely the morphological pattern spectrum and its alternatives. We discuss the obtained results and investigate the usefulness of the spectra in this research domain. Although it is not easy to find much correlation between the spectral parameters and the experimental settings, we can correlate the behaviour of the pattern spectrum with some other experimental findings, such as transitions at certain temperatures or loads.

1.3 Image interpolation

The third topic we address, is image interpolation. When we magnify a bitmap image, the number of pixels covered by such an image increases. Interpolation is performed to compute values for the newly added pixels. Such a process can be needed in several applications: digital zooming in cameras, printing, displaying low-resolution footage on a high definition television, etc.

Several interpolation techniques exist, each with their own type of artefacts. The most common interpolation methods are fast but simple linear techniques, such as pixel replication, bilinear and bicubic interpolation. While the first one produces jagged edges, the other two give rise to a blurred result in textured areas or at edges.

Adaptive techniques incorporate prior knowledge about images to achieve better interpolation results. They are not as straightforward to implement as the linear techniques, but they are able to avoid the blur or jagged artefacts. Nevertheless, new kinds of artefacts might be introduced, such as effects of segmentation or painting, or other visual degradations. The adaptive methods can be classified, according to the method used. Edge-based techniques do not allow interpolation across the edges in the image, or interpolation has to be performed along the edges. The so-called restoration methods try to remove the artefacts obtained by interpolation with a linear technique. The example-based

approaches map blocks of the low-resolution image into pre-defined interpolated patches, or exploit the self-similarity property of an image.

These techniques work quite well on images containing natural scenes with smooth gradients, but they fail when applied to binary images or images containing sharply defined edges, such as cartoons, maps or logos. Our contribution is the introduction of an interpolation technique that is able to satisfactorily interpolate this kind of images. It uses the morphological hit-miss transform to detect jagged corners in a pixel-replicated image. A speed improvement and an extension to greyscale images is also presented. Several experiments show that this technique is superior to existing ones.

1.4 Thesis outline

The organization of this dissertation is as follows. Chapter 2 explains the theory of mathematical morphology, its power and possibilities. We use different morphological tools throughout this thesis. In this chapter, we discuss the binary and greyscale morphology, and we take a look at some of the many problems that can be solved using mathematical morphology.

In chapter 3, we discuss the extension of mathematical morphology to colour images. First, we give an introduction to colour and colour spaces. Afterwards, we discuss some colour ordering schemes that enable us to apply the theory of greyscale morphology to colour images. We also introduce our own ordering scheme, the *majority ordering*.

Chapter 4 covers the theory of granulometries. We explain what granulometries are and introduce the morphological pattern spectrum. From this pattern spectrum, several parameters can be derived. They can be used to extract properties from the objects in images, or to perform a classification. Because of the high computational cost of the pattern spectrum, several alternatives are discussed, some existing and some new. The computation times of the different techniques are compared.

In chapter 5, we investigate the applicability of the pattern spectra in the field of tribological research on composite materials and polymers. These materials are used as sliding bearings. First, we give an introduction to composite materials and tribology. Afterwards, we examine the different pattern spectra and their parameters for images containing debris particles. We correlate the spectral parameters with the parameters of the experimental set-up.

Finally, before we state some general conclusions in chapter 7, we dedicate chapter 6 to the discussion of image interpolation. First we explain the difference between vector and raster graphics. Then we discuss some standard linear interpolation techniques and state-of-the-art non-linear techniques. Afterwards, we propose a novel technique, based on mathematical morphology, that interpolates binary images. We perform several qualitative and quantitative experiments that compare our method with existing ones. To reduce the

computational cost, we introduce an adaptation of our interpolation technique which is much faster. At last, we propose an extension to greyscale images.

1.5 Contributions and publications

During the work on this thesis, several contributions have been made. The main contributions, extensively discussed in this dissertation, are:

- The development of a new ordering scheme, the *majority ordering* (MSS), that allows us to perform greyscale morphology on colour images [Ledda and Philips, 2005b, Ledda and Philips, 2005a];
- A study of the debris particles from polymers with morphological techniques (the pattern spectrum), in order to investigate the usefulness of morphological operations in the field of material science [Ledda and Philips, 2002, Ledda et al., 2003, Ledda et al., 2004, Samyn et al., 2004, Samyn et al., 2005];
- The invention of an interpolation technique, MMINT, based on mathematical morphology. It works on binary images and we also developed a version that works on greyscale images containing sharp edges [Ledda et al., 2005, Ledda et al., 2006b, Ledda et al., 2006a].

Contributions to other people's work are:

- The segmentation of flares in ultrasound images using mathematical morphology [Vansteenkiste et al., 2003];
- Research on polymers and composites [Quintelier et al., 2004, Quintelier et al., 2005];
- The development of new methods for image interpolation [Luong et al., 2006b, Luong et al., 2006a, De Witte et al., 2006].

Four papers have been published in Springer's Lecture Notes in Computer Science, of which two as first author [Ledda and Philips, 2005a, Ledda et al., 2006a] and two as co-author [Luong et al., 2006a, De Witte et al., 2006]. Another paper appeared in the journal of Materials Science Forum: [Quintelier et al., 2005]. Nine other papers appeared in the proceedings of international conferences: four as first author [Ledda et al., 2003, Ledda et al., 2004, Ledda and Philips, 2005b, Ledda et al., 2006b] and five as co-author [Vansteenkiste et al., 2003, Samyn et al., 2004, Quintelier et al., 2004, Samyn et al., 2005, Luong et al., 2006b]. Two papers were presented at a local conference [Ledda and Philips, 2002, Ledda et al., 2005], of which the last one received the best presentation award.

Chapter 2

Mathematical Morphology

The main tool used throughout this thesis is mathematical morphology. This chapter explains the theory of mathematical morphology and its power and possibilities.

The mathematical foundations of morphology are quite elaborate, but we will only introduce the concepts necessary in this thesis. Mathematical deductions and proofs are already available in literature, so the interested reader is referred to the books mentioned in the bibliography [Haralick and Shapiro, 1992, Heijmans, 1994, Nachtegael, 2002]. In this chapter, we discuss the binary and greyscale morphology, but we also extend the theory to colour images in chapter 3. We use mathematical morphology as a tool for different purposes, like the morphological pattern spectrum for the investigation of debris particles (chapters 4 and 5), and we use morphology for image interpolation (chapter 6).

Firstly, we will discuss the morphological operators and their properties for binary images. Secondly, we take a look at some extensions to greyscale morphology. We also discuss some of the many applications that can be performed using mathematical morphology.

2.1 Introduction to image processing

An image can be represented as a function $f(x, y)$, with (x, y) the coordinates of the pixels (*picture elements*) in the image [Gonzalez and Woods, 2002]. The function's output is the value of the image pixel, which is a logical value (0 or 1) for binary images, a grey value $(0, \ldots, 255)$, or a colour vector, as we will discuss in chapter 3. In image processing we alter an input image with an image processor, which is designed for a specific task, e.g. noise removal, in order to obtain an output image, e.g. a noise free image. The *image processor* Ψ is thus a

transformation of an input image $f(x, y)$ to an output image $g(x, y)$. We obtain the *input-output equation* [Goutsias and Batman, 2000, Goutsias, 2001]:

$$g(x, y) = \Psi(f(x, y)) \ . \tag{2.1}$$

The choice of image processors can be reduced by assuming two fundamental properties:

- *Distributivity*: the operator Ψ produces the same result when applied to two individual images, and then combined, or to the combination of both images;
- *Translation invariance*: the operator Ψ produces the same result when a translation is applied to the image or to the result of the operation.

The operator Ψ is called *linear* when it satisfies the following properties:

$$\Psi(f_1(x, y) + f_2(x, y)) = \Psi(f_1(x, y)) + \Psi(f_2(x, y)) \ , \tag{2.2}$$
$$\Psi(cf(x, y)) = c\Psi(f(x, y)) \ , \tag{2.3}$$

where c is a constant value. Property (2.2) is the distributivity for the linear operator. It states that the sum of two images can be processed by the operator, or that the results of the (separate) processing of those two images can be added. Both approaches return the same output image. For a translation invariant linear operator we can rewrite the input-output equation (2.1) in terms of the *convolution operator*:

$$\Psi(f)(x, y) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} h(x - \xi, y - \eta) f(\xi, \eta) d\xi d\eta \ . \tag{2.4}$$

Linear image processors are often used because of their simplicity and because the convolution equation is equivalent to a multiplication in the Fourier domain:

$$G(u, v) = H(u, v)F(u, v) \quad \text{with } -\infty \leq u, v \leq +\infty \ , \tag{2.5}$$

with $H(u, v)$ the *frequency response* of the linear operator Ψ, and u and v the spatial frequency coordinates.

Linear image operators cannot be applied to binary images, because they assume that the images are combined by standard addition. This is not possible in the binary case, since the image only allows the logical values 0 and 1, and $1 + 1 = 2$ for example is a pixel value that does not exist in binary images. The morphological operators were especially designed for binary images, using a set theoretical approach.

2.2 Binary morphology

Mathematical morphology (MM) [Serra, 1982, Haralick and Shapiro, 1992, Heijmans, 1994, Goutsias and Batman, 2000, Nachtegael, 2002, Soille, 2003] is a framework for image processing based on lattice theory and random geometry. It dates back to 1964 when it was introduced by J. Serra and G. Matheron [Matheron, 1975, Serra, 1982, Serra, 1988]. It is a tool for investigating geometric structures in binary and greyscale images. Morphological image processing can simplify images, preserve objects' essential shape characteristics, and can eliminate irrelevant objects. Further on in this chapter we will see some possible applications.

We first discuss the binary case, since originally the theory was developed for binary images. In section 2.3 we will discuss the extension to greyscale images. The theory uses operators and functionals based on topological and geometrical concepts.

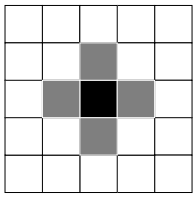
Binary images can be described in terms of sets of pixels of constant pixel value. Image pixels can assume the value 1 or 0. Thus, the image is uniquely defined by specifying the *set*:

$$A = \{\mathbf{r} \mid A_f(\mathbf{r}) = 1\} \ . \tag{2.6}$$

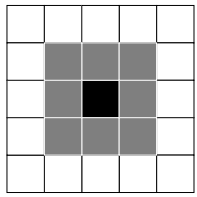
The vector $\mathbf{r}$ is the representation of the pixel coordinate (x, y). A is the set, while A_f is the binary image function that gives value 0 or 1 to the specified pixel. The term *binary image* is sometimes interchanged with the term *set*. The pixels with value 1 are foreground pixels (they are part of the set), while the background has value 0. When a binary image is displayed, usually the colours black and white are respectively used for background and foreground. Our schematic examples use black as foreground (i.e., value 1) pixels, though.

A set that represents a binary image can consist of several *objects*. Objects are connected areas of elements with pixel value 1, whereas the background elements have value 0.

We work in a discrete space, where an image is represented by pixels that are aligned on a grid. Usually this grid is rectangular. To determine if two pixels are part of the same object, a *connectivity* rule must be specified. The two most common connectivities are the 4- and 8-connectivity (figure 2.1), but 6-connectivity on a hexagonal grid is also possible. In the 4-connectivity case, a pixel is connected with another if the other pixel is one of the four nearest neighbours of this pixel. When we use the 8-connectivity, then also the four neighbouring corner pixels are connected with the pixel. Two pixels are part of the same object if it is possible to move between the two pixels using a path of connected pixels. In figure 2.2 there are one or two black objects, depending on which connectivity we assume, 8- or 4-connectivity respectively. The left black pixel and the right black pixel in the figure are only part of the same object if 8-connectivity is used.



(a) 4-connectivity



(b) 8-connectivity

Figure 2.1: Connectivity between pixels can be defined in different ways. The grey coloured pixels are connected with the black pixel.

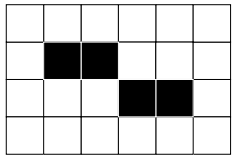
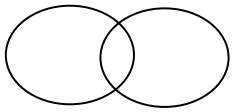
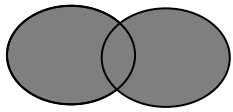


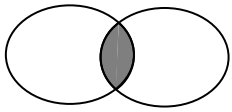
Figure 2.2: One or two objects, depending on the connectivity used. The white pixels are background.



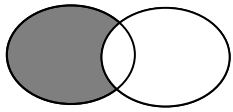
(a) Two sets A and B



(b) Set union $A \cup B$



(c) Set intersection $A \cap B$



(d) Set difference $A \backslash B$

Figure 2.3: The set operations.

The *complement* of a set A is defined as:

$$\begin{aligned} A^c &= \{\mathbf{r} \mid A_f(\mathbf{r}) = 0\} \\ &= \{\mathbf{r} \mid \mathbf{r} \notin A\} \ . \end{aligned} \tag{2.7}$$

Image A^c is obtained from A by changing black pixels into white ones, and vice versa.

We use the set operations *union* ($\cup$) and *intersection* ($\cap$), and *set difference* ($\backslash$), as illustrated in figure 2.3. The set difference $A \backslash B$ is defined by $A \cap B^c$. An empty set is represented by the symbol $\emptyset$.

Binary mathematical morphology is based on two basic operators: *dilation*

(section 2.2.1) and *erosion* (section 2.2.2). They are defined in terms of a *structuring element* (section 2.2.6). The input-output equation of a morphological image processor is denoted:

$$A' = \Psi_B(A) \ . \tag{2.8}$$

The structuring element B is an important parameter of the operator. It is also a set and thus can be thought of as a binary image, albeit a very small-sized one. By small we mean a binary image with sides below 10 pixels, while the input image can have sides of hundreds of pixels.[1] The function of the operator is determined completely by B. The shape of this structuring element reveals what kind of shapes are important in the following morphological operation. Some operations, like the hit-miss transform (section 2.4.4), need more than one structuring element.

In the following sections we will make use of the concepts of the translation and reflection. The *translation* of B by a vector $\mathbf{r}$ is defined as:

$$\begin{aligned} T_{\mathbf{r}}(B) &= \{\mathbf{b} \mid \mathbf{b} - \mathbf{r} \in B\} \\ &= \{\mathbf{b} + \mathbf{r} \mid \mathbf{b} \in B\} \ . \end{aligned} \tag{2.9}$$

A digital image is always described inside some area, defined by a number of pixels horizontally and vertically. This means the pixels inside this area have been assigned a value, but all other pixel values are unknown. A translation from the coordinates of a defined pixel can refer to a pixel with an undefined value. This must be taken into account.

Pixels outside the image area can be assigned the value 0, the so-called *zero padding*. All objects are then contained inside the visible area. A disadvantage is that we obtain an abrupt cut-off for objects that touch the border of the image. We can generalize this padding to other values, e.g., by assigning the maximum value to the outside pixels.

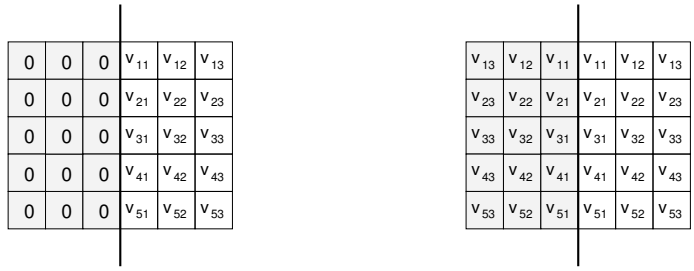
Another possibility is *border mirroring*. The image borders function as mirror lines, and the pixel values of the visible image area are reflected. Abrupt changes at the image borders are avoided this way, but the new values do not necessarily represent the real values of the undefined pixels.

Both methods are visualized in figure 2.4, where v_{xy} is a pixel value. For the morphological dilation and erosion, two different approaches are used. This will be explained in sections 2.2.1 and 2.2.2 respectively.

The *reflection* of B (around the origin) is denoted $\check{B}$ and defined as:

$$\check{B} = \{\mathbf{b} \mid -\mathbf{b} \in B\} \ . \tag{2.10}$$

[1] Actually, an image has an infinite range, but in practice the image size is limited.



(a) Zero padding (b) Border mirroring

Figure 2.4: Pixels outside the visible area (the shaded region) are given a default value.

2.2.1 Binary dilation

The binary dilation combines two sets by using the vector addition of set elements. This addition is called the *Minkowski addition*. The dilation is defined as:

$$A \oplus B = \{\mathbf{r} \mid \mathbf{r} = \mathbf{a} + \mathbf{b} \text{ for } \mathbf{a} \in A \text{ and } \mathbf{b} \in B\} \ . \tag{2.11}$$

Because the Minkowski addition is commutative, the dilation is also commutative, thus $A \oplus B = B \oplus A$. In practice, A is an image and B is the much smaller structuring element, and the operation is always described as the "dilation of the image by the structuring element", i.e., the terminology is non-symmetric.

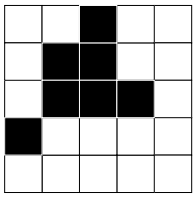
The dilation can also be explained in terms of a union operation. The definition of the dilation can therefore be rewritten as:

$$\begin{aligned} A \oplus B &= \textstyle\bigcup_{\mathbf{b} \in B} T_{\mathbf{b}}(A) \\ &= \{\mathbf{r} \mid T_{\mathbf{r}}(\check{B}) \cap A \neq \emptyset\} \ . \end{aligned} \tag{2.12}$$

Figure 2.5 shows the effect of the morphological dilation. The dilation adds pixels to the set A. The (origin of the) structuring element is positioned at every object pixel. Every pixel that is now part of the structuring element, will be part of the dilated image.

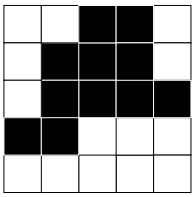
A dilation enlarges the objects in the image, by increasing the number of 1-pixels in the image. If we use a disc shaped structuring element, then the objects dilate isotropically. Figure 2.6 shows the dilation of a binary image (white pixels are part of the set) by a disc shaped element with a radius of 4 pixels. Notice that, besides the object expansion, holes are filled and contour lines appear smoother.

The dilation has some interesting properties. First of all, it is distributive with respect to the union operation. It is also translation invariant, so it has the



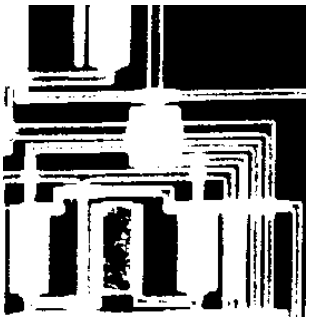
(a) A

(b) B



(c) $A \oplus B$

Figure 2.5: Binary dilation operation. The cross is the origin of the structuring element. The black blocks represent the object pixels.



(a) A

(b) $A \oplus B$

Figure 2.6: Binary dilation by B, with B a disc structuring element. White are object pixels, black is background.

properties of an image processor, according to the definition in section 2.1. The distributivity and translation invariance properties for the dilation are respectively:

$$(A_1 \cup A_2) \oplus B = (A_1 \oplus B) \cup (A_2 \oplus B) \ , \tag{2.13}$$

$$A \oplus T_{\mathbf{r}}(B) = T_{\mathbf{r}}(A \oplus B) \ . \tag{2.14}$$

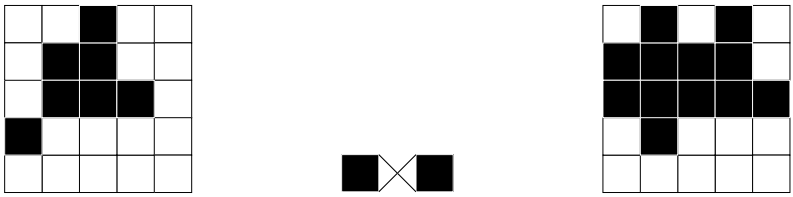
The operator is also *increasing*, which means that the dilation of a subset remains part of the dilation of the bigger set:

$$A_1 \subseteq A_2 \Rightarrow A_1 \oplus B \subseteq A_2 \oplus B \ . \tag{2.15}$$

The dilation operator is called an *extensive operator* if, for all images, the input image is part of its dilation:

$$\forall A: \ A \subseteq A \oplus B \ . \tag{2.16}$$

This is the case when the origin is part of the structuring element ($\mathbf{0} \in B$).



(a) A (b) B (c) $A \oplus B$

Figure 2.7: A binary dilation operation that is not extensive. White is background.

We then know that every pixel with value 1 keeps its value, some pixels with value 0 will change to value 1 after dilation, but no 1-pixels ever change to 0. For example, the dilation performed in figure 2.5 is extensive, but the dilation in figure 2.7 is not. The structuring element determines if the dilation is extensive.

In the given examples, the border effect must be taken into account. For the dilation, all pixels outside the image are set to 0.[2] This way, only the object pixels that are visible, i.e., explicitly defined, will contribute to the dilation. Otherwise, pixels outside the image boundaries can introduce unforeseen/unwanted object pixels in the image.

2.2.2 Binary erosion

The binary erosion is the morphological dual of the dilation. The erosion is defined in terms of the *Minkowski subtraction*:

$$A \ominus B = \{\mathbf{r} \mid \mathbf{r} + \mathbf{b} \in A, \forall \mathbf{b} \in B\} \ . \tag{2.17}$$

Unlike the dilation, the erosion is not commutative. As with the dilation, it is possible to rewrite the erosion, this time using the intersection operation:

$$\begin{aligned} A \ominus B &= \textstyle\bigcap_{\mathbf{b} \in B} T_{-\mathbf{b}}(A) \\ &= \{\mathbf{r} \mid T_{\mathbf{r}}(B) \subseteq A\} \ . \end{aligned} \tag{2.18}$$

A schematic example of the erosion can be seen in figure 2.8. The (origin of the) structuring element is translated to every object pixel, one at a time. If every pixel that is part of the translated structuring element is also part of the object, then the pixel at the location of the (origin of the) structuring element will be part of the eroded image.

[2] This zero padding of the image is the common approach for the dilation. Generally, the padding value is the minimum value a pixel can assume (0 in this case).

(a) A (b) B (c) $A \ominus B$

Figure 2.8: Binary erosion operation. The cross is the origin of the structuring element. The black blocks represent the object pixels.

(a) A (b) $A \ominus B$

Figure 2.9: Binary erosion by B, with B a disc structuring element. White are object pixels, black is background.

An erosion shrinks objects; also small objects may disappear and objects connected by a small bridge will be disconnected (figure 2.9).

The distributivity (now with the intersection operation) and the translation invariance are also properties of the erosion, which makes it an image processor, according to the definition in section 2.1. The distributivity and translation invariance properties for the erosion are respectively:

$$(A_1 \cap A_2) \ominus B = (A_1 \ominus B) \cap (A_2 \ominus B) \ , \tag{2.19}$$

$$A \ominus T_{\mathbf{r}}(B) = T_{-\mathbf{r}}(A \ominus B) \ . \tag{2.20}$$

The erosion is also increasing:

$$A_1 \subseteq A_2 \Rightarrow A_1 \ominus B \subseteq A_2 \ominus B \ . \tag{2.21}$$

The erosion is called an *anti-extensive operator* if, for all images, the erosion of the input image is part of the input image itself:

$$\forall A: \ A \ominus B \subseteq A \ . \tag{2.22}$$

This is the case when the origin is part of the structuring element ($\mathbf{0} \in B$).

In the given examples, the border effect must be taken into account. For the erosion, all pixels outside the image are set to the maximum value, i.e., value 1 for a binary image.[3] This way, only the pre-defined pixels contribute to the erosion. The border pixels are then part of a large object outside the visible area. This might lead to the preservation of these border pixels, while this is not necessarily desired. Border mirroring could also be used, but this is generally not done. Padding ensures that no pixel outside the image contributes to the dilation or erosion.

2.2.3 Binary closing and opening

The dilation and erosion are the primary building blocks of the other morphological operators. The basic operators can be combined in several different ways. The simplest combination is the chaining of one basic operator with the other. These are the *secondary morphological operators*.

2.2.3.1 Closing

The morphological closing is defined as a dilation followed by an erosion:

$$\begin{aligned} A \bullet B &= (A \oplus B) \ominus B \\ &= \{\mathbf{r} \mid \mathbf{r} \in T_{\mathbf{s}}(\breve{B}) \ \Rightarrow T_{\mathbf{s}}(\breve{B}) \cap A \neq \emptyset\} \ . \end{aligned} \tag{2.23}$$

Notice that the same structuring element B is used as well for the dilation as for the erosion. We have to remark that the morphological operators are defined with the Minkowski notation. Alternative definitions exist (see section 2.2.5), which explains why some books use a reflected structuring element for the second operation.

The erosion partially compensates for the effect of the dilation: the contour lines appear smoother after a dilation, but this effect is partially undone by the erosion. Also the swelling of the objects is cancelled by the shrinking due to the erosion step. Only, when small holes are filled, the erosion is not able to make them appear again. The closing is therefore a *smoothing filter*. Figure 2.10(a) shows an example.

Every closing is increasing, extensive and idempotent, which is respectively:

$$A_1 \subseteq A_2 \Rightarrow A_1 \bullet B \subseteq A_2 \bullet B \ , \tag{2.24}$$

$$A \subseteq A \bullet B \ , \tag{2.25}$$

$$A \bullet B = (A \bullet B) \bullet B \ . \tag{2.26}$$

[3] This padding of the image is the common approach for the erosion. Generally, the padding value is the maximum value a pixel can assume (1 in this case).

(a) $A \bullet B$ (b) $A \circ B$

Figure 2.10: Binary closing and binary opening of figure 2.9(a) by B, with B a disc structuring element. White are object pixels, black is background.

The *idempotency* is an important property (equation (2.26)). A second closing by the same structuring element produces the same image as closing the image only once.

2.2.3.2 Opening

The morphological opening is defined as an erosion followed by a dilation:

$$\begin{aligned} A \circ B &= (A \ominus B) \oplus B \\ &= \{\mathbf{r} \mid \mathbf{r} \in T_{\mathbf{s}}(B) \text{ and } T_{\mathbf{s}}(B) \subseteq A\} \ . \end{aligned} \tag{2.27}$$

The dilation will try to undo the erosion operation, but objects that disappear because of the erosion cannot be recovered by the dilation. Like the closing, the opening is a smoothing filter. An example of the binary opening is shown in figure 2.10(b).

Every opening is increasing, anti-extensive and idempotent, which is respectively:

$$A_1 \subseteq A_2 \Rightarrow A_1 \circ B \subseteq A_2 \circ B \ , \tag{2.28}$$

$$A \circ B \subseteq A \ , \tag{2.29}$$

$$A \circ B = (A \circ B) \circ B \ . \tag{2.30}$$

2.2.4 Properties

In the previous sections we mentioned some properties of the four basic morphological operators, like the distributivity and translation invariance, the monotonicity, (anti-)extensiveness and idempotency. This section contains a list of some additional properties of and relations between the different operators.

2.2.4.1 Duality

We mentioned that the erosion is the morphological dual of the dilation. This is also true for the closing and the opening. The erosion can be written in function of the dilation, or vice versa, and the opening can be written in function of the closing, and vice versa. As a consequence, the morphological theory can be described in terms of a single basic operator. The duality relations [Serra, 1982] are obtained using the set complement:

$$(A \oplus B)^c = A^c \ominus \check{B} \ , \tag{2.31}$$
$$(A \ominus B)^c = A^c \oplus \check{B} \ , \tag{2.32}$$
$$(A \bullet B)^c = A^c \circ \check{B} \ , \tag{2.33}$$
$$(A \circ B)^c = A^c \bullet \check{B} \ . \tag{2.34}$$

2.2.4.2 Distributivity and translation invariance

The distributivity property exists for the structuring elements too:

$$A \oplus (B_1 \cup B_2) = (A \oplus B_1) \cup (A \oplus B_2) \ , \tag{2.35}$$
$$A \ominus (B_1 \cup B_2) = (A \ominus B_1) \cap (A \ominus B_2) \ . \tag{2.36}$$

These properties can be turned to good account to improve the computational cost of a morphological operation. An image can be processed with different smaller structuring elements in parallel on different computer processors. The results are then combined using the distributivity property.

Some more translation invariance rules are available [Nachtegael, 2002]:

$$A \bullet T_{\mathbf{r}}(B) = A \bullet B \ , \tag{2.37}$$
$$A \circ T_{\mathbf{r}}(B) = A \circ B \ , \tag{2.38}$$
$$T_{\mathbf{r}}(A) \oplus B = T_{\mathbf{r}}(A \oplus B) \ , \tag{2.39}$$
$$T_{\mathbf{r}}(A) \ominus B = T_{\mathbf{r}}(A \ominus B) \ , \tag{2.40}$$
$$T_{\mathbf{r}}(A) \bullet B = T_{\mathbf{r}}(A \bullet B) \ , \tag{2.41}$$
$$T_{\mathbf{r}}(A) \circ B = T_{\mathbf{r}}(A \circ B) \ , \tag{2.42}$$
$$T_{\mathbf{r}}(A) \oplus T_{-\mathbf{r}}(B) = A \oplus B \ , \tag{2.43}$$
$$T_{\mathbf{r}}(A) \ominus T_{\mathbf{r}}(B) = A \ominus B \ , \tag{2.44}$$
$$T_{\mathbf{r}}(A) \bullet T_{\mathbf{r}}(B) = T_{\mathbf{r}}(A \bullet B) \ , \tag{2.45}$$
$$T_{\mathbf{r}}(A) \circ T_{\mathbf{r}}(B) = T_{\mathbf{r}}(A \circ B) \ . \tag{2.46}$$

2.2.4.3 Scaling invariance

The morphological definitions and properties are valid for discrete binary images, as well as for continuous binary images. The scaling invariance is an

exception and needs two different definitions. We first define the scaling for a continuous image.

The homogeneity properties are stated below. The *scaling* of an image B with a factor λ is:

$$H_\lambda(B) = \{\lambda \mathbf{b} \mid \mathbf{b} \in B, \lambda \neq 0\} \ , \tag{2.47}$$

with $\lambda \in \mathbb{R}$. When $\lambda = 0$, the scaled result is the origin $\mathbf{0}$.

The following property holds:

$$\breve{H}_\lambda(B) = H_{-\lambda}(B) = H_\lambda(\breve{B}) \ . \tag{2.48}$$

For the special case $\lambda = -1$ we have:

$$H_{-1}(B) = H_1(\breve{B}) = \breve{B} \ . \tag{2.49}$$

The homogeneity properties state that absolute size does not matter, only relative size and shape:

$$H_\lambda(A) \oplus H_\lambda(B) = H_\lambda(A \oplus B) \ , \tag{2.50}$$
$$H_\lambda(A) \ominus H_\lambda(B) = H_\lambda(A \ominus B) \ , \tag{2.51}$$
$$H_\lambda(A) \bullet H_\lambda(B) = H_\lambda(A \bullet B) \ , \tag{2.52}$$
$$H_\lambda(A) \circ H_\lambda(B) = H_\lambda(A \circ B) \ . \tag{2.53}$$

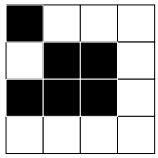
With $\lambda = -1$ we obtain the following special cases:

$$\breve{A} \oplus \breve{B} = (A \oplus B)\breve{} \ , \tag{2.54}$$
$$\breve{A} \ominus \breve{B} = (A \ominus B)\breve{} \ , \tag{2.55}$$
$$\breve{A} \bullet \breve{B} = (A \bullet B)\breve{} \ , \tag{2.56}$$
$$\breve{A} \circ \breve{B} = (A \circ B)\breve{} \ . \tag{2.57}$$

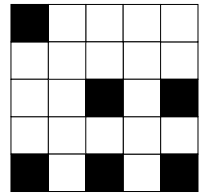
Equation (2.47) is the general definition for continuous sets. If we consider a discrete image, then another type of scaling applies. First of all, the pixels of a digital image are aligned on a grid, so the pixels of the scaled image must have discrete coordinates. Therefore, λ is restricted to an integer. Secondly, the magnification of a discrete object results in a set of disconnected pixels (figure 2.11). To avoid this, the objects must be dilated instead of magnified. The definition for scaling by factor n (equation (2.47)) is changed into:

$$nB = \underbrace{B \oplus B \oplus \cdots \oplus B}_{n \text{ times}} \ . \tag{2.58}$$

When $n = 0$, the scaled result is the origin $\mathbf{0}$.



(a) A



(b) nA, with $n = 2$

Figure 2.11: Connected pixels become disconnected after scaling, when using definition (2.47) for a discrete image.

2.2.4.4 Commutativity and associativity

The Minkowski addition is a commutative operation, which implies that the dilation is also commutative: $A \oplus B = B \oplus A$. We discussed this in section 2.2.1. Because in practice A is an input image and B is a small structuring element, the commutativity rule has no practical purpose. More useful are the following equations. They state that the order in which different structuring elements are used, is irrelevant:

$$(A \oplus B_1) \oplus B_2 = (A \oplus B_2) \oplus B_1 \ , \tag{2.59}$$

$$(A \ominus B_1) \ominus B_2 = (A \ominus B_2) \ominus B_1 \ . \tag{2.60}$$

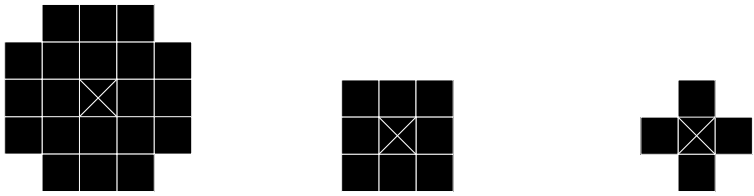
The associativity equations are:

$$(A \oplus B_1) \oplus B_2 = A \oplus (B_1 \oplus B_2) \ , \tag{2.61}$$

$$(A \ominus B_1) \ominus B_2 = A \ominus (B_1 \oplus B_2) \ . \tag{2.62}$$

The associativity rules are important to know. Equation (2.61) states that the dilation of an image, once by B_1 and the result by B_2, is equivalent to one dilation of the image by the bigger structuring element $B_1 \oplus B_2$. The first calculation is much less computational intensive than the second: a sequence of morphological operations with small structuring elements is more advantageous than one operation with one large structuring element. Indeed, a non-optimized algorithm would need to dilate or erode every pixel in the image, for example $M \times N$ pixels. If the set of the structuring element B contains P pixels, then the number of operations would be $MN(P-1)$. If we can write this structuring element B as $B_1 \oplus B_2 \oplus \ldots \oplus B_p$, i.e., we can *decompose* this structuring element into p smaller elements with size P_i $(i = 1, \ldots, p)$, then the number of operations would be $MN(\sum_i (P_i - 1))$, which is often smaller than $MN(P-1)$.

A 5×5 square structuring element can be decomposed into two 3×3 square structuring elements. The number of operations decreases from $24MN$ to



(a) B_c (b) B_s (c) B_r

Figure 2.12: Structuring element decomposition: $B_c = B_s \oplus B_r$.

$16MN$. For larger structuring elements the difference in number of operations is even more clear.

Another example is the circle shaped structuring element, shown in figure 2.12(a). It contains 21 pixels and the computational cost is $20MN$ operations. Its decomposition into a square (9 pixels) and a cross element (5 pixels) has a cost of $12MN$ operations.

2.2.4.5 Idempotency

The idempotency is a useful property of the secondary basic morphological operators. We repeat equations (2.26) and (2.30):

$$A \bullet B = (A \bullet B) \bullet B \ , \tag{2.63}$$
$$A \circ B = (A \circ B) \circ B \ . \tag{2.64}$$

A consequence of the idempotency is the invariance of the dilation under opening and the erosion under closing [Haralick and Shapiro, 1992]:

$$A \oplus B = (A \oplus B) \circ B \ , \tag{2.65}$$
$$A \ominus B = (A \ominus B) \bullet B \ . \tag{2.66}$$

Notice that the above equations can respectively be rewritten as:

$$A \oplus B = (A \bullet B) \oplus B \ , \tag{2.67}$$
$$A \ominus B = (A \circ B) \ominus B \ . \tag{2.68}$$

If the closing or opening by a certain structuring element has no effect on the input image, then the image is said to be respectively *B-closed* or *B-open*:

$$A \bullet B = A \ , \tag{2.69}$$
$$A \circ B = A \ . \tag{2.70}$$

2.2.4.6 Union and intersection

Note that the following properties can be generalized: instead of a combination of two sets (a union or intersection of images or structuring elements), more sets can be combined.[4]

$$(A_1 \cap A_2) \oplus B \subseteq (A_1 \oplus B) \cap (A_2 \oplus B) \ , \tag{2.71}$$
$$(A_1 \cup A_2) \ominus B \supseteq (A_1 \ominus B) \cup (A_2 \ominus B) \ , \tag{2.72}$$
$$A \oplus (B_1 \cap B_2) \subseteq (A \oplus B_1) \cap (A \oplus B_2) \ , \tag{2.73}$$
$$A \ominus (B_1 \cap B_2) \supseteq (A \ominus B_1) \cup (A \ominus B_2) \ , \tag{2.74}$$
$$(A_1 \cup A_2) \bullet B \supseteq (A_1 \bullet B) \cup (A_2 \bullet B) \ , \tag{2.75}$$
$$(A_1 \cup A_2) \circ B \supseteq (A_1 \circ B) \cup (A_2 \circ B) \ , \tag{2.76}$$
$$(A_1 \cap A_2) \bullet B \subseteq (A_1 \bullet B) \cap (A_2 \bullet B) \ , \tag{2.77}$$
$$(A_1 \cap A_2) \circ B \subseteq (A_1 \circ B) \cap (A_2 \circ B) \ . \tag{2.78}$$

2.2.4.7 Adjunction

Another form of duality between the dilation and erosion is the following property:

$$A_1 \oplus B \subseteq A_2 \Leftrightarrow A_1 \subseteq A_2 \ominus B \ . \tag{2.79}$$

The couple $(\oplus, \ominus)$ is called an *adjunction*. Note that this equation states that the erosion is not the exact inverse of the dilation: the erosion generally does not completely undo the effect of the dilation, and vice versa. If this had been the case, then the closing and opening would both be identity operators, since they are a dilation followed by an erosion, and an erosion followed by a dilation, respectively.

2.2.4.8 Monotonicity

We have seen before that the four basic morphological operators are *monotonic*, i.e., they are entirely non-increasing or non-decreasing.[5] To be more specific, they are all (strictly) increasing (i.e., they do not remain constant or decrease). From the point of view of the structuring element, the dilation and erosion are monotonic too. The dilation is increasing in its second argument, but the erosion is decreasing.

$$B_1 \subseteq B_2 \Rightarrow A \oplus B_1 \subseteq A \oplus B_2 \ , \tag{2.80}$$

[4] The properties can be rewritten using $\bigcup_i A_i$ and $\bigcap_i A_i$ instead of $A_1 \cup A_2$ and $A_1 \cap A_2$, respectively.

[5] A function f is *non-increasing* if: $\forall a, b : a < b \Rightarrow f(a) \geq f(b)$. A function f is *non-decreasing* if: $\forall a, b : a < b \Rightarrow f(a) \leq f(b)$.

$$B_1 \subseteq B_2 \Rightarrow A \ominus B_1 \supseteq A \ominus B_2 \ . \tag{2.81}$$

2.2.4.9 Inequalities

A dilation followed by an erosion (closing) is not the same as an erosion followed by a dilation (opening). The same is true for the combination of the closing with the opening:

$$(A \circ B) \bullet B \neq (A \bullet B) \circ B \ . \tag{2.82}$$

Some other inequalities [Nachtegael, 2002]:

$$A \oplus (B_1 \ominus B_2) \subseteq (A \oplus B_1) \ominus B_2 \ , \tag{2.83}$$
$$(A \ominus B_1) \oplus B_2 \subseteq (A \oplus B_2) \ominus B_1 \ , \tag{2.84}$$
$$A \ominus B \subseteq A \oplus \breve{B} \ , \tag{2.85}$$
$$A \circ B \subseteq A \bullet B \ . \tag{2.86}$$

Notice that the last equation is a special case of equation (2.84), with $B_1 = B_2$. We can also derive equation (2.86) from the extensivity (equation (2.25)) and anti-extensivity (equation (2.29)) property of the closing and opening, respectively. Equation (2.83) shows that the associativity rule, that holds for the dilation and in a slightly different form for the erosion (see subsection 2.2.4.4), is not extendible to combinations of the dilation and erosion. Equation (2.84) shows something similar regarding the commutativity rules (see subsection 2.2.4.4).

If the origin is part of the structuring element ($\mathbf{0} \in B$), then we have some more relations between the different operators; they follow directly from the extensiveness and anti-extensiveness properties:

$$A \ominus B \subseteq A \subseteq A \oplus B \ , \tag{2.87}$$
$$A \bullet B \subseteq A \oplus B \ , \tag{2.88}$$
$$A \circ B \subseteq A \oplus B \ , \tag{2.89}$$
$$A \ominus B \subseteq A \circ B \ , \tag{2.90}$$
$$A \ominus B \subseteq A \bullet B \ . \tag{2.91}$$

In all the above equations, it is allowed to replace the structuring element B by its reflective counterpart $\breve{B}$, on only one or both sides of the equation.

As a conclusion, we mention the following relationship between the different operators, assuming the condition $\mathbf{0} \in B$ is fulfilled:

$$A \ominus B \subseteq A \circ B \subseteq A \subseteq A \bullet B \subseteq A \oplus B \ . \tag{2.92}$$

When $\mathbf{0} \in B$, the erosion and opening are anti-extensive operators, the dilation and closing are extensive operators. We can also conclude that, for the same

structuring element, the erosion removes most pixels, while the dilation adds most pixels.

2.2.5 Alternative definitions

In the literature it is possible to find alternative definitions for the basic morphological operators. We can distinguish two frequently used definitions: the *Serra definition* and the *Minkowski definition* (the one used in this thesis). The difference between these notations is whether or not the reflection of the structuring element is used.

The definitions according to Serra are:

$$\textbf{Dilation} : A \oplus B = \bigcup_{\mathbf{b}\in B} T_{-\mathbf{b}}(A) \ , \tag{2.93}$$

$$\textbf{Erosion} : A \ominus B = \bigcap_{\mathbf{b}\in B} T_{-\mathbf{b}}(A) \ , \tag{2.94}$$

$$\textbf{Closing} : \ A \bullet B = (A \oplus B) \ominus \breve{B} \ , \tag{2.95}$$

$$\textbf{Opening} : \ A \circ B = (A \ominus B) \oplus \breve{B} \ . \tag{2.96}$$

The Serra definitions $(\cdot_S)$ are related to the Minkowski definitions $(\cdot_M)$ in the following way:

$$(A \oplus B)_M = (A \oplus \breve{B})_S \ , \tag{2.97}$$

$$(A \ominus B)_M = (A \ominus B)_S \ , \tag{2.98}$$

$$\left.\begin{matrix}(A \bullet B)_M \\ ((A \oplus B)_M \ominus B)_M\end{matrix}\right\} = \begin{cases}(A \bullet \breve{B})_S \\ ((A \oplus \breve{B})_S \ominus B)_S\end{cases} \ , \tag{2.99}$$

$$\left.\begin{matrix}(A \circ B)_M \\ ((A \ominus B)_M \oplus B)_M\end{matrix}\right\} = \begin{cases}(A \circ B)_S \\ ((A \ominus B)_S \oplus \breve{B})_S\end{cases} \ . \tag{2.100}$$

In equation (2.99) we used the property $\breve{\breve{B}} = B$. Note that the definition of the erosion remains the same, but for the dilation a reflected structuring element is used.

When the structuring element B is symmetric, then $B = \breve{B}$ and the Serra and Minkowski definition generate the same result.

2.2.6 The structuring element

The structuring element is a probe that scans the image and alters the pixels based on its content. Just like the input image, it is a binary one. The task of the structuring element is to alter the input image in a certain way, and this by taking into account local information. The structuring element is therefore in most cases a very small set, not only in number of pixels (or area in the

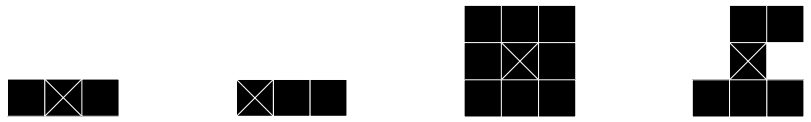


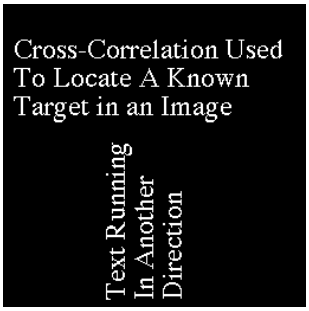
Figure 2.13: A few examples of structuring elements. The cross is the origin.

continuous case), but the pixels are also close to each other. A typical (discrete) structuring element consists of a few pixels (less than 10) that are connected to each other. A typical (discrete) input image has hundreds by hundreds or thousands by thousands pixels. When we work with discrete images, the structuring element is a small rectangular window with certain pixels set to 1 and the others to 0. Figure 2.13 shows a few examples of structuring elements. They can be linear, symmetric, have a specific shape, be tilted, have different origins, ... The shape of the element is chosen in function of the task it has to perform.

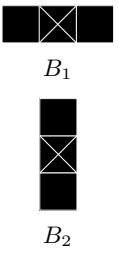
The crosses in the examples resemble the origin of the structuring element. The location of this origin is very important. It sets the reference position of the window, and thus the structuring element, when we move it over the image pixels.

In figure 2.14 the same image has undergone a morphological opening operation with two different structuring elements. In figure (c) an elongated horizontal element is used, in figure (d) an elongated vertical element is used. In figure (c) the small vertically oriented objects in the image are removed and the small horizontally oriented objects are kept. In figure (d) the opposite is true. Anisotropic structuring elements are therefore useful for the extraction of orientation-dependent information from the image.

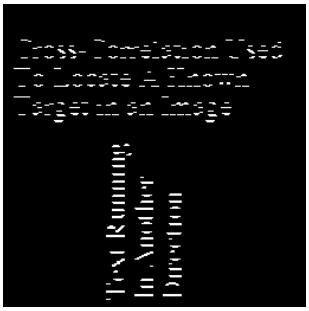
Structuring elements can also be combined. Small and simple structuring elements can be used as the building blocks to construct a larger and more complex structuring element. It is then possible to perform the morphology, either by using the large structuring element or by using a sequence of small elements. This is a consequence of the associativity relations mentioned in section 2.2.4.4, where a big structuring element is the dilation of one small structuring element with another (or a sequence of dilations). Some examples of combinations are shown in figure 2.15. As discussed before, we emphasize that a sequence of morphological operations using small structuring elements is computationally more efficient than one operation using a large structuring element. It is, however, possible that such a sequence requires more computation time, if the algorithms for the basic morphological operations are optimized. For example, the commercial package Matlab needs about 3.5 seconds to dilate a binary image (size 280×272) 50 times successively by a 3×3 structuring element, and only 1/100 of that time to perform the dilation by the composition of those elements, i.e., a 101×101 square.



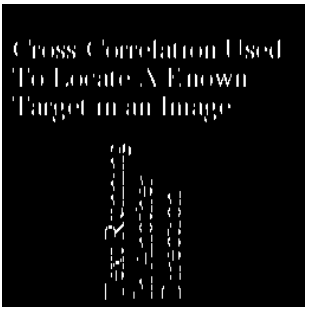
(a) A



(b) B_1 and B_2



(c) $A \circ B_1$

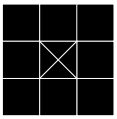


(d) $A \circ B_2$

Figure 2.14: Morphological opening by different structuring elements.

(a) B_1

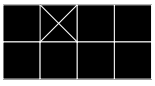
(b) B_2



(c) $B_1 \oplus B_2$

(d) B_3

(e) B_4



(f) $B_3 \oplus B_4$

Figure 2.15: Structuring elements can be combined.

2.3 Extension to greyscale images

The theory of mathematical morphology was developed for binary images. Union and intersection operations are the operations used for the construction of the morphological operators. These are logical operations working on logical images: a point in the space, i.e., a pixel in the discrete image, is part of the set or it is not. This is the equivalent of saying that a pixel has the logical value 1 or 0.

Greyscale images are a different matter. While a binary image is represented by the colours black and white, a greyscale image consists of different shades of grey. In image processing, the range of these grey values is usually between 0 and 255, taking only integer values, with 0 being the darkest shade (black) and 255 being the lightest shade (white). This results in an 8-bit image, with $2^8 = 256$ possible grey values. The logical union and intersection cannot be used like in the binary case. An extension of the morphological theory is therefore imposed.

We keep in mind that a binary image is a special case of a greyscale image, but with only two levels of grey. The extension will therefore be quite natural: by replacing the union (or logical OR) and intersection (or logical AND) with the MAX en MIN operations, binary morphology will be obtained as a specific case of greyscale morphology.

This section will discuss two approaches for extension: the *threshold approach* and the *umbra approach*.[6]

2.3.1 The threshold approach

A greyscale image can be represented by the collection of cross sections of the image. The *cross section* (or *level set*) at level t is a set defined as:

$$A_{\times}(t) = \{\mathbf{r} \mid A(\mathbf{r}) \geq t\} \ . \tag{2.101}$$

t is the threshold value, which can be any possible grey value. The image A should now be seen as a function: the function value at coordinate $\mathbf{r}$ (for a two-dimensional image this is (x, y)) is the grey value $A(\mathbf{r})$. Figure 2.16 shows a one-dimensional function $A(x)$ that is thresholded at level t.

The *threshold decomposition* is the collection of all non-empty cross sections:

$$\mathcal{A} = \{A_{\times}(t) \mid A_{\times}(t) \neq \emptyset\} \ . \tag{2.102}$$

Notice that we can reconstruct the original image from its cross sections:

$$A(\mathbf{r}) = \max\{t \mid \mathbf{r} \in A_{\times}(t)\} \ . \tag{2.103}$$

[6] In this thesis it is implicitly assumed that the *threshold approach* is used, unless stated otherwise.

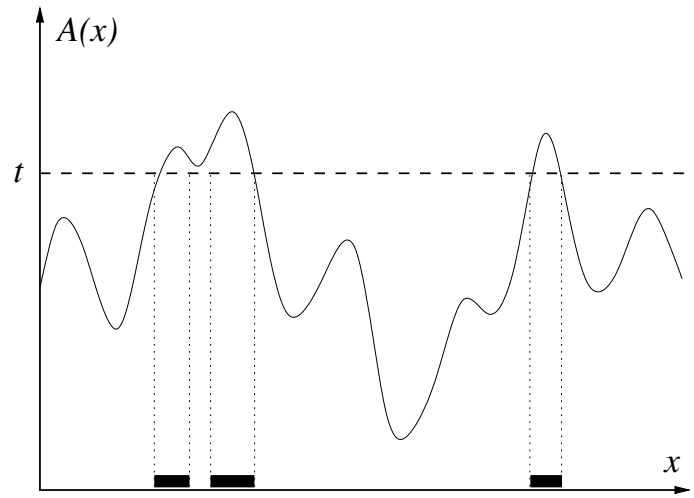


Figure 2.16: The cross section $A_{\times}(t)$. For the sake of simplicity, we show a one-dimensional image: the coordinates are the abscissa values, the grey values are the ordinate values. The coordinate values between the dotted lines are part of the cross section for the given t.

The principle of the threshold approach (sometimes referred to as the t-approach) is to perform the binary morphological operations on the cross sections of the image/function A. Every cross section is a binary image, the structuring element is binary too. The basic morphological operators are defined in a similar way, except that the union is replaced by MAX and intersection by MIN. Actually, the replacements would be the supremum and infimum, but this mathematical correctness only makes sense when continuous images are used. Since in practice in image processing discrete images are used, we will further use the maximum and minimum.

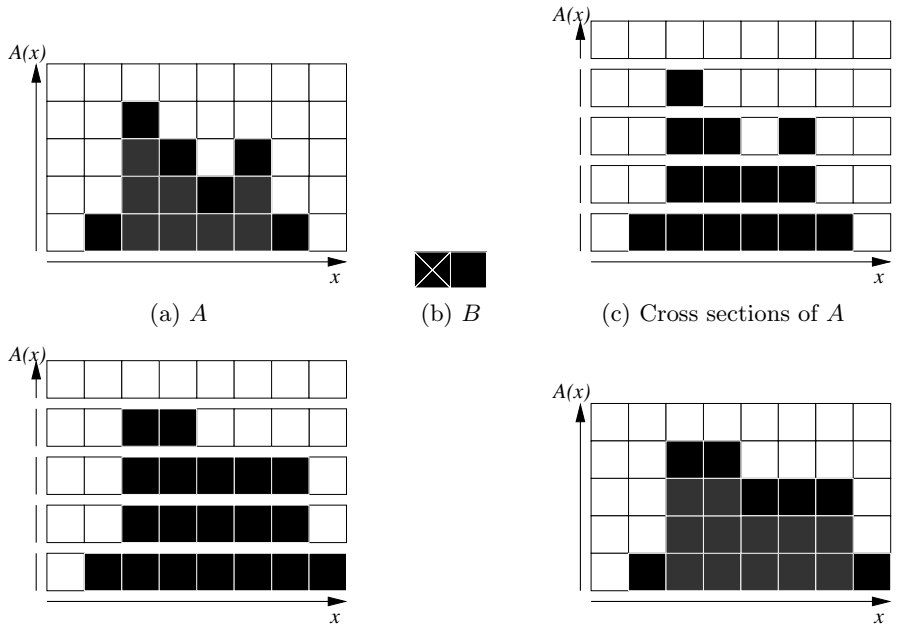
2.3.1.1 Greyscale t-dilation

As in the binary case, the dilation has the properties of an image processor, i.e., it is distributive and translation invariant. The dilation is defined as:

$$\begin{aligned}(A \oplus B)(\mathbf{a}) &= \max\{A(\mathbf{r}) \mid \mathbf{a} - \mathbf{r} \in B\} \\ &= \max\{A(\mathbf{r}) \mid \mathbf{r} = \mathbf{a} - \mathbf{b}, \mathbf{b} \in B\} \ .\end{aligned} \tag{2.104}$$

This greyscale dilation has the same complexity as a convolution operation (section 2.1). However, the summation of products is replaced by a maximum of sums.

The threshold or flat dilation can be visualized by taking every cross section from the threshold decomposition as an input image. The input greyscale im-



(a) A (b) B (c) Cross sections of A

(d) $A \oplus B$ on cross sections (e) $A \oplus B$

Figure 2.17: Greyscale t-dilation on a one-dimensional greyscale image. The abscissa is the pixel coordinate, the ordinate the grey value or t-value.

age, which is in fact a function, is now transformed into slices of binary images. On every cross section the binary morphological dilation by a binary structuring element is applied. The results are then combined using equation (2.103). An example is shown in figure 2.17.

2.3.1.2 Greyscale t-erosion

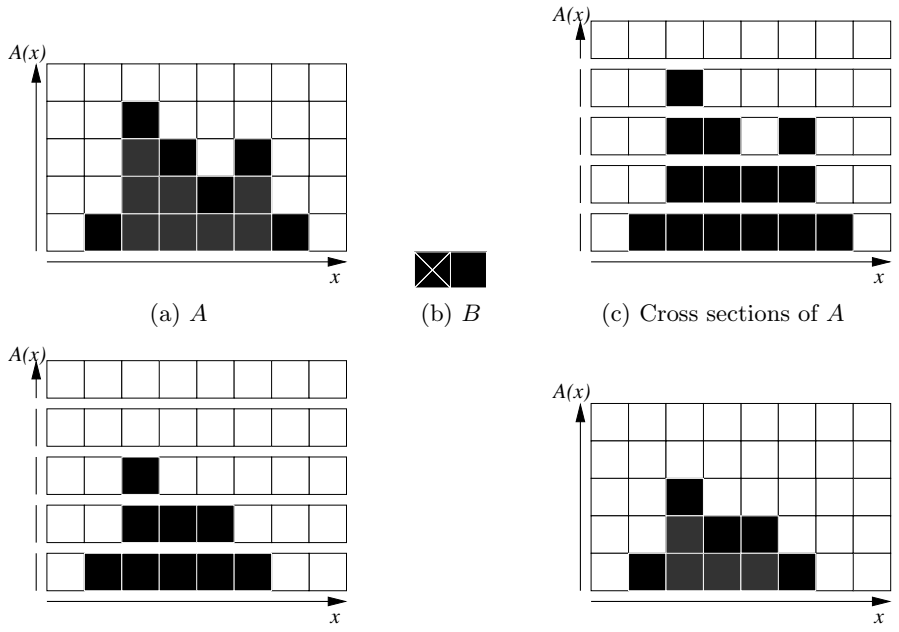
The t-erosion (figure 2.18) is defined in a similar way as the greyscale t-dilation:

$$\begin{aligned}(A \ominus B)(\mathbf{a}) &= \min\{A(\mathbf{r}) \mid \mathbf{r} - \mathbf{a} \in B\} \\ &= \min\{A(\mathbf{r}) \mid \mathbf{r} = \mathbf{a} + \mathbf{b}, \mathbf{b} \in B\} \ .\end{aligned} \tag{2.105}$$

2.3.1.3 Greyscale t-closing and t-opening

The secondary basic operators for greyscale morphology with the threshold approach are defined like their binary equivalents. In this case, they are a MAX operation followed by a MIN operation, and vice versa:

$$\textbf{Closing} : A \bullet B = (A \oplus B) \ominus B \ , \tag{2.106}$$



(a) A (b) B (c) Cross sections of A

(d) $A \ominus B$ on cross sections (e) $A \ominus B$

Figure 2.18: Greyscale t-erosion on a one-dimensional greyscale image. The abscissa is the pixel coordinate, the ordinate the grey value or t-value.

$$\textbf{Opening} : A \circ B = (A \ominus B) \oplus B \ . \tag{2.107}$$

An example on a real greyscale image is shown for all four operators in figure 2.19. The swelling or shrinking of the image objects and the filling of the holes or the disconnection of thin lines is now translated to the greyscale domain. A dilation increases the overall brightness of the image, while the erosion darkens the image. A hole in a greyscale image is a small dark area surrounded by lighter grey values. A closing operation would then increase the greyscale value of the pixels of the hole. The opposite happens with the opening operation.

2.3.1.4 Properties using the threshold approach

The t-dilation and t-erosion are also distributive and translation invariant, just as in the binary case. All operators are still increasing. The dilation and closing are extensive, the erosion and opening anti-extensive. The secondary operators are idempotent.

All other properties are extendable to the greyscale case. The commutativity is an exception: in the binary case, both A and B are binary sets, but in

(a) A

(b) $A \oplus B$

(c) $A \ominus B$

(d) $A \bullet B$

(e) $A \circ B$

Figure 2.19: Greyscale morphology using the threshold approach. B is a disc structuring element (with a radius of 4 pixels).

the greyscale extension with the threshold approach, the input image A is a greyscale image and the structuring element B is considered to be a flat, binary, element. In definition (2.104), A and B cannot be switched.

An extra property is the *greyscale translation invariance*. With the (spatial) translation invariance, the absolute position of the image content is irrelevant. Now, it is also possible to increase or decrease the grey values with a constant value. We define the *greyscale translation* of image A by a scalar value v as:

$$T_v(A) = \{A(\mathbf{r}) + v\} \ . \tag{2.108}$$

This greyscale translation T_v may not be confused with the spatial translation $T_{\mathbf{r}}$. Similar properties can now be written down like the equations in section 2.2.4.2. The *greyscale translation invariance* for the basic operators is then:

$$T_v(A) \oplus B = T_v(A \oplus B) \ , \tag{2.109}$$
$$T_v(A) \ominus B = T_v(A \ominus B) \ , \tag{2.110}$$
$$T_v(A) \bullet B = T_v(A \bullet B) \ , \tag{2.111}$$
$$T_v(A) \circ B = T_v(A \circ B) \ . \tag{2.112}$$

The morphological t-operators produce the same results on a binary image as the binary morphology. In that case, background pixels have grey value 0 and object pixels have grey value 1.

2.3.2 The umbra approach

The umbra approach, or u-approach, is an extension which allows greyscale structuring elements, but which introduces a new problem, as we will discuss later.

This approach introduces a few new concepts that transform an image set into an image function, or vice versa. Let us assume an N-dimensional set A_s. The *top surface* (or *top*) is defined by:

$$T[A_s](\mathbf{r}) = \max\{z \mid (\mathbf{r}, z) \in A_s\} \ . \tag{2.113}$$

An example is shown in figure 2.20 for a two-dimensional set. The first $N-1$ coordinates of such a set are the coordinates in the spatial domain of the set, the Nth coordinate is for the so-called *surface* of A_s. For our purposes we have $N = 3$, with the first two coordinates the (x, y)-coordinates of the pixels.

In practice, when working with greyscale images, the input image A is already a top surface: $A(\mathbf{r}) = T[A_s](\mathbf{r}) = z$. A is the functional representation of the image, A_s is the three-dimensional set representation of the image.

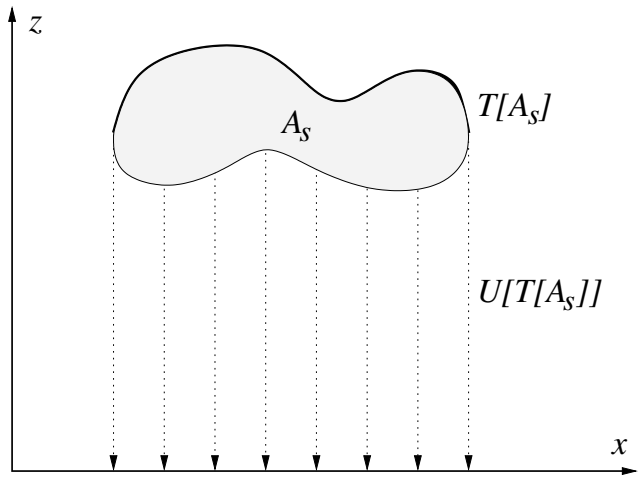


Figure 2.20: A set A_s in two dimensions (one spatial coordinate and one surface coordinate) and its top surface $T[A_s]$ and umbra $U[T[A_s]]$.

The *umbra* (sometimes referred to as the *subgraph*) of the image function A is:

$$U[A] = \{(\mathbf{r}, z) \mid z \leq A(\mathbf{r})\} \ . \tag{2.114}$$

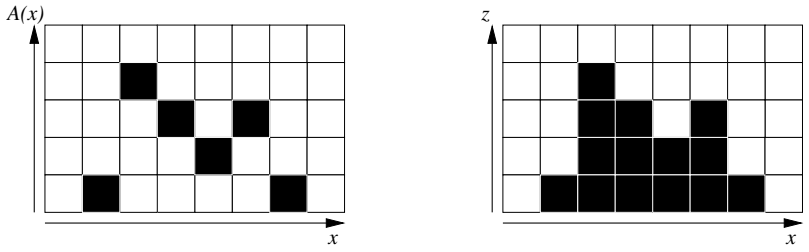
The umbra (figure 2.20) is thus a set containing the top surface of a function A and everything "below" that top surface if we interpret $z = A(\mathbf{r})$ as a height of a topographic landscape at position $\mathbf{r}$. The umbra operation drops a shadow on everything on and below the top surface, hence the name umbra, which means shade or shadow. The image A, with grey values $A(\mathbf{r})$, is transformed into a set $U[A]$. This set is N-dimensional, with $N-1$ coordinates that constitute the spatial domain and the Nth coordinate the surface of the set.

The umbra of the one-dimensional function $T[A_s]$ is visualized in figure 2.20. The schematic example in figure 2.21 illustrates the difference between the function and its umbra.

The definitions of the top surface and the umbra allow us to define the extension of mathematical morphology to greyscale images. With this umbra approach it is possible to use greyscale structuring elements. The morphological operations are performed on the umbrae of the images. The structuring element used is an umbra version of the original structuring element (figure 2.22).

2.3.2.1 Greyscale u-dilation

The greyscale image A and the greyscale structuring element are functions. We take the umbra of both images and perform a binary dilation between the umbrae. The dilated result is a set, which is transformed to a greyscale image,



(a) A (b) $U[A]$

Figure 2.21: A one-dimensional greyscale image and its umbra. The ordinate is the grey value in figure (a) and an extra dimension in figure (b).

(a) B (b) $U[B]$

Figure 2.22: A one-dimensional greyscale structuring element and its umbra.

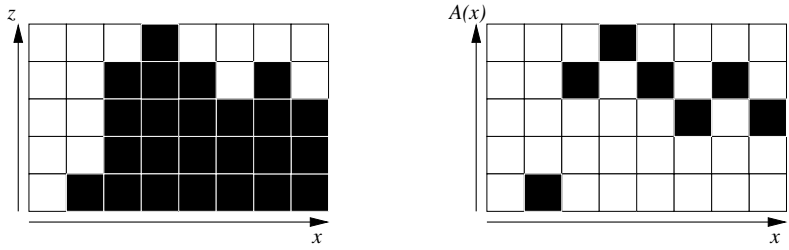
by taking the top surface:

$$A \oplus B = T[U[A] \oplus U[B]] \ , \tag{2.115}$$

$$(A \oplus B)(\mathbf{a}) = \max\{A(\mathbf{a} - \mathbf{b}) + B(\mathbf{b}) \mid \mathbf{b} \in \Omega(B)\} \ , \tag{2.116}$$

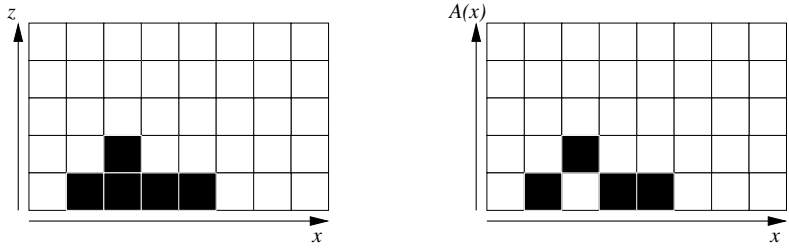
with $\Omega(B)$ the *support* of B. The support is the set of all pixels $\mathbf{r}$ in the image (function) that are defined. In practice, the undefined pixels are given value $-\infty$, and by convention $s + t = -\infty$ if $s = -\infty$ or $t = -\infty$, and $s - t = +\infty$ if $s = +\infty$ or $t = -\infty$ are used in case of ambiguity.

The one-dimensional example (figure 2.23) illustrates what happens: the image and the structuring element are one-dimensional, i.e., every pixel has one spatial coordinate with a respective grey value. The umbra transforms the one-dimensional greyscale image into a set, which is a two-dimensional binary image. The same applies for the structuring element. Now we can dilate the two-dimensional binary image by a two-dimensional structuring element. The result is also a two-dimensional binary image. By taking the top surface, we transform the set into a greyscale image, which is one-dimensional. This explanation remains valid for higher dimensions (but is not so easy to visualize).



(a) $U[A] \oplus U[B]$ (b) $T[U[A] \oplus U[B]] = A \oplus B$

Figure 2.23: Greyscale u-dilation on a one-dimensional greyscale image. A and B are shown in figures 2.21 and 2.22.



(a) $U[A] \ominus U[B]$ (b) $T[U[A] \ominus U[B]] = A \ominus B$

Figure 2.24: Greyscale u-erosion on a one-dimensional greyscale image. A and B are shown in figures 2.21 and 2.22.

2.3.2.2 Greyscale u-erosion

The u-erosion (figure 2.24) is defined in a similar way as the greyscale dilation:

$$A \ominus B = T[U[A] \ominus U[B]] \ , \tag{2.117}$$

$$(A \ominus B)(\mathbf{a}) = \min\{A(\mathbf{a}+\mathbf{b}) - B(\mathbf{b}) \mid \mathbf{b} \in \Omega(B)\} \ . \tag{2.118}$$

2.3.2.3 Greyscale u-closing and u-opening

The secondary basic operators for greyscale morphology with the umbra approach are defined like their binary equivalents:

$$\textbf{Closing} : A \bullet B = (A \oplus B) \ominus B \ , \tag{2.119}$$

$$\textbf{Opening} : A \circ B = (A \ominus B) \oplus B \ . \tag{2.120}$$

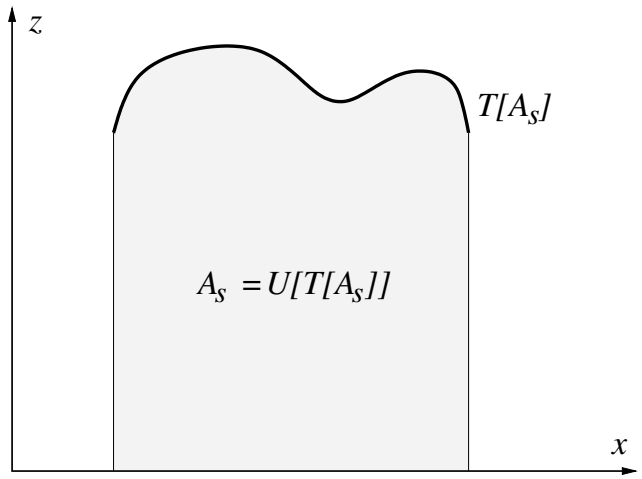


Figure 2.25: A set A_s in two dimensions (one spatial coordinate and one surface coordinate) is an umbra. Its top surface $T[A_s]$ and umbra $U[T[A_s]]$ are each other's inverse.

2.3.2.4 Properties using the umbra approach

The top surface operator is the *left inverse* of the umbra operator, so the top surface operation always cancels the umbra operation:

$$T[U[A]] = A \ . \tag{2.121}$$

They are not real inverses, because the same is not true for the opposite operation, i.e., the umbra is not the left inverse of the top surface. All we can say is that:

$$A_s \subseteq U[T[A_s]] \ . \tag{2.122}$$

When set A_s is an umbra, then the equality is true. In this case the umbra and the top surface are each other's inverse. Figure 2.25 illustrates this. Set A_s is already an umbra. The umbra of the top surface of A_s results back in A_s.

The *umbra homomorphism theorem* states:

$$U[A \oplus B] = U[A] \oplus U[B] \ , \tag{2.123}$$
$$U[A \ominus B] = U[A] \ominus U[B] \ . \tag{2.124}$$

A homomorphism is a structure-preserving map between two algebraic structures. In this theorem, the umbra of the result of a morphological dilation or erosion, using the greyscale morphology on functions, is the same as the binary

dilation or erosion with the umbrae (thus sets) as arguments. The homomorphism is also valid for the closing and opening:

$$U[A \bullet B] = U[A] \bullet U[B] \ , \tag{2.125}$$
$$U[A \circ B] = U[A] \circ U[B] \ . \tag{2.126}$$

This method makes it possible to extend properties for binary morphology to greyscale morphology. Most properties, like the duality, associativity, union/intersection, ... remain valid in the umbra approach. In contrast to the threshold approach, the commutativity rule still applies. The top surface and umbra are also monotone (and increasing) functions:[7]

$$A_1 \leq A_2 \Leftrightarrow U[A_1] \subseteq U[A_2] \ , \tag{2.127}$$
$$A_{s1} \subseteq A_{s2} \Rightarrow T[A_{s1}] \leq T[A_{s2}] \ . \tag{2.128}$$

And when A_{s1} and A_{s2} are umbrae, equation (2.128) becomes:

$$A_{s1} \subseteq A_{s2} \Leftrightarrow T[A_{s1}] \leq T[A_{s2}] \ . \tag{2.129}$$

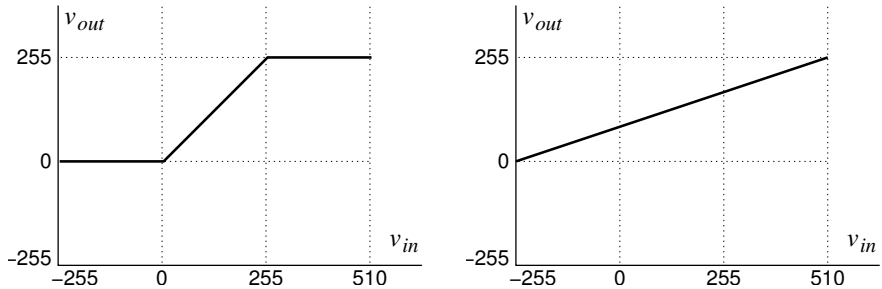
2.3.3 Comparison of the t- and u-approaches

In the threshold approach we take the cross sections of the greyscale image and use a binary morphological operator on every cross section. After reconstruction we get the resulting greyscale image. The structuring element used is *flat* (i.e., binary).

With the umbra approach we perform binary morphology on the umbra of the image and the structuring element. The top surface of the result is the output greyscale image. It is now possible to use *non-flat* (i.e., greyscale) structuring elements. The grey values of the structuring element can in theory be real values, positive or negative. Digital images usually have pixels with positive integer values in the range $[0, 255]$.

This is the problem that arises when using the u-operators: when taking this range into consideration, the use of a non-flat structuring element makes it possible to go outside this range, even if the range of the structuring element is also $[0, 255]$. A dilation can increase the value to 510 (a grey value of 255 dilated by a structuring element grey value of 255), an erosion can decrease the value to -255 (a grey value of 0 eroded by a structuring element grey value of 255). The dilation example in figure 2.23 shows a few pixels that have increased in value. This is also true with the t-dilation, but there the pixels only increase to the maximum value of the pixels in the structuring element window. With

[7] The relation $\leq$ is an *ordering* (see also chapter 3, section 3.2) on two functions: $A \leq B \Leftrightarrow \forall \mathbf{r} : A(\mathbf{r}) \leq B(\mathbf{r})$.



(a) Hard-limit approach (b) Rescaling approach

Figure 2.26: The u-operators can generate values outside the greyscale range $[0, 255]$, which can be avoided by cropping or remapping the grey values.

the umbra approach, the grey values can go beyond the extremal value in the image, which can be a value outside the range of the greyscale domain.

Since digital images usually have pixels with positive integer values in the range $[0, 255]$, we cannot visualize or store the values that go outside this range. This problem can be tackled in different ways. The easiest solution would be to hard-limit the result: values below 0 become 0, values above 255 are set to 255. A second possibility is a rescaling and translation of the result: the least possible value when using an 8-bit structuring element is -255 (using the erosion), and the maximum value is 510. Translating this result to the range $[0, 765]$ and rescaling this range by a factor $1/3$ to the range $[0, 255]$ would give an 8-bit greyscale image. A consequence of this approach is that the grey values are remapped, even the initially valid pixel values.

The transfer function for the hard-limit approach is shown in figure 2.26(a), the transfer function for the rescaling approach is shown in figure 2.26(b). Both approaches are not ideal: the former assigns a default value to every grey value outside the range; the latter remaps the whole range. Because of this, the umbra approach is rarely used in practice, and also because of the extra computations needed[8] and because it is difficult to set the greyscale weights of the structuring element according to the image intensity values [Soille, 2003].

The formulas for the u-dilation and u-erosion can be used for flat (i.e., binary) structuring elements. The values of B should then be set to 0, and all elements outside B must have value $-\infty$. The equations are now those of the t-approach.

There is also another possibility, for which we refer the reader to [Kerre and Nachtegael, 2000, Nachtegael, 2002]. Several morphological frameworks exist, based on fuzzy logic. It is possible to use fuzzy logic for the

[8] A non-flat (greyscale) structuring element introduces an extra cost: if we compare equations (2.104) and (2.105) with equations (2.116) and (2.118), respectively, we notice the extra term added or subtracted in the definition of the u-operators. This term increases the computational cost.

greyscale extension of mathematical morphology, using non-flat structuring elements. The pixel values can be kept in the range $[0, 255]$.

2.4 Applications

This section covers a few special operators that can be formed by combining several of the basic morphological operators. Each operator has its own functionality and usage. We have chosen to divide this part into subsections, where each subsection addresses a specific type of application, like image reconstruction, filtering, segmentation and shape detection. The selected special operators are introduced in the section for the application where they are useful.

2.4.1 Image reconstruction

Image reconstruction is a technique that tries to create a desired image from an inferior one. With an inferior image we mean an image that hides some of the desired information, but where this information is still somehow available. For example, a noise filter might remove the noise, but also some details in the image. An image reconstructor will approximate the original image by restoring the vanished details, but not the noise.

A morphological image reconstructor [Goutsias, 2001, Soille, 2003] is defined with the *conditional dilation* (a.k.a. *geodesic dilation*):

$$\delta(A|M) = \min\{(A \oplus B), M\} \ . \tag{2.130}$$

In the binary case, when using sets, the MIN operation is replaced by the intersection $\cap$. Structuring element B contains the origin. Likewise, the other conditional operators can be defined. The *conditional closing* and *conditional opening* are compositions of the conditional dilation and conditional erosion. The *conditional erosion* is defined by:

$$\varepsilon(A|M) = \max\{(A \ominus B), M\} \ . \tag{2.131}$$

A dilation by a structuring element containing the origin expands the objects. If this is repeated several times, then the objects will grow until no more background is present. The conditional dilation restricts this swelling by defining a *mask element* M. This mask defines the boundaries of the image growth. Figure 2.27(a) shows a binary image that is used as a mask for the conditional dilation of the image in figure (b). The input image is also called a *marker*. The marker element should be part of the mask ($A \leq M$).[9]

[9] For the conditional erosion we expect the mask to be part of the marker ($A \geq M$). If A and M are sets, we replace $\leq$ by $\subseteq$.

(a) Mask M

(b) Marker A

Figure 2.27: For the conditional dilation we need a *marker* and *mask* image.

(a) $\delta^{10}(A|M)$

(b) $\delta^{20}(A|M)$

(c) $\rho_B(A|M)$

Figure 2.28: Morphological image reconstruction of A, with mask M, both shown in figure 2.27. The dilation is performed by a structuring element B, a 5×5 square.

For the definition of the image reconstructor, we first need to define the *conditional dilation of size n*:

$$\delta^n(A|M) = \underbrace{\delta\delta\cdots\delta(A|M)}_{n \text{ times}} \ . \tag{2.132}$$

The *morphological image reconstruction* is then:

$$\hat{A} = \rho_B(A|M) = \max_{n\geq 1}\{\delta^n(A|M)\} \ . \tag{2.133}$$

In the binary case, when using sets, the MAX operation is replaced by the union $\cup_{n\geq 1}$. A morphological image reconstruction is thus a sequence of conditional dilations until idempotency. We defined the conditional dilation as a dilation of a marker image (A) that is bound by a mask image (M).

Figure 2.28 shows the image reconstruction using the conditional dilation of the marker in figure 2.27. The first conditional dilations might not show any difference with a normal dilation, but for higher sizes, as can be seen in figures 2.28(a) and (b) for $n = 10$ and $n = 20$, the dilated image does not exceed the boundaries of the mask. Figure (c) shows the reconstructed image.

In section 2.4.2, the opening and closing by reconstruction are used for noise filtering. The opening and closing can be used to remove small elements, i.e.,

noise, from the image. The noise is then filtered out of the image, but the remaining objects are smoothened, because the opening and closing are smoothing filters. Therefore, morphological image reconstruction is used to prevent this. The result from the opening (or closing) can be used as a marker image for image reconstruction. We introduce two special cases of image reconstructors. The *opening by reconstruction* is defined by:

$$\Psi_{oprec}(A) = \rho_C(A \circ B|A) \ . \tag{2.134}$$

Because $A \circ B \leq A$ ($A \circ B \subseteq A$ in the binary case), we can use the opening as a marker image. The input image is at the same time the mask image. While B is used for the making of the marker, the structuring element C is used in the reconstruction step. The *closing by reconstruction* is defined by:

$$\Psi_{clrec}(A) = (\rho_C(A^c \circ B|A^c))^c \ . \tag{2.135}$$

2.4.2 Image filtering

We previously mentioned that the morphological closing and opening are filter operators. Generally, an operator is said to be a *morphological filter* if it is increasing and idempotent.

The closing and opening can be combined in order to construct new filters. A combination of a closing with an opening, or vice versa, is called an *alternating filter* (AF):

$$\rho(A) = (A \bullet B) \circ B \ , \tag{2.136}$$
$$\pi(A) = (A \circ B) \bullet B \ . \tag{2.137}$$

Remember (equation (2.82)) that those two filters do not produce the same result. In general, we can specify a size n for a basic structuring element B, resulting in nB:[10]

$$\rho_n(A) = (A \bullet nB) \circ nB \ , \tag{2.138}$$
$$\pi_n(A) = (A \circ nB) \bullet nB \ . \tag{2.139}$$

These alternating filters can be used to construct a new class of filters, the *alternating sequential filters* (ASF):

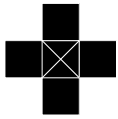
$$\nu_n(A) = \rho_n \rho_{n-1} \cdots \rho_1(A) \ , \tag{2.140}$$
$$\mu_n(A) = \pi_n \pi_{n-1} \cdots \pi_1(A) \ . \tag{2.141}$$

These filters first remove the smallest objects with filter $\rho(A)$ (or $\pi(A)$), using structuring element B. Visually this means that small darker or lighter

[10] Recall equation (2.58): $nB = B \oplus B \oplus \ldots \oplus B$ (n times).

(a) Original

(b) 10 % salt-and-pepper noise



(c) Structuring element

(d) ρ-filtered

Figure 2.29: Filtering salt-and-pepper noise from an image.

areas (compared to their surrounding intensity values) are smoothened. Every successive filter $\rho_n(A)$ (or $\pi_n(A)$) filters the image using a larger structuring element nB, removing the larger objects from the image.

We also mention an extension of the alternating filters, namely the combination of three filters:

$$\tau_n(A) = ((A \bullet nB) \circ nB) \bullet nB \ , \tag{2.142}$$

$$\sigma_n(A) = ((A \circ nB) \bullet nB) \circ nB \ . \tag{2.143}$$

Figure 2.29 shows an image contaminated with salt-and-pepper noise.[11] Applying an AF on this image will remove most noise speckles. We can compare the behaviour of the morphological filter with the more common *median filter*.

[11] *Salt-and-pepper noise* are random black or white pixels that appear in the image.

The median filter is a rank order filter that takes the middle grey value from the ordered list of neighbouring grey values, i.e., it takes the median inside a small window. Note that the morphological operators are also rank order functions: a dilation takes the maximum value of the neighbourhood defined by the structuring element, in other words, the output value is the largest value from the ordered list of neighbouring grey values. The same goes for the erosion, but now with the minimum value. Note that the dilation and erosion are not filters, because they are not idempotent. The median filter is also not idempotent, so it is not a *morphological* filter.

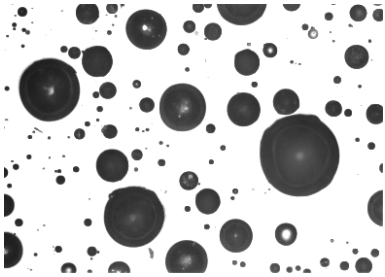
Morphological filters are good at removing salt-and-pepper noise, just like the median filter does. The effect on Gaussian noise[12] can be seen in figure 2.30, where the ρ-filter is compared to the classical *averaging filter*. The averaging filter is a low-pass filter and is good at removing Gaussian noise from an image. However it is a low-pass filter, all the high frequency information in the Fourier domain is filtered out of the image. Not only the noise, which is high frequency information, but also details, like sharp edges, will disappear. The image will look blurred, and the larger the filtering window is chosen, the more blur will emerge. The morphological filter does not average values, but takes the extremal value from a selection of pixels (i.e., the pixels covered by the structuring element), which leads to sharper edges.

In figure 2.31(a) we filtered the same image as in the previous example, but now the morphological filtering is a sequence of an opening by reconstruction (section 2.4.1), a closing by reconstruction, and an alternate filter ρ. The cross structuring element (figure 2.29(c)) is used. In figure (b) a *Wiener filter* has been applied. The Wiener filter filters out additive noise that has corrupted the signal by statistical means. Its objective is to estimate the noise-free image, by minimizing the difference between the filtered result and the optimal result. This is done by calculating the minimum mean squared error (chapter 6, section 6.4.5).[13] It produces quite good results with sharp edges. In the example the morphological filter produces somewhat better edges. Notice the cross shaped artefacts caused by the structuring element (a few of them are encircled in the figure).

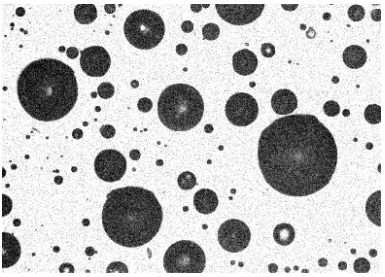
Besides for noise filtering, the morphological filters can be used for image smoothing. An alternate filter π will first open the image, which reduces the grey value of the brighter isolated pixels. The closing step will increase the value of the darker isolated pixels. Both steps smoothen the image.

[12] *Gaussian noise* is noise with a Gaussian amplitude distribution.

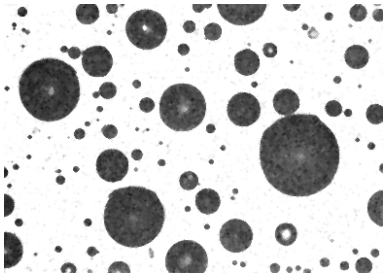
[13] Of course, the optimal result is usually not known, but the Wiener filter can be expressed in function of the *signal-to-noise ratio* (SNR) instead of the ideal image [Pratt, 2001]. The signal-to-noise ratio is a measure of the variance of the signal against the variance of the noise in the signal. The variance of the signal (i.e., the optimal image) can be calculated using the variance of the noise and the variance of the observed image.



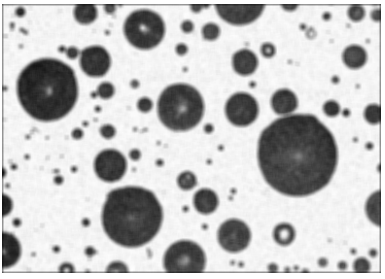
(a) Original



(b) Gaussian noise ($\mu = 0$, $\sigma = 0.01$)

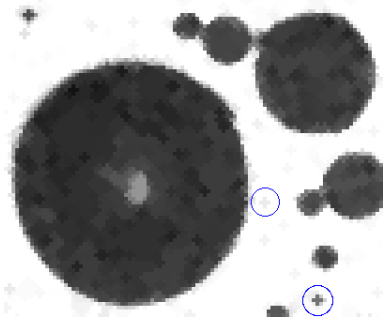


(c) ρ-filtered

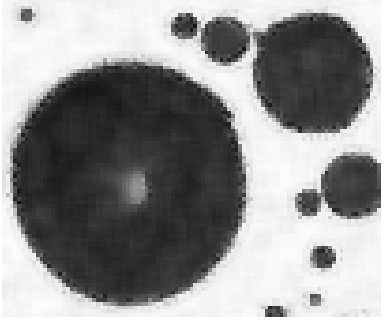


(d) Average filtered (window size 5)

Figure 2.30: Filtering Gaussian noise from an image.

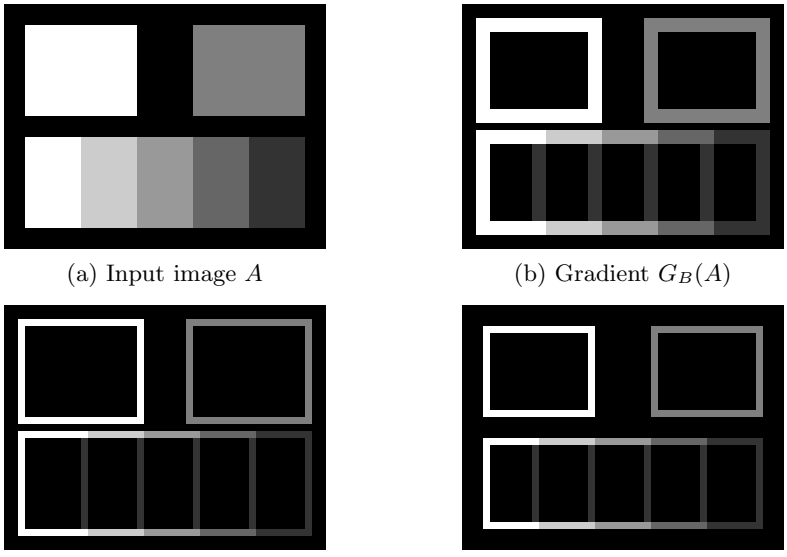


(a) Result of morphological filtering



(b) Result of Wiener filtering (window 5)

Figure 2.31: Filtering Gaussian noise from an image (crop of figure 2.30(a)).



(a) Input image A (b) Gradient $G_B(A)$

(c) External gradient $G_B^e(A)$ (d) Internal gradient $G_B^i(A)$

Figure 2.32: Morphological gradient operations on A, using a square structuring element B.

2.4.3 Segmentation

A morphological gradient can detect edges in an image. If objects are regions of rather homogeneous grey values, then object boundaries are located at high grey value differences. The morphological gradients enhance these differences in grey values. The *morphological gradient operator* is defined by:

$$G_B(A) = (A \oplus B) - (A \ominus B) \ . \tag{2.144}$$

The subtraction is the pixel-wise difference in grey value. In the binary case, only two grey values are present, 0 and 1. When the binary image is interpreted as a set instead of a function, then the set difference $\setminus$ is used. The dilation takes the maximum value in the local neighbourhood defined by the structuring element, the erosion gives the minimum value. The difference of the dilation with the erosion shows the object boundaries. Higher grey values indicate higher grey level variations. An example is shown in figure 2.32.

The *external morphological gradient operator* is defined by:

$$G_B^e(A) = (A \oplus B) - A \ . \tag{2.145}$$

As we can see in figure 2.32(c), the external gradient marks the outside of the object boundaries.

Finally, the *internal morphological gradient operator* is defined by:

$$G_B^i(A) = A - (A \ominus B) \ . \tag{2.146}$$

The internal gradient marks the inside of the object boundaries (see figure 2.32(d)). The gradient $G_B(A)$ marks both the outside and the inside boundary. To avoid negative grey values, the structuring element must contain the origin ($\mathbf{0} \in B$). This ensures the relationship $A \ominus B \subseteq A \subseteq A \oplus B$ (equation (2.92)) is valid. In practice, a symmetric structuring element is used (i.e., $B = \check{B}$).

Notice that the morphological gradient is the combination of the external and internal gradient:

$$G_B(A) = G_B^e(A) + G_B^i(A) \ . \tag{2.147}$$

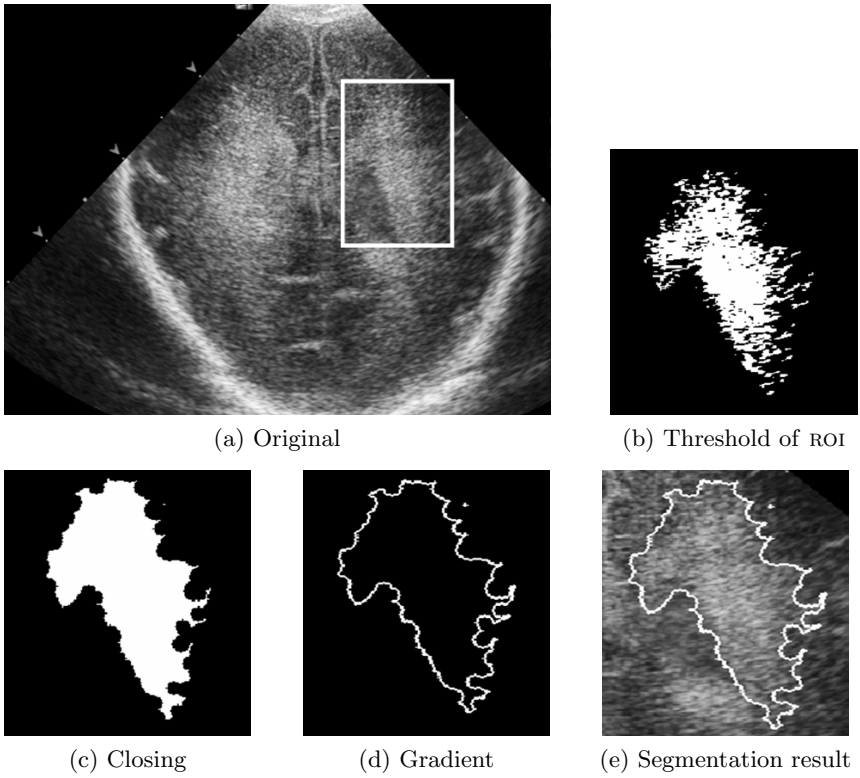
The external gradient marks the edges at the outside of the objects, while the internal gradient calculates the internal boundaries. The gradient G_B marks both the outside and the inside boundaries. This formulation is also valid in the binary case. When the binary image is interpreted as a set, then the addition is replaced by the union operation $\cup$.

An example of segmentation with the morphological gradient is shown in figure 2.33. Figure (a) shows an ultrasound (US) image of the brain of a *very low birth weight* (VLBW) newborn, i.e., less than 1500 g [Vansteenkiste et al., 2003, Vansteenkiste et al., 2006a]. The risk for such an infant to have *white matter damage* (WMD) or *Periventricular Leukomalacia* (PVL) is between 20 and 50 %. Due to a lack of oxygen in the brain, parts of the white matter die. In the US image this leads to *flaring*. Our goal is to draw the boundaries of these flares in a (semi-)automatic way. The *speckle* in the image, which is inherent in US images, hinders the segmentation.

Firstly, a rectangular *region of interest* (ROI) is drawn by the medical doctor. This ROI contains the flare and surrounding background speckle. This region is binarized using a threshold which is the average grey value in the ROI. The result is shown in figure (b).

Secondly, mathematical morphological techniques are used for the segmentation. A closing will fill the small holes in figure (b). The morphological gradient (equation (2.144)) is calculated from figure (c). Here, we have used different structuring elements for the dilation and the erosion. The dilation is done by a disc shaped structuring element of radius 3, the erosion is done with radius 2.

Finally, the gradient result (figure (d)) is placed on top of the ultrasound image, shown in figure (e).



(a) Original (b) Threshold of ROI

(c) Closing (d) Gradient (e) Segmentation result

Figure 2.33: Morphological segmentation of flares in an ultrasound image.

2.4.4 Corner detection

In some applications we want to extract specific shapes from the image, or we want to detect corners from objects, or edge points, or isolated points. The hit-and-miss transform is the ideal tool for this task. The *hit-miss operator* for binary images is defined by:

$$\begin{aligned} A \otimes (B, C) &= (A \ominus B) \cap (A^c \ominus C) \\ &= \{\mathbf{r} \mid T_{\mathbf{r}}(B) \subseteq A \text{ and } T_{\mathbf{r}}(C) \subseteq A^c\} \ . \end{aligned} \tag{2.148}$$

This operator needs two disjoint structuring elements: element B erodes the objects, element C erodes the background set. The combination of both structuring elements, as in the above equation, is a powerful tool to detect specific patterns.

For example, to detect the upper-left corner of an object, we erode the image by a structuring element B like the one in figure 2.34(a), resulting in $A \ominus B$ (see figure 2.35). If an object pixel has a right and lower object neighbour,

(a) B (foreground)

(b) C (background)

0	–1	0
–1	1	1
0	1	0

(c) Alternative representation

Figure 2.34: Upper-left corner detection with the hit-miss transform. Specific structuring elements are used.

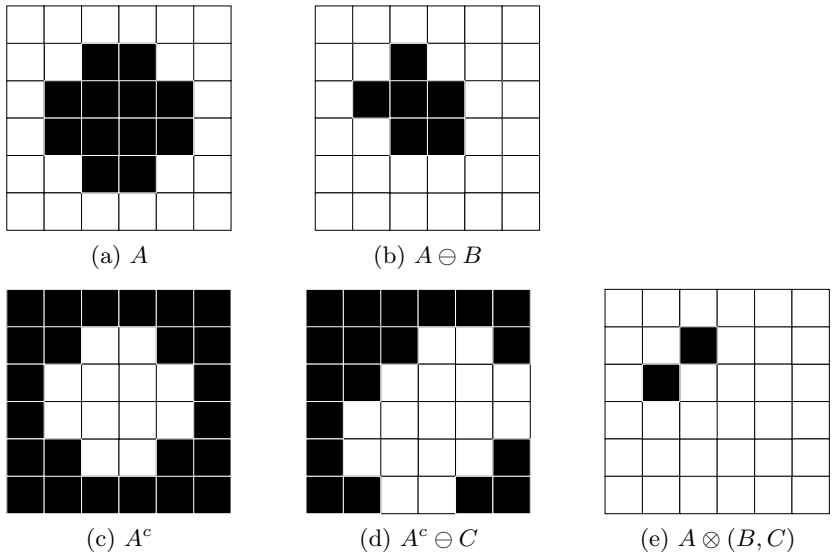
then this pixel is kept by the erosion. It is irrelevant if other neighbours are object pixels. We also erode the complement of the image by the structuring element C, shown in figure 2.34(b), resulting in $A^c \ominus C$. Notice that the origin of the structuring elements plays an important role: it sets the position of the possible corner pixel. Here, if the left and upper neighbour of the pixel are background (in the original image, foreground in A^c), then this pixel is kept by the erosion. Whether the pixel itself or its other neighbours are background pixels is not important.

The intersection of both erosions shows where upper-left corners are in the image. The result of the hit-miss transform is a set of the foreground pixels with at their right and below them foreground pixels and at their left and above them background pixels.

The hit-miss transform is capable of detecting very specific forms, and these forms are defined by the structuring elements B and C. If we want to detect a shape that is slightly different, we must slightly change the structuring elements. For example, the structuring elements from figure 2.34 detect an upper-left corner. No requirements are implied for the value of the pixel at the upper-left position, relative to the investigated pixel. We can change structuring element C by adding an extra pixel to its support, namely the pixel at the upper-left, relative to the origin. As a result, the upper-left pixel must now be a background pixel, otherwise the hit-miss transform does not produce a hit. The next paragraph introduces the *don't care* pixels, which embody the strictness of the hit-miss shapes. Unfortunately, the size of the shapes that will be detected are defined by the given structuring elements. For the detection of corners this is not a problem, but if a specific object shape must be found, then size does matter.

The hit-miss structuring elements can be represented in an alternative way (figure 2.34(c)): instead of structuring elements, a small window with values -1, 1 and 0 scans the image. In our example, a pixel is detected as a corner if the pixels at the locations with value 1 are foreground pixels, and the pixels at the locations with value -1 are background pixels. The value 0 represents the so-called *don't care* pixels: the pixel values at these positions are not important. This approach gives the same hit-miss results as the definition with the structuring elements.

The structuring elements must not intersect, otherwise the hit-miss transform



(a) A (b) $A \ominus B$

(c) A^c (d) $A^c \ominus C$ (e) $A \otimes (B, C)$

Figure 2.35: An example of the hit-miss transform, detecting upper-left corners (using the structuring elements from figure 2.34).

is useless:

$$B \cap C \neq \emptyset \Rightarrow A \otimes (B, C) = \emptyset \ . \tag{2.149}$$

Some properties of the hit-miss operator are:

$$T_{\mathbf{r}}(A) \otimes (B, C) = T_{\mathbf{r}}(A \otimes (B, C)) \ , \tag{2.150}$$

$$A \otimes (T_{\mathbf{r}}(B), T_{\mathbf{r}}(C)) = T_{-\mathbf{r}}(A \otimes (B, C)) \ , \tag{2.151}$$

$$H_\lambda(A) \otimes (H_\lambda(B), H_\lambda(C)) = H_\lambda(A \otimes (B, C)) \ , \tag{2.152}$$

$$B = \emptyset \Rightarrow A \otimes (B, C) = A^c \ominus C \ , \tag{2.153}$$

$$C = \emptyset \Rightarrow A \otimes (B, C) = A \ominus B \ , \tag{2.154}$$

$$A \otimes (B, C) = A^c \otimes (C, B) \ . \tag{2.155}$$

Equations (2.150) and (2.151) show the translation invariance, while equation (2.152) shows the scale invariance property. Equations (2.153) and (2.154) state special cases of the hit-miss transform, when one of the structuring elements is an empty set. The last equation shows the relation of the hit-miss transform of the image with that of its complement.

The hit-miss transform has originally been developed for the detection of shapes in binary images. The pixels are either part of the background or part of the foreground, and the elements from the output set are pixels where the transform

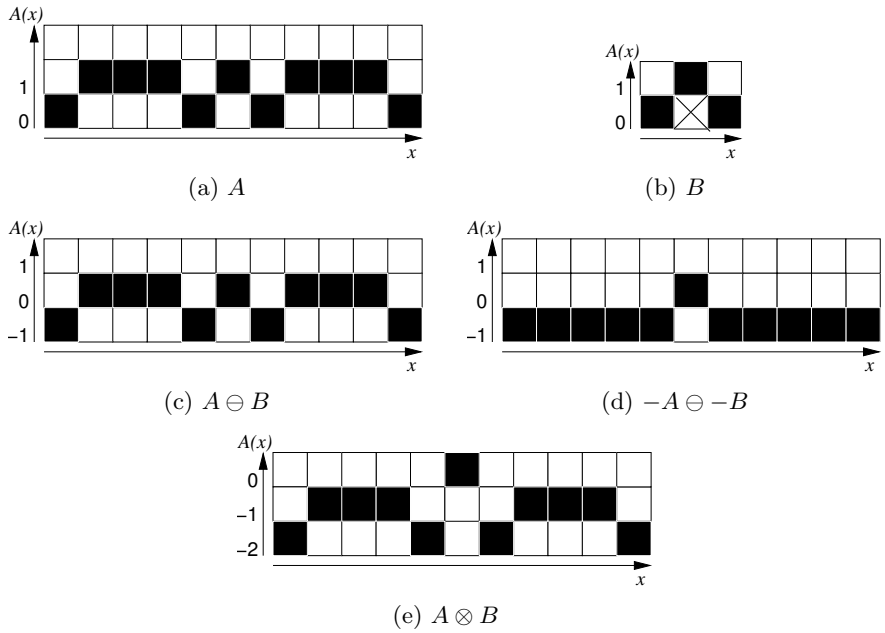


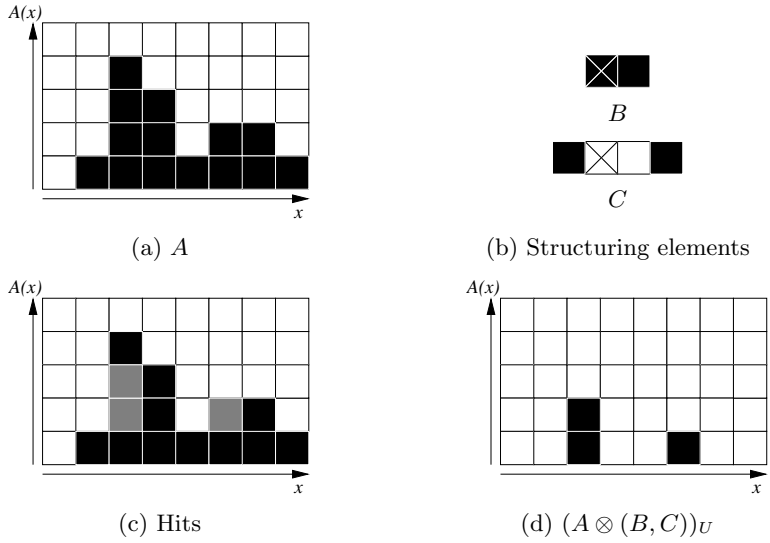
Figure 2.36: A one-dimensional example of the greyscale hit-miss transform using template matching. Note that the range of the grey values on the ordinate changes for each figure.

produces a perfect hit. With greyscale images, we have more grey values at our disposal. We might want to make a distinction between identical shapes found at different grey levels, or on the contrary do not want this to affect the hit-miss transform at all. We can also use greyscale structuring elements and try to match them to the greyscale content in the image. Several extensions to a *greyscale hit-miss transform* exist. In [Khosravi and Schafer, 1996] an extension, with only one structuring element, for the u-approach is proposed:

$$\begin{aligned}(A \otimes B)(\mathbf{a}) &= (A \ominus B) + (-A \ominus (-B)) \qquad (2.156)\\ &= \min_{\mathbf{b}\in\Omega(B)} \{A(\mathbf{a}+\mathbf{b}) - B(\mathbf{b})\} + \min_{\mathbf{b}\in\Omega(B)} \{-A(\mathbf{a}+\mathbf{b}) + B(\mathbf{b})\}\\ &= \min_{\mathbf{b}\in\Omega(B)} \{A(\mathbf{a}+\mathbf{b}) - B(\mathbf{b})\} - \max_{\mathbf{b}\in\Omega(B)} \{A(\mathbf{a}+\mathbf{b}) - B(\mathbf{b})\}\ .\end{aligned}$$

The structuring element B is called the *template*. The result of this hit-miss transform is always ≤ 0. Pixels with value 0 are the locations where a perfect match has been discovered (figure 2.36).

In [Soille, 2003], two extensions are discussed (t-approach). The *unconstrained*



(a) A (b) Structuring elements

(c) Hits (d) $(A \otimes (B,C))_U$

Figure 2.37: A one-dimensional example of the unconstrained greyscale hit-miss transform. The grey blocks in figure (c) denote the pixels and their respective cross sections where the hit-miss finds a match.

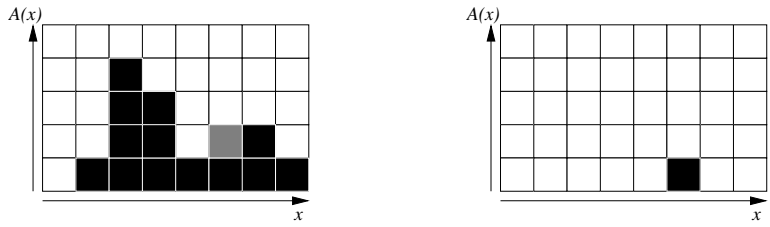
hit-miss transform is defined as:

$$(A \otimes (B,C))_U(\mathbf{a}) = \begin{cases} (A \ominus B)(\mathbf{a}) - (A \oplus \breve{C})(\mathbf{a}), \text{if } (A \oplus \breve{C})(\mathbf{a}) < (A \ominus B)(\mathbf{a}) \\ 0, \qquad \text{otherwise.} \end{cases} \tag{2.157}$$

This transform gives for every pixel the number of cross sections that produce a hit. We illustrate the unconstrained hit-miss transform in figure 2.37 using a one-dimensional greyscale image. At several pixels and cross sections a hit is found. The final hit-miss result is shown in figure (d). This method finds exact matches for the desired (flat) shape, for every cross section.

The *constrained hit-miss transform* is defined as:

$$(A \otimes (B,C))_C(\mathbf{a}) = \begin{cases} A(\mathbf{a}) - (A \oplus \breve{C})(\mathbf{a}), \text{if } A(\mathbf{a}) = (A \ominus B)(\mathbf{a}) \\ \qquad\qquad\qquad\qquad \text{and } (A \oplus \breve{C})(\mathbf{a}) < A(\mathbf{a}) \\ (A \ominus B)(\mathbf{a}) - A(\mathbf{a}), \text{if } A(\mathbf{a}) = (A \oplus \breve{C})(\mathbf{a}) \\ \qquad\qquad\qquad\qquad \text{and } (A \ominus B)(\mathbf{a}) > A(\mathbf{a}) \\ 0, \qquad\qquad\qquad\qquad \text{otherwise.} \end{cases} \tag{2.158}$$



(a) Hits (b) $(A \otimes (B, C))_C$

Figure 2.38: A one-dimensional example of the constrained greyscale hit-miss transform. A, B and C are shown in figure 2.37. The grey blocks in figure (a) denote the cross sections and pixels where the hit-miss finds a match.

If $\overline{FG}(\mathbf{r}) = \{\mathbf{a} \mid A(\mathbf{a}) \geq A(\mathbf{r})\}$ and $\overline{BG}(\mathbf{r}) = \{\mathbf{a} \mid A(\mathbf{a}) \leq A(\mathbf{r})\}$, the extra constraint here is that the structuring element containing the origin must match the foreground set $\overline{FG}$ if $\mathbf{0} \in B$, or the background set $\overline{BG}$ if $\mathbf{0} \in C$. These are sets defined by a threshold for every pixel. An example of the constrained greyscale hit-miss transform is shown in figure 2.38.

Both the unconstrained and the constrained greyscale hit-miss transform are equivalent in the binary case and result in the binary hit-miss operation, equation (2.148). The constrained hit-miss is always part of the unconstrained result, i.e., $(A \otimes (B, C))_C \leq (A \otimes (B, C))_U$.

Finally, we briefly mention a fourth variant of the greyscale hit-miss transform (u-approach) [Ronse, 1996]:

$$(A \otimes (B, C))(\mathbf{a}) = \begin{cases} (A \ominus B)(\mathbf{a}), \text{if } (A \oplus (-\check{C}))(\mathbf{a}) \leq (A \ominus B)(\mathbf{a}) \\ -\infty, \qquad \text{otherwise.} \end{cases} \tag{2.159}$$

The result depends on the absolute values in the input image, while the result of the previous definitions depends on the relative values in the input image.

2.5 Conclusion

In this chapter, we introduced the concepts of mathematical morphology, a theory based on lattice theory and random geometry, and used for the investigation of geometric structures. We defined the basic operators, *dilation* and *erosion*, as well as the secondary operators, *closing* and *opening*. The morphological operators perform their duty using one or more *structuring elements*. The choice of size and shape of these elements depends on the application and image content. We also discussed several properties of the morphological operators.

We explained two greyscale extensions of the binary theory: the *threshold approach* and the *umbra approach*. The t-approach builds on the concept of cross sections, while the u-approach uses concepts like the top surface and umbra. The umbra approach allows the use of greyscale structuring elements, but it has a higher computational cost and the grey values can lie outside the initial range.

In the final section, we discussed a few morphological operators used in practice, such as the *morphological reconstructors* and *filters*, the *gradient* for segmentation and the *hit-miss transform* for the detection of specific shapes. The hit-miss transform will play an important role in chapter 6.

Chapter 3

Colour Mathematical Morphology

In chapter 2, we introduced the concepts of mathematical morphology. Originally, this theory was developed for binary images, using set theory. The mathematical concepts were extended to greyscale images, represented by functions. This was quite straightforward, since the only essential change in the theory is the replacement of the union and intersection by the maximum and minimum operation, respectively.

Extending the theory to colour is a different matter because colour is vector data (e.g., each pixel has three colour components, rather than one grey value). One approach to extend greyscale morphology is to define a complete ordering on the possible colour vectors. This ordering can then be used instead of taking the grey values. This chapter will explain the problem that is involved and will discuss some possible solutions. First, an introduction to colour representation is given, along with some colour space definitions. Different solutions to colour ordering exist, as we will see in section 3.2. Finally, we will introduce our own ordering scheme, the *majority ordering*.

3.1 Theoretical background

Chapter 2 dealt with binary and greyscale images. The pixel values in those images describe intensity. The human eye is also able to distinguish different colours [Trussell et al., 2005, Gonzalez and Woods, 2002, Pratt, 2001, Poynton, WWW]. Colour images can provide valuable information, not available in greyscale images. We will therefore describe how to deal with colour images. We first give a definition of colour, how we perceive it, and how we can represent it.

Figure 3.1: The visible spectrum, going from violet (380 nm) to red (780 nm). Neighbouring lower wavelengths are in the ultraviolet range, neighbouring higher wavelengths are in the infrared range.

3.1.1 The human perception of colour

Sunlight is perceived as white light, but when a beam is passed through a glass prism, then the emerging beam is not white. Instead, it is broken up, *dispersed*, in a continuous spectrum of colours. This effect can be seen when a rainbow appears in the sky, caused by the dispersion of sunlight as it is refracted by raindrops. This colour spectrum is most commonly cited as the seven colour regions red, orange, yellow, green, blue, indigo, and violet. In reality, however, an infinite number of colours are present.

Each spectral component is characterized by a frequency ν, or alternatively by a wavelength λ, defined by $\lambda = c/\nu$, with c the speed of electromagnetic radiation, which is about 300 000 km/s in vacuum.[1] [2] The visible spectrum for humans lies between 380 nm and 780 nm (see figure 3.1). The range of colours corresponding to this spectrum goes from blue, over green, to red. Electromagnetic signals with a wavelength outside this range are not detectable by the human eye. The wavelengths below 380 nm are the ultraviolet (UV) waves (on the left in figure 3.1). Further away from this area are the X-rays, γ-rays and cosmic rays, which have the highest energy per photon.[3] The wavelengths above 780 nm are the infrared (IR) waves (on the right in figure 3.1). Electromagnetic waves with even less energy per photon are the microwaves, television and radio signals, and electricity.

The eye is a lens system, where the incident light is focussed by the *cornea* and *lens* to form an image of the object being viewed on the *retina* at the back of the eyeball. The retina contains receptors that translate the incident light to a nerve signal, that is interpreted by the brain. There are two kinds of receptors:

- *Rod cells* or *rods* are very sensitive photoreceptor cells, and are thus able to respond to very weak light stimuli. They are very numerous, the retina has about 100 million rods. Their purpose is night vision, or *scotopic* view, and they confer the *achromatic* or monochromatic view.

[1] The speed of light in vacuum is exactly defined as 299 792 458 m/s [NIST, WWWc].

[2] Note that the wavelength λ of a certain spectral component depends on the medium, while the frequency ν does not. Often, λ is used to define a certain spectral component, and it is then implicitly assumed that the medium is vacuum.

[3] The energy of a *photon* (the smallest amount of electromagnetic radiation, a massless bundle of energy) has an energy of $E = h\nu$, with h *Planck's constant* ($6.626\,0693 \cdot 10^{-34}$ J s) [NIST, WWWb]. This energy is usually expressed in electron volt (eV), where 1 eV = $1.602\,176\,53 \cdot 10^{-19}$ J [NIST, WWWa].

- *Cone cells* or *cones* function in relative bright light and allow the perception of colour. There are about 6 million cones present on the retina, so they are less numerous than the rods. They are also less sensitive, since they only respond to a specific wavelength range. There are three types of cones:
  - *Short* (S) cones are sensitive in the blue region. Their response function has a peak wavelength at (about) 445 nm.[4]
  - *Medium* (M) cones respond to green colours, with a peak wavelength at (about) 540 nm.
  - *Long* (L) cones respond to the long wavelengths, i.e., yellow-green light. Their peak wavelength is at (about) 565 nm.

The (normalized) spectral absorption curves for the three types of cones and the rod cells (R) are visualized in figure 3.2. The combined sensation of the cells is what we define as *colour*. Black, white and the different grey values are also colours, but the achromatic ones. This system of three different cones makes the human vision a *trichromatic* system, i.e., a colour perceived by the human eye can be described as a weighted sum of three pre-defined colours. The colour yellow, for example, is perceived when the L-cone is stimulated slightly more than the M-cone. There are fewer S-cones and they are less sensitive than the other cones, so the detection of blue is more difficult.[5] The human eye is also more sensitive to the colour green, because then both the M- and L-cones are stimulated almost equally.

3.1.2 Colour reproduction

3.1.2.1 Additive systems

In additive devices, colours are produced by combining several primary light sources. It is sufficient to have three well-chosen primary colours (red, green and blue) or *primaries* to produce most colours that are visible to the human eye. The choice of the primaries defines the range of colours that are physically obtainable. This range is called the *gamut*. Each primary has a *colour matching function*, which defines the weight, or *tristimulus value*, for its respective primary that is needed to mimic a spectral light source with a specific wavelength. A colour matching function can be negative in a certain wavelength domain, in which case a spectral light source with that wavelength cannot be reproduced. Any colours containing such spectral components are outside the gamut.

[4] The values of the peak wavelengths are from [Stiles and Burch, 1959]. Other experiments [Dartnall et al., 1983] show peak wavelengths at 419 nm, 531 nm and 558 nm for the S-, M- and L-cones, respectively.

[5] Figure 3.2 shows the normalized absorption spectra of the cones. Relatively, the maximum absorption value of the S-cone is much lower than that of the other two cones.

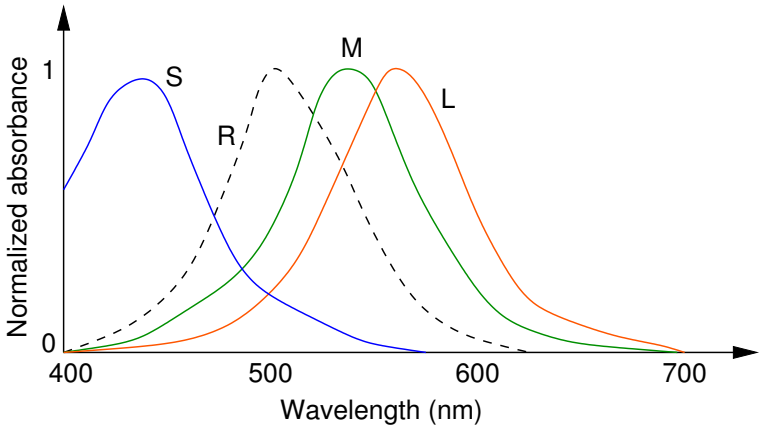


Figure 3.2: The cone sensitivities for the S-, M- and L-cones. The response function of the rod cells (R) is also shown.

Figure 3.3 shows how the colours cyan, magenta, yellow and white are generated by adding the primaries. If the intensities of those primaries red, green and blue are changed, then other colours are produced. An example of this is shown in figure 3.4 with the "Lena" image.

A television or computer monitor with a CRT (*cathode ray tube*) is an example of an additive colouring system. The screen is covered with phosphoric dots that emit light with a specific spectrum when hit by an electron beam that scans the screen. The dots are clustered in groups containing the three primaries. The emitted spectra of neighbouring primaries are added and the result is captured by the eye.

3.1.2.2 Subtractive systems

In an additive colour system, primaries are superimposed to obtain a specific colour, i.e., their light intensities are added. In a subtractive colour system, on the other hand, a portion of the electromagnetic spectrum is removed from initially white light in a filtering process, before reaching the observer, which results in a certain colour. The filter is a semi-transparent coloured medium. The colours used are cyan, magenta and yellow. The thickness and properties of the dye layer define how much light can pass. This way, a wide range of colours can be obtained. Figure 3.5 shows the subtraction of colours. For example, in order to obtain green, the white light must pass a cyan and yellow filter.

A colour inkjet printer is a typical example of a subtractive device, using the CMYK model. The different ink cartridges contain semi-transparent ink with a certain primary colour, i.e., cyan, magenta and yellow (and black). The ink drops are printed onto a white piece of paper. When we look at the paper,

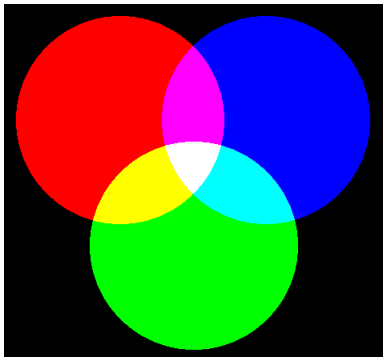


Figure 3.3: The addition of colours. The combination of the primaries red, green and blue generates new colours, such as cyan, magenta, yellow and white.

(a) Red colour band

(b) Green colour band

(c) Blue colour band

(d) Additive result

Figure 3.4: The additive colour system. The sum of the three colour bands results in a colour image.

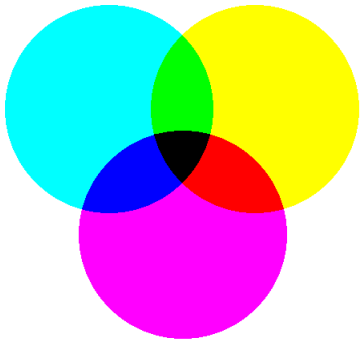


Figure 3.5: The subtraction of colours. The combination of the primaries cyan, magenta and yellow generates new colours, such as red, green, blue and black.

white light passes the ink layers and is changed into coloured light, which is reflected on the paper and reaches our eyes through the layers of ink. An extra cartridge with a black primary (the letter K in CMYK) is used for a better and less expensive reproduction of the "colour" black.

The subtractive system enables us to observe objects and distinguish their colours. Light is reflected on objects and certain spectral parts are absorbed by these objects. The other parts reach the eye and define the perceived colour.

3.1.3 Colour spaces

All visible colours can be represented by the combination of three default colours with their respective colour matching function, the *primaries*. Every colour system has its own set of primaries. In theory, the number of possible sets of primaries is infinite.

In practice, several different colour spaces are used, and also several different sets of primaries. The choice of the colour space and primaries depends on the device and application used. A good set of primaries is one that has a large gamut, i.e., many colours can be obtained by combining the primaries. Therefore, the preferred colour matching functions are non-negative. We now discuss some commonly used colour spaces.

3.1.3.1 CIE's XYZ space

The *Commission Internationale de l'Eclairage* (CIE) [CIE, WWW] defined some standard colour coordinates X, Y and Z. The XYZ coordinates are associated to three artificial primaries that cannot be realised by light sources. The coordinate values are linear to the intensity of light and embed the spectral properties for human colour vision.

The spectra of the primaries of the XYZ colour system have the following restrictions:

- The Y-coordinate is the *luminance* component. Luminance is like intensity,[6] but it is now radiant power weighted by a spectral sensitivity function that is characteristic of vision [Poynton, 1997b]. The unit of luminance is cd/m^2.[7]
- Uniform white light is obtained when $X = Y = Z$. As a consequence, greyscale values can be represented by the luminance component Y, knowing that the other two components have the same value.
- The colour matching functions of the three primaries must be positive.

3.1.3.2 CIE's xy space

Until now, we have discussed the colours in function of how the human eye captures colour. It can be convenient to have a representation of "pure" colour in the absence of luminance. Therefore, the CIE defined the normalized chromaticity coordinates x, y and z:

$$\begin{cases} x = \frac{X}{(X+Y+Z)} \\ y = \frac{Y}{(X+Y+Z)} \\ z = \frac{Z}{(X+Y+Z)} \end{cases} . \tag{3.1}$$

Since z can be described as a linear combination of x and y, i.e., $x + y + z = 1$, only the first two coordinates are used, along with the luminance Y. The colour space is represented by the *chromaticity diagram* (figure 3.6). The gamut is the area inside the shark fin shaped curve. The curved edge of the gamut is called the *spectral locus* and corresponds to monochromatic light. The gamut of real devices is much smaller than the xy-gamut. Also, their white point is not the same as the *equal energy white point*, where $x = y = 1/3$.

3.1.3.3 The RGB colour space

In section 3.1.2 we discussed additive and subtractive colour systems. The RGB (Red-Green-Blue) colour system is an additive colour system based on the trichromatic theory. This is an easy to implement, but visually non-linear system (i.e., a displacement in its three-dimensional space is not linearly proportional to the perceived colour difference). It is also device dependent. It is often used in computer monitors and video cameras. The space is represented as a cube, with on each axis one of the primaries red, green or blue (see figure 3.7). The

[6] *Intensity* (unit: W/m^2) is the rate at which radiant energy is transferred over some time interval per unit area, without correlation to human perception.

[7] The symbol "cd" stands for *candela*, the unit of luminous intensity.

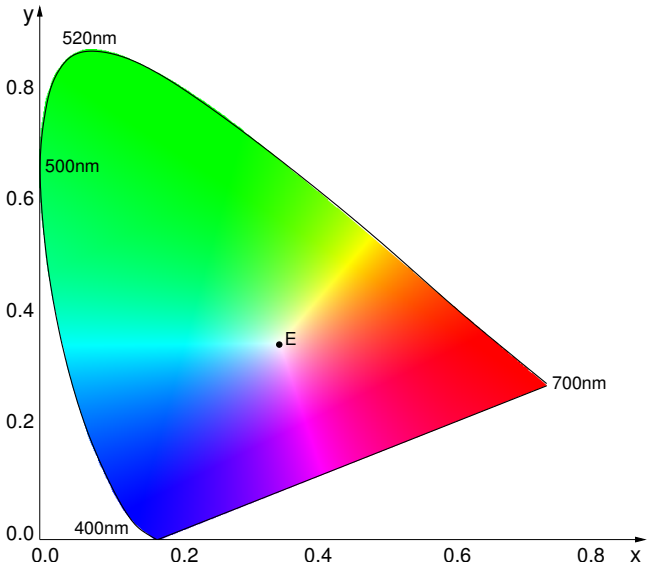


Figure 3.6: The CIE xy chromaticity diagram. The point E is the *equal energy white point*, i.e., $x = y = 1/3$. (Part of the graph was made using software from [efg, WWW].)

coordinates $(1, 1, 1)$, i.e., when all primaries are at their maximum, produce bright white (W). When all primaries have the same value, then grey values are generated. When two of the components are set to zero, then the colour ranges from black (K) to bright red (R), green (G) or blue (B).

The conversion between the RGB space and XYZ space is:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} X_r & X_g & X_b \\ Y_r & Y_g & Y_b \\ Z_r & Z_g & Z_b \end{bmatrix}^{-1} \cdot \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}. \tag{3.2}$$

The matrix coefficients depend on the definition of the RGB space. For computer monitors, studio video and high definition television (HDTV), a specific transformation matrix is used:

$$\begin{bmatrix} X_r & X_g & X_b \\ Y_r & Y_g & Y_b \\ Z_r & Z_g & Z_b \end{bmatrix}^{-1}_{709} = \begin{bmatrix} 3.240479 & -1.537150 & -0.498535 \\ -0.969256 & 1.875992 & 0.041556 \\ 0.055648 & -0.204043 & 1.057311 \end{bmatrix}. \tag{3.3}$$

This is ITU-R Recommendation BT. 709 (a.k.a. standard RGB or sRGB) [Poynton, 1997b]. The white point from this space is the so-called D_{65} white

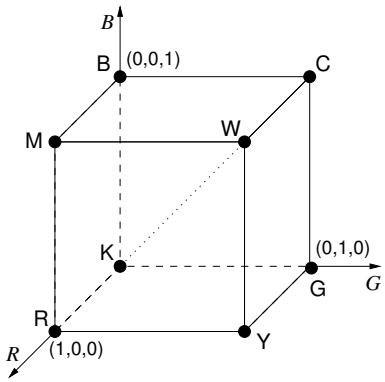


Figure 3.7: RGB colour space.

point, which differs slightly from the equal energy white point.[8] For PAL and NTSC television, two other sets of primaries are used.

Notice the negative coefficients in the transformation matrix. The RGB values can thus become negative. These points lie outside the gamut, so there are colours that cannot be displayed using this colour system.

On a computer, the colours are usually stored in 24-bit format. This means there are three colour bands, each 1 byte (= 8 bits) large. Every colour band can thus store $2^8 = 256$ possible values; together there are $2^{24} \approx 16.7$ million colours. Not every colour stored can be distinguished by the human eye. No exact number exists, but [Leong, WWW, Expert, WWW] sum several sources that give different results, going from 100 000 to about 10 million distinguishable colours. Thus several different colour vectors will be perceived as the same colour. On the other hand, many noticeable colours are not produced by the computer monitor, because of its limited gamut.

3.1.3.4 The CMY(K) colour space

The subtractive colour system CMYK is used in printing. It is device dependent,[9] non-linear with visual perception and quite unintuitive. A simple trans-

[8] The chromaticities xy of Rec. 709 are: $R = (0.64, 0.33)$, $G = (0.30, 0.60)$, $B = (0.15, 0.06)$ and $W = (0.312713, 0.329016)$ [Ford and Roberts, WWW]. The non-linearity or gamma of the monitor is 2.2 (see also p. 73).

[9] Not only the printing device influences the result, but also the inks used and even the printing paper. No wonder why the inks and paper from the printer manufacturers are so expensive.

formation between RGB and CMY exists:[10]

$$\begin{cases} C = 1 - R \\ M = 1 - G \\ Y = 1 - B \end{cases} . \tag{3.4}$$

So, cyan is the complementary colour of red, magenta is the complement of green, and yellow is that of blue.

Unfortunately, this transform is not useful in practice. In reality, the transformation from the desired RGB to the CMY required to reproduce this RGB is more complex. The absorption spectra of the colourants overlap in practice, which means that, for example, the cyan filter also attenuates some of the magenta parts of the spectrum. Also, the conversions are device and application dependent. They can be described in ICC *profiles* [ICC, WWW]. We must also keep in mind that, when transforming an RGB image (screen image) into a CMYK image (print image), that both gamuts are not the same.

3.1.3.5 HSL cylindrical space

One of the many colour spaces that aim to separate the luminance and chrominance information, is the HSL (Hue Saturation Luminance) space. Variants of this space are HSV (V for Value) and HSI (I for Intensity). The HSL coordinates are the following:

- H: *hue*: this is the tint of the colour in the visible spectrum. People would refer to a hue as blue, red, orange, purple, ...
- S: *saturation*: this corresponds to the purity of the colour: less saturation results in a more pastel colour. When there is no saturation, then grey values are obtained.
- L: *luminance*: see section 3.1.3.1. It is proportional to the radiant power.

The HSL space is a double-cone space (HSL_{dc}), as will be discussed in the next subsection, but a cylindrical variant (HSL_c) is also used (figure 3.8). The transformation from RGB to HSL_c is [Hanbury, 2001]:

$$L = \frac{m + M}{2} , \tag{3.5}$$

$$S_c = \begin{cases} \frac{M-m}{M+m}, & \text{if } L \leq \frac{1}{2} \\ \frac{M-m}{2-M-m}, & \text{if L } > \frac{1}{2} \end{cases} , \tag{3.6}$$

[10] A transformation between CMY and CMYK is given in [Ford and Roberts, WWW].

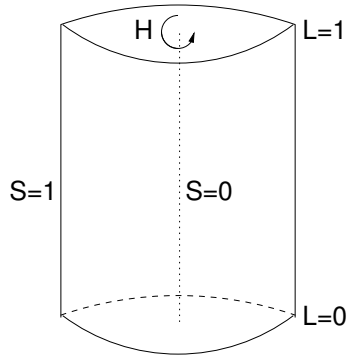


Figure 3.8: HSL cylindrical space.

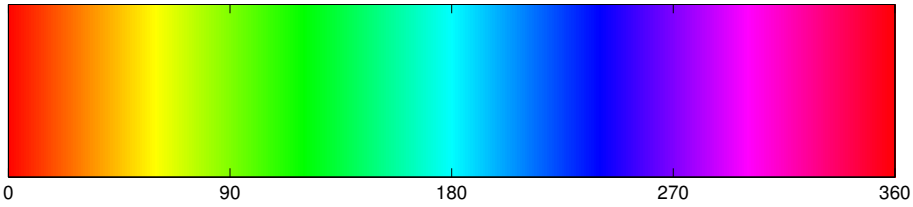


Figure 3.9: The hue is defined on a circle, in the interval $[0°, 360°[$. Here, it is shown on a strip (for $L = 1/2$ and $S = 1$).

$$H = \begin{cases} 60°(\frac{G-B}{M-m} \bmod 6), & \text{if } R = M \\ 60°(2 + \frac{B-R}{M-m} \bmod 6), & \text{if } G = M \\ 60°(4 + \frac{R-G}{M-m} \bmod 6), & \text{if } B = M \end{cases} , \tag{3.7}$$

with $m = \min\{R, G, B\}$ and $M = \max\{R, G, B\}$ and $a \bmod b$ is the modulo of a to b (i.e., the remainder of a/b). L and S_c have values in the interval $[0, 1]$, just as the coordinates R, G and B. The hue H is defined in the interval $[0°, 360°[$ (see figure 3.9).

A high saturation value denotes a more pure colour. When $S_c = 0$, then the hue is undefined and grey values are obtained.

The backward transformation from cylindrical HSL to RGB is described by the following algorithm [Hanbury, 2001]:

1. First, remap the hue to the interval $[0, 6[$:

$$H_t = H/60° \ ; \tag{3.8}$$

2. Then, if $S = 0$ and $H_t = 0$:

$$\begin{cases} R = L \\ G = L \\ B = L \end{cases} ; \tag{3.9}$$

3. Else:

   (a) Define $f = H_t - \lfloor H_t \rfloor$;

   (b) If $L \leq 1/2$:

      i. $m = L(1 - S)$;
      ii. $M = L(1 + S)$;
      iii. $\text{mid}_1 = L[2Sf + (1 - S)]$;
      iv. $\text{mid}_2 = L[2S(1 - f) + (1 - S)]$;

   (c) Else:

      i. $m = L(1 + S) - S$;
      ii. $M = L(1 - S) + S$;
      iii. $\text{mid}_1 = 2[L(1 - f) - (\frac{1}{2} - f)M]$;
      iv. $\text{mid}_2 = 2[Lf - (f - \frac{1}{2})M]$;

   (d) Further, if $\lfloor H_t \rfloor = 0$:

$$\begin{cases} R = M \\ G = \text{mid}_1 \\ B = m \end{cases} ; \tag{3.10}$$

   (e) Else, if $\lfloor H_t \rfloor = 1$:

$$\begin{cases} R = \text{mid}_2 \\ G = M \\ B = m \end{cases} ; \tag{3.11}$$

   (f) Else, if $\lfloor H_t \rfloor = 2$:

$$\begin{cases} R = m \\ G = M \\ B = \text{mid}_1 \end{cases} ; \tag{3.12}$$

(g) Else, if $\lfloor H_t \rfloor = 3$:

$$\begin{cases} R = m \\ G = \text{mid}_2 \\ B = M \end{cases} ; \tag{3.13}$$

(h) Else, if $\lfloor H_t \rfloor = 4$:

$$\begin{cases} R = \text{mid}_1 \\ G = m \\ B = M \end{cases} ; \tag{3.14}$$

(i) Else, if $\lfloor H_t \rfloor = 5$:

$$\begin{cases} R = M \\ G = m \\ B = \text{mid}_2 \end{cases} . \tag{3.15}$$

3.1.3.6 HSL double-cone space

The HSL cylindrical space does not agree well with the perceived properties of colour, such as saturation. HSL is therefore represented in a different way, namely a double-cone representation (figure 3.10). The luminance value goes from 0 to 1, exactly as we described in the previous section concerning the cylindrical representation. The same is true for the hue value, which is angular. The saturation, however, is defined differently. The maximum possible saturation value now depends on the luminance. The saturation S has values between 0 and 1 for $L = 1/2$, but the maximum S decreases linearly as L goes to 0 or 1. At $L = 0$ and $L = 1$ the saturation can only be 0. As a consequence, the colours black and white are independent of the value of S and H, and narrowing of the cones shows that colours with $L < 1/2$ and $L > 1/2$ cannot take on the full range of saturation values.

The cylindrical HSL representation can be useful for adjustments to the luminance or saturation channels. It is more convenient to alter the luminance while the saturation is kept constant. In the double-cone space this can result in coordinates that lie outside the HSL space. Also, if we want to change the saturation in the cylindrical space, we know that the range is always $[0, 1]$. The altered colour is then transformed back to the double-cone space. The

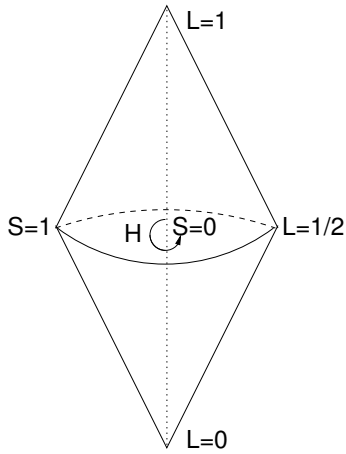


Figure 3.10: HSL double-cone space.

transformation from cylindrical HSL to double-cone HSL is:[11]

$$\begin{cases} H = H_{(c)} \\ S = S_c(1 - 2|\frac{1}{2} - L_{(c)}|) \\ L = L_{(c)} \end{cases} . \tag{3.16}$$

The transformation from double-cone HSL to cylindrical HSL is:

$$\begin{cases} H_{(c)} = H \\ S_c = S/(1 - 2|\frac{1}{2} - L|) \\ L_{(c)} = L \end{cases} . \tag{3.17}$$

3.1.3.7 CIE's $L^*a^*b^*$ space

The problem with most colour spaces is the fact that they are not *perceptually uniform*. A space is perceptually uniform if we can use the Euclidean distance as a measure for the difference in colour, perceived by the eye. The *Euclidean distance* is defined as:

$$d(\mathbf{r}, \mathbf{s}) = \sqrt{(r_1 - s_1)^2 + (r_2 - s_2)^2 + (r_3 - s_3)^2} , \tag{3.18}$$

with, in this context, $\mathbf{r}$ and $\mathbf{s}$ two points in the colour space. r_i and s_i (for $i = \{1, 2, 3\}$) are the vector components.

[11] The subscript $_{(c)}$ in equations (3.16) and (3.17) denotes the component in HSL_c, but this value equals that of the respective component in HSL_{dc}.

Some colour spaces have been constructed to be perceptually uniform. The L*a*b* (CIELab)[12] colour space is derived from the XYZ space. It is device independent. The transformation from XYZ to L*a*b* is [Ford and Roberts, WWW]:

$$L^* = \begin{cases} 116\ (\frac{Y}{Y_n})^{\frac{1}{3}} - 16, \text{if } \frac{Y}{Y_n} > 0.008856 \\ 903.3\ (\frac{Y}{Y_n}), \qquad \text{if } \frac{Y}{Y_n} \leq 0.008856 \end{cases} , \tag{3.19}$$

$$a^* = 500\ \left[f\left(\frac{X}{X_n}\right) - f\left(\frac{Y}{Y_n}\right) \right] , \tag{3.20}$$

$$b^* = 200\ \left[f\left(\frac{Y}{Y_n}\right) - f\left(\frac{Z}{Z_n}\right) \right] , \tag{3.21}$$

with

$$f(t) = \begin{cases} t^{\frac{1}{3}}, & \text{if } t > 0.008856 \\ 7.787\ t + \frac{16}{116}, & \text{if } t \leq 0.008856 \end{cases} . \tag{3.22}$$

The *lightness* L^* has values in the range $[0, 100]$. The non-linear relationship is intended to mimic the logarithmic response of the eye. The lightness is the perceptual response to luminance, which behaves non-linearly regarding human vision. a^* and b^* are the chromaticity coordinates. The subscript n denotes the white point of the system.[13]

We can transform the chromaticity coordinates to polar coordinates:

$$C^* = \sqrt{a^{*2} + b^{*2}} , \tag{3.23}$$

$$h_{ab} = \arctan\left(\frac{b^*}{a^*}\right) , \tag{3.24}$$

where h_{ab} is the hue and C^* is called the *chroma*.

The *colour difference* between colours $\mathbf{r}$ and $\mathbf{s}$ is expressed using the Euclidean distance, as stated in equation (3.18):

$$\Delta E^*_{ab}(\mathbf{r}, \mathbf{s}) = \sqrt{(L^*_\mathbf{r} - L^*_\mathbf{s})^2 + (a^*_\mathbf{r} - a^*_\mathbf{s})^2 + (b^*_\mathbf{r} - b^*_\mathbf{s})^2} . \tag{3.25}$$

A newer definition for the colour difference exists, taking into account the hue and chroma components [Trussell et al., 2005]. It is denoted ΔE^*_{94}.

[12] There also exists another Lab space, *Hunter Lab* [HunterLab, WWW].

[13] If we want to represent an sRGB image in L*a*b*, we transform first to XYZ using the inverse of the matrix in equation (3.3). The white point in RGB is $(R, G, B) = (1, 1, 1)$, which equals $(X_n, Y_n, Z_n) = (0.950455, 1.0, 1.088754)$.

3.1.3.8 CIE's L*u*v* space

Another perceptually uniform colour space is L*u*v* (CIELuv). Just as L*a*b*, it is derived from XYZ:

$$L^* = \begin{cases} 116\ (\frac{Y}{Y_n})^{\frac{1}{3}}, \text{ if } \frac{Y}{Y_n} > 0.008856 \\ 903.3\ (\frac{Y}{Y_n}), \text{if } \frac{Y}{Y_n} \leq 0.008856 \end{cases} , \tag{3.26}$$

$$u^* = 13L^*(u - u_n) , \tag{3.27}$$

$$v^* = 13L^*(v - v_n) , \tag{3.28}$$

with

$$u = \frac{4X}{X + 15Y + 3Z} , \tag{3.29}$$

$$v = \frac{9Y}{X + 15Y + 3Z} , \tag{3.30}$$

$$u_n = \frac{4X_n}{X_n + 15Y_n + 3Z_n} , \tag{3.31}$$

$$v_n = \frac{9Y_n}{X_n + 15Y_n + 3Z_n} . \tag{3.32}$$

Notice that the lightness is the same as the one for L*a*b* (equation (3.19)).

We can transform the chromaticity coordinates u^* and v^* to polar coordinates:

$$C^* = \sqrt{u^{*2} + v^{*2}} , \tag{3.33}$$

$$h_{uv} = \arctan\left(\frac{v^*}{u^*}\right) , \tag{3.34}$$

where h_{uv} is the hue and C^* is the chroma. A *psychometric saturation* is also defined:

$$s_{uv} = \frac{C^*}{L^*} . \tag{3.35}$$

The colour difference between colours $\mathbf{r}$ and $\mathbf{s}$ is expressed as:

$$\Delta E^*_{uv}(\mathbf{r}, \mathbf{s}) = \sqrt{(L^*_{\mathbf{r}} - L^*_{\mathbf{s}})^2 + (u^*_{\mathbf{r}} - u^*_{\mathbf{s}})^2 + (v^*_{\mathbf{r}} - v^*_{\mathbf{s}})^2} . \tag{3.36}$$

3.1.3.9 Other spaces

The list of existing colour spaces is very long, which makes an overview of colour spaces a difficult task. We confined ourselves to the above mentioned colour spaces. The colour spaces can be classified into four groups [Poynton, 1997a] that are related by different kinds of transformations:

- *Linear-light tristimulus* spaces: these are the spaces where the defined luminance is proportional to intensity. The CIE XYZ space (section 3.1.3.1) and the linear RGB space (e.g., sRGB (section 3.1.3.3)) belong to this group.
- *(x,y) chromaticity* spaces: this is the CIE xyY colour space (section 3.1.3.2).
- *Perceptually uniform* spaces: these are the spaces that take into account the non-linear response of the eye. CIELab (section 3.1.3.7), CIELuv (section 3.1.3.8) and non-linear R'G'B' are spaces where the Euclidean distance is a good measure for the perceptual colour difference. R'G'B' is a gamma-corrected RGB,[14] so it is still device dependent. It is also not really perceptual uniform.
- *Hue-oriented* spaces: one of the components is the hue. HSV, HSI and HSL (discussed in sections 3.1.3.5 and 3.1.3.6) are examples of hue-oriented spaces, as well as the polar versions of the L^* spaces (sections 3.1.3.7 and 3.1.3.8).

Gamma

The detector of a camera produces electrical signals that are proportional to the intensity of the incident light. However, if the captured images are shown on a CRT, the obtained intensity is not linearly related to the applied voltage:

$$I(\mathbf{r}) \propto V(\mathbf{r})^{\gamma} \ . \tag{3.37}$$

$I(\mathbf{r})$ is the obtained intensity at position $\mathbf{r}$ in the image, $V(\mathbf{r})$ is the applied voltage in the CRT and γ is a number that denotes the power-like relation. The latter symbol explains why we speak of *gamma* and *gamma-correction*. The value of γ lies between 2.3 and 2.6 [Poynton, 1996].

Because the acquisition is a linear relationship while the reproduction is non-linear, a correction is needed. This is the γ-correction: a modification of the image signal, s, is performed before it is sent to the phosphoric screen:

$$s(\mathbf{r}) \propto I(\mathbf{r})^{1/\gamma} \ . \tag{3.38}$$

This γ-correction can be done right after acquisition[15] or right before displaying. Since different monitors have different gamma, although there are standards, it is important to know the value of γ of the correction and the value of γ of the monitor, otherwise the result of the images might not look good (too

[14] The next paragraph explains gamma.

[15] For a television program, the correction is done in the studio, before broadcasting. A digital image is usually stored as a gamma-corrected one. By convention, γ-correction is done at this stage.

bright or too dark, for example). The Rec. 709 (sRGB) gamma value is 2.2 and is recommended when some of the above information is not available.

An advantage of the γ-correction (equation (3.38)) is that the relationship resembles very good the logarithmic response of the human eye. Storing an image after γ-correction (nearly) minimizes the perceptibility of noise and (nearly) maximizes the perceptual uniformity. This is useful since colour or grey values are quantized, i.e., the image pixels have a certain bit depth.

3.1.4 Colour quantization

Sometimes we want to reduce the number of colours in a digital image. This can be because of technical reasons (e.g., a GIF image has a palette of only 256 colours), colour reduction is needed for compression, it improves segmentation of objects, ... In section 3.3 we will also need colour reduction.

Several quantization techniques exist. The most straightforward one is *uniform quantization*. Each colour band is subdivided in a number of bins of equal width. The colour of each bin is the centre colour of the bin. This technique does not take the image content or the image colours into account. Some bins can be almost empty, while other bins can contain many colours that are all transformed into one single colour.

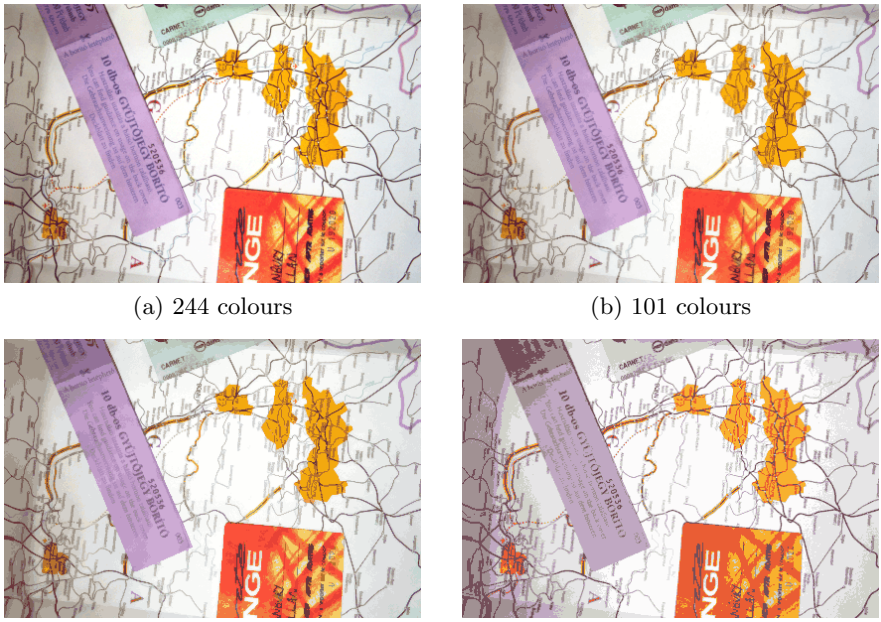
A better approach would be to take the N most frequent colours in the image, by calculating a colour histogram. The disadvantage of this technique, which is called the *population algorithm*, is that rare colours, even if they are important for the representation of the image content, will be thrown out of the colour palette after reduction. Colour reduction by a *median cut* is yet another possible technique. It produces quite good results.

More sophisticated quantization methods use *non-uniform quantization*. Similar colours are grouped together into a *cluster*. Every pixel in such group receives a new colour value, the quantized colour.

An example of colour quantization is shown in figure 3.11. The image in figure (a) contains 244 different colours. This number is decreased with colour quantization, and the last figure only contains 8 unique colours. The quantization technique that is used here, is *peer group filtering*. We also use it in section 3.3. We will therefore take a closer look at this method.

3.1.4.1 Peer group filtering (PGF)

Peer group filtering (PGF) [Deng et al., 1999] is a technique that filters out salt-and-pepper noise (a.k.a. impulse noise) from colour images. Image pixels are replaced by a weighted average of their peer group (see next paragraph). PGF can also be used as a pre-processing step for colour quantization. Local statistics obtained from the PGF procedure are then used as weights in the quantization step. These weights will suppress more colours in detailed regions,



(a) 244 colours (b) 101 colours

(c) 32 colours (d) 8 colours

Figure 3.11: Figure (a) is colour reduced from 244 different colours to 101, 32 and 8 colours. While there is a big difference in number of colours, the visual difference between the images is small.

which is good, since human perception is less sensitive to colour differences in these areas. In smooth areas, the quantization is less severe.

A *peer group* is a set of pixels with similar characteristics. In our case, we discriminate based on colour. Let us denote $\mathbf{x_0}(\mathbf{r})$ the colour information of the image pixel at position $\mathbf{r}$, and the centre of a $w \times w$ window (e.g., $w = 10$). We sort the pixels inside this window, according to their colour similarity. For this, we measure the Euclidean distance of the colours to the colour of the centre pixel (for instance in RGB space):

$$d_i(\mathbf{r}) = ||\mathbf{x}_0(\mathbf{r}) - \mathbf{x}_i(\mathbf{r})|| \ , \tag{3.39}$$

with $i = 0, \ldots, w^2 - 1$. $\mathbf{x}_i(\mathbf{r})$ are the pixels inside the window and they are sorted from small to large distance to $\mathbf{x}_0(\mathbf{r})$, i.e.:

$$d_0(\mathbf{r}) \leq d_1(\mathbf{r}) \leq \ldots \leq d_{w^2-1}(\mathbf{r}) \ . \tag{3.40}$$

We distill a peer group from this set of pixels:

$$P(\mathbf{r}) = \{\mathbf{x}_i(\mathbf{r}) \mid i = 0, \ldots, m(\mathbf{r}) - 1\} \ . \tag{3.41}$$

The size of the peer group, $m(\mathbf{r})$, can be different for every image pixel. We define this threshold further on.

We want to perform a filtering of the image, by replacing the pixels with a weighted average of their peer group. We confine ourselves to the averaging of the peer group, and not of the entire local window, in order to avoid edge blurring. First, we remove the impulse noise. Therefore we define the first-order difference of the distance $d_i(\mathbf{r})$:

$$f_i(\mathbf{r}) = d_{i+1}(\mathbf{r}) - d_i(\mathbf{r}) \ . \tag{3.42}$$

To detect impulse noise, we check the following condition on the first and the last $M = \lceil w/2 \rceil$ points of the set $\mathbf{x}_i(\mathbf{r})$:

$$f_i(\mathbf{r}) \leq \alpha \ . \tag{3.43}$$

The parameter α can be chosen freely. For images with a lot of noise, a higher value for α is used. We set $\alpha = 5$. When we test the first M points, and $f_i(\mathbf{r}) > \alpha$, then the end points $\mathbf{x}_j(\mathbf{r})$, with $j \leq i$, are considered impulse noise and removed from the set. When we test the last M points and $f_i(\mathbf{r}) > \alpha$, then we remove $\mathbf{x}_j(\mathbf{r})$, with $j > i$ from the set. As we can see, if $\mathbf{x}_0(\mathbf{r})$ itself is impulse noise, then it is removed from the peer group and its new value will be determined by the other remaining pixels in the set. We now continue with the remaining set of points $\mathbf{x}_j(\mathbf{r})$ and distances $d_j(\mathbf{r})$.

The threshold $m(\mathbf{r})$ for defining the peer group is calculated with discriminant analysis. We divide the set of pixels into two clusters and maximize *Fisher's linear discriminant criterion* [Duda et al., 2000]:

$$J(i) = \frac{|a_1(i) - a_2(i)|^2}{s_1^2(i) + s_2^2(i)} \ , \tag{3.44}$$

with $i = 0, \ldots, w^2 - 1$, but without the pixels we considered as impulse noise. The averages of the clusters are:

$$a_1(i) = \frac{1}{i} \sum_{j=0}^{i-1} d_j(\mathbf{r}) \ , \tag{3.45}$$

$$a_2(i) = \frac{1}{w^2 - i} \sum_{j=i}^{w^2-1} d_j(\mathbf{r}) \ , \tag{3.46}$$

and the variances are:

$$s_1^2(i) = \frac{1}{i} \sum_{j=0}^{i-1} |d_j(\mathbf{r}) - a_1(i)|^2 \ , \tag{3.47}$$

$$s_2^2(i) = \frac{1}{w^2 - i} \sum_{j=i}^{w^2-1} |d_j(\mathbf{r}) - a_2(i)|^2 \ . \tag{3.48}$$

We calculate the discriminant criterion for every i (that is still available) and take the threshold on the maximal $J(i)$:

$$m(\mathbf{r}) = \arg[\max_i\{J(i)\}] \ . \tag{3.49}$$

With this threshold we can define the peer group $P(\mathbf{r})$ (equation (3.41)). The pixels in the peer group are similar in colour. We use the points in this peer group for the calculation of the new value of pixel $\mathbf{r}$.

The new pixel value is the weighted average of its peer group members:

$$x_0'(\mathbf{r}) = \frac{\sum_{i=0}^{m(\mathbf{r})-1} w_i \mathbf{x}_i(n)}{\sum_{i=0}^{m(\mathbf{r})-1} w_i} \ , \tag{3.50}$$

with $\mathbf{x}_i(\mathbf{r}) \in P(\mathbf{r})$ and w_i are standard Gaussian weights depending on the relative position of $\mathbf{x}_i(\mathbf{r})$ with respect to $\mathbf{x}_0(\mathbf{r})$.

The above procedure filters noise from a colour image and smoothens this image by using colour distances. We now detail the steps necessary to quantize the colours. We use the results from the PGF to determine in which areas of the image the colours may change and how much. We exploit the property that the human visual perception is more critical about colour changes in smooth areas than in textured ones.

For each pixel $\mathbf{x}_0(\mathbf{r})$ we have determined a peer group $P(\mathbf{r})$ containing $m(\mathbf{r})$ elements. The maximum distance in the peer group is:

$$T(\mathbf{r}) = d_{m(\mathbf{r})-1} \ . \tag{3.51}$$

This value gives an indication for the smoothness of the local region. If this value is low, the colours are very similar and thus the area is smooth. The weight for the pixel at position $\mathbf{r}$ is defined as:

$$v(\mathbf{r}) = \exp\left(-T(\mathbf{r})\right) \ . \tag{3.52}$$

So, a colour in a smooth area (low $T(\mathbf{r})$) will receive a higher weight than a colour in a textured area.

For the quantization we start with a number of clusters. This number N depends on the smoothness of the overall image:

$$N = \beta\langle T\rangle \ , \tag{3.53}$$

with $\beta = 2$ and $\langle T\rangle$ the average of $T(\mathbf{r})$ for the entire image.

We use the so-called *splitting initialization algorithm* for the determination of the initial N clusters. We start with one colour cluster, containing all the colour pixels, which we split up into smaller clusters, until we obtain N clusters. The *centroid* c_i and *distortion* D_i of cluster C_i are defined by:

$$c_i = \frac{\sum_{\mathbf{x}(\mathbf{r})\in C_i} v(\mathbf{r})\mathbf{x}(\mathbf{r})}{\sum_{\mathbf{x}(\mathbf{r})\in C_i} v(\mathbf{r})} \ , \tag{3.54}$$

$$D_i = \sum_{\mathbf{r}\in I} v(\mathbf{r})||\mathbf{x}(\mathbf{r}) - c_i||^2 \ . \tag{3.55}$$

The cluster with the highest distortion (e.g., C_j) is divided into two clusters. The centroid and distortion of the new clusters are calculated. This splitting algorithm is repeated until the initial number of clusters N is obtained. The weights $v(\mathbf{r})$ cause the centroids to move towards the pixels of a smooth region. As a result, more colours are obtained in the smooth areas, and colours are merged in detailed regions.

For the vector quantization, the *generalized Lloyd algorithm* (GLA) is used [Lloyd, 1982]. The generalized Lloyd algorithm is a clustering technique. It consists of a number of iterations, each one recomputing the set of more appropriate partitions of the input vectors, and their centroids.

The colour vector of each pixel is mapped onto the closest remaining centroid. To obtain the true cluster centres, the centroids for this final step are calculated without the weights $v(\mathbf{r})$.

3.2 Colour ordering

A binary image is usually interpreted as a two-colour image, the colours typically being black and white. We can apply set theory on this kind of images, because we can simply state that one of the colours is background (i.e., is not part of the set) and the other one represents the objects (i.e., is part of the set). Mathematical morphology was originally developed for binary images, as it was in essence a set theory where the colours white and black can be interpreted as "the pixel belongs to a set" or "the pixel does not belong to a set".[16] We covered this in the previous chapter.

Binary images are a special case of greyscale images. Grey values range from black to white, with different grey values in between. A binarized image is a quantized greyscale image, resulting in a two-colour (black and white) image. The space of the grey values, $\mathcal{G}$, is *totally ordered*, i.e., we can define a relation $\leq$ that orders the elements in $\mathcal{G}$. The following properties hold

[16] Notice that for most examples in chapter 2 black is the colour given to the set members and white the colour of the background.

[Weisstein, WWW]:

$$\textbf{Reflexivity}: \quad \forall a \in \mathcal{G} : a \leq a \ , \tag{3.56}$$

$$\textbf{Antisymmetry}: a \leq b \text{ and } b \leq a \Rightarrow a = b \ , \tag{3.57}$$

$$\textbf{Transitivity}: a \leq b \text{ and } b \leq c \Rightarrow a \leq c \ , \tag{3.58}$$

$$\textbf{Comparability}: \forall a, b \in \mathcal{G} : a \leq b \text{ or } b \leq a \ . \tag{3.59}$$

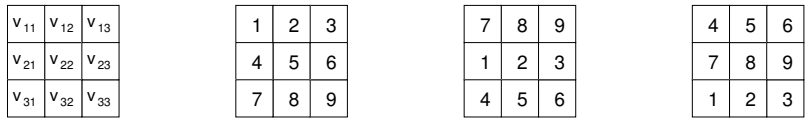
Because of this ordering, we can say whether grey value g_1 is darker or lighter than grey value g_2. The theory of mathematical morphology has been extended, as we explained in chapter 2, from binary images to greyscale images. The basic operators, *dilation* and *erosion*, are then described with a maximum and minimum operation, respectively, instead of the union and intersection operation, respectively.

For colour images, the situation is more complicated. A colour space, like for example the RGB space, cannot be totally ordered in a sensible way. We can represent a colour as a vector in a three-dimensional space. In the RGB space, the origin $(R, G, B) = (0, 0, 0)$ is located on black and the three axes represent the colour bands red, green and blue. This is illustrated in figure 3.7. There exists no obvious "logical" ordering for such vectors. Colour vectors cannot be totally ordered in an obvious way. Several total orderings can be defined, such as the lexicographical ordering, discussed in section 3.2.2. However, in general, such orderings have no physical meaning. Sometimes, only the ordering of one of the components is useful, e.g., $(R, G, B) \leq (R', G', B')$ if $R \leq R'$. We then have a *partial ordering*. The properties stated before are still valid, except for the comparability property, equation (3.59). This property only holds when we consider two colours in the same colour band.

The goal of colour morphology is to extend the morphological theory to colour images by defining a useful order relationship. Since we do not have some natural basis for the ordering of a colour space and cannot define some "universal" total ordering, we need to restrict ourselves to a *sub-ordering* (i.e., a restricted ordering). Four groups of sub-ordering for multivariate data have been proposed [Barnett, 1976]:

- *Marginal ordering* (M-ordering): the data are ordered within one or more components, the marginal samples, of the multivariate data;
- *Reduced (aggregate) ordering* (R-ordering): each multivariate observation is reduced to a single value, which is a function of the component values;
- *Partial ordering* (P-ordering):[17] the observations are classified into different groups that are ordered one to another, but not internally;

[17] This partial ordering or *P-ordering* must not be confused with the previously mentioned *partial ordering* of the colour space. The latter term describes a relation $\leq$ with the properties (3.56), (3.57) and (3.58). The *P-ordering*, although also called a partial ordering, is a specific type of sub-ordering.



(a) Colour image (b) R-band (c) G-band (d) B-band

Figure 3.12: An image and its colour components. A morphological erosion using M-ordering transforms $\mathbf{v}_{22} = (5, 2, 8)$ into $\mathbf{v}'_{22} = (1, 1, 1)$. This colour is not available in the original image and is called a *false colour*.

- *Conditional (sequential) ordering* (C-ordering): the ordering is conducted on a marginal set, in terms of other ordered marginal sets.

We now discuss the M-ordering and C-ordering more in depth.

3.2.1 Component-wise ordering (marginal ordering)

In marginal ordering, the ordering takes place within one or more of the components (i.e., marginal samples). In colour images represented in RGB space, there are three marginal sets, namely the three colour bands R, G and B. Within a colour band it is possible to order the colours. We can order each component independently and perform operations on the separate components. The three results are combined afterwards, to obtain a colour image again.

However, this is not without problems. Regarding mathematical morphology, we dilate or erode the image. If we use the marginal ordering, we consider each colour band to be a greyscale image and perform greyscale morphology on each colour band separately. Afterwards, we combine the different greyscale results to a colour image. This approach can introduce *false colours*. A false colour is a colour that did not exist in the original image, or in the considered image region. For example, figure 3.12(a) shows a part of a colour image with the following colours: $\mathbf{v}_{11} = (1, 7, 4)$, $\mathbf{v}_{12} = (2, 8, 5)$, $\mathbf{v}_{13} = (3, 9, 6)$, $\mathbf{v}_{21} = (4, 1, 7)$, $\mathbf{v}_{22} = (5, 2, 8)$, $\mathbf{v}_{23} = (6, 3, 9)$, $\mathbf{v}_{31} = (7, 4, 1)$, $\mathbf{v}_{32} = (8, 5, 2)$ and $\mathbf{v}_{33} = (9, 6, 3)$. The colour components are shown in figures (b), (c) and (d). If we perform a component-wise erosion operation on the middle pixel, $\mathbf{v}_{22}$, using a 3×3 structuring element, then the resulting component values are 1, 1 and 1, respectively. Our erosion result is thus the colour $\mathbf{v}'_{22} = (1, 1, 1)$, which is not present in the original figure (a), and is therefore a false colour.

Also, the high correlation of the image components makes the marginal ordering a non-ideal ordering solution. As we can see in figure 3.2, the human visual system (and also RGB sensor devices) has overlapping spectral responses. There is a lot of redundancy in the RGB space and the different image components are highly correlated. A change of one component, independent of the other components, is therefore not a good decision. Nevertheless, M-ordering is widely used, because of its simplicity. The M-ordering can be applied for image

filtering or motion estimation in video sequences. To obtain better results, the component-wise ordering is preferably applied in a chromaticity space.

3.2.2 Lexicographical ordering (conditional ordering)

Suppose we have two colours: $\mathbf{v}_1 = (a_1, b_1, c_1)$ and $\mathbf{v}_2 = (a_2, b_2, c_2)$. We can order them using a *lexicographical ordering*[18] scheme. It is defined as follows:

$$(a_1, b_1, c_1) < (a_2, b_2, c_2) \text{ if } \begin{cases} a_1 < a_2 \\ \text{or } a_1 = a_2 \text{ and } b_1 < b_2 \\ \text{or } a_1 = a_2 \text{ and } b_1 = b_2 \text{ and } c_1 < c_2 \end{cases} . \quad (3.60)$$

First, we order the colour vectors by their first components (the rule $a_1 < a_2$). If the components are equal, then we compare the second components. Only if both the first and the second components are equal, we compare the third component. For higher-dimensional elements, this ordering can be easily generalized.

Since the three colour bands in RGB space are correlated and none of the components is considered superior to the others, a lexicographical ordering in RGB will not produce an ordering with a physical meaning. More useful colour spaces are the ones where the chromaticity information is separated from the luminance, like HSL/I/V or $L^*a^*b^*$ [Hanbury and Serra, 2001b]. We will discuss the lexicographical ordering in HSL [Hanbury and Serra, 2001a].

3.2.2.1 Ordering by luminance

The *ordering by luminance* or *L-ordering* is the lexicographical ordering where the luminance component in double-cone HSL space (see section 3.1.3.6) is ordered first, followed by the saturation component and finally the hue:[19]

$$\mathbf{v}_1 > \mathbf{v}_2 \text{ if } \begin{cases} L_1 > L_2 \\ \text{or } L_1 = L_2 \text{ and } S_1 < S_2 \\ \text{or } L_1 = L_2 \text{ and } S_1 = S_2 \text{ and } H_1 \div H_0 < H_2 \div H_0 \end{cases} . \quad (3.61)$$

For the last row, we must take into account that the hue is an angular function defined on the unit circle. We compare the hue value to a value H_0, which can

[18] The term *lexicographical ordering* is sometimes called the *dictionary ordering*, since this is how words are ranked in a dictionary.

[19] It is also possible to order by hue before ordering by saturation. The order of the second and third rule is chosen somewhat arbitrarily.

be chosen arbitrarily. $H \div H_0$ is the acute angle between those hues:

$$H \div H_0 = \begin{cases} |H - H_0| & \text{if } |H - H_0| \leq 180^\circ \\ 360^\circ - |H - H_0| & \text{if } |H - H_0| > 180^\circ \end{cases} . \tag{3.62}$$

For many images, most colours present in the image can be ordered using only the first two rules in equation (3.61). The definition of H_0 is therefore not an important issue for the L-ordering. An ordering by luminance is often used, because in general the most interesting differences are obtained from the luminance component. The edges of objects, for example, are for the most part detectable as a difference in luminance.

3.2.2.2 Ordering by saturation

The *S-ordering* is defined by:

$$\mathbf{v}_1 > \mathbf{v}_2 \text{ if } \begin{cases} S_1 > S_2 \\ \text{or } S_1 = S_2 \text{ and } |L_1 - \frac{1}{2}| < |L_2 - \frac{1}{2}| \\ \text{or } S_1 = S_2 \text{ and } |L_1 - \frac{1}{2}| = |L_2 - \frac{1}{2}| \text{ and } H_1 \div H_0 < H_2 \div H_0 \, . \end{cases} \tag{3.63}$$

The most saturated (i.e., the purest) colour is ranked highest. In case of a tie, the colour with luminance closest to $L = 1/2$ is ranked higher, since at this luminance value the saturation can assume its largest range of values.[20] This is the choice in [Hanbury and Serra, 2001a], and seems logical because e.g. a dilation should take, using the S-ordering, the value of the pixel with the largest saturation as output; if the saturations are equal, we take the pixel with the highest potential to increase further its saturation, i.e., the pixel with L closest to $1/2$. Also for the S-ordering, the definition of H_0 is not an important issue. The ordering by saturation can be used when the background colour is some tint of grey, i.e., a colour with a very low saturation. If the objects are coloured, then it is very likely that their saturation is relatively high. These colours are then ranked higher than the background colour.

3.2.2.3 Ordering by hue

A possible lexicographical scheme for the *H-ordering* could be:

$$\mathbf{v}_1 > \mathbf{v}_2 \text{ if } \begin{cases} H_1 \div 0^\circ < H_2 \div 0^\circ \\ \text{or } H_1 \div 0^\circ = H_2 \div 0^\circ \text{ and } S_1 > S_2 \\ \text{or } H_1 \div 0^\circ = H_2 \div 0^\circ \text{ and } S_1 = S_2 \text{ and } |L_1 - \frac{1}{2}| < |L_2 - \frac{1}{2}| \, . \end{cases} \tag{3.64}$$

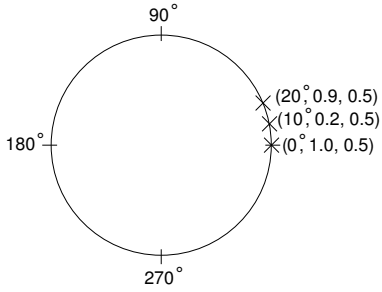
[20] Remember that we work in the double-cone HSL space, not the cylindrical one.

(a) $\mathbf{a} = (0°, 1.0, 0.5)$

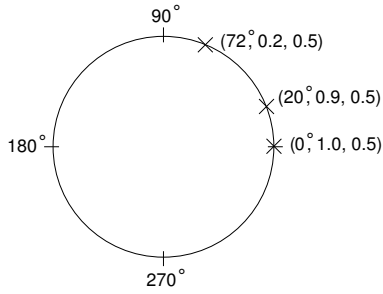
(b) $\mathbf{b} = (20°, 0.9, 0.5)$

(c) $\mathbf{c} = (10°, 0.2, 0.5)$

Figure 3.13: Three sample colours. While colour (a) looks more like colour (b) than like colour (c), the latter will be ordered between the former two, when using H-ordering.



(a) Only hue



(b) Saturation-weighted hue

Figure 3.14: The positions of some colours on the hue circle, (a) before and (b) after taking the saturation value into account.

The problem here is that there is a close interaction between the hue and the saturation component [Hanbury, 2001]: figures 3.13 and 3.14 illustrate this. In figure 3.13, three colours and their HSL-values are shown. These colours are also shown on the unit circle of the hue, figure 3.14(a). If we compare the colours $\mathbf{b} = (20°, 0.9, 0.5)$ and $\mathbf{c} = (10°, 0.2, 0.5)$ to the red colour $\mathbf{a} = (0°, 1.0, 0.5)$,[21] then equation (3.64) will position the grey-purple colour $\mathbf{c}$ closer to this red than the orange $\mathbf{b}$.

When we order in function of the hue, we expect red and orange to be close to each other. The under-saturated colour $\mathbf{c}$ is not perceived as a red-like or orange-like colour (more some grey-purple tint), although it is positioned between $\mathbf{a}$ and $\mathbf{b}$. To solve this problem, a *saturation-weighted H-ordering* is suggested, where an alternative hue in function of the saturation is defined. If the saturation value of a colour is high, the original hue value is retained. In case we have a low saturation value, the hue is shifted away from the reference hue H_0, towards $H_0 + 90°$ or $H_0 - 90°$. This will reduce the probability of being chosen as an extremum when a morphological dilation or erosion is performed.

[21] This means that $H_0 = 0°$.

To simplify the notations, we first redefine H such that $H_0 = 0°$:

$$H'' = \begin{cases} H - H_0 & \text{if } H - H_0 \geq 0° \\ 360° + (H - H_0) & \text{if } H - H_0 < 0° \end{cases} . \tag{3.65}$$

Then, we take the value of the saturation into account:

$$H' = \begin{cases} \sup\{H'', 90°(1 - S)\} & \text{if } 0° \leq H'' < 90° \\ \inf\{H'', 90°(1 + S)\} & \text{if } 90° \leq H'' < 180° \\ \sup\{H'', 90°(3 - S)\} & \text{if } 180° \leq H'' < 270° \\ \inf\{H'', 90°(3 + S)\} & \text{if } 270° \leq H'' < 360° \end{cases} . \tag{3.66}$$

So, low-saturation colours with a hue in the range $[0°, 180°[$ will change hue towards 90°; low-saturation colours with a hue in the range $[180°, 360°[$ will change hue towards 270°.

Finally, we obtain the saturation-weighted hue ordering:

$$\mathbf{v}_1 > \mathbf{v}_2 \text{ if } \begin{cases} H_1' \div 0° < H_2' \div 0° \\ \text{or } H_1' \div 0° = H_2' \div 0° \text{ and } S_1 > S_2 \\ \text{or } H_1' \div 0° = H_2' \div 0° \text{ and } S_1 = S_2 \text{ and } |L_1 - \tfrac{1}{2}| < |L_2 - \tfrac{1}{2}| \end{cases} . \tag{3.67}$$

As we can see in figure 3.14(b), colour **c** has been given a new hue value of 72°. Now, the orange colour **b** is positioned closer to **a** (red). The saturation-weighted hues H' are only used to *order* the colours, the colours themselves keep their original hue. This way, no false colours are introduced.

3.2.2.4 Definition of H_0

The hue is the angle (in the range $[0, 360°[$) of a point on the unit circle. $H = 0°$ represents the colour red, by convention (see also figure 3.9). The hue component cannot be sensibly ordered like the luminance or saturation. For the lexicographical ordering, the angle between the hue and some reference hue H_0 is taken. This way, we can order the hue component, but such ordering of the hue introduces inconsistencies: angles $H \div H_0$ that are (almost) the same can belong to very different colours. To illustrate this statement, consider $H_0 = 180°$ (cyan). If $H \div H_0 = 45°$, the hue can be $H = 135°$ or $H = 225°$. These hues, green and blue, respectively (see figure 3.9), are perceived as very different colours, although their distance to H_0 is the same.

Another problem we face when using the H-ordering, is the choice of H_0: the principal ordering is done in function of the reference hue. A well chosen H_0 is therefore a must. With the L-ordering, the colours with the highest ranking are those with the highest luminance value. With the S-ordering, these are the

colours with the highest saturation value. In the case of the H-ordering, the colours with their hue closest to H_0 are ranked highest. Since there is no hue value that has a logical or intuitive preference over the other hues, we should choose H_0 in function of the image content.[22]

A possible choice for H_0 could be the hue that appears the most in the image. If the background hue is the most present in the image, then all hue values are referenced to the hue of the background.[23]

Another possibility [Hanbury, 2001] is to take the hue value of the average chromaticity vector as H_0. The chromaticity vector is the vector representation of the hue: the hue can be seen as a point on a circle with an angle θ (the actual hue value) and with radius $r = 1$. This hue "vector" in polar coordinates is transformed into its Cartesian coordinates:

$$\begin{cases} x = r\cos\theta \\ y = r\sin\theta \end{cases} . \tag{3.68}$$

The sum of two vectors (in Cartesian coordinates) is:

$$\mathbf{a} + \mathbf{b} = (x_a, y_a) + (x_b, y_b) = (x_a + x_b, y_a + y_b) \equiv (x', y') \ . \tag{3.69}$$

We can generalize this to take the vector sum of all hue vectors. Finally, we transform back to polar coordinates:

$$\begin{cases} \theta' = \arctan\left(\frac{y'}{x'}\right) & \text{if } x' > 0,\ y' > 0 \\ \theta' = \arctan\left(\frac{y'}{x'}\right) + \pi & \text{if } x' < 0 \\ \theta' = \arctan\left(\frac{y'}{x'}\right) + 2\pi & \text{if } x' > 0,\ y' < 0 \\ r' = \sqrt{x'^2 + y'^2} \end{cases} . \tag{3.70}$$

The resultant average angle θ' is the origin of the hue (H_0).

If we take the saturation into account, as we did in the previous subsection, the transformation to Cartesian coordinates (equation (3.68)) becomes:

$$\begin{cases} x_S = rS\cos\theta \\ y_S = rS\sin\theta \end{cases} . \tag{3.71}$$

The average hue now depends on the saturation S.

We can also obtain θ' (i.e., H_0) from equation (3.70) in a different way, by taking the histogram of the hue in the image and then assign the histogram values to the radius r of the respective chromaticity vector. Then we also transform

[22] For the L-ordering and S-ordering this choice is not a major drawback, because the hue component is the last criterion in the lexicographical ordering. Most colour pairs can be ordered using the first and the second rule, so the choice of H_0 will not have a big influence.

[23] The opposite is also possible: the reference hue lies 180° from the most prevalent hue.

to Cartesian coordinates and average the vectors and transform this average vector back to polar coordinates. This approach is similar to the previous method, but in a histogram the data are grouped into bins. For example, if 36 bins are chosen, the accuracy of the hues is $\pm 10°$, which is less accurate than using the previous method (where the original accuracy of the hues is used).

We propose yet another method for the calculation of the average hue. The basic principle is to shift the obtained histogram of the hue. Let us first consider a straightforward example: the average of $H = 170°$ and $H = 190°$ is $H_0 = 180°$. In this case, the average hue is very similar to the hue of the two original colours, which is what we expect from the average. Now, let us assume we have two colours with hue $H = 10°$ and $H = 350°$. These are two tints of red. The big difference in hue value is misleading, because the hues have to be interpreted modulo 360°. The average of these two values, without any conversion from polar to Cartesian coordinates and back, is 180°. This cyan tint is much different from the hue of the input colours and not desired.

In such case, we shift the histogram and take into account the fact that the hue is defined on the unit circle and in the interval $[0°, 360°[$. If we shift the histogram with $+10°$ (i.e., 1 bin-unit when there are 36 bins), then the two mentioned hue angles ($H = 10°$ and $H = 350°$) become 20° and 0°, respectively, so the average hue is 10°; minus the shift ($-10°$) this would give us the correct result, i.e., $H_0 = 0°$. We must of course know if the histogram needs to be shifted, and how much. Therefore, we calculate the variance of every shifted spectrum, and finally calculate the (weighted) average hue from the histogram with the smallest variance.

This method is not very efficient, because we must shift the histogram with every possible value and calculate the variance for each shifted histogram. Nevertheless, it has an advantage over the previously defined approach (equation (3.70)). Consider the hues 0° and 180°. Averaging using equations (3.68), (3.69), and (3.70) leads to an undefined angle. On the other hand, the shifted histogram method always returns an average value (90° in the example).

There is a remark to make: in our example, the result 270° is equally valid,[24] which creates an ambiguous situation. Also, in practice, the situations where the angles annihilate each other (when using equation (3.70)) are rare.

3.2.2.5 Morphology using lexicographical ordering

When we perform a morphological operation, we calculate a maximum or minimum of a number of pixel values. Therefore, we must be able to order the pixel values. Colours can be ordered, for example, using a lexicographical ordering (a C-ordering) in HSL. Practically, we can perform mathematical morphology on colour images in two ways.

[24] We know that $0° = 360°$. A shift by $+180°$ changes the hues to 180° and 360°. Their average is 270° and the variance is the same as for shift $+0°$.

The first possibility is to order the colours only locally instead of globally. That is, for a dilation by a small structuring element we order, at every pixel, the pixels that are covered by the structuring element. The colour that has the highest rank is considered the maximum and thus the dilation result for the considered pixel. A similar approach is taken for the erosion.

This local operation can be quite intensive because a lot of comparisons have to be made. To avoid the recalculations of the order of one pixel compared to another, it is better to transform the HSL image into a scalar image, based on the lexicographical ordering.

A 24-bit RGB image[25] will be transformed into a double-cone HSL image and subsequently into a 24-bit "greyscale" image. We subdivide the 24-bit scalar into three 8-bit scalars.[26] The highest 8 bits are used for the ordering of the colours by the first rule in the lexicographical ordering scheme (equation (3.60)). The following 8 bits are reserved for the ordering by the second rule, and the lowest 8 bits are used for the ordering by the least important rule. This way, each pixel gets some value between 0 and $2^{24} - 1$.

For example, using the L-ordering, the index value of a colour $\mathbf{v} = (H, S, L)$ will be:

$$v = \left\lfloor ((2^8-1)L)2^{16} + ((2^8-1)(1-S))2^8 + ((2^8-1)(1-\frac{H \div H_0}{180^\circ}))2^0 \right\rfloor . \quad (3.72)$$

Notice that this indexing is in fact an R-ordering. We used the lexicographical ordering (C-ordering) to obtain a scalar value for each vector (R-ordering).

3.3 Majority ordering

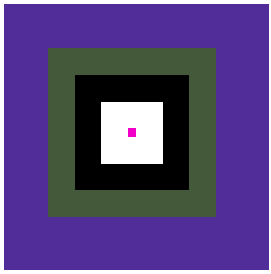
In this section we propose a new type of sorting colours, the *majority sorting scheme* (MSS) [Ledda and Philips, 2005b]. The approach of the majority ordering is quite different from the previously discussed sub-orderings. The basic idea is to count the number of pixels present in the image for each colour and order the colours accordingly.

3.3.1 Basic approach

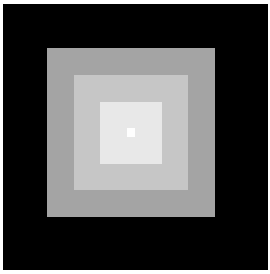
The following assumptions do not hold for all images, but they do for a certain type of images. We consider the background of an image to be mostly made out of a single or a few colours. Also we assume that the background colours are the most prevalent colours in the image. The other most frequent colours

[25] Each colour band (red, green, blue) contains 2^8 values.

[26] This is not the same as is done in a quantized RGB image: in RGB, the three colour components have no mutual ordering, it is still a vector; here, the 24 bits form one scalar value.



(a) Input image



(b) MSS ordering map

Figure 3.15: A colour image is transformed into an image with ordered values, that can be treated as a greyscale image.

are those that are part of the objects and that constitute the basic shapes of the objects. On the other hand, details or noise are rare colours. The "Lena" image, figure 3.4(d), is not a very good example of such type of images. On the other hand, the "Barbara" image, figure 3.23, shows a large background of a few colours, the subject (Barbara) contains a lot of white, black and skin-colour. The detail colours are less prominent (the colours of the books, the eyes, ...).[27]

We use this assumption to construct a new kind of colour ordering which we call the *majority sorting scheme* (MSS) or *majority ordering*.[28] We count the number of times each (quantized) colour appears in the image and order the colours accordingly. A background colour will then probably have the highest number of pixels and the details the lowest number of pixels. This way, we can construct an *ordering map* (or MSS-*map*). The values in this newly obtained image are scalar index values, and not vector coordinates. We have ordered the colours in the image. We could say we performed an R-ordering, although the new scalar is now an image-dependent function. For the moment, we presume that there is a bijective relationship between the different image colours and the indices of the MSS-map. What happens when different colours have the same number of pixels, will be discussed in section 3.3.3.

Figure 3.15 shows a simple example of a colour image. In figure (a) we see that purple is the background, since this colour appears most in the image. The pink pixel, on the other hand, is a small detail or maybe even some noise. The MSS results in figure (b). The black colour represents the colour most present (purple), the white colour represents the most rare colour (pink).

The approach of the majority ordering is illustrated in figure 3.16. The colour input image (I_{in}) is ordered using the MSS, which results in an ordered map

[27] Of course, such real life scene contains thousands of different colours, thus saying that the background only exists of a few colours is an optimistic statement. Reduction of the number of colours is a solution to the problems that this can impose to our ordering technique, as we will discuss in section 3.3.3.2.

[28] A more correct name would be *majority ordering by total area*.

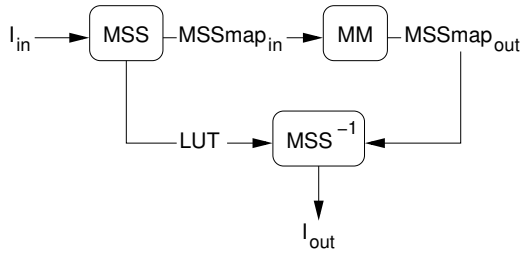


Figure 3.16: The majority sorting scheme (MSS) is conducted on the input image. A morphological operation (MM) is then applied on the ordering map.

($MSSmap_{in}$). This ordered map behaves as if it is a greyscale image, but the number of possible values is not necessarily 256. It could be much more or much less, depending on the number of different colours in the input image. We can now perform any kind of operation that is possible for greyscale images, like mathematical morphological operations (MM). The resulting image is still an ordering map ($MSSmap_{out}$). We note that a morphological operation with a flat structuring element does not introduce new values in the image (here, the ordering map). Other operations, such as an averaging operation or morphology with a greyscale structuring element, can introduce non-integer values, non-existing index values or values that lie above the maximum or below the minimum index value. Such values should be changed into the closest valid (i.e., existing) index values.[29] To obtain a colour image again (I_{out}), we transform the ordering map using an inverse MSS. This inverse transformation simply looks up the colour value that belongs to a certain index in the ordering map. Therefore we need to store this information in a lookup table (LUT) when we construct the MSS-map. The resulting image does not contain any false colours (see section 3.2.1), since we obtain all values from the LUT.

In a greyscale image, black is given the value 0, while white is given the value 255 (for an 8-bit greyscale image). The background is usually a dark shade of grey, i.e., it has lower grey values compared to the other pixels. For this reason, we illustrate the MSS-map by giving the background the grey value black and the rarest colour the grey value white (as in figure 3.15(b)).

Notice that the background has the biggest number of pixels, and the details the smallest number. If we use these colour frequencies to fill the ordering map, then we have values that are opposite to what we expect (i.e., black has the lowest value, white has the highest value). A dilation is a maximum operation and increases the values in the ordering map, which is in this case an increase in background colour (the dominant colour). In a greyscale image (no MSS-map), where the background is assumed dark and the details bright, the opposite happens, namely a decrease in background colour. To solve this inconsistency, we have two options:

[29] In a normal greyscale image, values below 0 would become 0, and values above 255 would become 255. A non-integer value is rounded to the nearest integer.

1. Redefine *dilation* and *erosion*, i.e., switch their definitions (dilation becomes a minimum operation and erosion becomes a maximum operation);
2. Switch the values in the MSS-map: $v_{new} = v_{min} + v_{max} - v_{old}$.

So, actually we should perform a *minority* ordering instead of a majority ordering, but we will keep the original terms "majority ordering" and "majority sorting scheme".

The majority ordering is content-dependent, i.e., image-specific, as opposed to traditional orderings which are known and static, and do not take any knowledge about the image into account. For example, if blue is ranked lower than green, then this is the case for every image. The MSS, on the other hand, is highly dependent on the colours that are present in the image. If blue is ranked lower than green in one image, it is not necessarily the case in another image.

We also point out that only the colours that are present in the image can be ordered. Although the colour space contains many more colours,[30] the MSS-map only contains the indices for the colours in use.[31]

We need to emphasize an important property of the MSS. As we can see in figure 3.16, the morphological operations are performed on the ordered image, $MSSmap_{in}$. We could say that the morphological operations are applied in MSS-*space*. The inverse MSS transformation on $MSSmap_{out}$ produces an output image. We can perform a sequence of morphological operations in two ways:

1. We use I_{out} (figure 3.16) as the input image for the second morphological operation;
2. We remain in MSS-space and use $MSSmap_{out}$ as the input map for the second morphological operation.

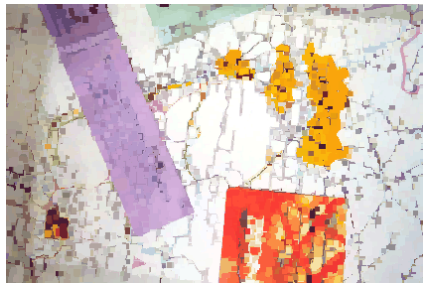
The two approaches seem similar, but in the second case no inverse MSS and no second MSS need to be performed between the two morphological operations. Because the ordering map depends on its input image, the results between the two approaches are different.

For example, the closing of figure 3.11(a) by a 5×5 structuring element can be performed in two ways. The result using the first approach is shown in figure 3.17(a), the second approach outputs figure 3.17(b). If we dilate first, transform back to "real" space (RGB, HSL, ...), and use this result I_{out} as input for the erosion, then we must generate a new ordering map. This MSS-map is not the same as $MSSmap_{out}$ after dilation!

For all the morphological properties discussed in chapter 2 to hold,[32] only the second approach is correct. We must perform all morphological operations at once in MSS-space without converting back to "real" space first. The morphological properties only hold in the MSS-space.

[30] In a 24-bit colour image there are more than 16.7 million different colours.
[31] Which can be at most $M \times N$, the size of the image.
[32] Properties such as increasingness, idempotency, extensivity, ...

(a) Dilation, followed by erosion



(b) Closing

Figure 3.17: An erosion on the output image of a dilation is not the same as a closing, i.e., an erosion on the output MSS-map of a dilation.

3.3.2 Examples

In this section we investigate how the MSS behaves, and how it relates to other ordering schemes. We take the HSL ordering, discussed in section 3.2.2, as a comparison.

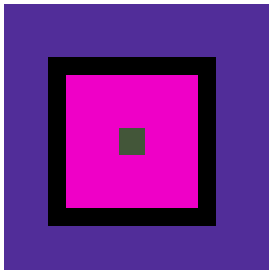
3.3.2.1 Objects of one colour

We first discuss the most trivial situation. If all objects in the image have the same colour and also the background consists of only one colour, then the image can be seen as a binary image. The binary morphological operations will produce the results we expect with the majority sorting scheme as well as with the lexicographical orderings (H-, S- or L-ordering).

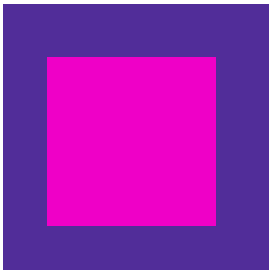
We have to make some remarks, though. In order to obtain the same results as with binary morphological operators (thus, with the set theoretical approach), the background colour should be the most dominant colour for the MSS-ordering. With the lexicographical orderings, it is possible that the background colour is ranked higher than the foreground colour (which means the background is then treated as objects). In the case of the H-ordering, the choice of H_0 is important in order to avoid such situations. In the case of the S- and H-ordering, the comparison of the luminances (see section 3.2.2) is referenced to $L = 1/2$, which means that there is no difference between colours with luminance L and luminance $1 - L$ (and having the same saturation and hue angle). Background and objects are then treated as the same colour.

3.3.2.2 Objects of multiple colours

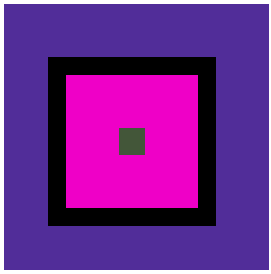
Figure 3.18(a) represents a pink object with a black border. We consider the purple colour to be background. In the middle of the image we notice a small cluster of green pixels. The isolated cluster is probably noise, so we want to



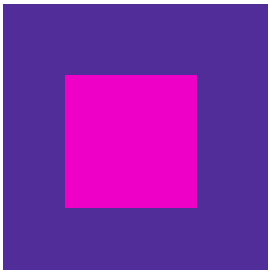
(a) Original image



(b) MSS-ordered opening



(c) $L/S/H$-ordered opening ($H_0 = 0°$)



(d) H-ordered opening ($H_0 = 105°$)

Figure 3.18: Opening (by a 5×5-square) of figure (a).

remove it. The black border is part of the object, but we also want to get rid of this border. A morphological opening seems most suited to perform this task: in a greyscale image the opening would darken the image by removing bright (isolated) objects and thin lines.

When we use the MSS-ordering (figure 3.18(b)), it is possible to remove the artefacts, because the background (purple) and the object (pink) are more prevalent than the border (black) and the isolated cluster (green). The result for the lexicographical ordering technique is shown in figure 3.18(c). If we take for the reference hue $H_0 = 0°$, then the H-, S- and L-ordering do not remove these artefacts. If we know the "correct" H_0-value, then we obtain a good result (figure 3.18(d)). We discussed some possible definitions of H_0 in section 3.2.2.4. It can be a problem to define the right value for H_0.

There is a difference between figure (b) and figure (d): the MSS-ordering considers the black border to be noise or detail and the opening removes this border in favour of the (pink) object. The H-ordering ranks purple higher than pink, which results in a different output as with the majority ordering. This is not necessarily a bad result, but it illustrates that the two methods produce different output images. If we consider the black border to be part of the object and not of the background, then MSS is the ordering of preference.

3.3.2.3 Results on real images

Figure 3.19(a) shows the "jelly beans" image, a colour image with 252 colours. Our goal here is to extract the yellow letters "I P I" by emphasizing them. The yellow from the letters and the colours of the background are the dominant colours. An erosion operation will increase the area covered by these colours. This is shown in figure 3.19(c), while the ordering map is displayed in figure 3.19(b). The grey values are not linearly related to the values of the MSS-map. Darker grey values are dominant colours in figure 3.19(a) and the lighter grey values are rare colours.

The results with the L- and S-ordering are disappointing. These orderings look at the luminance or the saturation values of the colours, respectively. The problem is that the image is not a real life scene, where we can expect some correlation between the luminances and saturations of different pixels. As a consequence, luminance and saturation do not give useful information for colour ordering. The same is true for the H-ordering, but if we choose the reference hue well (here, $H_0 = 250°$, the average hue $+ 180°$),[33] then we also get a nice result.

3.3.3 Variations on the basic method

3.3.3.1 Discrimination of equally frequent colours

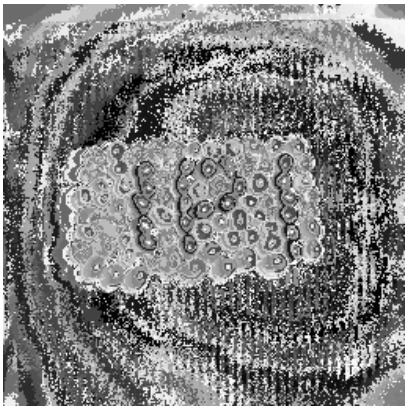
Until now, we considered images where the number of pixels with one colour in an image is different from the number of pixels with another colour. In practice, colours can have the same frequency of occurrence, and therefore have the same rank order in the MSS-map. We present different solutions for discriminating between such colours:

- It is possible to change the rank order of a colour by adding a constant to its frequency of occurrence. This way, we can give each colour a unique index in the ordering map. This approach can be handy if for example a specific background colour has to be chosen. This would be the case when the background colour is not the most prominent colour, as we assume for the MSS.
- The output colour is chosen in function of neighbouring pixel colours.[34] In case of a tie, the output colour is chosen according to the difference of the possible values with that of the input pixel. This difference can be a difference in grey value or hue, for example.

[33] The average hue of the image is very similar to the average hue of the background colours. It is also reasonably similar to the hue of the yellow beans. If this average hue is chosen as H_0, then the yellow and background colours will be ranked higher than the other colours. An erosion will decrease the presence of the pixels with these colours. Therefore, we shift the average hue with 180°.

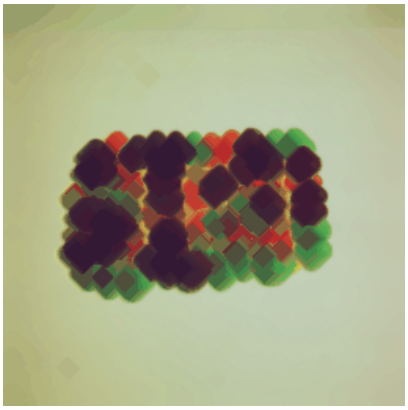
[34] In the case of mathematical morphology, it are the pixels covered by the structuring element that define the output colour.

(a) Original image



(b) MSS ordering map

(c) MSS erosion



(d) L-ordered erosion

(e) S-ordered erosion

(f) H-ordered erosion ($H_0 = 250°$)

Figure 3.19: Erosion (by a diamond with diameter 13) of figure (a).

- Another possibility is to prefer the colour of the pixel closest (in distance) to the current pixel. When there is still a tie, thus in case of equidistant and equally frequent colours, an extra comparison should be made or one of these colours must be chosen arbitrarily.
- A comparison with the background colour is also an option (e.g., choose the colour most similar to the background colour).

In the second proposed solution, we take the difference in grey value or hue with that of the input pixel as an extra criterion to decide which colour will be the output colour for the pixel. It is best to retain the colour with the smallest difference to the input colour, in order to keep the difference in grey value or hue between the original and morphological processed image small.

We performed experiments on several images like the ones in figures 3.19 and 3.11, but they show no (significant) visual differences with the basic MSS approach when this extra discrimination criterion is used. Also, the *Peak Signal-to-Noise Ratio*[35] shows that the differences are small: if we take the morphologically treated image *without* the extra ordering as the reference image and we compare the morphological treatment of the same input image *with* the extra ordering against it, then we obtain PSNR-values of about 35 dB and more. This high value indicates that the differences between the basic MSS approach and the approach with the extra ordering are very small.

3.3.3.2 Quantization of colours

For practical reasons, a quantization of the colours is necessary and can also improve results: if many different colours are present, then the pixel count per colour will be low for many colours.[36] As a consequence, many colours will be assigned the same ranking, which cancels the purpose of the MSS, namely to construct an MSS-map that orders the colours in function of their frequency (which is related to their importance). Therefore, in most cases a quantization is desired.[37] The majority sorting scheme therefore becomes as illustrated in figure 3.20. Before the majority ordering, the image colours are first quantized. We discussed a few quantization methods in section 3.1.4.

In figure 3.21 we illustrate the effect of a dilation on the same image, but each time with a different number of colours present (we quantized the image using peer group filtering (section 3.1.4.1)). We notice that the result of the dilation depends strongly on the number of colours in the input image. If we want to detect the roads in the image, then it is best to reduce the number of colours strongly.

[35] For a definition of the PSNR we refer to chapter 6, section 6.4.5.

[36] In a 24-bit image that displays a real life scene, it is not uncommon to have ten thousands or more colours that appear only a few times (less than 10).

[37] The colour quantization is one possibility. Another option is to rank a colour whose pixels are clustered more lower than a colour whose pixels are scattered more throughout the image.

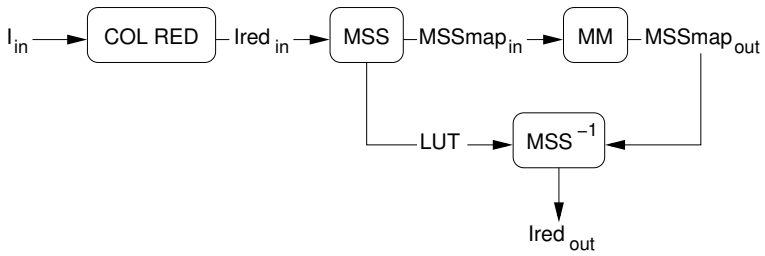


Figure 3.20: The majority sorting scheme (MSS) is conducted on the input image, after colour reduction. A morphological operation (MM) is then applied on the ordering map.

The ideal number of colours after quantization depends on the source image and the application. As we just mentioned, a strong colour reduction is desired for the extraction of roads from a map. Text extraction also works better when there are not many colours present. However, we have to make sure that the useful colour differences in the image are retained.

It is not easy to define the ideal level of colour reduction. It is possible to limit the number of colours to a value selected by the user (e.g., 100 or 32). Another solution we propose is the following: we count the number of colours C in the original image, and we count the number of levels L in the ordering map. If we find more colours than there are levels, then we reduce the number of colours to the number of levels:

$$\begin{cases} C > L \Rightarrow \text{reduce to } L \text{ colours} \\ C = L \Rightarrow \text{ok} \end{cases} . \tag{3.73}$$

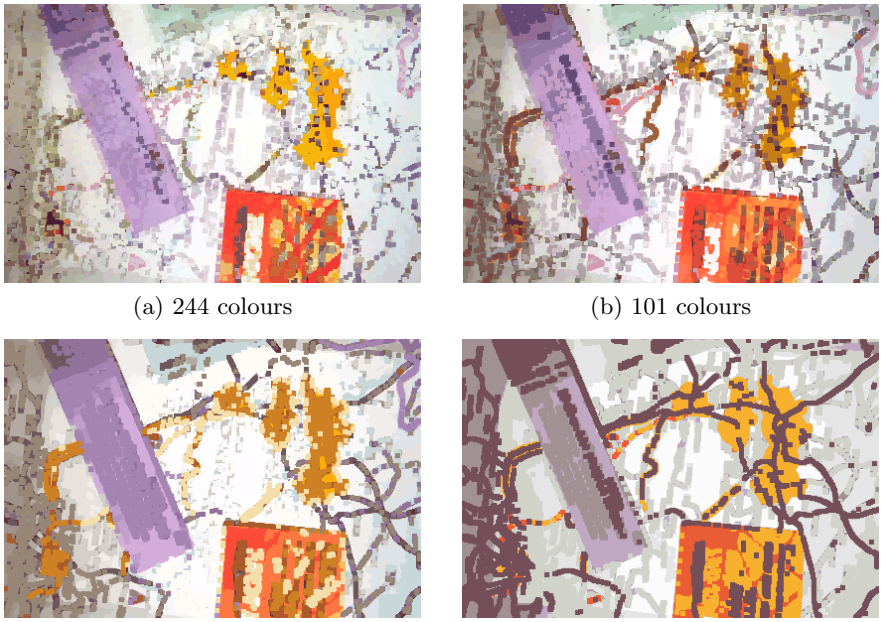
We repeat this procedure until $C = L$. As the number of colours decreases, the number of levels also decreases (or remains constant). The reason is the colour reduction, that implies a new and different ordering map. Since we decrease the number of colours to the number of levels, the newly obtained number of levels will never be more than the previous one:

$$L_{new} \leq L_{old} . \tag{3.74}$$

When $L_{new} = L_{old}$, then also $C = L$ and the colour reduction is completed.

3.3.3.3 Avoiding false colours

When we compare the quantized image to the original one, we notice that false colours have been introduced. This should and can be avoided. The colour reduction is necessary to let the majority ordering work properly. Colours whose mutual difference is perceptually irrelevant, are changed into one new colour, which will influence the pixel counts and thus the MSS-map. Because



(a) 244 colours (b) 101 colours

(c) 32 colours (d) 8 colours

Figure 3.21: MSS-dilation (by a 5×5-square) of the respective images in figure 3.11.

the operations are performed on the MSS-map, the colour values themselves are not important during the calculations. This is advantageous, because we can put the original colours back in the image (actually, in the lookup table) while retaining the new majority ordering index value. This is done before performing some operation on the MSS-map. So, an operation (in the MSS-space) is performed on a quantized image, but with an output image containing only colours originally present.[38]

The scheme from figure 3.20 can be updated to the one in figure 3.22. Figure 3.24 shows the result of four morphological operations on figure 3.23. The "Barbara" image originally contains 242 colours, but after the colour reduction described in subsection 3.3.3.2 (equation (3.73)), the number of colours is reduced to 79. In the end result, no false colours are present, because the original colours were extracted from a lookup table from the original input image.

3.3.3.4 Merging of colours

In some situations it is an advantage to merge the pixel counts of different colours into one single value in the MSS-map. Colour quantization is a sug-

[38] We have a similar situation as in subsection 3.3.3.1: different colours get the same ranking. It is thus possible to discriminate between the different colours (that are quantized into one colour) by using an extra discrimination criterion, as explained in subsection 3.3.3.1.

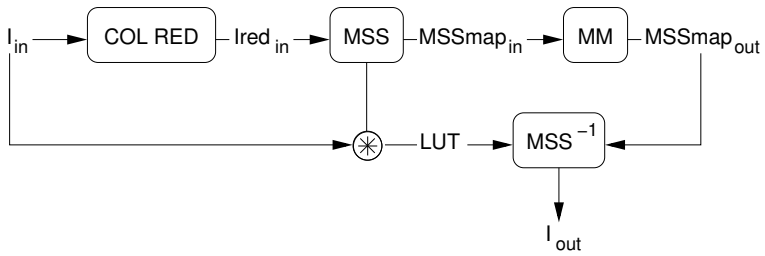


Figure 3.22: The majority sorting scheme (MSS) is conducted on the input image, after colour reduction. A morphological operation (MM) is then applied on the ordering map. False colours are avoided.

gested possibility, as we have seen. In some cases, almost equally important colours (i.e., they have almost the same number of pixels) in the image are too different to be merged by a colour quantization procedure. For example, if the background has two equally prominent colours with almost the same ranking, then one colour would be rated higher than the other in an arbitrary manner. If this is not desired, then we can merge these two colours:

$$\frac{|L_i - L_j|}{\max\{L_i, L_j\}} < \alpha \Rightarrow \begin{cases} L_i \rightarrow L'_i \\ L_j \rightarrow L'_j \\ L'_i = L'_j \end{cases}, \tag{3.75}$$

where L_i and L_j are the MSS-index for the colours i and j, respectively. The value α is a small percentage, say 5 %. $a \rightarrow b$ means that a is replaced by b. It is possible to generalize equation (3.75) for more than 2 colours. The pixels retain their respective colours, but their MSS-index changes to L'_i. Again, an extra discrimination criterion can be used (subsection 3.3.3.1) if a choice has to be made between the two merged colours.

The new index value $L'_i (= L'_j)$ can be calculated in two different ways:

1. We take the sum of the values of the separate colours, i.e., $L'_i = L_i + L_j$. This way, the different colours have merged into one bigger area, although it contains different colours.

2. We calculate the average area, i.e., $L'_i = (L_i + L_j)/2$. This way, the two colours are considered equally important, just like in the first approach, but compared to the other colours in the image their importance does not change much.

3.3.4 Properties

Traditional colour ordering techniques use colour information to obtain an ordering. This colour information can be of all sorts: hue, saturation, luminance

Figure 3.23: The "Barbara" image.

[Hanbury and Serra, 2001a], reference to black or white [De Witte et al., 2005], order inside a colour band, etc. The majority ordering does not take this information into account; it counts the number of times a colour appears in the image. The colour value itself is of no importance, only the ability to distinguish different colours is. We also know that the MSS is a content-based ordering: every image has its own ordering map, the ordering is not pre-defined. We summarize some properties that can be advantageous or disadvantageous.

The technique can be used for *binary*, *greyscale* and *colour* images, since the colour itself is not important, but its cardinality in the image. In fact, an extension to multispectral images is possible with the same majority ordering scheme.

Colour images with only *two colours* are treated as if they are binary images. Also, the *colour* of the background does not matter. This is an advantage if we expect the morphological operators to perform as we expect intuitively. That is, if we expect a dilation to let the objects grow and an erosion to let the objects shrink, the background must be black and the objects white. On the complement of the image, the opposite results will be obtained. With the MSS, the colour itself is no issue anymore.

The *colour space* used to represent the colours, is irrelevant. The majority ordering produces the same result in every colour space, if no colour quantization is used. Also, the technique is quasi invariant for colour and greyscale transformations (e.g., γ-compensation), if no colour reduction has been applied; when all colours change in a new map of unique colours (the transformation is *bijective*), then there is no difference in the majority ordering.

An advantage over the lexicographical HSL ordering discussed in section 3.2.2 is that we do not have to define a reference hue value H_0. The definition of H_0 is rather arbitrary (see section 3.2.2.4), but with the MSS we do not have to bother about this.

(a) Dilation (b) Erosion

(c) Closing (d) Opening

Figure 3.24: Morphological operations using the MSS on the "Barbara" image (figure 3.23).

The assumptions made about images, namely that the colour of the background is the most prominent colour and that the rare colours are details or noise, are at the same time the restrictions of this technique. If the background colour is not the colour most prevalent in the image, then it can be chosen by the user. Too many colours in the image can make the MSS-technique useless (as discussed in section 3.3.3). Finally, if colours have almost the same number of pixels, then the technique can become inaccurate, because maybe one colour is regarded object and another colour noise. The ranking is then not reliable.

3.4 Conclusion

In this chapter, we gave an introduction to the theory of colour; we defined colour, discussed how we perceive it and how we can reproduce it. Different colour spaces exist, such as XYZ, RGB, HSL or $L^*a^*b^*$. They all have their advantages, disadvantages and purposes.

Colours are not totally ordered in an obvious and unambiguous way. Several ways of ordering exist, like marginal ordering or lexicographical ordering. We explained the latter for the HSL colour space. A problem here is that the ordering of the angular hue component poses potential problems. When the colours are ordered, the image can be treated as if it was a greyscale image. Greyscale mathematical morphology can now be applied to the colour image.

We proposed an original colour ordering, the majority sorting scheme (MSS). It is a content-dependent ordering that ranks the colours depending on the number of pixels with those colours in the image. The method assumes that background colours are highly present, while details and noise are pixels with rare colours. Morphological operations using the MSS perform well on such images. A practical problem is the number of colours present. Colour reduction is a suggested solution to limit the amount of colours and to obtain a useful ordering.

Chapter 4

Granulometries

This chapter covers the theory of granulometries. We explain what granulometries are and introduce the morphological pattern spectrum. From this pattern spectrum, several parameters can be derived. They can be used to extract properties from the objects in images, or to perform a classification.

Because of the high computational cost of the pattern spectrum, several alternatives are discussed, such as the area pattern spectrum and the Fourier pattern spectrum. The computation times of the different techniques are compared.

4.1 The morphological pattern spectrum

The concept of *granulometry* is well known in material science [Soille, 2003]. A granulometry is the process of sieving a sample through sieves of increasing mesh size. The smallest objects will be filtered out first, followed by the bigger ones, until finally the largest objects are removed. This way, particles can be sorted form fine to rough. The principle in image processing is similar.

Such a sieving process has the following properties:

- If we sieve a subsample of a larger sample, then the sieving result is also a subsample of the sieving result of the larger sample;
- The residue after a sieving is part of the input sample;
- If we sieve repeatedly using the same mesh size, then we will not filter out new objects.

When we take a closer look at these properties, we notice that these are the properties of the morphological *opening* $A \circ B$ (both binary and greyscale):

- *Increasingness* (equation (2.28)): $A_1 \subseteq A_2 \Rightarrow A_1 \circ B \subseteq A_2 \circ B$;

- *Anti-extensivity* (equation (2.29)): $A \circ B \subseteq A$;
- *Idempotency* (equation (2.30)): $A \circ B = (A \circ B) \circ B$.

Moreover, if we sieve twice, using two different mesh sizes, the residue is what we get when sieving once using the larger mesh. This is the *absorption property* and in the case of the opening this formalizes as:

$$(A \circ B) \circ C = (A \circ C) \circ B = A \circ (\max\{B, C\}) \ , \tag{4.1}$$

where $\max\{B, C\}$ means the structuring element that is the superset of the other structuring element. Equation (4.1) is equivalent to:

$$B \subseteq C \Rightarrow A \circ B \supseteq A \circ C \ . \tag{4.2}$$

A *granulometric curve* is defined as:

$$GC(A; B)(n) = \sharp[A \circ nB], \ n \geq 0 \ . \tag{4.3}$$

The index n increases the size of the structuring element. The operator $\sharp$ computes the sum of all grey values.[1] The *normalized granulometric curve* is the granulometric curve, but with the results scaled (i.e., divided) by the sum of grey values in the original image.

In practice, we look at the part that is sieved out, not the residue. This is called the *pattern spectrum* (PS) or *size distribution*:[2]

$$PS(A; B)(n) = \sharp[A \circ nB - A \circ (n+1)B], \ n \geq 0 \ , \tag{4.4}$$

where the subtraction $-$ is the pixel-wise difference of the grey values. In the binary case, the interpretation of the pattern spectrum is straightforward: the value of a bin n in the size histogram indicates then how many pixels have been sieved out between the opening by the structuring element nB and the structuring element $(n+1)B$.

In the case of a greyscale image, the value in bin n of the size histogram indicates what amount in *grey value* has been sieved out between the opening by the structuring element nB and the structuring element $(n+1)B$. A value v can represent v pixels that decreased one unit in grey value, it can represent one pixel that decreased v units in grey value, but in general it is a combination of both. So, we cannot tell from the pattern spectrum whether a large value indicates that the opening caused many pixels to decrease a little in grey value, or a few pixels a lot. On the other hand, the pattern spectrum of a

[1] In other words: $\sharp[A] = \sum_{\mathbf{a} \in \Omega(A)} A(\mathbf{a})$, with $\Omega(A)$ the support of A. In the case of a binary image (i.e., a set), $\sharp[A]$ is the cardinality of the set A. The $-$-operation in the following equations is then replaced too, with the set difference $\setminus$.

[2] The pattern spectrum is sometimes referred to as the *pecstrum*.

greyscale image reveals subtle changes in grey values, while in the binary case these changes are gone after the binarization of the input image. For example, textured objects will produce another PS than flat objects, which is useful for classification purposes.

We can also define a *normalized pattern spectrum*, similarly as for the granulometric curve (i.e., divide the result by the sum of grey values in the original image). Alternatively, we can define a *structuring element-normalized* PS:

$$PS_N(A;B)(n) = \frac{1}{\sharp[nB]}\ \sharp[A \circ nB - A \circ (n+1)B],\ n \geq 0\ . \tag{4.5}$$

So, the values in the bins n of the size distribution indicate the number of objects that are sieved out, assuming that the objects have the same size and shape as the structuring element nB.

The definition for the pattern spectrum given in equation (4.4) can also be defined for *continuous* functions or sets [Maragos, 1989]. It is a differential size distribution function:

$$PS_c(A;B)(r) = -\frac{d\sharp[A \circ rB]}{dr},\ r \geq 0\ , \tag{4.6}$$

where $rB \equiv H_r(B)$ is defined as in equation (2.47).

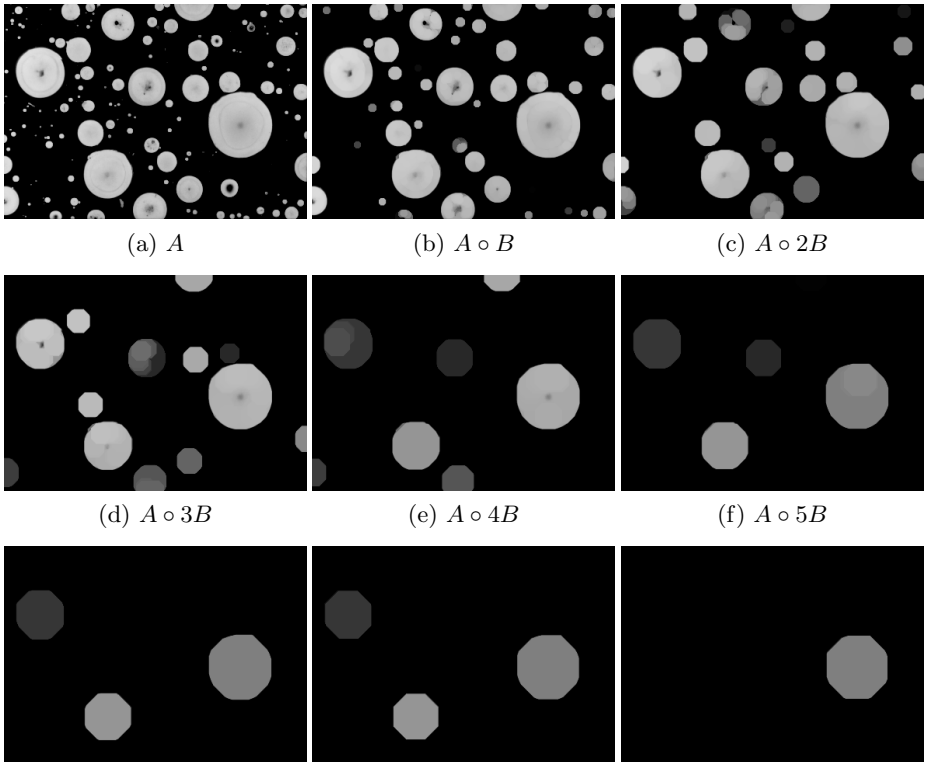
Figure 4.1 shows an example of an image that is sieved using disc-shaped structuring elements with increasing size. First, the smallest objects disappear, the largest object remains until the 10th opening. Figure 4.2 shows its granulometric curve and pattern spectrum. Notice that the granulometric curve is the inverse of a cumulative size distribution, in the sense that the value $GC(A;B)(n)$ decreases from $\sharp[A]$ to 0 for increasing n. The peaks in the pattern spectrum indicate when a lot of object pixels decrease value, thus telling us how many pixels belong to a region of size nB.

4.1.1 Discrete size transform

The morphological pattern spectrum is actually derived from statistics computed on the *discrete size transform* (DST) [Goutsias and Batman, 2000], which is:

$$DST(A;B)(n) = A \circ nB - A \circ (n+1)B,\ n \geq 0\ . \tag{4.7}$$

This is a set of images which have only non-negative values, because of the absorption property (equation (4.2)). The discrete size transform is thus the difference of two successive sievings, and the pattern spectrum is the "magnitude" of the DST. Notice that we can reconstruct the input image from the

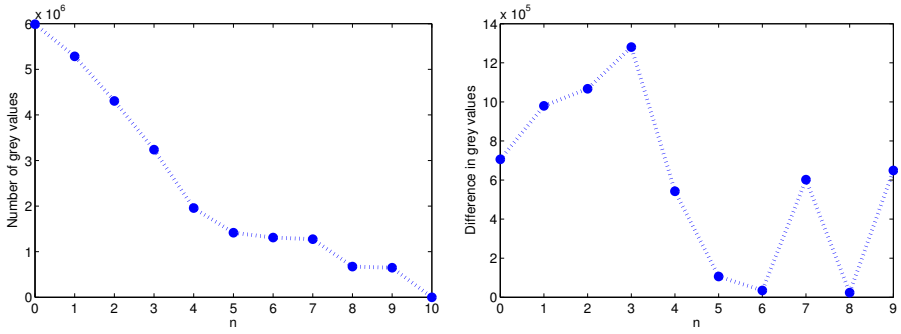


(a) A (b) $A \circ B$ (c) $A \circ 2B$

(d) $A \circ 3B$ (e) $A \circ 4B$ (f) $A \circ 5B$

(g) $A \circ 6B$ (h) $A \circ 7B$ (i) $A \circ 8B$

Figure 4.1: The first eight residues of the granulometry. The structuring element B is a disc with radius 5 pixels.



(a) $GC(A; B)$ (b) $PS(A; B)$

Figure 4.2: The granulometric curve and the pattern spectrum from figure 4.1(a).

elements of the DST, i.e.:

$$A = \sum_{n \geq 0} DST(A; B)(n) \ . \tag{4.8}$$

The DST is a multi-resolution image decomposition scheme, a morphological transform that can be seen as an analogue of the Fourier transform (which will be discussed in section 4.2.5). As is the case with the Fourier transform, it is common practice to work only with the magnitude of the transform, which here is the pattern spectrum.

4.1.2 Oriented pattern spectrum

Different structuring elements can be used in computing the pattern spectrum. The pattern spectrum with a square structuring element performs openings by square structuring elements with increasing size. It indicates what the sizes of the square objects in the image are. When we use a linear structuring element (oriented horizontally, vertically, diagonally, ...), the pattern spectrum indicates what the sizes of the objects in a specific direction are. For example, using a one-dimensional horizontal structuring element for the computation of the pattern spectrum reveals the width of the objects but not the height. So, the pattern spectrum depends both on the input image and the structuring element used for the openings.

In the case of a linear structuring element, or in fact any *anisotropic* structuring element (i.e., being not the same in all directions), the pattern spectrum is the result of a search for objects with a specific orientation. A square is an isotropic object, because we work on a square discrete grid, not in a continuous 2D-space. If we want to find anisotropic objects, but with unknown orientation, we must use the (discrete) *oriented pattern spectrum* (OPS) [Maragos, 1989]:

$$OPS(A; B_\theta)(n) = \sharp[\max_\theta\{A \circ nB_\theta\} - \max_\theta\{A \circ (n+1)B_\theta\}] \ , \tag{4.9}$$

with $n \geq 0$. B_θ are line segments with orientations θ. Typically only a discrete set of angles is considered, for example, $\theta = \{0°, 45°, 90°, 135°\}$. If the input image A is interpreted as a set (i.e., it is a binary image), we replace the maximum operation with a union.

Figure 4.3 shows a structuring element and its rotated versions. Note that for the calculation of the OPS, equation (4.9) must be used. It is not possible to produce the same oriented pattern spectrum using the standard pattern spectrum by combining linear structuring elements. For example, dilating a horizontal by a vertical structuring element results in a square element, which is no more anisotropic.

Let us show how the OPS works differently from the regular pattern spectrum, using a simple example. Consider the image A in figure 4.4. The pattern

(a) $B_{\theta_1} = B_{0^\circ}$ (b) $B_{\theta_2} = B_{45^\circ}$ (c) $B_{\theta_3} = B_{90^\circ}$ (d) $B_{\theta_4} = B_{135^\circ}$

Figure 4.3: Elongated structuring elements B_θ, of different orientations θ.

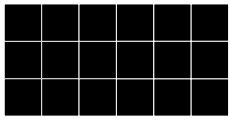


Figure 4.4: A sample image: a 3×6 pixels large black object on a white background. The pattern spectra are different when different structuring elements are used.

spectra for a square structuring element B with size 3×3 pixels, B_{θ_1} and B_{θ_3} (from figure 4.3) are respectively: $PS(A;B) = \{0, 18\}$, $PS(A;B_{\theta_1}) = \{0, 0, 18\}$ and $PS(A;B_{\theta_3}) = \{0, 18\}$. The regular pattern spectrum for a square structuring element reveals square-shaped objects in the image (namely, two touching 3×3 squares). The regular pattern spectrum for a linear structuring element (B_{θ_1} or B_{θ_3}) yields other results. The use of B_{θ_1} reveals horizontally-shaped objects. For figure 4.4, the spectrum indicates the presence of 3 touching lines of 6 pixels in length. The use of B_{θ_3} allows us to detect 6 vertical touching lines of 3 pixels in length.

When we use the oriented pattern spectrum with the structuring elements B_{θ_1} and B_{θ_3}, the result is $PS(A;B_\theta) = \{0, 0, 18\}$. With the OPS, we find the horizontal elongated lines of 6 pixels in length. The oriented pattern spectrum does not detect the smaller vertical objects, since equation (4.9) contains a maximum operation.

If we rotate the input image A over 90°, the regular pattern spectra for the linear structuring elements will change. This is caused by the anisotropy of the structuring elements.[3] The pattern spectrum with the square structuring element B remains the same, because it is an isotropic structuring element. The oriented pattern spectrum also does not change, it is rotation invariant. The big difference with the pattern spectrum for a square or elongated structuring element is that the OPS is able to find anisotropic (e.g., elongated) shapes, regardless of the orientation of the shapes.

4.1.3 Granulometries by closing

The granulometric curve, discrete size transform and (oriented) pattern spectrum are defined in terms of the morphological opening. With these techniques

[3]Notice that the results $PS(A;B_{\theta_1})$ and $PS(A;B_{\theta_3})$ are interchanged.

we can detect and quantify bright objects on a darker background. It is possible to use the closing operator to measure dark objects on a brighter background (such as holes inside objects) using the *granulometry by closing*, a.k.a. *anti-granulometry*. This is because the closing is an extensive operator, instead of an anti-extensive one. The index for the anti-granulometric curve or pattern spectrum by closing is given a negative value ($-n$). When both the opening and closing granulometries are calculated, the anti-granulometry is shown on the left (negative index) and the granulometry is shown on the right (positive index).

A *granulometric curve by closing* is defined as:

$$GC(A;B)(-n) = \sharp[A \bullet nB],\ n > 0\ . \tag{4.10}$$

The *pattern spectrum by closing* is defined as:

$$PS(A;B)(-n) = \sharp[A \bullet nB - A \bullet (n-1)B],\ n > 0\ . \tag{4.11}$$

We recall the duality relationship between the secondary morphological operators (chapter 2, section 2.2.4.1): $(A \bullet B)^c = A^c \circ \breve{B}$ and $(A \circ B)^c = A^c \bullet \breve{B}$.[4] The pattern spectrum by closing is simply the standard pattern spectrum by opening, but for the complement of the image, i.e.:[5]

$$\begin{aligned} PS(A;B)(-n) &= \sharp[A^c \circ (n-1)\breve{B} - A^c \circ n\breve{B}],\ n > 0 \\ &= PS(A^c;\breve{B})(n-1)\ . \end{aligned} \tag{4.12}$$

The *discrete size transform by closing* is defined as:

$$DST(A;B)(-n) = A \bullet nB - A \bullet (n-1)B,\ n > 0\ . \tag{4.13}$$

Notice that we can also reconstruct the input image from the elements of this DST, i.e.:

$$A = [\sum_{n>0} DST(A;B)(-n)]^c\ . \tag{4.14}$$

The *oriented pattern spectrum by closing* is, for $n > 0$:

$$OPS(A;B_\theta)(-n) = \sharp[\min_\theta\{A \bullet nB_\theta\} - \min_\theta\{A \bullet (n-1)B_\theta\}]\ . \tag{4.15}$$

[4] A^c is the complement of the (greyscale) image A. For a greyscale image, the complement v' of a grey value v in an 8-bit image is: $v' = 255 - v$.

[5] Since we use the Minkowski definition, we must also take the reflection of the structuring element into account. For a symmetric element, $B = \breve{B}$, which is generally the case when calculating a pattern spectrum.

4.1.4 Parameters

From the pattern spectrum different parameters can be calculated [Maragos, 1989]. These parameters provide statistical information about the content of the image, like mean object size, shape, direction, variation, ... Using a different structuring element results in another pattern spectrum. This also implies other values for the parameters.

4.1.4.1 Maximal size

The size of an object in the pattern spectrum is defined as $\sharp[nB]$, so we can take the bin-index n as a measure for the size. We define N_{max} as the *maximal size*, i.e., the last bin (the highest n-value, when all image objects are sieved out) of the pattern spectrum histogram. Thus:

$$N_{max}(A;B) = \min\{n \mid \forall n' > n : PS(A;B)(n') = 0\} \tag{4.16}$$

$$= \max\{n \mid \exists \mathbf{a} \in \Omega(A) : \mathbf{a} \in \Omega(A \circ nB)\} \ . \tag{4.17}$$

4.1.4.2 Average size

The *average size* is:

$$S(A;B) = \frac{\sum_{n=0}^{N_{max}} nPS(A;B)(n)}{\sharp[A]} \ . \tag{4.18}$$

As with N_{max}, this value is directly related to the actual *mean object size* or *mean area* $\sharp[S(A;B)B]$. Let us take figure 4.1(a) as an example (its pattern spectrum is shown in figure 4.2). The average size is $S(A;B) = 3.36$ ($N_{max} = 9$), which means that on average the objects have the size $\sharp[3.36B]$, with B a disc with radius 5 pixels, thus the actual mean object size or mean area is a disc with radius 16.8 pixels.[6]

4.1.4.3 Average roughness (entropy)

Another parameter is the *(average) entropy* of the size distribution, also known as the *average roughness*. Its definition comes from information theory [Shannon, 1948].

$$E(A;B) = -\frac{\sum_{n=0}^{N_{max}} PS(A;B)(n) \log_2 \left(\frac{PS(A;B)(n)}{\sharp[A]} \right)}{\sharp[A]} \ , \tag{4.19}$$

[6]Note that the value n in nB (B is a discrete set) is supposed to be an integer (see equation (2.58) in chapter 2). $S(A;B)$ can be a non-integer value and is not meant to reconstruct an image or structuring element with.

where $P \log_2(P) = 0$ if $P = 0$.[7]

If the entropy is small, then the objects in the image are very similar in size. The minimal value of $E(A; B)$ is 0 and is obtained when all objects are sieved out during one opening. In other words:

$$\exists! n \geq 0 : PS(A; B)(n) > 0 \Longrightarrow E(A; B) = 0 \ . \tag{4.20}$$

The larger the entropy, the more different sizes and/or shapes are present in the image. The maximal entropy value is attained when the pattern spectrum is flat, i.e., $\forall 0 \leq n, n' \leq N_{max} : PS(A; B)(n) = PS(A; B)(n')$. This maximal entropy value is $\log_2(N_{max} + 1)$.

The average roughness is a quantification of the shape-size complexity of the image A.

In our example, figure 4.1(a), the average roughness of $PS(A; B)$, with B a disc with radius 5 pixels, is $E(A; B) = 2.88$. The maximal obtainable entropy for $N_{max} = 9$ is about 3.32.

4.1.4.4 Normalized average roughness

It is also possible to calculate the *normalized average roughness*:

$$E_N(A; B) = \frac{E(A; B)}{\log_2(N_{max} + 1)} \ . \tag{4.21}$$

So, it is simply the entropy divided by the logarithm of the number of bins in the pattern spectrum (which is the maximal entropy value). The value of the normalized average roughness lies between 0 and 1.

In our example, figure 4.1(a), the average roughness of $PS(A; B)$, with B a disc with radius 5 pixels, is $E_N(A; B) = 0.87$. Maximal entropy is obtained when $E_N(A; B) = 1$. The example image apparently contains objects of many different sizes.

4.1.4.5 B-shapiness

The *B-shapiness* is the maximal degree to which A contains the shape B, or shapes like B:

$$BS(A; B) = \frac{PS(A; B)(N_{max})}{\sharp[A]} \ . \tag{4.22}$$

[7] In [Maragos, 1989], the use of the base 2 logarithm is not explicitly stated, but it is used in the examples thus assumed implicitly.

It is the pattern spectrum value at N_{max}, divided by the total amount of grey values. The B-shapiness is also considered a quantitative measure of the resemblance of the objects to the shape of the structuring element.

The B-shapiness takes values between 0 and 1. When this value is low, the object shapes should be very dissimilar to the shape of the structuring element. The opposite is true when the B-shapiness is close to 1.

In our example, figure 4.1(a), the B-shapiness of $PS(A;B)$, with B a disc with radius 5 pixels, is $BS(A;B) = 0.11$. If we use a square structuring element B' with size 3×3, then $BS(A;B') = 0.079$.

4.1.5 Computational cost

The pattern spectrum is computed by a sieving operation with openings by increasing structuring elements. We discussed in chapter 2, section 2.2.4.4 that a dilation or erosion takes $MN(P-1)$ calculations for an $M \times N$ image and a structuring element support containing P pixels. Since the opening is an erosion followed by a dilation, the calculation time doubles. For every n we must calculate an opening, and the size of the structuring element increases. One can imagine that the calculation time can become a real problem. In large images it is likely to find large objects, which implies the need for large structuring elements.

To reduce the computational cost, it is important that we implement the opening by taking advantage of the associativity rules (equations (2.61) and (2.62)). An opening by structuring element nB is:

$$A \circ nB = (A \ominus nB) \oplus nB \tag{4.23}$$

$$= (A \ominus (\underbrace{B \oplus B \oplus \ldots \oplus B}_{n \text{ times}})) \oplus (\underbrace{B \oplus B \oplus \ldots \oplus B}_{n \text{ times}}) \tag{4.24}$$

$$= A \ominus \underbrace{B \ominus \ldots \ominus B}_{n \text{ times}} \oplus \underbrace{B \oplus B \oplus \ldots \oplus B}_{n \text{ times}} . \tag{4.25}$$

The last equation reduces the calculation time considerably.

For the calculation of the value of the pattern spectrum $PS(A;B)$ at bin n, we must compute $A \circ nB$ as well as $A \circ (n+1)B$ (see equation (4.4)). For the value at bin $(n+1)$, we must calculate $A \circ (n+1)B$ and $A \circ (n+2)B$. So, when we have computed $PS(A;B)(n)$, we can re-use $A \circ (n+1)B$ for the calculation of $PS(A;B)(n+1)$. This advantage should be taken into account when implementing an algorithm for the pattern spectrum. At bin n, two images must be stored in memory. When $PS(A;B)(n)$ has been calculated, one of the images can be cleared $(A \circ nB)$ and the other one $(A \circ (n+1)B)$ will be used for the calculation of $PS(A;B)(n+1)$.

Unfortunately, it is not possible to use $A \circ nB$ as a starting point for the calculation of $A \circ (n+1)B$. While it is possible to erode $A \ominus nB$ by B to

obtain $A \ominus (n+1)B$, and similarly for the dilation, this is not the case with the opening (as well as the closing). Indeed, adding an extra opening at the end of equation (4.25) does not change the result, because of the idempotency and the absorption property. This means that we must compute an opening for every n. When n increases, the structuring element increases, which results in high computation times when N_{max} is high.

4.2 Other pattern spectra

In the next chapter, we use the pattern spectrum for the processing of large images (size 1030×1300). The calculation time of these images is very large. The calculation of the pattern spectrum for a 512×512 image already takes more than 3 hours.

Therefore we look at alternatives for the pattern spectrum, in order to reduce the calculation time. These alternatives can be an efficient implementation of the pattern spectrum, or another type of histogram of the image that reveals shape information.

In this section, we discuss several alternative "pattern spectra". In section 4.3 we compare the computational cost of the different spectra.

4.2.1 The colour pattern spectrum

Chapter 3 dealt with colour morphology. Greyscale morphology can be used on colour images, if we perform a sub-ordering on the image, such as a component-wise ordering or a lexicographical ordering. We also introduced the *majority sorting scheme* (MSS), an ordering based on the number of times the colours (or grey values) appear in the image.

Calculating the pattern spectrum of colour images is then possible, although the interpretation becomes quite abstract. The values in the histogram do not resemble differences in grey value, as is the case with greyscale images. A greyscale pattern spectrum is also abstract, in the sense that we cannot tell whether an obtained value of the spectrum represents a small decrease in grey value of many pixels or a large decrease of a few pixels. In the case of a colour image, however, not only luminance but also saturation and hue contribute to the spectrum. This complicates the interpretation.

With the MSS, we are able to look at images in a different way, namely to order the colours in function of the image content. We can also apply the MSS to greyscale images, resulting in another pattern spectrum than the traditional greyscale spectrum. The successive openings are performed in MSS-space, i.e., on the ordering maps.

These pattern spectra, the colour pattern spectrum (CPS) and the MSS-PS, are not faster than the standard pattern spectrum (PS). They use the same

algorithm, but it works on different kinds of data. In the case of the CPS using the lexicographical ordering, the colour image must be converted into a 24-bit scalar image using the lexicographical ordering rules. In the case of the MSS-PS, the colours or grey values are re-indexed according to their presence in the image. A colour quantization step might also be necessary, which can increase the total calculation time.

4.2.1.1 Experimental results

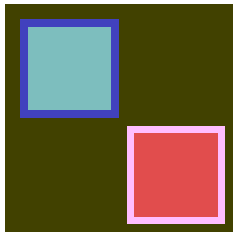
We now show some differences between the spectra obtained with the MSS-PS and the standard PS or CPS.

Objects of one colour If all the objects in the image have the same colour and the background also consists of only one colour, then the image can be seen as a binary image. The binary morphological openings produce the same results as with the majority sorting scheme or with the lexicographical orderings (H-, S- or L-ordering).[8] Therefore: $PS = CPS = PS_{MSS}$. Similar remarks can be made as in the previous chapter, section 3.3.2.1.

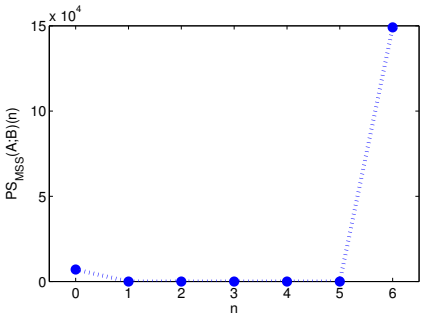
Objects of multiple colours Figure 4.5(a) shows two objects with a small, differently coloured border. The small borders are regarded less important than the objects. The spectra using the lexicographical ordering depend on the colour of the objects and the borders. When we use the majority ordering, the prevalence of these colours defines the ordering of the colours. Pattern spectra for a square structuring element (3×3) are shown in figures 4.5(b) and (c).

The majority sorted spectrum shows a peak at the lowest bin, which indicates pixels that do not specifically belong to a big object. In this case, these are the pixels from the small borders. The large peak at $n = 6$ indicates the disappearance of the two objects. The L-ordered spectrum, figure 4.5(c), has similar peaks as the MSS-PS, but it also has a non-zero value at bin $n = 5$. This extra bin value comes from pixels that change colour after an opening with a 13×13 square structuring element. These pixels are the pixels from the upper object in figure (a), inside the small border. They change into the colour of the surrounding border, which has a lower luminance value. Although this is just an artificial example, such extra peaks can make the interpretation of the pattern spectrum less trivial. We expect the colours to be ordered in a specific way. That is, we expect the borders in figure 4.5 to be part of the square objects. An opening operation should remove the borders in favour of the objects. A colour ordering that follows this assumption is therefore preferred, since it will result in a more meaningful pattern spectrum.

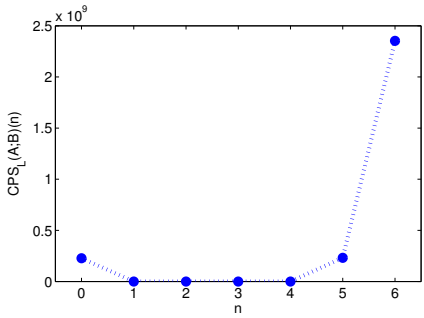
[8] H: hue; S: saturation; L: luminance.



(a) Original image



(b) MSS-PS



(c) CPS (L-ordered)

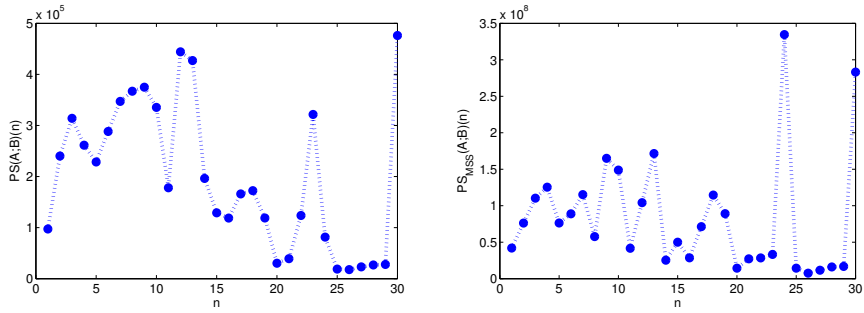
Figure 4.5: Different pattern spectra for an artificial colour image.

Results on real images The pattern spectrum of the image in figure 4.1(a) for a square structuring element (size 3×3) is shown in figure 4.6(a). When we use the majority sorting scheme to re-order the grey values, the spectrum looks like the one in figure 4.6(b).

As we can see, there are similarities between the two spectra. Especially for higher n values, we notice the same behaviour. The MSS-map that the majority ordering generates, is shown in figure 4.7. The colours of the dark centres and the borders of the blobs are apparently less prevalent than those of the blobs and the background. Therefore, these centre and border pixels will be the first to decrease value when calculating the pattern spectrum.

Another example is shown in figure 4.8. Figure (a) is a medical colour image. It is a histological section of a tissue containing several cells. Figure (b) visualizes the MSS-map. The image size is 239×328 pixels and there are 244 different colours present.

Most cells have a diameter between 10 and 20 pixels. This is confirmed by the peak in the range $n = \{5, \ldots, 10\}$ for the pattern spectrum of the majority sorted image, figure 4.9(a). The structuring element used is a 3×3 square. The colour pattern spectra with the L-, S- and H-ordering show peaks at much higher n-values. These peaks suggest the presence of much larger objects in the image, although this is not the case. For this image, the MSS-PS produces the most realistic graph. If we calculate the spectral parameters from the pattern



(a) Pattern spectrum (PS) (b) MSS-PS

Figure 4.6: The spectra PS and MSS-PS for figure 4.1(a).

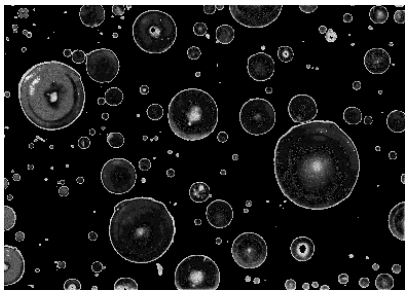


Figure 4.7: The MSS-map of figure 4.1(a), used for the calculation of the MSS-PS.

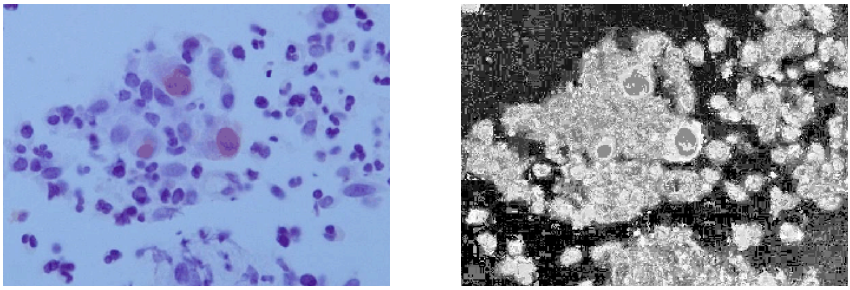
spectrum of the majority sorted image, then we can draw several conclusions. For example, the maximal size calculated is $N_{max} = 34$, which means the largest image structures are sieved out after an opening by a 69×69 square element. The average size is $S = 10.8$ and the normalized entropy is $E_N = 0.9$. This high value indicates the presence of objects of many different sizes. These results can be affirmed by a visual inspection of figure 4.8(a).

4.2.2 The area pattern spectrum

The necessary properties for a granulometry are stated at the beginning of this chapter. A granulometry function is *increasing*, *anti-extensive* and *idempotent*. The *absorption property* also holds. The morphological opening satisfies all these properties, and we defined the pattern spectrum using the opening function.

Some other functions exist that can replace the opening and are called *attribute openings*. One of them is the *area opening*.[9] We measure the area of connected

[9]Other attribute openings are openings in function of the diagonal length, the perimeter or area of the convex hull, ...



(a) Original image (b) MSS ordered map

Figure 4.8: A colour image and its MSS-map.

components, first for the binary case, afterwards we generalize for grey values.[10] Components with an area equal or greater than some threshold value λ are kept, but components with smaller areas are removed from the image. The *binary* area opening of set A by parameter λ is given by:

$$A \circ_\sharp \lambda = \{\mathbf{a} \in A \mid \sharp[C_\mathbf{a}(A)] \geq \lambda\} \tag{4.26}$$

$$= \bigcup_{i \in I} \{A_i \mid \sharp[A_i] \geq \lambda\} \ . \tag{4.27}$$

$C_\mathbf{a}(A)$ is the binary *connected opening*, which is the connected component of set A containing $\mathbf{a}$, if $\mathbf{a} \in A$, otherwise the connected opening is empty. $(A_i)_{i \in I}$ are all the connected components (i.e., objects) of set A.

Figure 4.10(a) shows a binary image (black is object, white is background). In figure (b), objects with an area smaller than 100 pixels are removed from the image (the eyebrows). In figure (c), objects with an area smaller than 500 pixels are removed from the image (the eyebrows and the eyes).

In the case of greyscale images, the grey value decreases to the grey value of the underlying connected component that satisfies the λ-condition. We first define $T_h(A)$, the threshold of the function A at level (i.e., grey value) h:

$$T_h(A) = \{\mathbf{a} \mid A(\mathbf{a}) \geq h\} \ . \tag{4.28}$$

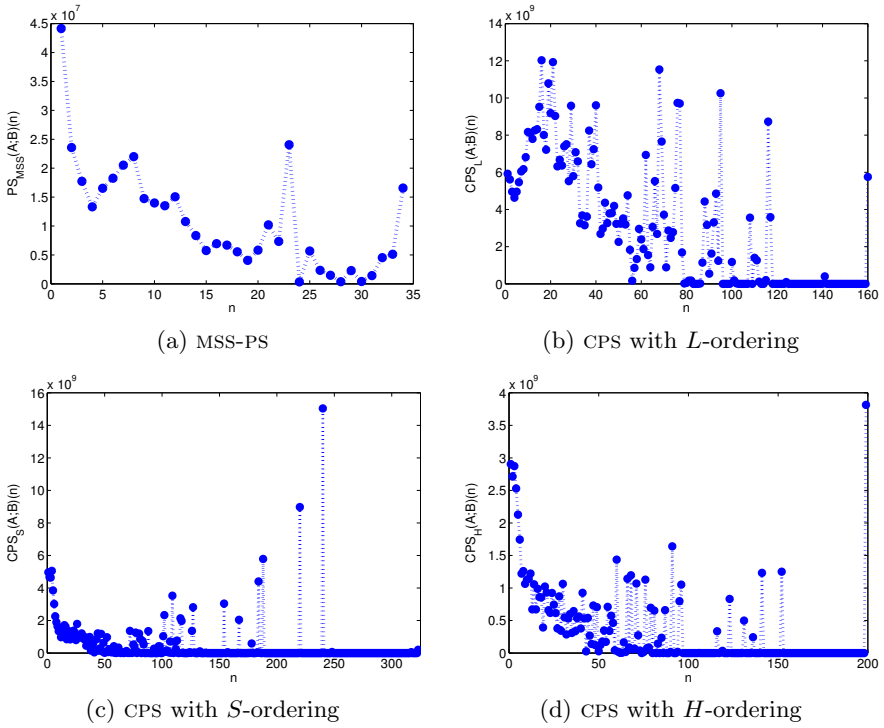
$A(\mathbf{a})$ is the grey value at position $\mathbf{a}$. The *greyscale* area opening of image function A by parameter λ is then given by:

$$(A \circ_\sharp \lambda)(\mathbf{a}) = \max\{h \leq A(\mathbf{a}) \mid \sharp[C_\mathbf{a}(T_h(A))] \geq \lambda\} \tag{4.29}$$

$$= \max\{h \leq A(\mathbf{a}) \mid \mathbf{a} \in T_h(A) \circ_{\sharp_b} \lambda\} \ . \tag{4.30}$$

The operator $\circ_{\sharp_b}$ in equation (4.30) denotes the binary area opening operator.

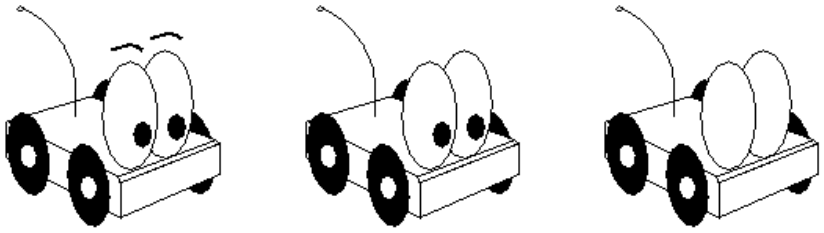
[10] One can choose for 4- or 8-connectivity when connecting the pixels.



(a) MSS-PS (b) CPS with L-ordering

(c) CPS with S-ordering (d) CPS with H-ordering

Figure 4.9: Different pattern spectra for figure 4.8(a).



(a) Original image (b) Area opened ($\lambda = 100$) (c) Area opened ($\lambda = 500$)

Figure 4.10: Area opening of a binary image, with $\lambda = 100$ and $\lambda = 500$. White is considered background.

In figure 4.11(a), we see the greyscale "Lena" image. Figure (b) shows the result after area opening this image with $\lambda = 10\ 000$. As with the traditional opening, the overall brightness has decreased (see for example the hair and feathers on the hat of Lena).

The *area pattern spectrum* (APS) is defined in a similar way as the standard

(a) Input image (b) Area opened image

Figure 4.11: Area opening of a greyscale image, with $\lambda = 10\ 000$.

pattern spectrum:

$$APS(A;\lambda)(n) = \sharp[A \circ_\sharp n\lambda - A \circ_\sharp (n+1)\lambda],\ n \geq 0\ . \tag{4.31}$$

The smaller components (in area) are sieved out first, the largest connected components are sieved out last.

The above equation states that the threshold ($n\lambda$) increases linearly with n. If desired, we can increase this threshold quadratically or in some other way. A generalization of the APS is therefore:

$$APS(A;\lambda)(n) = \sharp[A \circ_\sharp \Psi(n;\lambda) - A \circ_\sharp \Psi(n+1;\lambda)],\ n \geq 0\ . \tag{4.32}$$

The function $\Psi(n;\lambda)$ is increasing.

If we want to approximate the regular pattern spectrum PS for a specific structuring element B, we can increase $\Psi(n;\lambda)$ in such way that its value is the area of nB. A few examples:

- *Linear*: an elongated structuring element B has a length of λ pixels. nB is B dilated $n-1$ times by itself. We approximate the area of such element with: $\Psi(n;\lambda) = n(\lambda - 1) + 1$.

- *Square*: a square structuring element B has sides of λ pixels. For every n we have: $\Psi(n;\lambda) = (n(\lambda - 1) + 1)^2$.

- *Diamond*: a diamond shaped structuring element contains 5, 13, 25, 41, ... pixels if its diagonal is 3, 5, 7, 9, ... pixels, respectively. In a generalized form, this is: $\Psi(n;\lambda) = ((2n\lambda + 1)^2 + 1)/2$, where $2\lambda + 1$ is

the length of the diagonal (and λ is the distance from the origin to the points of the diamond).[11]

- *Disc*: because we work in a discrete space, a discrete disc structuring element is used, which is an approximation of an analytic disc. The disc is the set of pixels that are no more than λ away from the origin, with λ the radius of the disc. The area of the discrete disc can be approximated using the rounding of the area of the analytic disc: $\Psi(n;\lambda) = 1 + \text{round}((n\lambda)^2\pi)$.[12]

The big difference between the standard opening and the area opening is, of course, that the opening tries to match a specific shape of a specific size to the objects in the image, while the area opening calculates the area of connected components, which does not depend on the shape of those connected components.

Efficient implementations for the area opening and the area pattern spectrum exist [Vincent, 1992, Breen and Jones, 1996, Meijster and Wilkinson, 2002]. For the area pattern spectrum we have implemented an algorithm based on Tarjan's *union-find method*. We refer to [Wilkinson and Roerdink, 2000, Meijster and Wilkinson, 2001, Meijster and Wilkinson, 2002] for more details on the implementation of the algorithm. In the following subsection we will give a description of the union-find problem.

The union-find implementation is very fast, because the image is scanned only once. It is practically independent of the value of λ, or the image content. Comparisons of the calculation times of the different pattern spectra will be discussed in section 4.3.

4.2.2.1 The union-find method

In [Meijster and Wilkinson, 2001], the area pattern spectrum is calculated by representing the image pixels as *nodes* in a *tree*. A set can be subdivided into smaller, disjoint sets. Figure 4.12 shows disjoint sets represented as trees. Each set is uniquely identified by the so-called *canonical element* of the set, an arbitrary but unique representative for the set. Three set operations are defined [Tarjan, 1975, Tarjan, 1983]:

- **MakeSet(x)**: a new set is created for the element x, which previously did not belong to any set;
- **Find(x)**: the canonical element of the set containing element x is returned;

[11] If we want λ to be the length of the diagonal instead of $2\lambda + 1$, then we substitute $2\lambda + 1 \rightarrow \lambda$, which results in $\Psi(n;\lambda) = ((n(\lambda - 1) + 1)^2 + 1)/2$.

[12] The discrete disc will contain $5, 13, 29, 49, 81, \ldots$ pixels, for $\lambda = 1, 2, 3, 4, 5, \ldots$. The calculation of Ψ results in areas of $4, 14, 29, 51, 80, \ldots$ pixels, respectively.

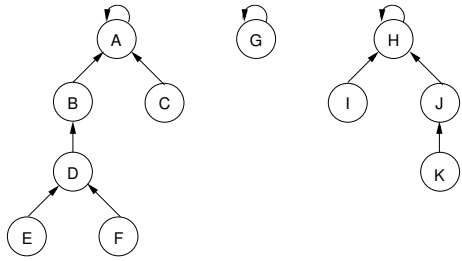


Figure 4.12: The 3 sets $\{A, B, C, D, E, F\}, \{G\}, \{H, I, J, K\}$ are represented as trees, with the elements the nodes in the trees.

- **Link**(x, y): two sets, with canonical elements x and y ($x \neq y$), are united. The original sets are replaced by a new set, with its own canonical element.

As we can see in figure 4.12, each set has a *root* (elements A, G and H, respectively), which is at the same time the canonical element. Each element is a node that points to another node, its *parent*. Notice that the roots point to themselves.

So, the operation `MakeSet(`x`)` would construct a pointer $p(x) = x$. The `Find(`x`)`-operation would look for the parent of x, and then for that element's parent, and so on, until the canonical element has been found. In the figure, `Find(`E`)` would first find D, then B and finally A, the root of the tree. The operator `Link(`x, y`)` merges the sets x and y. The pointer $p(x) = x$ becomes $p(x) = y$ and y becomes the canonical element of the new set.

This union-find method is used for the area opening. First, the pixels in the image are ranked from large grey value to small grey value. The pixels are processed in this order. The function `MakeSet` is called when the pixel under investigation is processed for the first time. Previously processed neighbouring pixels are merged with the function `Link` if they have the same grey value and thus a connected set is created. The grey value can be called with the function `Find`. In the algorithm, a check for the λ-condition is included. A more elaborate description of this technique can be found in [Wilkinson and Roerdink, 2000].

The worst case computational complexity for the union-find approach is on the order of $N \log N$, with N the number of pixels in the image [Meijster and Wilkinson, 2002]. Concerning the memory usage, the union-find needs $2N$ integers: N for the array containing the pixels' parents (which becomes the output image), and N for the sorted pixel array in the algorithm. An extra N memory elements is needed for the input image.

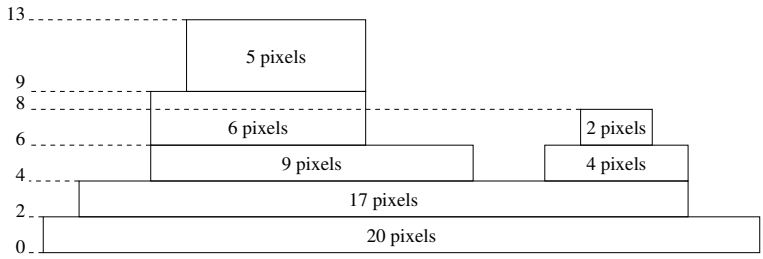


Figure 4.13: A one-dimensional greyscale image. The grey values are shown on the left.

4.2.3 The pattern spectrum using opening trees

The calculation of the morphological pattern spectrum (PS) has a high computational cost. Several attempts have been made to develop efficient morphological algorithms, for basic morphological operations as well as for granulometries. Most of the developed granulometry algorithms are restricted to binary images [Yuan, 1991, Vincent, 1994b, Läy, 1987, Haralick et al., 1992]. They are capable of using two-dimensional structuring elements, i.e., these algorithms are not restricted to process each row or column independently. Fewer techniques also work on greyscale images, and only for one-dimensional (i.e., linear) structuring elements [Vincent, 1994a, Sivakumar et al., 2000].

We have implemented a fast greyscale technique that works with linear structuring elements, but that is extendible to two-dimensional elements [Vincent, 1994a, Vincent, 1995, Vincent, 2000]. It introduces the concept of *opening trees*. We will now explain this approach. For more details about the implementation, we refer to the aforementioned articles.[13]

Consider the example of the one-dimensional greyscale image in figure 4.13. This image contains two so-called *horizontal maxima*, one with length $l(M_1) = 5$ and one with length $l(M_2) = 2$. When we open this image by linear structuring elements of increasing length, i.e., when we perform a granulometry, the pixel values of the upper plateaus[14] will decrease to the grey value of the plateau below them. This sequence of grey value changes for each pixel can be visualized with the *opening tree* (OT) (figure 4.14).[15]

An opening tree consists of the following elements:

- **Tree** T: each image line can be represented as a tree;
- **Nodes** (h, n): the nodes contain a grey value h and an opening size n;
- **Leaves**: these are the image pixels of the tree.

[13] Articles from L. Vincent are available at `http://www.vincent-net.com/luc/papers/`.
[14] The upper plateaus are the horizontal maxima.
[15] Note that this technique has nothing to do with the tree structure from the previous section.

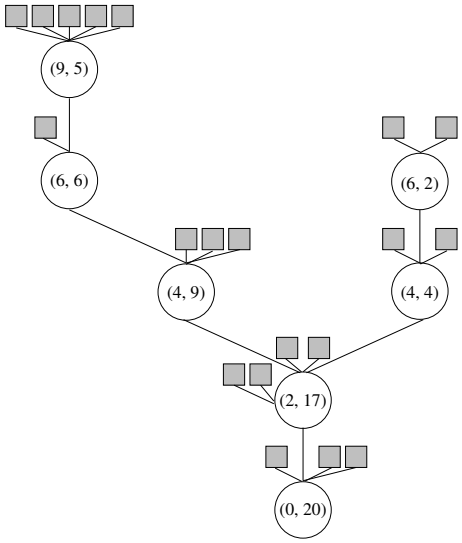


Figure 4.14: The tree representation of figure 4.13. The circles are the nodes (h, n), the squares are the leaves (the pixels).

The opening tree representation of figure 4.13 is shown in figure 4.14. The squares are the leaves and represent the 20 different pixels in the image. Each leave has at least one node (h, n). h is a grey value and n is an opening size. Opening operations are done by nB, where B is a horizontal elongated 1D-structuring element with a length of 2 pixels. Node (h, n) tells us that an opening by nB will result in the grey value h, for any pixel that we can reach by climbing up the tree starting at node (h, n). For example, from this representation we know that the opening of the 5 pixels of the horizontal maximum M_1 by nB, with $n = 9$, changes their grey values to $h = 4$. This information is stored in the node $(4, 9)$.

We can also interpret the tree as follows: the node (h, n) states that there is a plateau of n pixels with a grey value $> h$. The leaves attached to this node point to the pixels that have grey value h', with h' the grey value of the node(s) (h', n') that lie(s) directly above node (h, n). The remaining pixel values are stored in the successive nodes when climbing up the tree. Only the values of the horizontal maxima are not stored in a node (only in the leaves).

When going down to the *root* of the tree, the successive nodes (h_1, n_1), (h_2, n_2), ... satisfy, $\forall i > 0$:

$$h_i > h_{i+1} \ , \tag{4.33}$$

$$n_i < n_{i+1} \ . \tag{4.34}$$

0	0	0	0	0
0	1	2	2	0
0	0	0	0	0

(a) Example 1

1	1	1	1	1	1
1	5	5	2	2	1
1	4	3	1	3	1
1	4	2	2	2	1
1	3	2	2	1	1
1	1	1	1	1	1

(b) Example 2

Figure 4.15: A few sample greyscale images. The numbers are the grey values of the respective pixels.

We also know that:

$$(A \circ nB)(\mathbf{a}) = h_j, \text{ for } n_j \leq n < n_{j+1} \ . \tag{4.35}$$

An opening tree is constructed using the horizontal maxima in the image. The length of a horizontal maximum defines the opening size n. An opening by nB creates a new horizontal maximum with lower grey value. This grey value defines the value h. By remembering the end points of the horizontal maxima, we can avoid redundancy by processing every pixel once and thus significantly increase the processing speed. A more extensive description of this technique can be found in the original papers.

In order to calculate the pattern spectrum with an elongated structuring element B, we must execute the following algorithm:

```
1.  Set each bin of the pattern spectrum PS(A;B) to 0;
2.  For each pixel a of A do:
3.      v = A(a);
4.      (h,n) = node pointed at by a;
5.      While (h,n) exists, do:
6.          PS(A;B)(n-1) = PS(A;B)(n-1) + (v-h);
7.          v = h;
8.          (h,n) = next node down in tree T.
```

As an example, let us calculate the pattern spectrum for figure 4.15(a). We get $PS(A;B) = \{0, 2, 3\}$, like the result of the standard pattern spectrum, equation (4.4). As a second example, we calculate the pattern spectrum of the 3rd column of figure 4.15(b).[16] The result is $PS(A;B) = \{2, 2, 0, 4\}$. Notice that grey value 1 is the background (it is the lowest value of the considered column), so the pixels with value 1 are not linked to a node.

[16]Remember that a tree T is a representation of an image line. To calculate the pattern spectrum with a vertical elongated structuring element, using the OT, every column must be treated independently. Afterwards, the resulting pattern spectra are added.

Computation times between the different techniques are compared in section 4.3, but previous experiments [Vincent, 1994a] already suggest an improvement by a factor of about 1000, compared to the naive implementation of the pattern spectrum. The calculation time of the opening tree pattern spectrum increases linearly with the number of pixels in the image. In the worst case, one node per pixel is needed. In practice, between 0.3 and 0.9 nodes per pixel are needed.

4.2.3.1 Two-dimensional granulometry

The previously mentioned algorithm allows us to compute pattern spectra of greyscale images very quickly. The restriction, however, is that it only works for one-dimensional structuring elements, because the algorithm works on a row-by-row or column-by-column basis. In this subsection, we will show how the representation of the opening tree is used to calculate granulometries using maxima or minima of linear openings in several orientations [Vincent, 2000].

The algorithm can provide results for several orientations at once. The following algorithms are for two orientations.[17] We must perform the following steps:

1. Compute the opening tree for each horizontal image line;
2. Compute the opening tree for each vertical image line;
3. For each pixel, combine its associated horizontal opening tree T_1 and vertical opening tree T_2 and extract its contribution to the pattern spectrum. The trees are followed to their respective roots and along the way the desired values are calculated and added to the pattern spectrum.

In the last step, different trees are combined using maximum or minimum operations. This way, the rows and columns are not independent anymore, the results for the openings by linear horizontal and vertical structuring elements are merged. We show the algorithms used to combine the trees and to compute the pattern spectrum in the following paragraphs. We discuss the resulting spectra in the following subsection.

Granulometry by maxima of linear openings

```
1.  Set each bin of the pattern spectrum OT_max(A;B) to 0;
2.  For each pixel a of A do:
3.      v = A(a);
4.      (h1, n1) = node of T1 pointed at by a;
5.      (h2, n2) = node of T2 pointed at by a;
6.      While (h1, n1) and (h2, n2) exist, do:
```

[17] By default, we take horizontal and vertical structuring elements, but it is also possible to construct the opening tree for a diagonal image line.

```
 7.          s = max{n1, n2};
 8.          While (h1, n1) exists and n1 ≤ s, do:
 9.              (h1, n1) = next node down in tree T1;
10.          While (h2, n2) exists and n2 ≤ s, do:
11.              (h2, n2) = next node down in tree T2;
12.          OTmax(A; B)(s−1) = OTmax(A; B)(s−1)+(v−max{h1, h2});
13.          v = max{h1, h2}.
```

On line 12, the pattern spectrum is updated. The greyscale difference of the pixel value with the grey value after opening with size n is added to bin $n-1$. The opening size is that of the tree with the largest opening size (for that pixel), i.e., a granulometry by maxima of linear openings is computed.

Pseudo-granulometry by minima of linear openings We speak of a pseudo-granulometry, because from the properties stated in section 4.1, idempotency is not satisfied. Minima of linear openings are thus not openings. The algorithm is the following:

```
 1.  Set each bin of the pattern spectrum OTmin(A; B) to 0;
 2.  For each pixel a of A do:
 3.      v = A(a);
 4.      (h1, n1) = node of T1 pointed at by a;
 5.      (h2, n2) = node of T2 pointed at by a;
 6.      While (h1, n1) or (h2, n2) exist, do:
 7.          s = min{n1, n2};[18]
 8.          If (h1, n1) exists and n1 = s, do:
 9.              l1 = h1;
10.              (h1, n1) = next node down in tree T1;
11.          Else:
12.              l1 = +∞;
13.          If (h2, n2) exists and n2 = s, do:
14.              l2 = h2;
15.              (h2, n2) = next node down in tree T2;
16.          Else:
17.              l2 = +∞;
18.          OTmin(A; B)(s − 1) = OTmin(A; B)(s − 1) + (v − min{l1, l2});
19.          v = min{l1, l2}.
```

4.2.3.2 Adaptation of the two-dimensional granulometries

The previous two algorithms produce (pseudo-)granulometries by maxima/minima of linear openings, and the pattern spectra are named $OT_{max}(A;B)$ and $OT_{min}(A;B)$, respectively. The "MAX-opening" approach actually produces an oriented pattern spectrum (see section 4.1.2): for each pixel, the maximum opening size, when comparing between the different orientations,

[18] If n_1 does not exist, then $s = n_2$, and vice versa.

is kept as the opening size that contributes to the pattern spectrum. In the "MIN-opening" case, the minimum opening size is kept.

For binary images, the above algorithms produce spectra as expected, i.e., by means of the maxima or minima of linear openings. For greyscale images, however, we get unexpected results. The following paragraphs explain this. The pattern spectra using opening trees for figure 4.15(a) are: $OT_{max}(A;B) = \{0,4,1\}$ and $OT_{min}(A;B) = \{5,-2,2\}$.[19]

$OT_{max}(A;B)(1) = 4$ indicates a total decrease of 4 grey value units after opening by $2B$. This can be due to one pixel which grey value decreases 4 units, two pixels which grey values decrease with 2, etc. The two pixels with grey value 2 contribute to bin $n = 1$, they are completely removed by a structuring element of 3 pixels in length. However, a lower plateau of value 1 lies below. This plateau has a length of 3 pixels. An opening drops a pixel value to that of its lower plateau (that is kept intact by the opening). This is not the case here. While the author claims his algorithms are correct and work as expected, they behave differently from what we expect (that is, reduce the value to that of its lower (remaining) plateau). In the papers, no example is given to verify the outcome of the algorithm. Further on, we will describe an adaptation of OT_{max}, so that its output equals that of the oriented pattern spectrum (OPS), just like in the binary case.

A strange result is obtained with the algorithm of the pseudo-granulometry. Indeed, the negative values in OT_{min} suggest that some pixel values have increased at a certain stage of the sieving. This means that the minima of openings by elongated structuring elements of increasing length do not constitute a decreasing family of image operators, as expected from a granulometry[20] and stated in the papers.

Therefore, we now present two algorithms that are slightly modified versions of the original ones. The MAX-opening approach now produces the oriented pattern spectrum, while no negative values are obtained with the updated MIN-opening algorithm.

Granulometry by maxima of linear openings

```
1.  Set each bin of the pattern spectrum OT_max(A;B) to 0;
2.  For each pixel a of A do:
3.      v = A(a);
4.      (h1,n1) = node of T1 pointed at by a;
5.      (h2,n2) = node of T2 pointed at by a;
6.      While (h1,n1) and (h2,n2) exist, do:
7.          s = max{n1,n2};
8.          l = max{h1,h2};
```

[19] Applying these algorithms on figure 4.15(b) gives us $OT_{max}(A;B) = \{2,8,7,10\}$ and $OT_{min}(A;B) = \{10,9,0,8\}$.

[20] The property of a decreasing family of operators is contained in the absorption property, equation (4.2).

```
 9.          If h1 = l, do:
10.              (h1, n1) = next node down in tree T1;
11.          If h2 = l, do:
12.              (h2, n2) = next node down in tree T2;
13.          OTmax(A; B)(s − 1) = OTmax(A; B)(s − 1) + (v − l);
14.          v = l.
```

Line 8 is new and lines 9 and 11 are updated versions of lines 8 and 10 in the original algorithm (the while-loop is replaced by an if-block). What the algorithm does, is the following: for each pixel, it looks at the size of the plateau it belongs to, in both horizontal and vertical direction. The bin of the pattern spectrum corresponding to the maximal size (between the two orientations) is updated. The opening is performed in the direction of the tree with the node with the maximal size, so this node is replaced by the next node down in the tree.

The pattern spectrum for figure 4.15(a) is now $OT_{max} = \{0, 2, 3\}$, which equals the output of the OPS. The same is true for figure 4.15(b): $OT_{max} = \{1, 5, 5, 16\}$.

Pseudo-granulometry by minima of linear openings

```
 1.  Set each bin of the pattern spectrum OTmin(A; B) to 0;
 2.  For each pixel a of A do:
 3.      v = A(a);
 4.      (h1, n1) = node of T1 pointed at by a;
 5.      (h2, n2) = node of T2 pointed at by a;
 6.      While (h1, n1) and (h2, n2) exist, do:
 7.          s = min{n1, n2};
 8.          l = max{h1, h2};
 9.          If h1 = l, do:
10.              (h1, n1) = next node down in tree T1;
11.          If h2 = l, do:
12.              (h2, n2) = next node down in tree T2;
13.          OTmin(A; B)(s − 1) = OTmin(A; B)(s − 1) + (v − l);
14.          v = l.
```

This pseudo-granulometry is an exact copy of our adapted granulometry by maxima of linear openings, except that the size on line 7 is obtained by taking a minimum instead of a maximum. Since $l \leq v$, the contribution to the pattern spectrum (line 13) is always positive. The pattern spectrum will therefore contain no negative values.

The pattern spectra for figure 4.15(a) and figure 4.15(b) are, respectively: $OT_{min} = \{5, 0, 0\}$ and $OT_{min} = \{10, 9, 4, 4\}$.

The OT-based pattern spectra still use linear structuring elements. A granulometry by maxima of linear openings is actually an oriented pattern spectrum, using one-dimensional structuring elements. The characteristics of a pseudo-granulometry by minima of linear openings are often similar to those of a

standard morphological pattern spectrum with square structuring elements. An example will be shown in figure 4.19.

4.2.4 The erosion pattern spectrum

We define the *erosion pattern spectrum* (EPS) as:

$$EPS(A;B)(n) = \sharp[A \ominus nB - A \ominus (n+1)B],\ n \geq 0\ . \tag{4.36}$$

The definition is very similar to the pattern spectrum definition (equation (4.4)), but the erosion replaces the opening. The *erosion pattern spectrum* (EPS) is not a granulometry though, because it does not possess all the properties of a granulometry, stated in section 4.1: the erosion is increasing and it is anti-extensive if the origin is part of the structuring element ($\mathbf{0} \in B$). We assume this condition is met. Unfortunately, the erosion is not idempotent, so the absorption property does not hold.

There is a big advantage to this non-idempotency: we can re-use the calculations from $EPS(A;B)(n)$ for $EPS(A;B)(n+1)$. Indeed:

$$A \ominus nB = A \ominus (\underbrace{B \oplus B \oplus \ldots \oplus B}_{n \text{ times}}) \tag{4.37}$$

$$= A \ominus \underbrace{B \ominus \ldots \ominus B}_{n \text{ times}}\ . \tag{4.38}$$

Consequently, we have:

$$A \ominus (n+1)B = A \ominus (nB \oplus B) \tag{4.39}$$

$$= A \ominus nB \ominus B\ . \tag{4.40}$$

Thus, to calculate $A \ominus (n+1)B$, we can re-use $A \ominus nB$ by eroding this result again by B. This advantage improves the calculation speed considerably.

The interpretation of the EPS is much more abstract. While the (opening) pattern spectrum only removes pixels that are part of objects that are too small, compared to the structuring element used by the opening operation, now there will be pixels filtered out of the image for every n.[21]

4.2.5 The Fourier spectrum

The previously discussed granulometries or pseudo-granulometries stem from mathematical morphology. We now take a look at *Fourier analysis* [Pratt, 2001] and how we use the *Fourier spectrum* as an alternative for the pattern spectrum.

[21] In the case of a greyscale image, grey values decrease.

4.2.5.1 Theoretical background

A time signal or spatial signal can be represented in the frequency or spatial frequency domain, respectively. The Fourier transform converts a signal from the time or spatial domain to the frequency domain. The spectrum obtained from this transform, allows us to examine the periodic nature of the signal. The signal can be one-dimensional or higher-dimensional, and continuous or discrete. A greyscale image, for example, is a two-dimensional discrete spatial signal.

Continuous Fourier transform Consider a continuous one-dimensional function $f(x)$. The *Fourier transform* (FT) of this function, $\mathcal{F}\{f(x)\}$, is defined as [Gonzalez and Woods, 2002, Castleman, 1996]:

$$F(u) = \int_{-\infty}^{+\infty} f(x) e^{-i2\pi ux} dx \ , \tag{4.41}$$

where $i = \sqrt{-1}$. The *inverse Fourier transform*, $\mathcal{F}^{-1}\{F(u)\}$, is defined as:

$$f(x) = \int_{-\infty}^{+\infty} F(u) e^{i2\pi ux} du \ . \tag{4.42}$$

Equations (4.41) and (4.42) form a *Fourier transform pair*, because:

$$\mathcal{F}\{f(x)\} = F(u) \Longleftrightarrow \mathcal{F}^{-1}\{F(u)\} = f(x) \ . \tag{4.43}$$

The Fourier transform can be easily extended to multivariate continuous functions. For a two-dimensional function $f(x, y)$, equation (4.41) becomes:

$$F(u, v) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} f(x, y) e^{-i2\pi(ux+vy)} dxdy \ . \tag{4.44}$$

The inverse transform, $\mathcal{F}^{-1}\{F(u, v)\}$, is:

$$f(x, y) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} F(u, v) e^{i2\pi(ux+vy)} dudv \ . \tag{4.45}$$

Discrete Fourier transform We mostly work with discrete images. Therefore, we must use the *discrete Fourier transform* (DFT). In one dimension, this transform is:

$$F(u) = \frac{1}{M} \sum_{x=0}^{M-1} f(x) e^{-i2\pi ux/M} \ , \tag{4.46}$$

where M is the number of samples.[22] The *inverse discrete Fourier transform* is:

$$f(x) = \sum_{u=0}^{M-1} F(u) e^{i2\pi ux/M} \ . \tag{4.47}$$

As with the continuous FT, equations (4.46) and (4.47) can be extended to multivariate discrete functions. For two-dimensional discrete functions, such as digital images, the DFT is:[23]

$$F(u,v) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) e^{-i2\pi(ux/M+vy/N)} \ . \tag{4.48}$$

Notice that $F(0,0)$ is the average grey value if $f(x,y)$ is a greyscale image. The inverse DFT is:

$$f(x,y) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u,v) e^{i2\pi(ux/M+vy/N)} \ . \tag{4.49}$$

An important property is the *separability* of the Fourier transform, i.e., we can factor equation (4.48):

$$F(u,v) = \frac{1}{M} \sum_{x=0}^{M-1} \left[\frac{1}{N} \sum_{y=0}^{N-1} f(x,y) e^{-i2\pi vy/N} \right] e^{-i2\pi ux/M} \ . \tag{4.50}$$

Analogously, the inverse 2D-DFT can be separated into a horizontal and a vertical component:

$$f(x,y) = \sum_{u=0}^{M-1} \left[\sum_{v=0}^{N-1} F(u,v) e^{i2\pi ux/M} \right] e^{i2\pi vy/N} \ . \tag{4.51}$$

We therefore can perform a 1D-DFT on every row of the image, followed by a 1D-DFT on every column of the partly transformed image. The separability reduces the computational cost of the 2D-DFT (see also chapter 6, p. 190).

Another important property is the *periodicity* of the DFT:

$$F(u,v) = F(u+\alpha M, v+\beta N) \ , \tag{4.52}$$

$$f(x,y) = f(x+\alpha M, y+\beta N) \ , \tag{4.53}$$

[22] Sometimes, the constant $1/M$ in front of the Fourier transform is put in front of the inverse Fourier transform, equation (4.47). Sometimes, $1/\sqrt{M}$ is put in both equations.

[23] As in the one-dimensional case, the constant $1/MN$ can be placed in equation (4.49), or $1/\sqrt{MN}$ is placed in both equations.

with α and β integers. This property can be derived from equations (4.48) and (4.49), because $\exp(\pm i2\pi(\alpha+\beta)) = 1$. Thus, we only need an area of $M \times N$ to define all the frequency components of a Fourier transformed image.

Polar representation We can write the Fourier transform in polar coordinates, i.e., $F(u,v) = |F(u,v)| \exp(-i\phi(u,v))$. The *Fourier spectrum* is:[24]

$$\begin{aligned} |F(u,v)| &= \sqrt{F(u,v)F^*(u,v)} \\ &= \sqrt{\mathcal{R}^2(u,v) + \mathcal{I}^2(u,v)} \ , \end{aligned} \tag{4.54}$$

with $\mathcal{R}$ and $\mathcal{I}$ the real and imaginary part of F, respectively. F^* is the complex conjugate of F. The *phase angle* or *phase spectrum* is:

$$\phi(u,v) = \arctan\left(\frac{\mathcal{I}}{\mathcal{R}}\right) . \tag{4.55}$$

Both spectra are needed to reconstruct the original image. Mostly, only the amplitude spectrum (or Fourier spectrum) is visualized. This spectrum specifies *how strongly* each Fourier component contributes.

The *power spectrum* is the square of the amplitude spectrum:

$$\begin{aligned} P(u,v) &= |F(u,v)|^2 \\ &= \mathcal{R}^2(u,v) + \mathcal{I}^2(u,v) \ . \end{aligned} \tag{4.56}$$

Some properties

- *Addition theorem*: $af(x) + bg(x) \longleftrightarrow aF(u) + bG(u)$, i.e., multiplication with a constant or addition in the real domain corresponds to multiplication with a constant or addition in the frequency domain;
- *Similarity theorem*: $f(ax) \longleftrightarrow F(u/a)/|a|$, i.e., narrowing a function in the frequency domain broadens its Fourier transform, and vice versa;
- *Shift theorem*: $f(x-a) \longleftrightarrow F(u)\exp(-i2\pi au)$, i.e., a shift of the image content does not change the Fourier spectrum, only the phase spectrum;
- *Convolution theorem*: $f(x) * g(x) \longleftrightarrow F(u)G(u)$, i.e., costly convolution operations can be avoided by multiplying the Fourier transforms of the functions, at the expense of going back and forth between the Fourier and the spatial domain;
- The Fourier transform of a Gaussian function $f(x) = \exp(-\pi x^2)$ is also a Gaussian, $F(u) = \exp(-\pi u^2)$;

[24] Other names are *amplitude spectrum*, *magnitude* or simply *spectrum*.

- The Fourier transform of a real function is a so-called *Hermite function*, i.e., $F(u) = F^*(-u)$. As a consequence, the Fourier spectrum of a real function is symmetric about the origin $F(0)$.

Fast Fourier transform The one-dimensional discrete Fourier transform has a computational cost on the order of M^2, with M the number of samples.[25] [26] The calculation time can be highly reduced, when a *fast Fourier transform* (FFT) algorithm is used.

Consider $M = 2^p$, with p a positive integer. We can split equation (4.46) up into an even and an odd part:

$$\begin{aligned} F(u) &= \frac{1}{M} \sum_{x=0}^{M/2-1} f(2x) e^{-i2\pi u(2x)/M} + \frac{1}{M} \sum_{x=0}^{M/2-1} f(2x+1) e^{-i2\pi u(2x+1)/M} \\ &= \frac{1}{2} \left[\frac{1}{(M/2)} \sum_{x=0}^{(M/2)-1} f(2x) e^{-i2\pi ux/(M/2)} \right. \\ &\quad \left. + \frac{1}{(M/2)} \sum_{x=0}^{(M/2)-1} f(2x+1) e^{-i2\pi ux/(M/2)} e^{-i2\pi u/M} \right] \\ &\equiv \frac{1}{2} \left[F_{even}(u) + F_{odd}(u) e^{-i2\pi u/M} \right] . \end{aligned} \tag{4.57}$$

Both parts are a summation over $M/2$ samples. The extra complex factor $\exp(-i2\pi u/M)$ is called the *twiddle factor*. Because M is a power of 2, we can recursively split up the even and odd functions p times. This way, we can split up the Fourier transform into M Fourier transforms of length 1 (multiplied by some twiddle factor).

What we have done now, is actually rearranging the terms of the summation. This can be done in a very cost-effective way by using the concept of *bit reversal*. Successively putting the terms with even index to the left and the terms with odd index to the right can be seen as a binary problem. The index is translated into binary code and the binary digits are mirrored. Let us clarify this with an example.

Consider the case where $M = 8$, as illustrated in figure 4.16. The sample indices go from 0 to 7, the binary equivalent is also shown. When we reverse the bits, i.e., we make the least significant bit the most significant bit, and vice versa, then we obtain the order after $p = \log_2 M$ splits, using the principle stated in equation (4.57). Here, $p = 3$. Notice that all even indices are on the left, and the ones divisible by 4 on the extreme left.

[25] We must calculate M Fourier transforms and each Fourier transform contains a summation over M samples multiplied by a complex factor. This results in a computational cost on the order of M^2.

[26] A 2D-DFT has a cost on the order of M^2N^2.

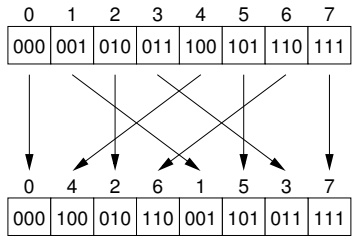


Figure 4.16: Bit reversal used for the FFT, for $M = 8$.

The actual cost reduction comes from the decrease of the number of samples per period. We know that the number of samples for the discrete Fourier transform decreases from M to $M/2$ when we split up into an even and an odd part (see equation (4.57)). This implies the following:

$$F_{even}(u) = F_{even}(u + M/2) \ , \tag{4.58}$$
$$F_{odd}(u) = F_{odd}(u + M/2) \ . \tag{4.59}$$

When we take the twiddle factor into account, we get:

$$F(u + M/2) = \frac{1}{2}\left[F_{even}(u) - F_{odd}(u)e^{-i2\pi u/M}\right] \ , \tag{4.60}$$

for $0 \leq u < M/2$. With this in mind, we calculate the complexity of the FFT (of the so-called Cooley-Tukey algorithm). Let us denote C_M the complexity of a DFT of length M. $C_1 = 0$, since the Fourier transform of length 1 is an identity operation. The split into an even and an odd part has the following computational cost:

$$C_M = 2C_{M/2} + \frac{M}{2} + M \ . \tag{4.61}$$

There are two DFTs to calculate, $M/2$ multiplications (with the twiddle factors) and M summations. These are the $M/2$ summations $F(u) = \frac{1}{2}\left[F_{even}(u) + F_{odd}(u)e^{-i2\pi u/M}\right]$ for $0 \leq u < M/2$ and the $M/2$ summations $F(u) = \frac{1}{2}\left[F_{even}(u) - F_{odd}(u)e^{-i2\pi u/M}\right]$ for $M/2 \leq u < M$ (because of equation (4.60)).

We can generalize equation (4.61) for a recursive split of the Fourier transform:

$$C_M = 2^q C_{M/2^q} + \frac{M}{2}q + Mq \ , \tag{4.62}$$

with $0 < q \leq p$ an integer denoting the split-level. If we split p times (i.e.,

$q = p$), then we have M Fourier transforms of length 1. We get:

$$C_M = MC_1 + \frac{M}{2}\log_2 M + M\log_2 M \tag{4.63}$$

$$= \frac{3}{2}M\log_2 M \ . \tag{4.64}$$

So, the computational cost of the fast Fourier transform is on the order of $M\log_2 M$ instead of M^2. For a 2D-FFT, the cost will reduce to $N(M\log_2 M) + M(N\log_2 N) = MN\log_2(MN)$. The transform of an image with size 1024×1024 pixels gets a speed bump by a factor of more than 50 000.

4.2.5.2 Pattern spectrum versus Fourier spectrum

From the Fourier spectrum, interesting information regarding the image content can be extracted. This is also true for the pattern spectrum. When we compare the behaviour of these two spectra, we find several similarities [Goutsias and Batman, 2000]:

1. (a) A smooth image, i.e., an image with no small details or rapid grey value variation, is characterized by large values in the Fourier spectrum at low frequencies and small or zero values at high frequencies.
   (b) An image that contains many large (smooth) objects and a few or no small objects, is characterized by large values in the higher part of the pattern spectrum (i.e., high n value) and small or zero values in the lower part of the pattern spectrum.
2. (a) An image with fast grey value variation has a lot of detail, noise and/or different content. In the Fourier spectrum this is characterized by small or zero values at low frequencies and large values at high frequencies.
   (b) An image that contains many small (rough) objects and no large (smooth) objects, is characterized by small or zero values for high n values of the pattern spectrum and large values for low n values of the pattern spectrum.
3. (a) The Fourier spectrum is a histogram of the distribution of energy over the frequency domain.
   (b) The pattern spectrum is a histogram of the distribution of sizes. These are the sizes of the various objects in the image.
4. (a) In order to reconstruct the input image from the Fourier spectrum, also knowledge of the phase spectrum is necessary.
   (b) We cannot reconstruct the input image from the pattern spectrum.[27]

[27] On the other hand, the discrete size transform (section 4.1.1) can be used to reconstruct the original image.

So, the Fourier spectrum and the morphological pattern spectrum have several things in common. They are both distributions that give us information about the objects in the image. The low frequency content of the Fourier spectrum reveals the global shapes in the image, just as the higher part of the pattern spectrum does. High values in the other part of both spectra indicate the presence of details and/or noise.

4.2.5.3 The Fourier pattern spectrum

The Fourier transform (FT) of an image gives us an amplitude spectrum (or spatial frequency spectrum) that reflects the greyscale variation in the image. Large image objects will contribute to the low-frequency part of the spectrum, small objects will contribute to the higher frequencies. The Fourier spectrum lacks the ability of spatial localisation, but this is also the case with the pattern spectrum. We assume that the size distribution of the objects in the image has its effect on the shape of the Fourier spectrum.

The Fourier transform of an image is two-dimensional. In order to calculate some first-order histogram features and to remove the orientation dependence of the Fourier spectrum, we transform the 2D-spectrum into a 1D-spectrum. On the abscissa, we put the spatial frequencies $\sqrt{u^2+v^2}$.[28] The resulting histogram is the *Fourier pattern spectrum* (FPS).

We must point out that in the DFT, redundant information is present. Equation (4.52) states the periodicity of the DFT, and we also mentioned that the Fourier transform of a real function is a Hermite function. We can merge these two properties:

$$F(u,v) = F^*(-u+\alpha M, -v+\beta N)\ . \tag{4.65}$$

As a result, the Fourier spectrum is symmetric around the origin, i.e., $|F(u,v)| = |F(-u,-v)|$. This property and the periodicity of the spectrum give us a redundancy of about 50 % for the Fourier spectrum, on an area $M \times N$ [Pratt, 2001]. We should take this into account when converting the Fourier spectrum to the FPS. This is shown in figure 4.17: the greyed-out regions in the amplitude spectrum can be generated from other samples, because of the above property. Also, the frequencies on the right can be seen as negative frequencies, i.e., $N/2+1$ would be $-N/2$ and $N-1$ would be -1, for example.

4.2.5.4 Parameters

Several parameters exist for the Fourier histogram [Pratt, 2001]. Some of these parameters have already been defined for the pattern spectrum.

[28] For example, the spectrum magnitudes at $(u,v) = (4,5)$ and $(u,v) = (5,4)$ are added and put in the bin at location $\sqrt{41}$.

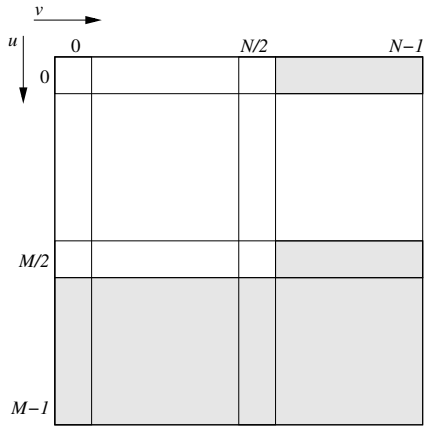


Figure 4.17: The frequency domain of the discrete Fourier transform. The grey area is redundant for a real function.

Mean The *average* value of the histogram is defined similarly as the average size (equation (4.18)):

$$S_M = \bar{b} = \sum_b bP(b) \ , \tag{4.66}$$

with b the histogram index and $P(b)$ the histogram value at index b divided by the total of the histogram.

Standard deviation The *standard deviation* defines the spread of the values:

$$S_D = \sigma_b = \sqrt{\sum_b (b - \bar{b})^2 P(b)} \ . \tag{4.67}$$

Skewness The *skewness* is a measure for the asymmetry of the histogram curve:

$$S_S = \frac{1}{\sigma_b^3} \sum_b (b - \bar{b})^3 P(b) \ . \tag{4.68}$$

Kurtosis The *kurtosis* matches the curve shape to the shape of a normal distribution:

$$S_K = \frac{1}{\sigma_b^4} \sum_b (b - \bar{b})^4 P(b) - 3 \ . \tag{4.69}$$

Energy The *energy* is defined by:

$$S_N = \sum_b P(b)^2 \ . \tag{4.70}$$

Entropy The definition for the *entropy* is similar to that of the average roughness (equation (4.19)):

$$S_E = -\sum_b P(b) \log_2 P(b) \ . \tag{4.71}$$

4.3 Comparison of computation times

We compare the processing time of the pattern spectrum (PS) from section 4.1 with the alternative suggestions introduced in section 4.2. As we will see, the difference in calculation time can be very high. The pattern spectrum needs larger and larger structuring elements, while other techniques only need to scan the image once or always use the same structuring element.

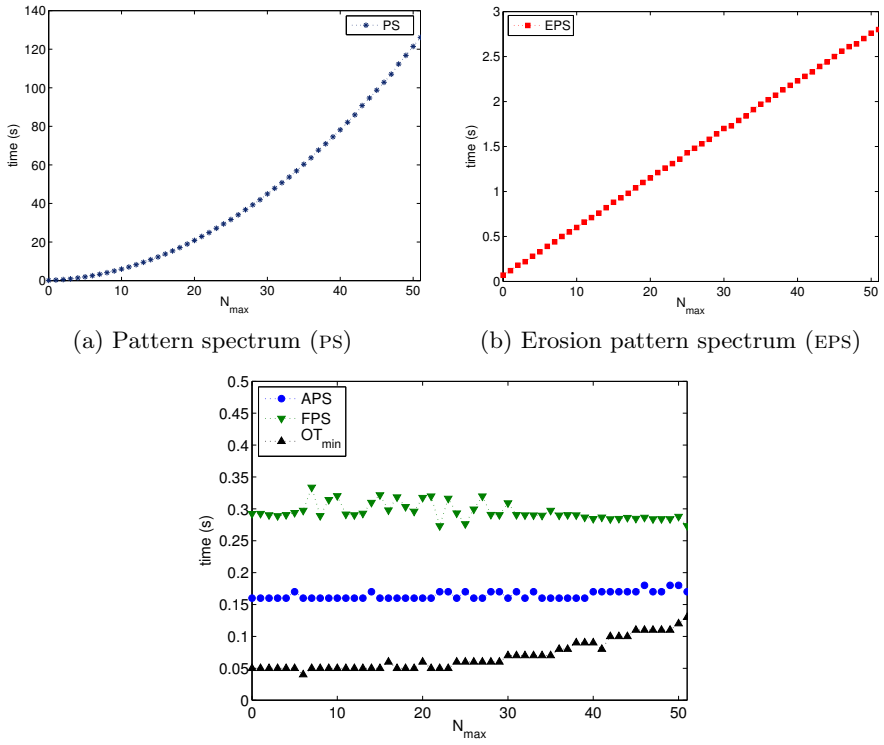
4.3.1 Influence of N_{max}

We process a set of artificial test images with different pattern spectrum techniques. These techniques are: the morphological pattern spectrum (PS), the erosion pattern spectrum (EPS), the area pattern spectrum (APS), the pseudo-granulometry (OT_{min}) and the Fourier pattern spectrum (FPS).

There are 52 binary test images of 512×512 pixels in size. The first image contains one square object with size 2×2 pixels. The second image contains an additional object of size 4×4, etc. The final image of the set contains 52 objects, with the largest one a square of size 104×104. If we calculate the pattern spectrum with a square structuring element with size 3×3, the first image will have $N_{max} = 0$. This number increases linearly until $N_{max} = 51$ for the last image of our test set.

Figure 4.18 shows the timing results for the different spectra. The timing experiments are performed on an AMD Athlon XP 2200+ (1.8 GHz, 1.5 GB RAM) running Linux, kernel v2.6.3. We used the `time` command in Linux to perform the measurements.[29] As we can see from figure (a), the calculation time of the pattern spectrum increases quadratically. The erosion pattern spectrum on the other hand, figure (b), increases linearly with N_{max}. The other three investigated spectra have a (quasi) constant computational cost. When the

[29] The calculation time (using `time`) is the addition of the measured CPU-seconds used by the system on behalf of the process (in kernel mode) and the CPU-seconds used directly by the process (in user mode).



(a) Pattern spectrum (PS)

(b) Erosion pattern spectrum (EPS)

(c) Other spectra

Figure 4.18: Calculation time to calculate the different pattern spectra.

opening tree is used (OT_{min}), a small quadratic increase is visible, but its calculation time is still the smallest of all considered spectra.

These artificial binary images already show the problem that arises when using a standard implementation for the pattern spectrum: when the input images become more complex, N_{max} can easily grow by 1 or 2 orders of magnitude, which results in a very inefficient algorithm.

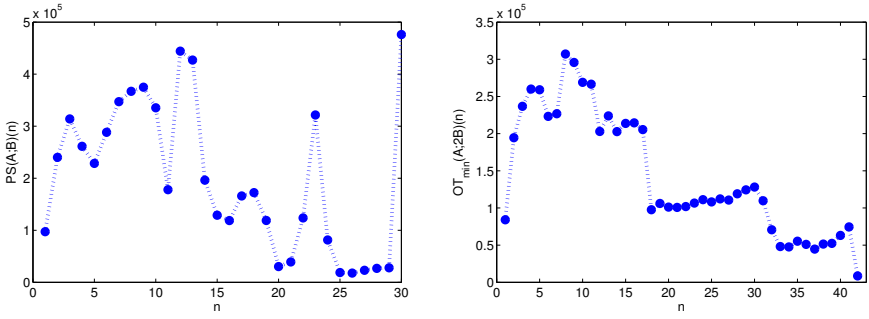
4.3.2 Computational cost on realistic images

We calculate different pattern spectra for the image shown in figure 4.1(a). The structuring element used is a 3×3 square element. For the area pattern spectrum we mimic a square structuring element and set $\lambda = 3$ (see section 4.2.2). The image has size 277×390 and contains 227 different grey values. The spectra are calculated 10 times to get a more accurate result.

The calculation times are shown in table 4.1. In this case, the area pattern spectrum is the fastest technique, followed by the pseudo-granulometry. The

Table 4.1: The calculation time for the different pattern spectra for figure 4.1(a), using a square structuring element (3×3) or an approximation of a square. The standard deviation is indicated for each result.

Spectrum	N_{max}	Calculation time (s)
PS	29	17.14 ± 0.38
EPS	29	0.651 ± 0.020
APS	39	0.0790 ± 0.0074
OT_{min}	83	0.0960 ± 0.0052
FPS	—	0.12 ± 0.059



(a) Pattern spectrum (PS) (b) Pseudo-granulometry (OT_{min})

Figure 4.19: Comparison of the pattern spectrum with the pseudo-granulometry.

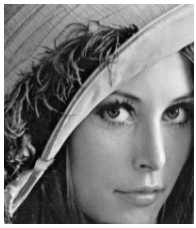
standard pattern spectrum PS is the only technique that needs more than a second to calculate its spectrum.

Figure 4.19 shows the pattern spectrum and the pseudo-granulometry.[30] Although they are not the same, similarities can be observed between the two spectra. Thus, the pseudo-granulometry by minima of linear openings has similar characteristics as the pattern spectrum with a square structuring element.

Another example is a cropped version of the "Lena" image, with a size of 234×202 pixels and 199 different grey values (see figure 4.20(a)). We also calculated the colour pattern spectrum (CPS) of figure 4.20(b), using the lexicographical ordering (H-, S- or L-ordering). There are 40 091 colours in the image.

Table 4.2 shows the calculation results. In this example, the Fourier pattern spectrum (FPS) is the fastest. Because of the conversions involved (i.e., transforming the RGB image into a HSL image and ordering the colours) and the

[30] The bins of OT_{min} have been merged two by two, in order to let the spectrum be like one where an elongated structuring element with a length of 3 pixels has been used (instead of length 2).



(a) Greyscale version

(b) Colour version

Figure 4.20: The input image, a cut-out of the "Lena" image.

Table 4.2: The calculation time for the different pattern spectra for the cropped "Lena" image, using a square structuring element (3×3) or an approximation of a square. The standard deviation is indicated for each result.

Spectrum	N_{max}	Calculation time (s)
PS	136	149.0 ± 1.2
EPS	136	1.2280 ± 0.0063
APS	108	0.0430 ± 0.0048
OT_{min}	200	0.2700 ± 0.0047
FPS	—	0.041 ± 0.013
CPS (H)	185	3134.8 ± 2.3
CPS (S)	223	4510.0 ± 5.8
CPS (L)	185	3105.9 ± 2.8

high value for N_{max}, the CPS is very slow (about one hour is needed). Note that a higher number of grey values or colours does not necessarily mean a higher computational cost for the (colour) pattern spectrum. The previously investigated image is larger and contains more grey values than the cropped "Lena", but only needs 17 s to calculate the pattern spectrum PS. For the current image, 2.5 minutes are needed. It is the complexity of the "Lena" image that increases the calculation time, i.e., larger greyscale plateaus are present, thus larger structuring elements are needed to sieve all the objects out.

The full "Lena" image (see figure 4.11(a)) has a size of 512×512 pixels and contains 216 different grey values. If we take the colour version of the "Lena" image (figure 3.4(d), p. 61), then 148 279 colours are present.

The results are shown in table 4.3. Again, the Fourier pattern spectrum is the fastest technique, followed by the area pattern spectrum (APS). The standard pattern spectrum is calculated in more than 3 hours.

As we can see, the computational cost for PS, as well as CPS, is very high. The calculation time increases non-linearly with N_{max}. This parameter mainly

Table 4.3: The calculation time for the different pattern spectra for the "Lena" image, using a square structuring element (3×3) or an approximation of a square. The standard deviation is indicated for each result.

Spectrum	$\mathbf{N}_{max}$	**Calculation time (s)**
PS	507	11530 ± 430
EPS	507	26.185 ± 0.027
APS	255	0.3670 ± 0.0048
OT_{min}	509	3.216 ± 0.034
FPS	—	0.2944 ± 0.0019
CPS (H)	388	7510 ± 310
CPS (S)	415	8580 ± 230
CPS (L)	312	4860 ± 150

depends on the image content: large objects, or large plateaus of grey values, imply a high value N_{max}. In the case of greyscale images, a small difference in background colour means that it is very probable that there are large plateaus in the image. The size of these plateaus are likely to increase when the image size is large.

As an approximation of the pattern spectrum, the pseudo-granulometry by minima of linear openings performs about 3 orders of magnitude faster than the regular pattern spectrum. The erosion pattern spectrum is about 2 orders of magnitude faster. In general, the Fourier pattern spectrum performs best, immediately followed by the area pattern spectrum (4 orders of magnitude faster).

Most techniques work with integers, because only (integer) grey values are compared, added or subtracted. An exception is the Fourier pattern spectrum, where we need floats. When we convert from RGB to HSL for the calculation of the colour pattern spectrum, or when we perform a colour quantization before applying the majority ordering for the MSS-PS, then we must use floats too.

The different techniques have different memory requirements. Some have been discussed before. If an image contains N pixels, then we always need at least N integers for the storage of the input pixels. The final spectrum also needs a certain amount of memory, depending on the image content.

For the pattern spectrum (PS) an extra N integers are needed for the storage of $A \circ nB$, as well as N integers for the storage of $A \circ (n+1)B$. The resulting difference of both openings can be temporarily stored in the first array. An extra small amount of memory is required for the storage of the structuring element B.

The area pattern spectrum (APS), implemented with the union-find method, needs an extra $2N$ integers: N for the array containing the pixels' parents

(which becomes the output image), and N for the sorted pixel array in the algorithm.

The opening tree algorithms (OT) need at most one node per pixel. A node contains a value h and a value n. For the 2D-(pseudo-)granulometries we need to scan the image twice (horizontally and vertically). Therefore, the maximum number of additional integers needed is $4N$.

For the erosion pattern spectrum (EPS) we can re-use the input array for $A \ominus nB$, but we need an extra N integers for the storage of $A \ominus (n+1)B$. The resulting difference of both erosions can be temporarily stored in the first array. An extra small amount of memory is required for the storage of the structuring element B.

The Fourier transform results in N complex values, but for the Fourier spectrum we know there is 50 % redundancy when the function is real. A colour image, finally, contains 3 colour bands, which triples the required amount of memory.

4.4 Conclusion

In this chapter, we introduced the morphological *pattern spectrum* (PS). It is a size distribution histogram of the objects in an image. The objects are sieved out and counted according to their shape and size. Several interesting statistical parameters can be obtained from the pattern spectrum, such as average size or entropy. The pattern spectrum is defined for binary and greyscale images.

We suggested the *colour pattern spectrum* (CPS) and *majority sorting scheme pattern spectrum* (MSS-PS) for the spectrum of a colour image. A colour spectrum is preferred over a greyscale spectrum if relevant information is stored in the colour. If we restrict ourselves to the luminance values, important information of saturation and/or hue will be ignored. The MSS-PS is based on the concept of ordering the colours according to their frequency of occurrence in the image. This approach can be used on images where the grey or colour values are not naturally ordered.

Because of the high computational cost, we have examined several other spectra. The *area pattern spectrum* (APS) is similar to the PS, but does not take the shape of the objects into account. This is useful if we do not look for a specific shape, but are interested in the size of the objects. We might be interested in the length of the objects to ensure us that they all have the same length or do not deviate too much. By objects we mean connected components. In the greyscale case, the connected components are treated per grey value.

The principle of *opening trees* allows us to calculate the pattern spectrum with one-dimensional structuring elements very quickly. It is the preferred method for classifying linear objects in function of their length and orientation. A granulometry by maxima of linear openings is in fact a very fast calculation

of the *oriented pattern spectrum* (OPS). The oriented pattern spectrum can be used for the examination of anisotropic objects with unknown orientation. The pseudo-granulometry by minima of linear openings can be used as an approximation of the pattern spectrum with a square structuring element.

We proposed the *erosion pattern spectrum* (EPS). It is not a granulometry, but it is fast compared to the pattern spectrum. We also suggested a one-dimensional histogram of the Fourier spectrum of the image, the so-called *Fourier pattern spectrum* (FPS). The usefulness of these two methods will be tested in the next chapter.

When comparing the computation times of the different spectra, we notice that the time for the calculation of the morphological pattern spectrum increases quadratically with N_{max}, and linearly for the erosion pattern spectrum. The Fourier pattern spectrum is the fastest technique, followed by the area pattern spectrum and the opening tree algorithm.

Chapter 5

Analysis of Sliding Bearings

This chapter deals with the applications of the pattern spectra, mentioned in chapter 4, to the analysis of composite materials and polymers. First, we will introduce the problem. Afterwards, we examine the different pattern spectra and their parameters. We investigate the correlation between the spectral parameters and the parameters of the experimental set-up. While the experiments have only been performed on polymers, other composite materials can be examined in a similar way.

5.1 Introduction

This section gives a brief overview of composites and tribology. We refer interested readers to [Halling, 1973, Matthews and Rawlings, 1993, Khonsari and Booser, 2001, van Beek, 2004] for more in-depth information about these topics.

5.1.1 Composites

Composites are heterogeneous materials, made out of at least two different and clearly distinct phases [Degrieck, 2002]. A phase is a distinct state of matter in chemical composition and physical state in the composite. In addition to the natural composites (e.g., wood, teeth, ...), there are a lot of artificial/technical composites. Examples are plywood (known in Dutch as triplex or multiplex), reinforced concrete, car tires, ... We are interested in the *fibre reinforced materials* in which fibres are combined with some plastic, metal or other matter. The *fibres* are added to strengthen the material. They are added to a binding agent, which is called the *matrix*. The fibres and matrix are separated by the *interface*, which is accomplished by a *finish* or *sizing* (i.e., some after-treatment) on the fibres. The interface binds the fibres with the matrix and

it also protects the fibres. The combination of fibres, matrix and interface determines the properties of the composite material.

5.1.1.1 The fibres

The fibres can be made of glass, carbon, metal, ceramic or an organic material. The fibres in the composite determine the stiffness and limit the growth of cracks in the matrix. They also need to carry the load applied on the composite.

Most used are glass fibres, because of their relative low price and good strength. Different types exist, with different properties (high electrical resistance, high chemical resistance, higher strength and stiffness, etc.).

Carbon fibres have a low strain, which is generally not desirable, but on the other hand their coefficient of thermal expansion is very low. This last property makes carbon fibres very useful for applications in astronautics.

Most of the organic fibres are polyaramide (a.k.a. aramide fibres). Kevlar and Twaron are well-known brands of aramide-based products that are used in bulletproof vests and fire-resistant clothing.

5.1.1.2 The matrix

The matrix for the composite is usually made of polymer (PMC), metal (MMC) or ceramic (CMC). The most popular materials used are the polymers, because they are easy to process, they have a low density and good chemical and dielectric properties. A distinction is made between thermoplastics and thermosets.

Thermoplastics can be recycled, melted and re-molded. This kind of polymers can be amorphous (e.g. polysulfone), semi-crystalline (e.g. polyamide) or liquid crystal (e.g. all-aromatic copolyester). Other well known examples of thermoplastic polymers are nylon, polyethylene and polypropene.

Thermosetting plastics are different from thermoplastics in that they do not melt at high temperatures, but remain hard (although they can decompose). They are also generally stronger than thermoplastics. Examples of thermosets are Bakelite, vulcanized rubber, polyester and epoxy. Also polyimides are thermosets and can sustain high temperatures.

The main task of the matrix is to keep the fibres together and in their place. The matrix limits the formation of cracks of the fibres and protects them from the environment.

5.1.1.3 Properties and applications

In the previous subsections we already mentioned some properties and possible applications for fibre reinforced materials.

The most important advantage of composites is that it is easy to tune their properties, like making them electric or dielectric, thermally insulating or not, or with a low or a high coefficient of thermal expansion. Another major advantage is the high stiffness- and/or strength-to-weight ratios.

Unfortunately, composites are costly, as well in raw materials as in production. The production process is still slow and labour-intensive and the composites allow only a low plastic deformation. Thermosetting plastics are not easy to recycle, which leads to environmental problems.

Despite the disadvantages mentioned, composite materials are part of our daily lives. In the transportation sector, composites are commonly used as part of the vehicle. For example, the shields of spacecrafts, the doors of airplanes, car spoilers or train seats are made of fibre reinforced materials. Sometimes, the whole vehicle is made out of a composite, like military planes, minesweepers or race boats. Another sector where composites are commonly used is that of (building) constructions: swimming pools, mobile homes or sewers are only a few examples where composites are used. Other examples of composite usage are computer cases, ladders, barriers, tennis rackets, surfing boards, ...

5.1.2 Tribology

Tribology is the science of interacting surfaces in relative motion and the practices related thereto. All aspects of surfaces and their relative motion is studied, such as friction, lubrication and wear.

Traditional parameters studied in tribology are the normal and the friction force. One of the most important parameters to collect is the *coefficient of friction*. If N is the normal force, then the friction force F is directly proportional to it:

$$F = \mu N \ , \tag{5.1}$$

where μ is the coefficient of friction. There exist two types of friction: *static friction*, when the two objects in contact are not moving relative to each other (thus, speed $v = 0$), and *dynamic friction* when $v > 0$. In general, the static friction F_s is higher than the dynamic friction F_d (for the same normal force and materials). There is also the *limiting friction*, which is the maximum value of static friction, obtained just before the objects move relative to each other. This coefficient is generally smaller than one, when we consider steel-plastic combinations.[1] Friction produces heat, so it is often better to have a low friction. The coefficient of friction is a perfect indication for this. This coefficient depends on many parameters other than the normal load, such as speed and temperature.[2]

[1] A tire on concrete can have a coefficient value of 1.7.

[2] Of course, the materials used are also important.

Another interesting parameter to measure is the *vertical displacement*. This gives an indication of the wear rate under certain conditions.

Nowadays it is not enough to know how fast materials wear, but we want to understand how they wear, what the major processes are, all in order to pick up the better material combination. There are lots of different processes that can take place. The most important mechanisms are *adhesion*, *abrasion* (wear of the contact surface by scratching), *corrosion*, *fretting*,[3] *pitting* (a form of extremely localized corrosion) and *delamination* of the composite layers.

5.2 Experimental set-up

Composite materials and polymers are useful as *friction bearing* materials. They are used in the motors of many household appliances, but also for telescopic arms or sluices. In comparison with traditional rolling bearings, the composite and polymer bearings have many advantages: low acoustic noise, simple assembly and disassembly, they are shockproof, high frequencies are possible and they have a long life expectation. As such, there is a great interest in experimentally evaluating their friction and wear properties and relating them to external parameters (motor speed, temperature, pressure, ...).

Compared to polymer bearings, the advantage of composite bearings is their load bearing capacity. Therefore, these materials are used in large machinery.

A microscopic study of the fibres in composites can help us understand what is happening physically, and allows us to choose the right materials to build bearing constructions. This is very important for composites when used as dry bearing materials, i.e., when no oil can be used, like in machinery in the food or pharmaceutical industry. The microscopic images have to be analysed, so image processing is needed for pre-processing and to do the analysis.

The subsequent experiments are performed on polymers, both thermosets and thermoplastics. The images and technical information about the experiments were provided by Pieter Samyn and Jan Quintelier of the Laboratory Soete (Department of Mechanical Construction and Production, Ghent University). The investigated materials are discussed in section 5.2.3, but first we give more information about the two types of experimental set-up used: the cylinder-on-plate small-scale tribotester and the flat-on-flat large-scale tribotester. Our experiments concern the investigation of the pattern spectra of the debris particles obtained from wear experiments. We investigate whether image processing, and the morphological pattern spectrum in particular, can contribute to tribology by giving additional information or by confirming other experimental findings. With image processing, it would be possible to (semi-)automate the investigations of the materials.

[3]Fretting is corrosion at the rough edges of the contact surface, induced under load and in the presence of repeated relative surface motion, as induced for example by vibration.

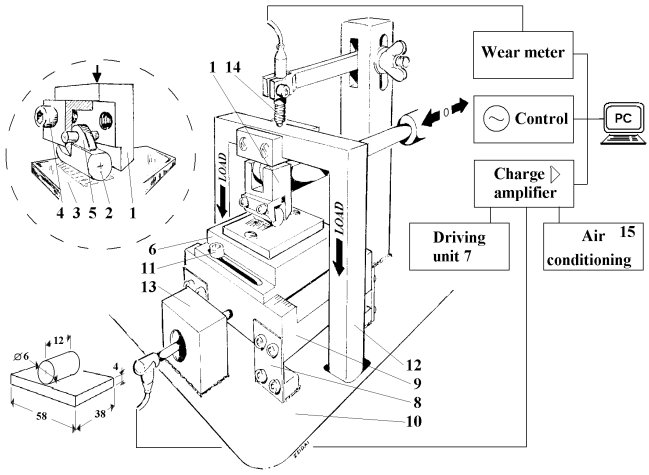


Figure 5.1: Small-scale cylinder-on-plate tribotester.

5.2.1 Small-scale testing

The small-scale testing device is illustrated in figure 5.1 [Zsidai et al., 2002]. The machine used is a PLINT TE 77 High Frequency reciprocating sliding tribotester. It is a *cylinder-on-plate* tester, which means the polymer is cylindrically shaped (measuring 15 mm in length and 5 mm in diameter) and allows dry reciprocating sliding[4] with a fixed steel plate. This counterface measures $58 \times 58 \times 4$ mm^3 and is made of HA-steel, a high alloy steel containing Cr, Mn, Mo and Ni.[5] It is much harder than normal steel.

The cylinder slides over the counterface, so it is prevented from rolling. The counterface can be heated. This way, the temperature can be regulated. If no heating is applied, we refer to this set-up as "*free T*". As can be seen in the illustration, a normal load can be applied on the polymer sample. The coefficient of friction μ, the vertical displacement Δh and the friction temperature T_f can be measured during the experiment. Offline, the difference in weight and diameter, before and after, is measured.

We summarize the test conditions used for the small-scale wear tests in table 5.1.

5.2.2 Large-scale testing

For high loads, a large-scale tribotester is used, which is illustrated in figure 5.2. This tester uses a *flat-on-flat* reciprocating motion set-up. The polymers are

[4]This means the polymer goes back and forth while contacting the plate. No lubricant is used during the wear tests.

[5]The elements Cr, Mn, Mo and Ni stand for Chromium, Manganese, Molybdenum and Nickel, respectively.

Table 5.1: Small-scale test conditions.

Test parameter	**Unit**	**Value(s)**
Normal load	N	50, 100, 150, 200
Single sliding stroke	mm	15
Total sliding distance	m	15 000
Sliding frequency	Hz	10, 20, 30, 40
Sliding velocity	m/s	0.3, 0.6, 0.9, 1.2
Counterface temp.	°C	free T, 60, 80, 100, 120, 140, 180, 220, 260
Ambient temperature	°C	23 ± 2 (standard atmosphere)
Relative humidity	%	60 ± 5 (standard atmosphere)

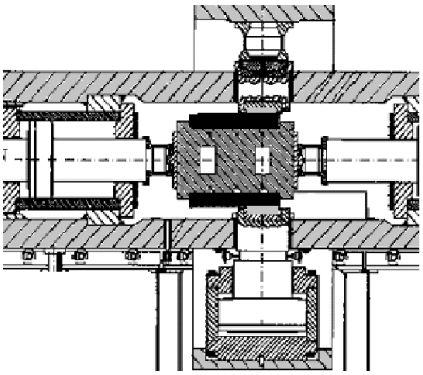


Figure 5.2: Large-scale flat-on-flat tribotester.

fixed at the top and the bottom in the figure. Each specimen measures $150 \times 150 \times 20$ mm^3. The sliding is done by the counterface plates, which measure $410 \times 200 \times 20$ mm^3 each. In the figure, the plates move from left to right (and back), and are shown as black rectangles. The counterface is continuously cooled by water. Initially, the counterface temperature is 15°C.

We summarize the test conditions used for the large-scale wear tests in table 5.2.

5.2.3 Polymers used in the study

Four different types of polymer have been investigated. Two are thermoplastics, two are thermosets. One of them is tested with the large-scale tribotester, in order to investigate the material under high pressure. The other three materials are tested with the small-scale tribotester at different temperatures.

Table 5.2: Large-scale test conditions.

Test parameter	Unit	Value(s)
Normal load	kN	190, 380, 560, 1260, 3380
Contact pressure	MPa	8, 16, 25, 55, 150
Single sliding stroke	mm	230
Total sliding distance	m	4.6 (short), 2500 to 3000 (long)
Sliding frequency	Hz	0.011
Sliding velocity	m/s	0.005
Counterface temp.	°C	no heating
Ambient temperature	°C	23 ± 2 (not controlled)
Relative humidity	%	40 to 60 ± 5 (not controlled)

5.2.3.1 POMH

POMH is the acronym for *polyoxymethylene homopolymer*, a hard and stiff thermoplastic with good abrasion resistance. It is used as sliding bearing in everyday appliances, such as gates. It can also be used for guide wheels in mechanical appliances.

This thermoplastic is tested under different high pressures. Therefore, the tests are performed with the flat-on-flat large-scale testing device.

5.2.3.2 SP-1

SP-1 is short for DuPont's Vespel SP-1 product. SP stands for *sintered polyimide*, which is a simple thermosetting plastic that is made in a sintering process. *Sintering* is the process of converting a powder into a solid polymer by a chemical reaction of the powder elements and by compressing the resulting substance under high pressure and temperature. Sintered polyimide is e.g. used in electrical engines.

This thermoset is specifically tested at different high temperatures. The experiments are performed with the cylinder-on-plate small-scale testing device.

5.2.3.3 SP-21

SP-21 is a variant of SP-1 and is also a thermoset. The powder contains 15 % graphite. This gives SP-21 a lower coefficient of friction μ than SP-1, as well as an increased wear resistance. SP-1 and SP-21 degrade both at 600°C.

5.2.3.4 TP

TP is the acronym for *thermoplastic polyimide*. The material is DuPont's Vespel TP-8054. The melting temperature of this thermoplastic, 388°C, is higher than average. It is used in electrical engines, but can be utilized in more demanding applications concerning temperature, because of its higher melting point. The fabrication procedure is called *injection moulding*: the powder is softened above 230°C and is afterwards pressed into a mould.

The experiments are performed with the cylinder-on-plate small-scale testing device.

5.2.4 Images of debris particles

We are interested in the debris particles obtained from the wear experiments. After the wear tests, the pieces that have come off the bulk material are collected and photographed. The debris particles are scattered onto a dark plate, in order to be photographed. They have no favourable orientation. The pictures taken are 24-bit RGB colour images with size 1030×1300 pixels.

For our experiments with the different granulometries, we must first process the images. First of all, we need to transform the RGB colour images into 8-bit greyscale images for most pattern spectrum techniques.[6] We also examine the usefulness of the processing of binary images. The binarization is done with the Otsu thresholding.[7] In some cases, we perform a histogram equalization.

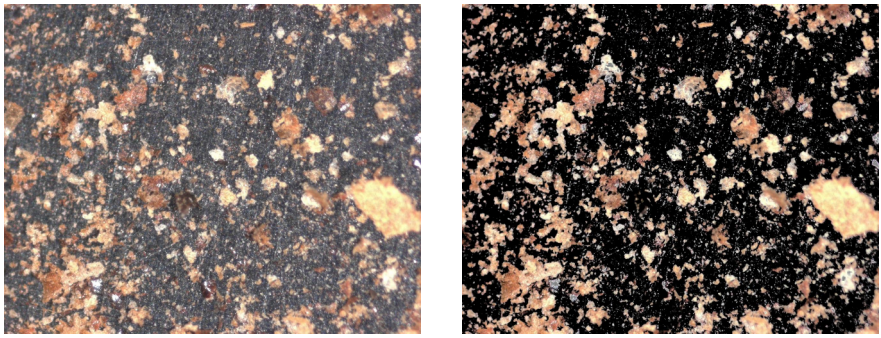
A pre-processing step we can perform, before any other transformation, is to blacken the background. The background plate on which the particles are scattered, is not homogeneously black. Therefore, we blacken this background in the following way:

1. We assign some pixel $\mathbf{p}$ the status of background;
2. We define some threshold t;
3. We calculate the maximum colour distance to the other pixels $\mathbf{p}'$, i.e., we calculate $d(\mathbf{p}, \mathbf{p}') = \max\{|R - R'|, |G - G'|, |B - B'|\}$;
4. We set the colour value of pixels $\mathbf{p}'$ with $d(\mathbf{p}, \mathbf{p}') < t$ to $RGB = (0, 0, 0)$.

In the first step, the user must point to a pixel with a colour that is characteristic for the background colour of the image. The threshold t depends on the image. Here, this value lies at about 50. An example of background blackening is shown in figure 5.3.

[6] Colour images can be used with the pattern spectrum using the majority sorting scheme MSS-PS and the colour pattern spectrum CPS (H-, S- or L-ordered).

[7] In chapter 6, section 6.4.3, we discuss the Otsu threshold.



(a) Before (b) After

Figure 5.3: Background blackening of a colour image of debris particles.

5.3 Results

This section deals with the analysis of the debris particles. We calculate the different pattern spectra for the different images and from these spectra we obtain some spectral parameters, as we discussed in chapter 4, section 4.1.4. We look for correlations with the experimental set-up, i.e., we investigate the correlation between the spectral parameters and the experimental parameters load and temperature. We also compare the behaviour of the spectral parameters with conclusions obtained from tribological experiments.

5.3.1 Choice of the structuring element

All the calculations of pattern spectra that allow the choice of a structuring element (PS, CPS, MSS-PS, EPS and APS) are done with a square structuring element with size 3×3 pixels. In the case of the area pattern spectrum (APS), we set λ to mimic a square structuring element. The opening trees (OT) use linear structuring elements and the Fourier pattern spectrum (FPS) is based on a different principle, so no structuring element has to be chosen here.

We choose a square structuring element because it is easy to implement and is isotropic on a square discrete grid. Also, it allows us to compare the standard pattern spectrum (PS) with the opening tree by minima of linear openings (OT_{min}).

5.3.2 Calculation times

In the previous chapter, section 4.3, we have investigated the computational cost of the different pattern spectra. It turned out that the implementation of the morphological pattern spectrum is computationally very inefficient. The other spectra, on the other hand, are calculated much faster.

This is of course also the case with the images we investigate in this chapter. The images measure 1030×1300 pixels, and contain objects of different size. The maximal size N_{max} can be up to 1300, which means a calculation time of up to several hours for the normal PS. On average,[8] the maximal size for a greyscale image is about 900. If we blacken the background of the images, N_{max} reduces to about 23.

The calculation of the PS for a greyscale image can take several days, if the spectrum is calculated in the background and other processes are running on the computer. If we only look at the resources needed by the algorithm, the calculation takes several hours. The second least efficient algorithm, that for the calculation of the erosion pattern spectrum (EPS) (see section 4.3), is finished in about 3 minutes, which is already a lot faster. The other algorithms, such as that of the opening tree and the area pattern spectrum, only need a few to several seconds.

5.3.3 Correlations

In this part, we discuss the correlations between the parameters from the different spectra and the experimental set-up. We therefore have to calculate the different spectral parameters, introduced in section 4.1.4 of chapter 4. These parameters are:

- average size, $S(A;B)$;
- average roughness or entropy, $E(A;B)$;
- normalized average roughness, $E_N(A;B)$;
- B-shapiness, $BS(A;B)$;
- maximal size, N_{max}.

These parameters are calculated for the following pattern spectra:

- morphological pattern spectrum (PS);
- *area* pattern spectrum (APS);
- *erosion* pattern spectrum (EPS);
- *colour* pattern spectrum using the lexicographical ordering (CPS);
- pattern spectrum using the *majority ordering* (MSS-PS);
- *granulometry* by maxima of linear openings (OT_{max}), which is an *oriented* pattern spectrum (OPS);

[8] The average is taken from the values of the parameter N_{max} of the pattern spectra of 15 images with debris particles from POMH.

- *pseudo-granulometry* by minima of linear openings (OT_{min}).

For the *Fourier* pattern spectrum (FPS) we calculate the parameters mentioned in section 4.2.5.4, i.e.:

- mean, S_M;
- standard deviation, S_D;
- skewness, S_S;
- kurtosis, S_K;
- energy, S_N;
- entropy, S_E.

First, we introduce the *correlation coefficient*.

5.3.3.1 Correlation coefficient

The *correlation coefficient* $\rho(x, y)$ gives us the quality of a linear fit of variable x against variable y. Generally, Pearson's correlation coefficient is used. It is defined as:

$$\rho(x,y) = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_i (x_i - \bar{x})^2 \sum_i (y_i - \bar{y})^2}} \ . \tag{5.2}$$

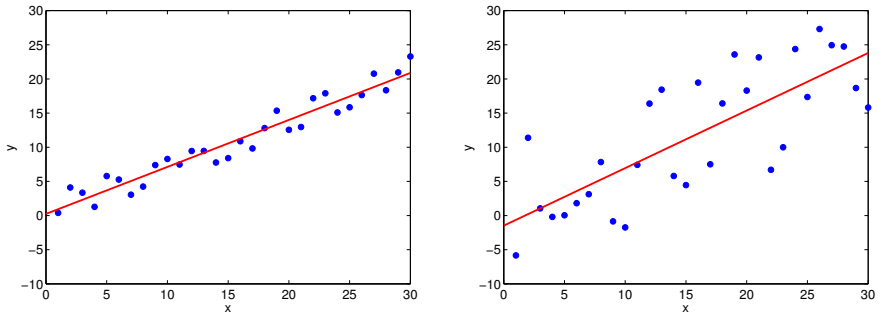
(x_i, y_i) are the data points that are fitted. $\bar{x}$ and $\bar{y}$ are the average values. Equation (5.2) is equivalent to:

$$\rho(x,y) = \frac{C(x,y)}{\sqrt{C(x,x)C(y,y)}} \ . \tag{5.3}$$

$C(x, y)$ is the *covariance* of x and y. The (first-order) *covariance matrix* element $C(x, y)$ is defined as:

$$C(x,y) = \frac{1}{N} \sum_{i=0}^{N-1} (x_i - \bar{x})(y_i - \bar{y}) \ , \tag{5.4}$$

with N the number of data points. The equivalence between equations (5.2) and (5.3) now becomes obvious. Notice that $C(x, x)$ and $C(y, y)$ are the *variance* of x and y, respectively. The square root of the variance (σ^2) is the *standard deviation* σ. When the variables are uncorrelated, then the covariance value will be 0.



(a) $\rho(x, y) = 0.97$ (b) $\rho(x, y) = 0.77$

Figure 5.4: Two example datasets with a different correlation coefficient of x and y.

The value of the correlation coefficient lies in the interval $[-1, +1]$. A value near 0 means that there is no linear correlation. A good linear fit is not possible then. Either the statistical deviation of the data is too high, the variables are correlated in a non-linear way, or both. A value of $\rho(x, y) = \pm 1$ indicates perfect correlation. When $\rho(x, y) = 1$, y is linearly correlated with x, while $\rho(x, y) = -1$ means y is anti-linearly correlated with x, i.e., y decreases when x increases. A visual example of different correlation coefficients is shown in figure 5.4.

There is no fixed rule that states which values for ρ indicate a good linear fit and which not. For $\rho \geq 0.7$ we can say that a linear trend becomes visible and reasonably accurate. Values above 0.9 are very good in our experiments. $\rho < 0.5$ does not give a good indication of a linear relationship between the variables, the fit is not trustworthy. As said before, this bad result can be due to the statistical deviation and/or the non-linearity of the relationship.

5.3.3.2 Correlation between the spectral parameters of the pattern spectrum and those of the alternative spectra

The morphological pattern spectrum is a tool that can be used to tell us something about the size and shape of the objects that are present in an image. In the previous chapter we have seen that this spectrum has a (very) high computational cost. For this reason, we looked at alternative pattern spectra and algorithms. Some of them calculate something completely different, such as the erosion pattern spectrum and the Fourier pattern spectrum, while we assume that some other techniques produce (to a certain extent) similar spectra (the area pattern spectrum or the pseudo-granulometry).

We now investigate how the parameters of these different spectra correlate with those of the default pattern spectrum (PS). To do this, we take the images used in the next subsection about POMH, i.e., 10 images for each of 5 different

Table 5.3: Correlation coefficients between the spectral parameters of the alternative spectra and the default pattern spectrum (PS). The inputs are blackened greyscale images.

Spectrum	ρ_S	ρ_E	ρ_{E_N}	ρ_{BS}	$\rho_{N_{max}}$
EPS	0.99	0.99	0.96	0.61	1.00
APS	0.81	0.73	0.66	0.69	0.44
OT_{min}	0.99	0.99	0.81	−0.35	0.59
OT_{max}	0.99	0.99	0.79	0.96	0.57
OT_h	0.99	0.99	0.80	0.95	0.57
OT_v	0.99	0.99	0.80	0.66	0.59

pressures. We calculate the different spectral parameters, as mentioned in the beginning of this section, and compute the correlation coefficients.

Table 5.3 lists the correlation coefficients between the spectral parameters of the different spectra and those of the default pattern spectrum.[9] OT_h and OT_v are pattern spectra using a horizontal or vertical structuring element.[10] The area pattern spectrum has the lowest scores (except for B-shapiness). The perfect correlation between the maximal sizes of PS and EPS is logical, because for both spectra the same number of openings/erosions is always needed to remove all the objects in the image. Overall, the correlation coefficients for size and roughness are very high, suggesting a good linear fit of these parameters for the PS with those for the alternative spectra. There is one negative value in the table, suggesting a negative linear fit, but the value is low ($\rho < 0.7$) so this fit is not accurate.

If we process every type of input image (i.e., greyscale and binary, blackened and non-blackened, histogram equalized and non-equalized images) and average all the obtained correlation coefficients, then we get the values listed in table 5.4.

We can partly make the same remarks as with the previous table: APS is least correlated with the PS, and EPS is most correlated. While the pseudo-granulometry by minima of linear openings should have similar features as the morphological pattern spectrum with a square structuring element [Vincent, 2000], the correlation coefficient is rather low, especially when compared to the correlation coefficients of other alternative pattern spectra. Even the pattern spectra with a one-dimensional structuring element correlate better with the parameters of the PS with a square.

The morphological pattern spectrum of the greyscale version of figure 5.6(a) is shown in the upper part of figure 5.5. The lower part shows the opening tree by minima of linear openings. Notice the logarithmic scale of the ordinate.

[9] The FPS is not included because we calculate other parameters from this spectrum.
[10] Using the algorithm of the opening tree.

Table 5.4: Correlation coefficients between the spectral parameters of the alternative spectra and the default pattern spectrum (PS), averaged.

Spectrum	ρ
EPS	0.90
APS	0.50
OT_{min}	0.73
OT_{max}	0.82
OT_{h}	0.83
OT_{v}	0.84

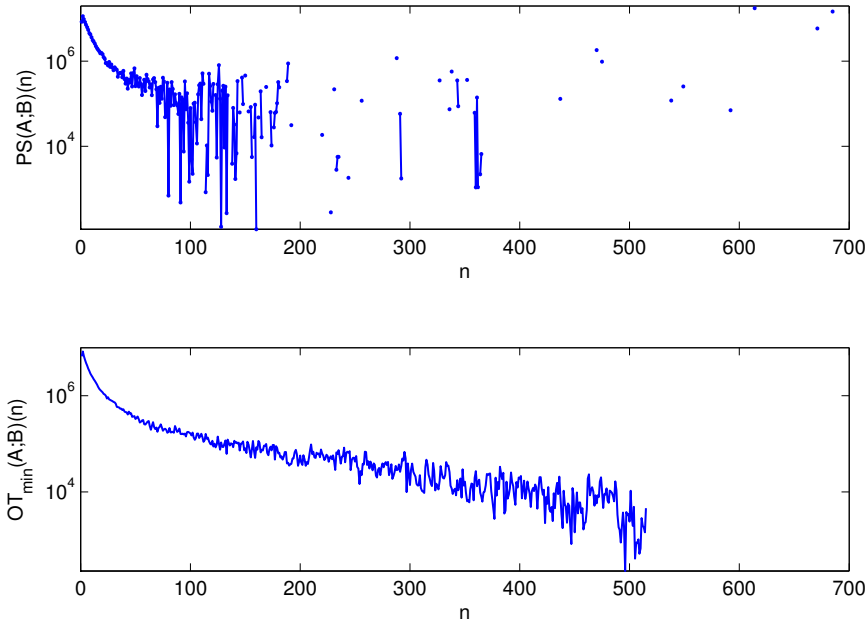


Figure 5.5: The pattern spectrum PS (upper) and pseudo-granulometry OT_{min} (lower) of figure 5.6(a).

The spectra are not very similar: at lower n, the spectra follow the same trend, but at higher n they are very different. This is not disadvantageous for our purposes, i.e., correlating spectral parameters with experimental parameters, but this means the pseudo-granulometry is not a good approximation of the pattern spectrum with a square structuring element. Both spectra can produce quite different parameters.

5.3.3.3 POMH

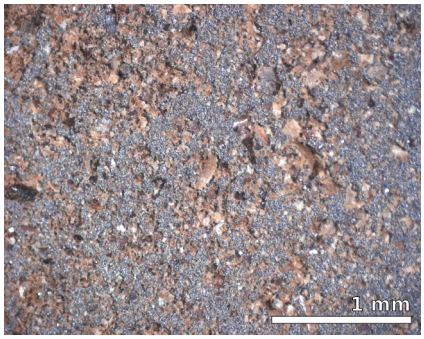
Tribological experiments on this material show that the wear at low pressures causes debris particles to crumble off the bulk material [Samyn and De Baets, 2005]. At higher pressures (16 MPa and above), crystallization occurs and also smaller debris particles in the shape of flakes are obtained. At 150 MPa, the contact surface melts, resulting in larger but filament-shaped particles. The applied contact pressures were 8 MPa, 16 MPa, 25 MPa, 55 MPa and 150 MPa. The sliding velocity is 5 mm/s (sliding frequency 11 mHz). The differences between the debris particles when different pressures have been applied, can be seen in figure 5.6. We now discuss the information we can retrieve from the pattern spectra from such images.

In the first experiment, we have 10 images containing debris particles, for each load. Their spectra are combined by adding them and then the parameters are calculated. Figure 5.7 shows plots of the behaviour of the spectral parameters with the applied pressure p. The images used are greyscale versions of the original images, with the background blackened (as explained in section 5.2.4).

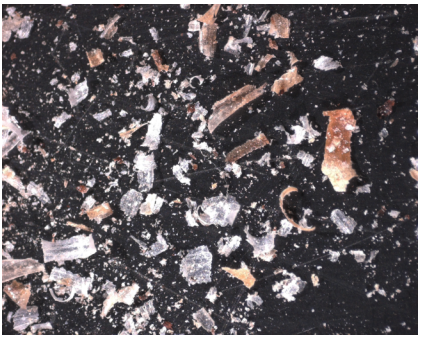
From the plots we can conclude that size and roughness both increase with the contact pressure, and in a similar manner. B-shapiness increases almost linearly with the load. The increase in entropy (roughness) with the load can be explained by the increasing diversity in size of the debris particles. At higher pressures, also smaller flakes are visible because of the crystallization process. The increase of the B-shapiness tells us that the shapes of the detected particles become more the shape of the structuring element used, which is a square. The absolute value of the B-shapiness is still very small, so the relative increase does not reveal much useful information. The variation of the average size states that the particles increase in size with the normal force. At 8 MPa, the actual mean object size is about 20×20 pixels and at the highest pressure the mean object size is about 120×120 pixels (i.e., about 0.27×0.27 mm^2). The flakes are in general smaller in size than the other fragments, but they are often clustered, which makes them appear bigger. Also, bigger particles come off the bulk material when high pressures are applied.

In table 5.5 we present the correlation coefficients between the spectral parameters of the default pattern spectrum and the applied pressure in the experiment. We investigate the effect of blackening of the images, binarization and histogram equalization. Overall, the histogram equalization has a negative impact on the correlation coefficients. For example, the correlation coefficient for average size, obtained from the standard pattern spectrum (PS), drops from about 0.87 to about 0.72 for the blackened greyscale images. There is no clear difference between the correlation coefficients for greyscale images and those of binary images.

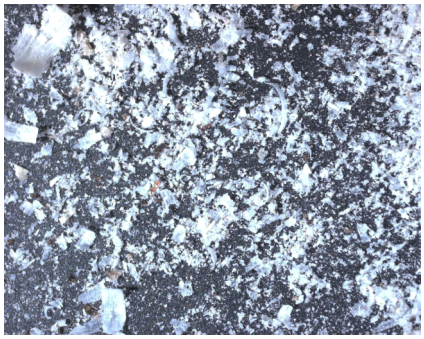
Table 5.6 lists the correlation coefficients for some pattern spectrum techniques. The processed images are greyscale and blackened. We can see that the correlation coefficients between the maximal sizes N_{max} and the applied pressure



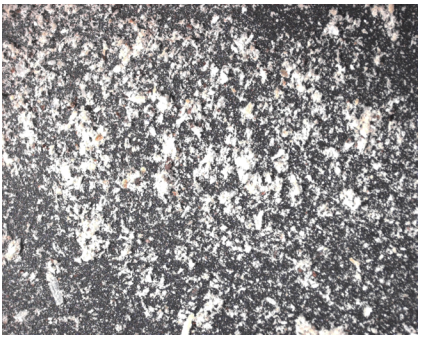
(a) $p = 8$ MPa



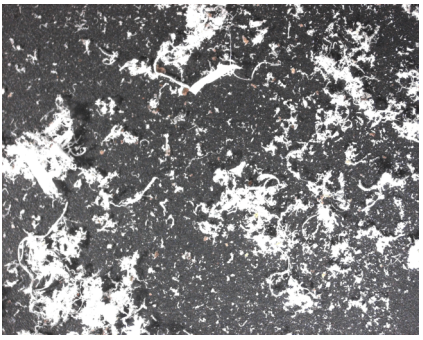
(b) $p = 16$ MPa



(c) $p = 25$ MPa

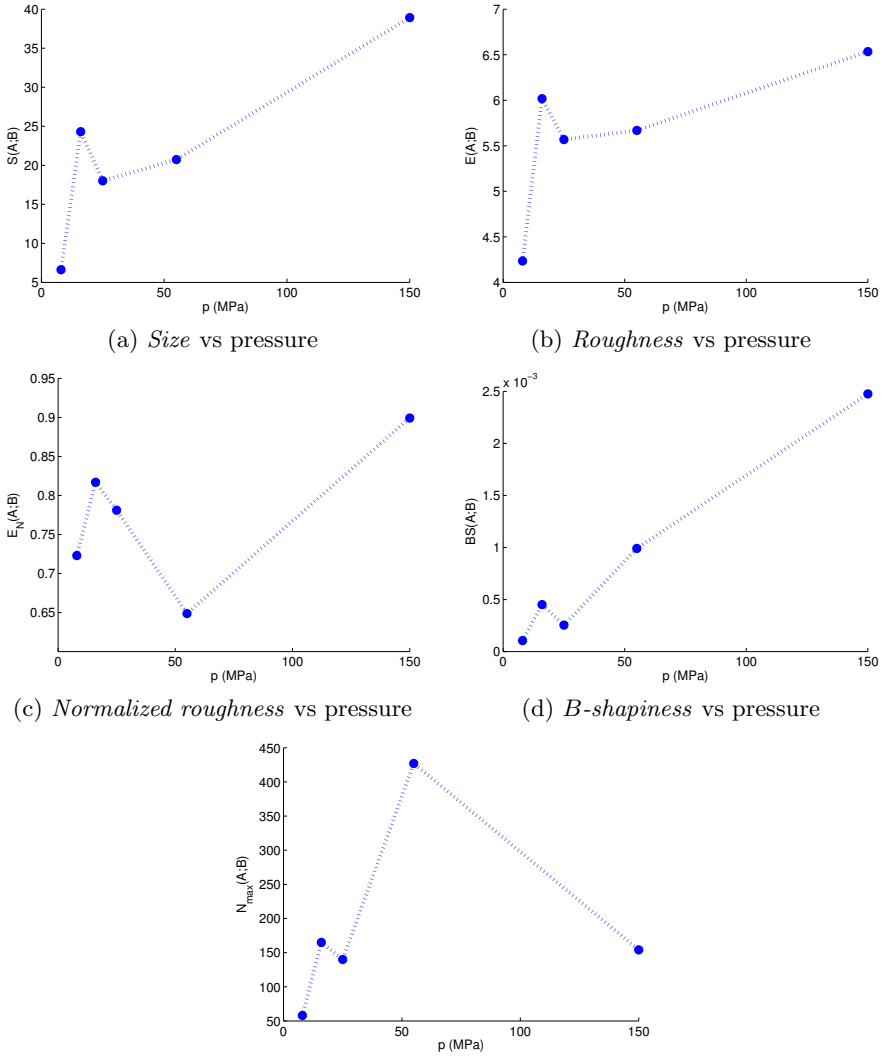


(d) $p = 55$ MPa



(e) $p = 150$ MPa

Figure 5.6: Debris particles from POMH, for different loads.



(a) *Size* vs pressure

(b) *Roughness* vs pressure

(c) *Normalized roughness* vs pressure

(d) *B-shapiness* vs pressure

(e) *Maximal size* vs pressure

Figure 5.7: The spectral parameters from the pattern spectrum (PS), for POMH (greyscale, blackened), plotted against the contact pressure.

Table 5.5: Correlation coefficients between spectral parameters of the pattern spectrum (PS) and the applied pressure, for differently treated images, for POMH. bl: blackened; gr: greyscale; bw: binary; eq: histogram equalized. Blackening is performed before greyscale conversion, histogram equalization is performed on the greyscale image.

Image	ρ_S	ρ_E	ρ_{E_N}	ρ_{BS}	$\rho_{N_{max}}$
bl–gr	0.87	0.69	0.56	0.99	0.16
gr–eq	0.75	0.58	0.33	0.86	0.65
bl–gr–eq	0.72	0.56	0.58	0.83	0.17
bw	0.86	0.65	−0.25	0.98	0.66
bl–bw	0.86	0.61	−0.31	0.99	0.54
eq–bw	0.74	0.38	−0.85	0.97	0.76
bl–eq–bw	0.67	0.49	0.37	0.96	−0.0092

Table 5.6: Correlation coefficients between spectral parameters and the applied pressure, for different spectra of blackened greyscale images of POMH.

Spectrum	ρ_S	ρ_E	ρ_{E_N}	ρ_{BS}	$\rho_{N_{max}}$
PS	0.87	0.69	0.56	0.99	0.16
EPS	0.87	0.71	0.68	0.60	0.16
APS	0.55	0.042	0.71	0.76	−0.70
OT_{min}	0.87	0.73	0.88	−0.36	0.029
OT_{max}	0.90	0.76	0.87	0.98	0.21

are very low. In general, we cannot state that N_{max} is linearly correlated with the applied pressure. An exception is the area pattern spectrum: its $\rho_{N_{max}}$ is much higher than that of the other spectra, but for the other parameters the correlation coefficient is amongst the smallest values. The other coefficients are quite high, often $\rho \geq 0.7$. Overall, the oriented pattern spectrum (OPS), calculated using the granulometry by maxima of linear openings (OT_{max}), scores best in this example. For the alternative spectra, the plots of the contact pressure against the spectral parameters is mostly similar to the plots shown in figure 5.7. Only in the cases where the correlation coefficient deviates significantly from that for the default pattern spectrum, the trend is different.

The parameters from the Fourier pattern spectrum (FPS) are not so well correlated with the experimental set-up, i.e., with the applied pressure. This can be seen in table 5.7: almost every correlation coefficient is less than 0.5. The best correlation is obtained when histogram equalized greyscale images are used as input. Here we can say that kurtosis has the highest (anti-)linear correlation with the applied load (−0.89 for histogram equalized greyscale images). We can conclude here that the Fourier pattern spectrum is not well suited for the

Table 5.7: Correlation coefficients between spectral parameters of the Fourier pattern spectrum (FPS) and the applied pressure, for differently treated images, for POMH. bl: blackened; gr: greyscale; bw: binary; eq: histogram equalized. Blackening is performed before greyscale conversion, histogram equalization is performed on the greyscale image.

Image	ρ_{S_M}	ρ_{S_D}	ρ_{S_S}	ρ_{S_K}	ρ_{S_N}	ρ_{S_E}
gr	0.40	0.90	−0.46	−0.74	0.29	0.13
bl–gr	0.14	0.61	−0.21	−0.29	0.12	0.049
gr–eq	0.69	0.87	−0.76	−0.89	0.087	0.46
bl–gr–eq	−0.38	0.33	0.37	0.33	0.26	−0.38
bw	−0.13	0.42	−0.079	−0.72	0.36	−0.22
bl–bw	0.047	0.30	−0.26	−0.68	0.17	−0.042
eq–bw	0.36	0.18	−0.65	−0.72	−0.099	0.23
bl–eq–bw	−0.31	0.44	0.21	−0.56	0.36	−0.34

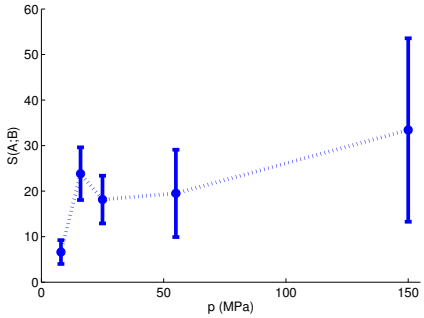
study of the debris particles.

Another approach is to calculate the parameters for each of the 10 spectra (for the same load) separately, and to average these results.[11] We have to remark, though, that the samples most probably do not follow a normal distribution. The particles were scattered on a surface and photographed. It is likely that in some areas (and thus in some pictures) more particles are present than in other areas. It is also possible that large particles dropped first and smaller objects later. Or vice versa. There is thus no certainty that the particles are evenly distributed over the different images.

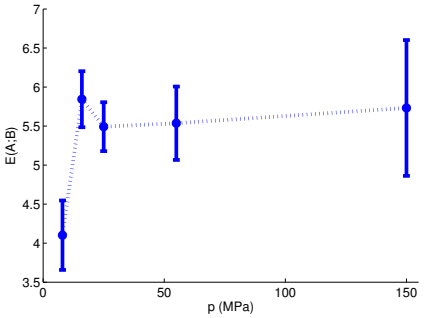
We will, however, show the graphs such as the ones in figure 5.7 for this approach, in order to better understand the usefulness of our method. Figure 5.8 shows the plots containing error bars. We notice that the standard deviations can be very big, going from about 10 % to more than 110 %. On average, the standard deviation is 45 % of the parameter value.

Figure 5.9 shows the same graphs, but the error bars are replaced by the actual data points. The green dotted line is the average, the red dashed line is the median. The median of the data points results in a more robust value, less sensitive to outliers. We notice several outlier values. In almost every graph, at least one outlier can be seen at $p = 55$ MPa. The outlier in figure 5.9(a) at 55 MPa is caused by one of the images where considerably larger objects are present. However, its influence on the average is minimal, since the difference of the average value with the median is small. A similar conclusion can be drawn from figure (b). For the normalized roughness, on the other hand, there is more than one outlier. The lower values for $E_N(A;B)$ at 55 MPa are from

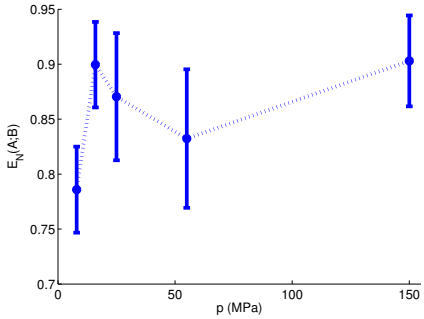
[11] Before, we combined the 10 spectra by adding them, and thus one set of parameters is obtained.



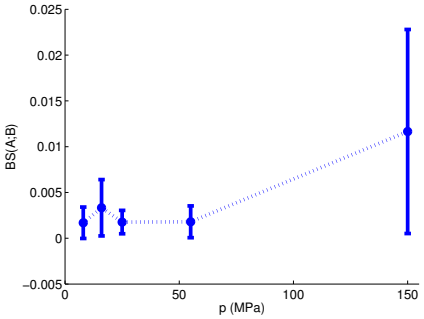
(a) *Size* vs pressure



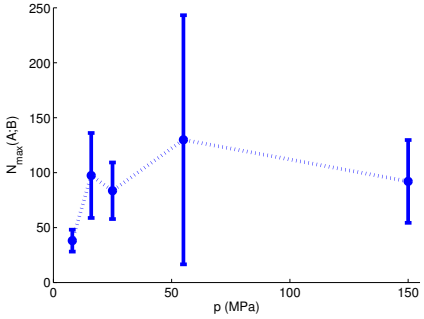
(b) *Roughness* vs pressure



(c) *Normalized roughness* vs pressure

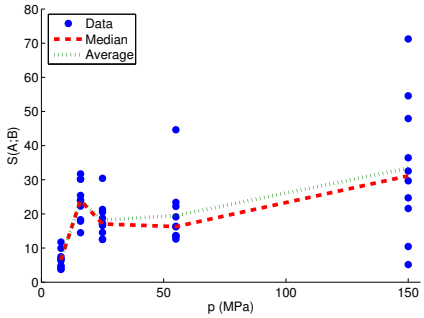


(d) *B-shapiness* vs pressure

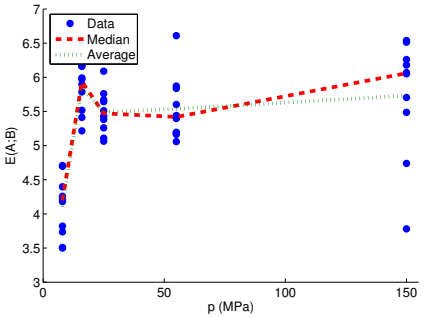


(e) *Maximal size* vs pressure

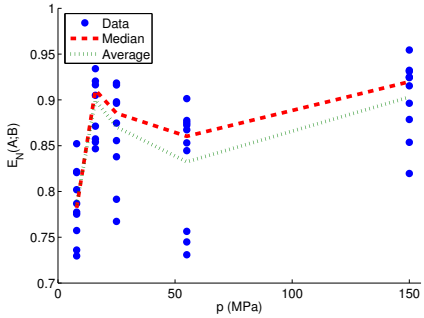
Figure 5.8: The spectral parameters from the pattern spectrum (PS), for POMH (greyscale, blackened), plotted against the contact pressure. The spectra are not added; instead, the parameters are averaged.



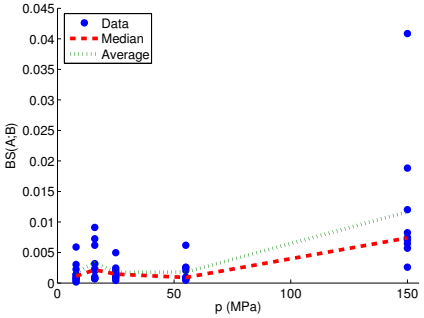
(a) *Size* vs pressure



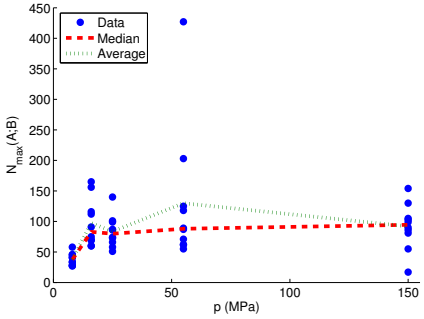
(b) *Roughness* vs pressure



(c) *Normalized roughness* vs pressure



(d) *B-shapiness* vs pressure



(e) *Maximal size* vs pressure

Figure 5.9: The spectral parameters from the pattern spectrum (PS), for POMH (greyscale, blackened), plotted against the contact pressure. The spectra are not added; instead, the parameters are averaged.

Table 5.8: Correlation coefficients between spectral parameters of the pattern spectrum (PS) and the applied pressure, for differently treated images, for POMH. The spectra are not added; instead, the correlation coefficients are median averaged. bl: blackened; gr: greyscale; bw: binary; eq: histogram equalized. Blackening is performed before greyscale conversion, histogram equalization is performed on the greyscale image.

Image	ρ_S	ρ_E	ρ_{E_N}	ρ_{BS}	$\rho_{N_{max}}$
bl–gr	0.61	0.48	0.44	0.89	0.35
gr–eq	0.53	0.43	0.35	0.65	0.62
bl–gr–eq	0.47	0.45	0.41	0.64	0.26
bw	0.43	0.27	−0.21	0.76	0.18
bl–bw	0.52	0.23	−0.48	0.75	0.41
eq–bw	0.48	0.19	−0.61	0.85	0.54
bl–eq–bw	0.44	0.19	−0.48	0.74	0.39

images where much more smaller debris particles are present. They are less numerous in the other images.

We notice large error bars at $p = 150$ MPa (figure 5.8). As we can see in figure 5.9(a), the average size $S(A;B)$ varies a lot at 150 MPa. This variety in size can be confirmed by looking at the photographs. Sometimes, only one or a few large objects are present, in other images more smaller or thinner particles are visible. We cannot define a certain size or type of particle as outlier, since they all are (approximately) equally present. The large spread of values is due to the uneven distribution of the particles (in size) over the different images.

In figure 5.9, the trend of the median curve is in accordance with other tribological experimental results. The increase in average size (figure 5.9(a)) from 8 MPa to 16 MPa is due to the increase in size of the debris particles. We notice a small decline in average size between 16 MPa and 55 MPa. Besides degradation, now also crystallization occurs, which leads to smaller debris. The average size is higher again at 150 MPa (melting pressure): long but clustered filaments are present.

When we compare the graphs in figure 5.7 with those in figure 5.8 or 5.9, we notice similarities, but also differences. For instance, the parameter values for average size and roughness are almost the same, except at 150 MPa where the values in figure 5.7 are bigger. For the other graphs on the other hand, the shape is very similar but not the values. This is logical, since the maximal size (N_{max}) in figure 5.7(e) is the maximum value N_{max} of the 10 different images. In figure 5.9(e), N_{max} is a mean or a median of several values. Because the normalized average roughness and the B-shapiness depend on N_{max},[12] their

[12]The B-shapiness depends on the value $PS(A;B)(N_{max})$ (see equation (4.22) in chapter 4).

Table 5.9: Correlation coefficients between spectral parameters and the applied pressure, for different spectra of blackened greyscale images of POMH. The spectra are not added; instead, the correlation coefficients are median averaged.

Spectrum	ρ_S	ρ_E	ρ_{E_N}	ρ_{BS}	$\rho_{N_{max}}$
PS	0.61	0.48	0.44	0.89	0.35
EPS	0.62	0.58	0.56	0.21	0.35
APS	0.43	−0.82	−0.57	0.39	−0.18
OT_{min}	0.76	0.62	0.51	0.90	0.21
OT_{max}	0.75	0.57	0.49	0.93	0.23

Table 5.10: Correlation coefficients between spectral parameters of the Fourier pattern spectrum (FPS) and the applied pressure, for differently treated images, for POMH. The spectra are not added; instead, the correlation coefficients are median averaged. bl: blackened; gr: greyscale; bw: binary; eq: histogram equalized. Blackening is performed before greyscale conversion, histogram equalization is performed on the greyscale image.

Image	ρ_{S_M}	ρ_{S_D}	ρ_{S_S}	ρ_{S_K}	ρ_{S_N}	ρ_{S_E}
gr	0.43	0.84	−0.49	−0.72	0.15	0.13
bl–gr	0.17	0.78	−0.25	−0.35	0.15	0.13
gr–eq	0.62	0.84	−0.64	−0.83	0.05	0.35
bl–gr–eq	−0.38	0.53	0.36	0.25	0.44	−0.29
bw	−0.052	0.47	−0.095	−0.55	0.44	−0.15
bl–bw	0.022	0.34	−0.21	−0.73	0.24	0.035
eq–bw	0.24	0.096	−0.52	−0.61	−0.12	0.13
bl–eq–bw	−0.36	0.48	0.28	−0.57	0.62	−0.35

respective values change too.

We list the equivalent of table 5.5 in table 5.8. Remember that now the spectra are not added, but 10 different sets of spectral parameters and correlation coefficients are calculated. The results are averaged. We use the median to suppress the influence of outliers. If we use the arithmetic mean, the correlation coefficient would be much lower.[13] We notice that almost all the correlation coefficients in table 5.8 are smaller than their respective values in table 5.5. Similar conclusions can be drawn when we compare table 5.6 with table 5.9 and table 5.7 with table 5.10.

In another experiment, 3 images were taken for each of the 5 different loads. Again, their resulting spectra are combined. We will not discuss the case where

[13]For example, the values in the first row of table 5.8 are respectively 0.46, 0.35, 0.36, 0.68 and 0.24 if we do not use the median.

Table 5.11: Correlation coefficients between spectral parameters and the applied pressure, for different spectra of blackened greyscale images of POMH (second experiment). The last row displays the correlation coefficients for the parameters of the Fourier pattern spectrum.

Spectrum	ρ_S	ρ_E	ρ_{E_N}	ρ_{BS}	$\rho_{N_{max}}$
PS	−0.46	−0.46	−0.36	−0.078	−0.42
EPS	−0.46	−0.45	−0.37	0.022	−0.42
APS	−0.31	−0.36	0.54	0.17	−0.49
OT_{min}	−0.45	−0.46	−0.26	−0.086	−0.47
OT_{max}	−0.38	−0.36	−0.031	−0.11	−0.47
ρ_{S_M}	ρ_{S_D}	ρ_{S_S}	ρ_{S_K}	ρ_{S_N}	ρ_{S_E}
−0.066	−0.54	0.22	0.29	−0.31	0.058

we treat each spectrum independently, because having only 3 images is not enough to obtain statistically relevant error bars. Also here we would obtain large error bars and lower correlation coefficients.[14]

The pattern spectra and parameters are quite different from the ones we showed in figure 5.7. Again, size and roughness have a similar trend mutually, but the highest parameter value for these parameters is obtained when $p = 16$ MPa. The photographs from this set confirm this: large objects, mixed with smaller particles, are visible for $p = 16$ MPa, and mostly only fine flakes can be seen at the higher pressures. These results are not the same as those of the previous experiment, but still explicable: there is no peak at 150 MPa, because the filaments are not clustered. They are very thin, which means that they are sieved out by the pattern spectrum operation when the structuring element nB is still small. If the peak at 16 MPa is due to some outlier particles present in the image, then we can exclude them and recalculate the spectra and their parameters. Unfortunately, too many possible outliers are present in the images and there are not enough debris particles to conclude that they are indeed outliers.

The correlations for the parameters of different pattern spectra, like those shown in table 5.6, are shown in table 5.11 for this second experiment. The correlation coefficients are much lower than in the first experiment, and also negative. The major reason for this difference with the coefficients from the first experiment is the lack of sufficient samples, i.e., if more debris particles would be present and processed, then more accurate correlations would be possible. Again, the erosion pattern spectrum (EPS) follows the regular pattern spectrum (PS) for the most part. The parameters from the FPS do not show a clear trend in function of the applied pressure (see the last row in table 5.11).

Concerning the colour images, i.e., the images containing debris particles

[14]The previous experiment is the only one where 10 images are available for each setting.

Table 5.12: Correlation coefficients between spectral parameters and the applied pressure, for different spectra of blackened colour images of POMH (second experiment).

Spectrum	ρ_S	ρ_E	ρ_{E_N}	ρ_{BS}	$\rho_{N_{max}}$
MSS-PS	−0.44	−0.51	−0.46	0.71	−0.42
CPS (L)	−0.47	−0.48	−0.37	0.058	−0.42
CPS (S)	−0.68	−0.74	−0.54	0.89	−0.69
CPS (H)	−0.29	−0.38	0.32	−0.38	0.69

without any greyscale conversion or binarization, we refer to table 5.12. We notice that the correlation between the applied pressure and the parameters from the pattern spectra of the blackened colour images, using the lexicographical ordering, is highest when the S-ordering is used (CPS (S)).[15] It is also interesting to note that the pattern spectrum of the majority sorted image (MSS-PS) produces better correlations than the other two lexicographical orderings (H and L). The plots of the spectral parameters from the S-ordered CPS against the contact pressure are shown in figure 5.10. We notice a decreasing behaviour in several graphs, which is in accordance with the coefficients from table 5.12. When we examine the accompanying photographs, we see that the debris particles are indeed large at the lower pressures, and that the flakes (that appear at higher pressures) are smaller. In contrast to the first experiment, here the flakes are not clustered.

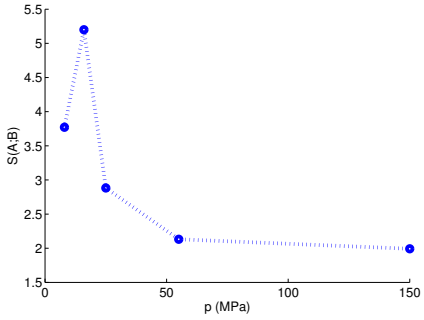
5.3.3.4 SP-1

Typical examples of debris particles of sintered polyimide are shown in figure 5.11. The analysis of the tribological experiments on this material show a transition at about 180°C [Samyn et al., 2003]. At higher temperatures, the debris particles become bigger. The reason for this is a chemical reaction called *imidization*, which causes polymer chains to be formed. Figure 5.11(a) shows debris particles of SP-1 without imidization. Figure 5.11(b) shows the debris particles when $T > 180$°C. There is also a transition between normal loads of 100 N and 150 N: larger particles are obtained at higher pressure, caused by mechanical fracture. The applied loads during the experiments were 50 N, 100 N or 200 N. Images of particles exist for counterface temperatures of "free T", 60°C, 80°C, 100°C, 140°C, 180°C, 220°C and 260°C. The sliding frequency is $\nu = 10$ Hz (or speed $v = 0.3$ m/s).

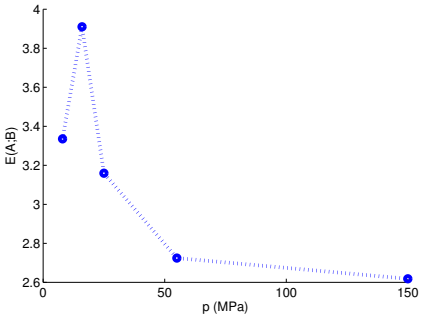
We show the behaviour of the spectral parameters related to the temperature in figure 5.12.[16] Data are shown for loads of 50 N, 100 N and 200 N. From most graphs we can conclude that a load of 50 N leads to different results

[15] H: hue; S: saturation; L: luminance.

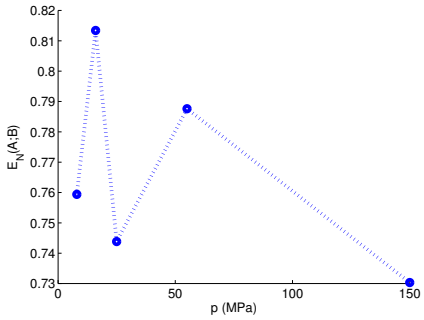
[16] Notice that we do not have images for every (N, T)-combination.



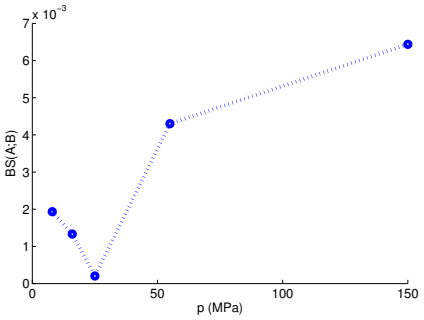
(a) *Size* vs pressure



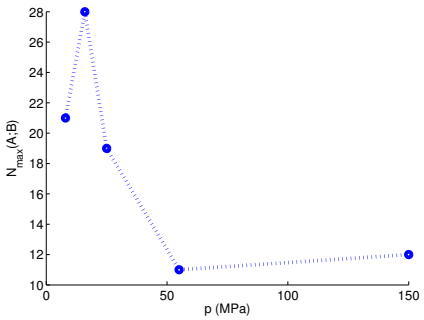
(b) *Roughness* vs pressure



(c) *Normalized roughness* vs pressure

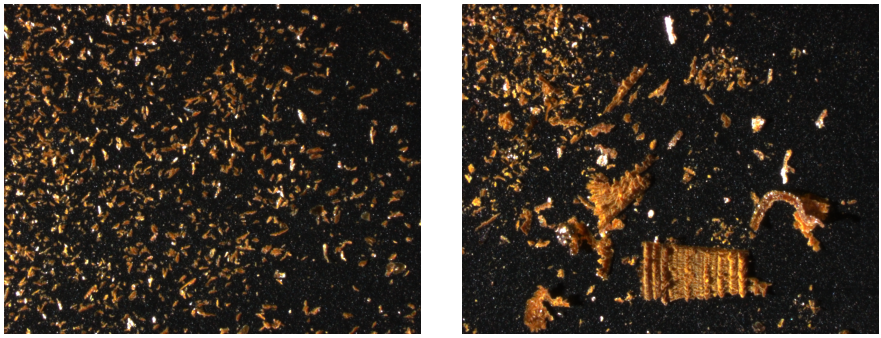


(d) *B-shapiness* vs pressure



(e) *Maximal size* vs pressure

Figure 5.10: The spectral parameters from the S-ordered CPS, for POMH (colour, blackened), plotted against the contact pressure (second experiment).



(a) $T = 100°C$ (b) $T = 220°C$

Figure 5.11: Debris particles after wear of SP-1. Both small and large particles are visible. $N = 100$ N and $\nu = 10$ Hz.

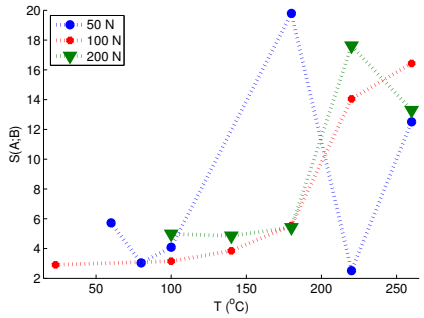
compared to the graphs for the other loads. This could suggest that there is a transition between 50 N and 100 N, but material science experiments revealed a transition above 100 N. When we examine the plots at 100 N and 200 N, we notice a significant increase of average size and average roughness above 180°C. The actual mean object size jumps from about 16×16 pixels to 48×48 pixels (i.e., about 0.11×0.11 mm^2), an increase by a factor of 3. The average roughness also increases by a large factor, about 1.4. These tendencies might be the indication for the imidization process.

Table 5.13 lists some correlation coefficients which indicate a quite good and reliable linear fit (although not always for normalized roughness and B-shapiness), if we consider $\rho \geq 0.7$ a good correlation value. We must point out that parameters from the pattern spectrum of the majority sorted images (MSS-PS) correlate better with the temperature than the pattern spectrum on the normal images (PS). The pseudo-granulometry also scores better than the PS.

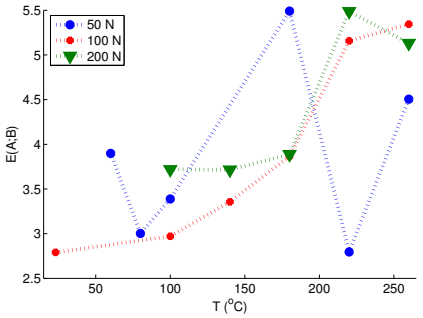
The Fourier pattern spectrum (FPS) gives us very high correlation coefficients (the last row in table 5.13). The mean, for example, is (anti-)linear with temperature, with $\rho(S_M, T) = -0.96$. This negative value is in accordance with the linear correlation of the spectral parameters with temperature, since high frequencies in the FPS are similar to low values of n in the PS, and vice versa.[17]

The plots in figure 5.13 for the FPS show a lot of similarity with those in figure 5.12. A load of 50 N leads mostly to different results compared to the graphs for the other loads. Again, a transition can be observed at 180°C.

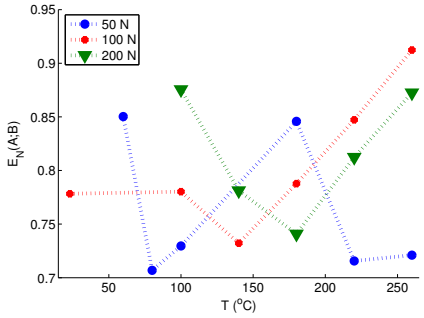
[17] We discussed the similarities between the Fourier spectrum and the pattern spectrum in chapter 4, section 4.2.5.2.



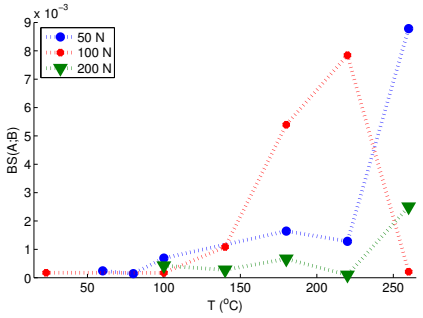
(a) *Size* vs temperature



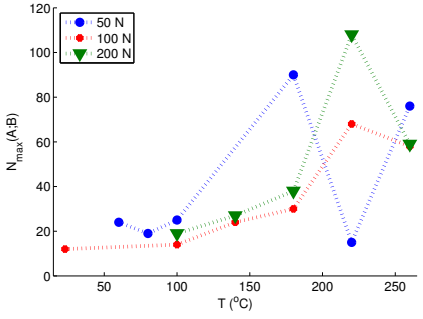
(b) *Roughness* vs temperature



(c) *Normalized roughness* vs temperature

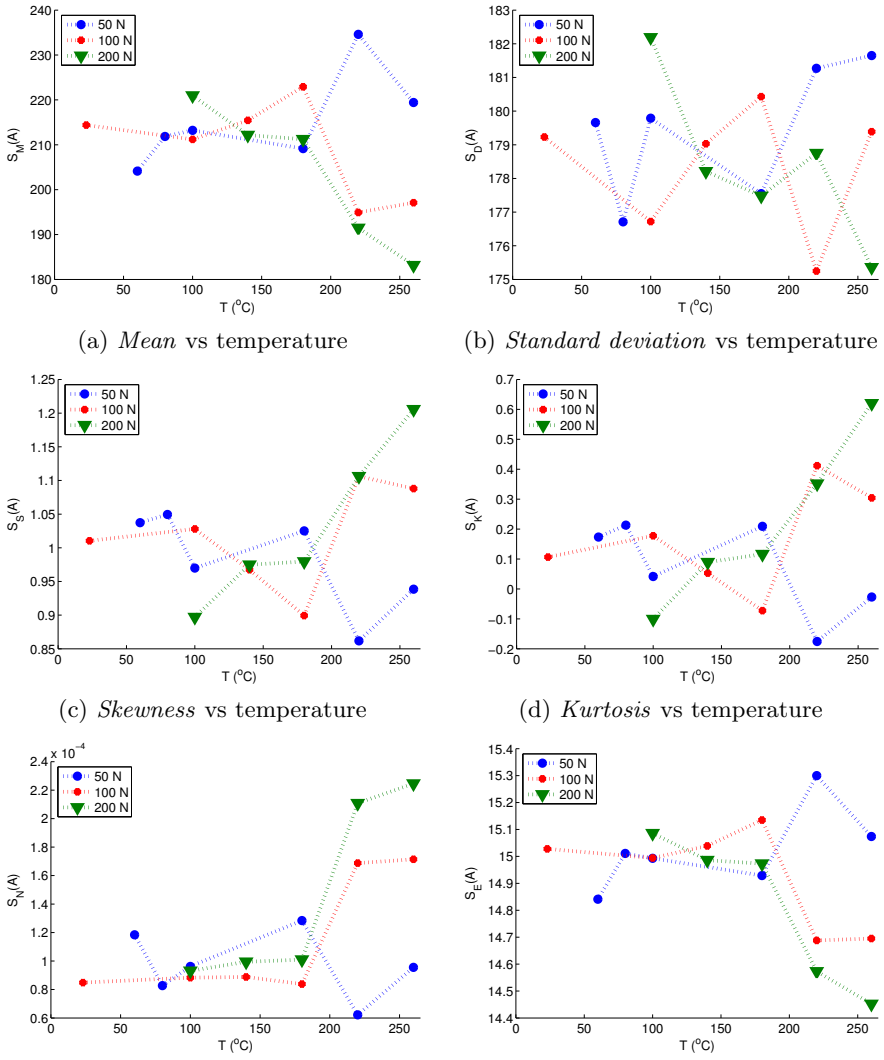


(d) *B-shapiness* vs temperature



(e) *Maximal size* vs temperature

Figure 5.12: The spectral parameters from the pattern spectrum (PS), for SP-1 at 50 N, 100 N and 200 N (greyscale, blackened), plotted against the counterface temperature.



(a) *Mean* vs temperature

(b) *Standard deviation* vs temperature

(c) *Skewness* vs temperature

(d) *Kurtosis* vs temperature

(e) *Energy* vs temperature

(f) *Entropy* vs temperature

Figure 5.13: The spectral parameters from the Fourier pattern spectrum (FPS), for SP-1 at 50 N, 100 N and 200 N (greyscale, blackened), plotted against the counterface temperature.

Table 5.13: Correlation coefficients between spectral parameters and the counterface temperature, for different spectra of blackened greyscale images of SP-1 at 200 N. The last row displays the correlation coefficients for the parameters of the Fourier pattern spectrum.

Spectrum	ρ_S	ρ_E	ρ_{E_N}	ρ_{BS}	$\rho_{N_{max}}$
PS	0.79	0.85	0.068	0.64	0.71
EPS	0.79	0.84	0.37	0.00	0.71
APS	0.76	0.85	0.50	−0.051	0.59
OT_{min}	0.80	0.86	0.62	−0.74	0.82
OT_{max}	0.78	0.81	0.77	−0.32	0.78
MSS-PS	0.83	0.88	0.14	0.30	0.72
ρ_{S_M}	ρ_{S_D}	ρ_{S_S}	ρ_{S_K}	ρ_{S_N}	ρ_{S_E}
−0.96	−0.84	0.97	0.97	0.90	−0.94

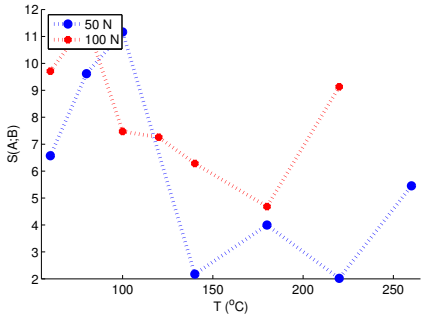
5.3.3.5 SP-21

Graphite filled sintered polyimide has less friction at higher temperatures (above 100°C) and therefore smaller debris, and above 180°C imidization occurs, resulting in a homogeneous polyimide film. The applied loads during the experiments were 50 N and 100 N. Images of particles exist for counterface temperatures of 60°C, 80°C, 100°C, 120°C, 140°C, 180°C, 220°C and 260°C. The sliding frequency is $\nu = 10$ Hz (or speed $v = 0.3$ m/s).

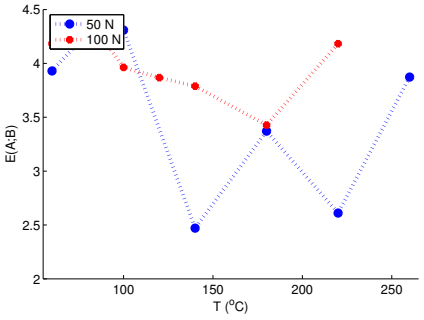
The curves for 50 N and 100 N are quite similar mutually, except for some outlier values (see figure 5.14). We do not observe a clear trend in function of the counterface temperature, but we notice several abrupt changes in the graphs at 140°C and/or at 180°C (especially at 100 N). The correlation coefficients in table 5.14 are almost always below 0.5.[18]

There is an interesting similarity between the graphs in figure 5.14 and figure 5.15 [Samyn, 2007]. Figure 5.15 is a plot of the coefficient of friction μ (see section 5.1.2) against the temperature. Only at the highest temperature the graphs are different: the average size and roughness increase above 200°C. Apparently, there is a direct relationship between the coefficient of friction and the size and diversity (i.e., average roughness) of the debris particles. The correlation coefficients of some parameters from the pattern spectrum and the coefficient of friction are, for a blackened greyscale image and a load of 50 N, $\rho(S, \mu) = 0.84$, $\rho(E, \mu) = 0.66$ and $\rho(E_N, \mu) = 0.75$.

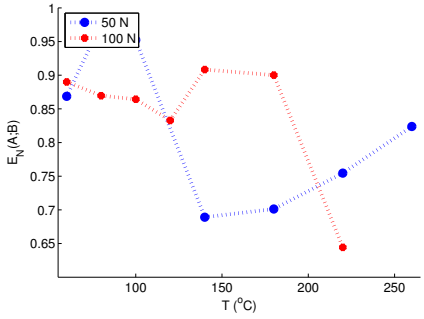
[18] For anti-linear correlations, the correlation coefficients are negative, so they are almost always higher than −0.5.



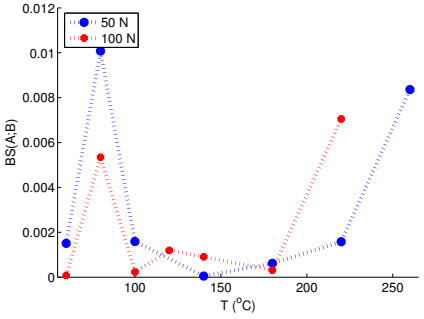
(a) *Size* vs temperature



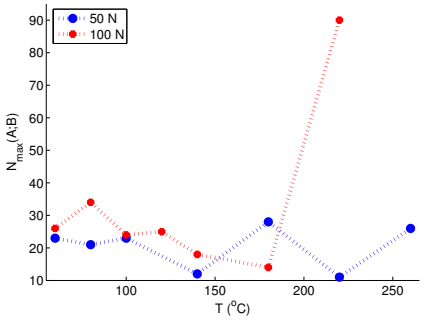
(b) *Roughness* vs temperature



(c) *Normalized roughness* vs temperature



(d) *B-shapiness* vs temperature



(e) *Maximal size* vs temperature

Figure 5.14: The spectral parameters from the pattern spectrum (PS), for SP-21 at 50 N and 100 N (greyscale, blackened), plotted against the counterface temperature.

Table 5.14: Correlation coefficients between spectral parameters and the counterface temperature, for different spectra of blackened greyscale images of SP-21 at 100 N. The last row displays the correlation coefficients for the parameters of the Fourier pattern spectrum.

Spectrum	ρ_S	ρ_E	ρ_{E_N}	ρ_{BS}	$\rho_{N_{max}}$
PS	−0.46	−0.42	−0.63	0.41	0.55
EPS	−0.41	−0.65	−0.86	0.59	0.55
APS	0.52	−0.026	−0.68	0.69	0.52
OT_{min}	−0.17	−0.15	−0.34	0.079	0.46
OT_{max}	−0.010	0.099	−0.12	0.86	0.37
MSS-PS	0.36	0.0031	−0.60	0.79	0.60
ρ_{S_M}	ρ_{S_D}	ρ_{S_S}	ρ_{S_K}	ρ_{S_N}	ρ_{S_E}
0.47	−0.90	−0.63	0.67	−0.54	0.49

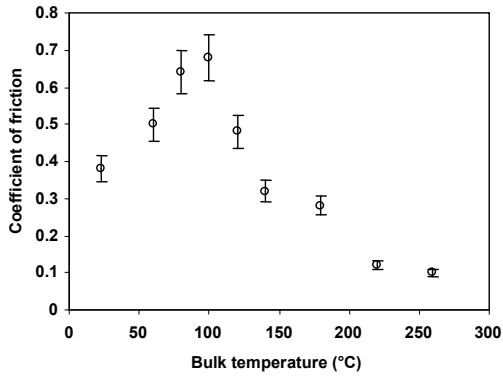
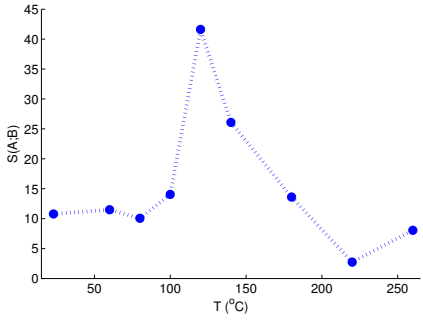


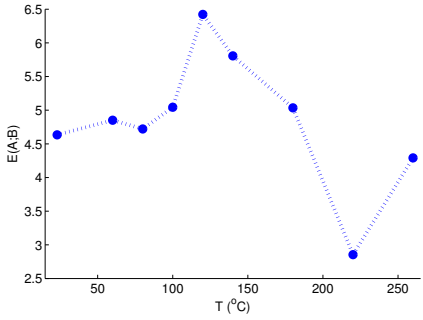
Figure 5.15: The coefficient of friction plotted against the counterface temperature, for SP-21 at 50 N and $\nu = 10$ Hz.

5.3.3.6 TP

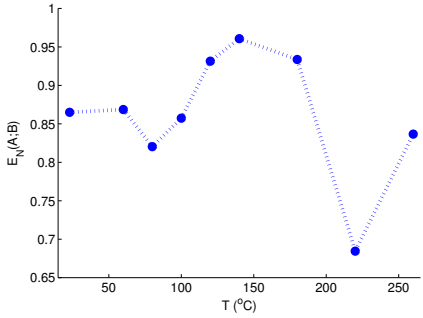
Tribological studies of this material show that the debris particles gradually become bigger at higher pressures and temperatures [Samyn et al., 2006]. There is imidization between 140°C and 200°C. The thermoplastic polyimide melts around 200°C–250°C during the wear experiment. Figure 5.16 shows the behaviour of the spectral parameters with the temperature, for a fixed load (50 N). The temperatures applied are "free T", 60°C, 80°C, 100°C, 120°C, 140°C, 180°C, 220°C and 260°C. Figure 5.18 shows the behaviour of the spectral parameters with normal load, at "free T". The loads applied are 50 N, 100 N, 150 N and 200 N. The sliding frequency is always $\nu = 10$ Hz (or speed $v = 0.3$ m/s). The material melts above 150 N.



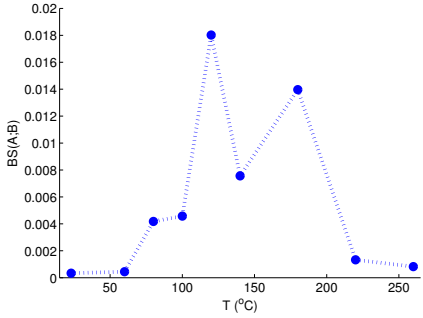
(a) *Size* vs temperature



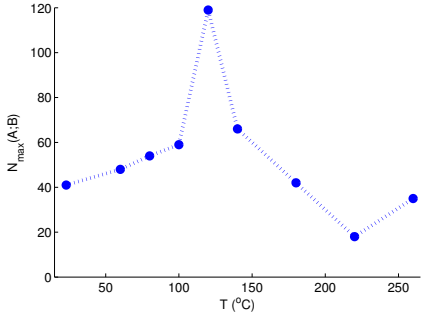
(b) *Roughness* vs temperature



(c) *Normalized roughness* vs temperature



(d) *B-shapiness* vs temperature



(e) *Maximal size* vs temperature

Figure 5.16: The spectral parameters from the pattern spectrum (PS), for TP at 50 N (greyscale, blackened), plotted against the counterface temperature.

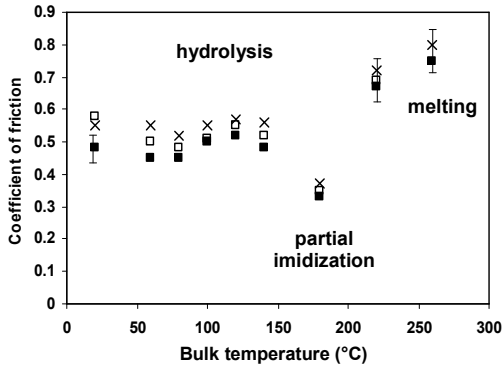


Figure 5.17: The coefficient of friction plotted against the counterface temperature, for TP at 50 N and $\nu = 10$ Hz. Cross: measurement after 30 m sliding; white square: measurement after 100 m sliding; black square: measurement after 15 000 m sliding or end-of-test.

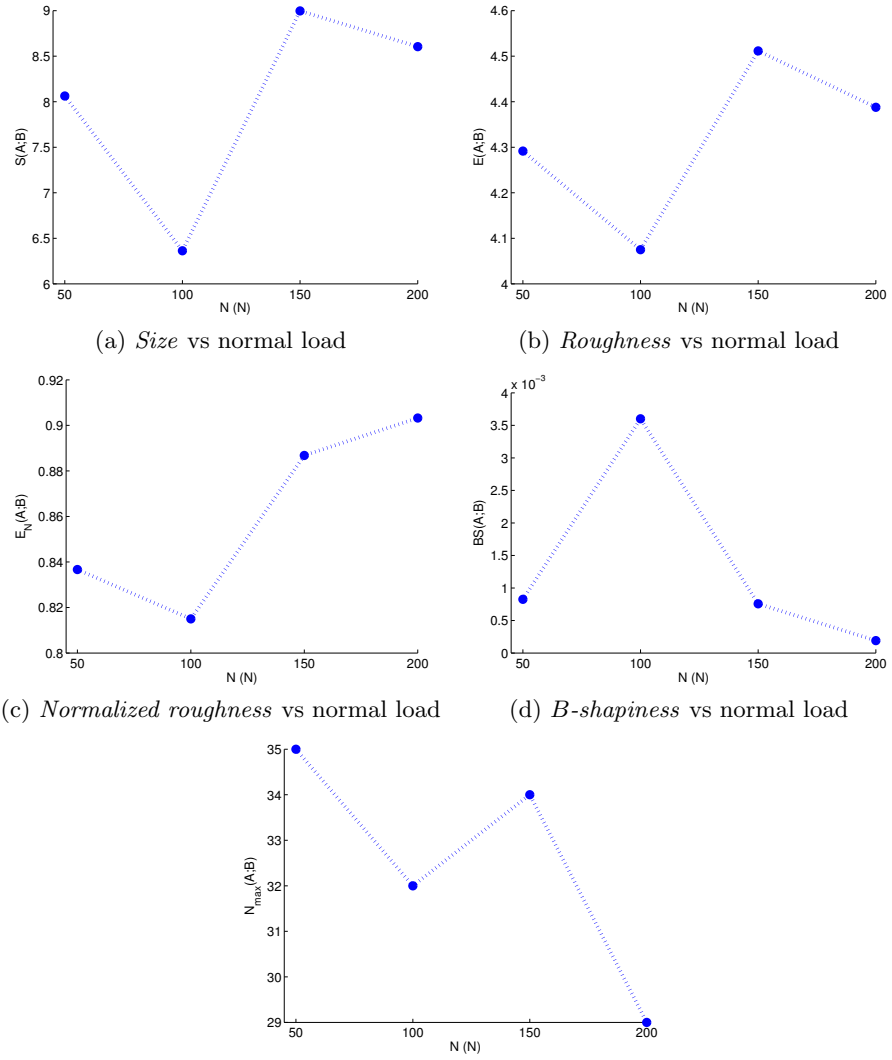
At 120°C, a peak is visible in several graphs (figure 5.16). This indicates that one or more large debris particles are present. The high value of the maximal size (N_{max}) indicates that the size of these large particles is very big compared to the size of the other particles. The roughness also increases, so there are also smaller particles present at the same time or the big objects are very rough and jagged. The peak is also visible in the graph for average size, which means that the larger object(s) have a significant influence on the average size of the particles. When we look at the pictures, we notice for $T = 120$°C and $T = 140$°C that very large and rough debris particles are present and only few small particles.[19]

As with SP-21, we notice some correlation between the plot of the coefficient of friction against the temperature (figure 5.17 [Samyn, 2007]) and the graphs in figure 5.16. The plots in figures 5.16(b) and (c) ((normalized) roughness) approximately correspond to the plot in figure 5.17. This visual correlation can mean that the friction is related to the diversity of the particles in the image. The correlation coefficients, however, are low and negative ($\rho < |-0.70|$).

Table 5.15 lists some correlation coefficients. In general, the absolute values are rather low, below 0.5. The peak in the graphs attributes strongly to these low ρ-values. A good linear fit of the spectral parameters against the counterface temperature is therefore not possible. This is also true for the Fourier pattern spectrum.

Figure 5.18 shows the spectral parameters from the pattern spectrum (PS), for TP at "free T" and for blackened greyscale images, plotted against the applied normal force. If the values at 100 N in the graphs are considered outliers, then we can conclude that the pattern spectrum indeed indicates that the

[19]These large particles are about ten times larger in area than the ones at the other temperatures.



(a) *Size* vs normal load

(b) *Roughness* vs normal load

(c) *Normalized roughness* vs normal load

(d) *B-shapiness* vs normal load

(e) *Maximal size* vs normal load

Figure 5.18: The spectral parameters from the pattern spectrum (PS), for TP at "free T" (greyscale, blackened), plotted against the applied normal force.

average size of the particles increases (but not monotonically), as well as the entropy. On the other hand, the maximal size N_{max} states that the size of the objects decreases with the temperature. This suggests that the biggest objects become smaller, but at higher temperatures there are more of those big objects, compared to the smaller debris. We can confirm this by looking at the pictures.

Table 5.15: Correlation coefficients between spectral parameters and the counterface temperature, for different spectra of blackened greyscale images of TP at 50 N. The last row displays the correlation coefficients for the parameters of the Fourier pattern spectrum.

Spectrum	ρ_S	ρ_E	ρ_{E_N}	ρ_{BS}	$\rho_{N_{max}}$
PS	−0.30	−0.49	−0.32	−0.10	−0.46
EPS	−0.28	−0.40	−0.30	0.44	−0.41
APS	−0.34	−0.61	0.20	−0.66	−0.59
OT_{min}	−0.28	−0.44	−0.23	0.42	−0.52
OT_{max}	−0.30	−0.48	−0.41	−0.20	−0.36
MSS-PS	−0.24	−0.47	−0.22	−0.38	−0.41
ρ_{S_M}	ρ_{S_D}	ρ_{S_S}	ρ_{S_K}	ρ_{S_N}	ρ_{S_E}
0.30	0.22	−0.24	−0.21	−0.20	0.28

Table 5.16: Correlation coefficients between spectral parameters and the normal force, for different spectra of blackened greyscale images of TP at "free T". The last row displays the correlation coefficients for the parameters of the Fourier pattern spectrum.

Spectrum	ρ_S	ρ_E	ρ_{E_N}	ρ_{BS}	$\rho_{N_{max}}$
PS	−0.28	−0.16	0.63	−0.26	−0.86
EPS	0.70	0.70	0.47	−0.86	0.41
APS	0.75	0.74	0.40	0.18	0.52
OT_{min}	0.78	0.84	−0.19	0.64	0.83
OT_{max}	0.59	0.70	−0.036	0.12	0.31
MSS-PS	0.75	0.87	0.72	−0.76	0.41
ρ_{S_M}	ρ_{S_D}	ρ_{S_S}	ρ_{S_K}	ρ_{S_N}	ρ_{S_E}
0.34	0.43	−0.45	−0.45	−0.14	0.34

Table 5.16 lists some correlation coefficients which sometimes indicate a good linear fit of the spectral parameters with the counterface temperature. This is the case for the average size and average roughness ($\rho \geq 0.7$), for most pattern spectra. The parameters of the default pattern spectrum parameters correlate worst of all with the normal load, while the parameters from MSS-PS correlate quite well. Also the parameters obtained from OT_{min} correlate well with the applied load, except for the normalized roughness. The correlation coefficients are not high enough to state a definite conclusion, but the values suggest an increase of the debris size with the normal force.

5.4 Conclusion

The biggest problem with our analysis of the pattern spectra is the small amount of available data. For each of the experimental settings, we had access to a few pictures of debris particles. In order to perform a more relevant statistical research, much more data would be needed. Also, the number of experimental settings is limited, which hinders the correlation accuracy.

Nevertheless, we have obtained some interesting results. We can confirm for one experiment that, using the pattern spectrum, the size of the debris particles of the wear of polymer POMH increases with the contact pressure.

The sintered polyimide SP-1 has a transition temperature above $T = 180°\text{C}$, when imidization starts to occur. We observe this through the behaviour of the spectral parameters with the temperature.

The parameters from SP-21 tell us something similar as with SP-1, but less unambiguously. However, for the graphite filled polymer SP-21, we can correlate the coefficient of friction with the average size and roughness obtained from the pattern spectrum.

For the material TP, we notice a correlation between the coefficient of friction and the (normalized) average roughness, when the load is fixed.

The morphological pattern spectrum (PS) has a high computational cost. Each of the suggested alternatives has its own characteristics. Most of them are much faster, so they could replace the PS. It is not clear which one is best suited as the replacement for PS. Interesting to note is that the parameters of the pattern spectrum based on the majority sorting scheme, MSS-PS, correlate quite well with the experimental settings of the tribological experiments.

For this kind of images, the influence of colour is minimal. Processing of the greyscale (or even the binary) versions of the images produces similar results.

Chapter 6

Image Interpolation

Bitmapped images sometimes need to be magnified. This means increasing the number of pixels covered by such an image. If the magnification is done by using simple interpolation techniques (which will be explained later on in this chapter), the result often looks jagged, because they are unable to increase the apparent resolution. This chapter deals with an interpolation technique that removes those jagged edges from the images. Mathematical morphology is used in a novel way to perform this task.

In this chapter, we first explain the difference between vector and raster graphics. Then we discuss some standard linear interpolation techniques and state-of-the-art non-linear techniques. Afterwards, we explain our morphological interpolation technique for binary images, as well as an improvement of this method. We propose an extension to greyscale images in the final section.

6.1 Introduction

6.1.1 Vector and raster graphics

When we consider image formats, we can distinguish between two classes of images: *vector graphics* and *raster graphics*. Vector graphics [Wikipedia, WWWb] are images in which the content (the *objects*) is described using geometrical primitives, such as points, lines, curves and polygons. The vector graphics format is used for line art drawings, such as logos, diagrams, graphs and text. There are different advantages to the use of vector graphics:

- Objects can be drawn with a minimum of information. Indeed, in order to draw a circle on a computer screen, we mainly only need to specify the type of object (a circle), the origin and radius of the circle, instead of every point on its boundary. Additional specifications would be the stroke style and fill style and their respective colour.

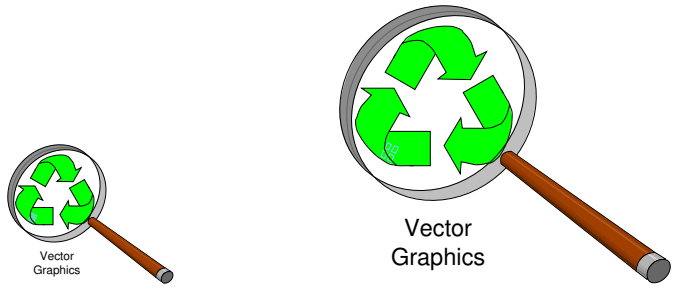


Figure 6.1: Vector graphics look good at any magnification.

- The objects can later be edited with a minimum of effort. We can translate, rotate, change colour, fill, ... the objects by only updating specific parameters.
- The image objects can be changed in size without limitation. Because of the geometrical character of the image, it is possible to zoom in on the objects in the image and still see smooth lines.

Figure 6.1 shows an image stored as a vector image. The magnified version of this image clearly looks smooth. Different software applications are available that produce vector graphics, like Adobe Illustrator, CorelDRAW, Xfig and others. The vector graphics can be stored in a variety of file formats. Best known are Encapsulated PostScript (EPS), Windows Meta File (WMF), Adobe Flash (SWF), and Adobe's PostScript (PS) and Portable Document Format (PDF). PS and PDF are both page description languages that use a vector graphics model, and are therefore device-independent. TrueType fonts are vector based and are thus scalable fonts, which makes it possible to print text well with an arbitrary point size.

The other class of images, raster graphics, is the one we take a closer look at now. It is the type of images used in image processing. A raster image is also called a *digital image* or *bitmap*. A bitmapped image consists of picture elements, *pixels*, aligned on a grid. Each pixel has its own colour or grey value. The difference with vector images is clear: the scene in the image is now described in terms of pixels, rather than by (the drawing of) geometric primitives. In general, more disk space is needed to store a bitmap (every pixel requires an amount of memory), but it is now possible to represent a real life scene, with a lot of texture, colour variation and detail. We can visualize a textured and detailed scene with a vector graphic, but then we must use dots as geometrical primitives. In that case, the advantage of using a vector graphic is lost. Therefore, vector graphics are used for artificial images (line art, graphs, charts, logos), while raster graphics are generally used for pictures.

Applications like Adobe Photoshop, Corel Paint Shop Pro or The GIMP are sophisticated computer applications for editing pictures, pixel by pixel, or as a

group. Vector graphics can also be edited with these programs, but they are then converted to raster format.

The quality of a raster image is determined by its *resolution*, which is the maximum amount of spatial detail in the image (which in turn is related to the total number of pixels that describe that raster image), and the precision of the colour information in each pixel (often called *colour depth*). The latter is not important in this discussion. When we wish to display this image on a computer screen or print it on paper using an inkjet or laser printer, the resolution is the number of *pixels per inch* (PPI, also called the *pixel density*) or *dots per inch* (DPI), respectively. The terms PPI and DPI are sometimes interchanged.

When an actual measure of the physical resolution is not necessary, we can define the image resolution just by the width and height of that image, e.g. $M \times N$, measured *in pixels*. It is not an uncommon practice to denote the image size as the image resolution and the physical resolution as the pixel density.

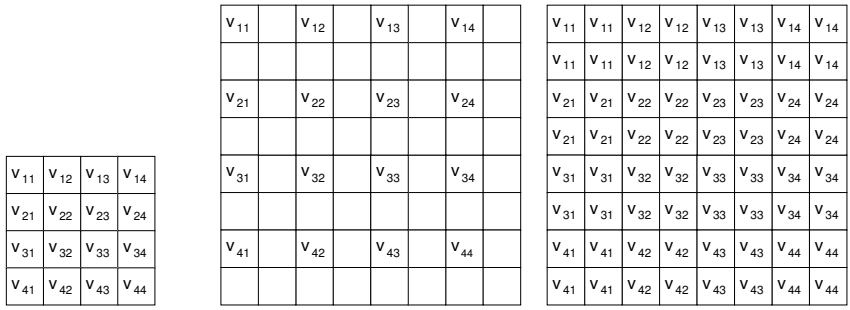
A consequence of the pixel-wise approach is the fact that we cannot scale up without limitation and without introducing artefacts. An example: consider a 15 inch computer display with a size of 1024×768 pixels. 15" is the length of the diagonal, so the size of the screen is 12" $\times$ 9", assuming that the aspect ratio is 4 : 3. The pixel density is then 85.3 PPI. An image with a size of 512×384 pixels that is displayed at the resolution of the monitor's pixel depth (i.e., a quarter of the screen is filled by the image) will be visually pleasing.[1] When we zoom in on the image, i.e., we magnify the image M times (e.g., to full-screen, $M = 2$), then the number of pixels should also be increased (M^2 times), in order to keep the pixel density constant. The problem is that we only know the pixel values of the original pixels. The values of the new pixels must be guessed using some intelligent calculation. This process is called *image interpolation*.

The simplest interpolation method is *pixel replication* or *nearest neighbour interpolation*. Every pixel from the original image now occupies $M \times M$ pixels. In other words, the pixel values in the enlarged image are copied from the pixels at the corresponding position in the original image, or, if that position does not correspond to a pixel centre, from the pixels nearest to that position. This is illustrated in figure 6.2. Figure 6.3 shows an example of a binary raster graphic that is magnified 4 times using pixel replication. The result contains unwanted jagged edges, called *jaggies*.[2]

To avoid or remove these jaggies (as has been done in figure 6.4), various interpolation techniques are at our disposal (sections 6.1.3 and 6.1.4). Most

[1] Not taking the content itself into account, that is.

[2] According to [Wikipedia, WWWa], the origin of the term "jaggies" is the following: in the video game *Rescue on Fractalus!* (1985), aliens that were called *Jaggi* made their appearance. They seem to have been given this name, due to the blocky/jagged graphics of this game.



(a) Original image (b) Intermediate result (c) Pixel replication

Figure 6.2: Pixel replication for 2× magnification: the original pixel values are copied to the newly available pixels.

of them work well on natural scenes, but some of them are suited for the interpolation of line art (i.e., two-colour drawings) or multi-colour images with sharp edges (e.g., cartoons). These techniques may seem superfluous because the preferred way to store line art is using vector graphics. This is true, indeed, but there are situations where bitmaps are unavoidable. E.g., to display an image on a website, the World Wide Web Consortium, or W3C [W3C, WWW], has approved three file formats, GIF, JPEG and PNG, all three raster graphics formats, and a standard format for vector graphics, SVG. The Scalable Vector Graphics format (SVG) can be used to display a logo or graph on a web page and adhere to the web standards, but its use is not yet widespread. This is because several browsers still need a plug-in, like Adobe's SVG Viewer, to display SVG. Microsoft's Internet Explorer and Apple's Safari are among them. Other browsers, like Opera and Firefox, only recently included native support for SVG. It is possible to display the picture using the popular Flash format, but this is not a web standard and also requires a browser plug-in. So, in most cases, line art and cartoons are converted to one of the raster graphics formats.

In other situations it happens that only the raster image is available, e.g., because the vector image file is lost, corrupt or unavailable, the line art image was created with a raster image editing application, or the vector image has been edited and then saved as a raster image.

Another example where interpolation can be advantageous is digital television. Footage from a standard definition camera has a smaller resolution than HDTV. Interpolation can improve the visual result of the images when viewed on a high definition display. Especially for text this can be an advantage.

A similar problem occurs with LCD projectors: presentations are beamed on a big screen, but if the resolution of the connected laptop is too low, the result does not look good without interpolation.

Another application is to use a segmentation mask in video motion estimation applications (especially in object based coding) at different magnifications.

(a) Original image (b) 4× magnified

Figure 6.3: Raster graphics: pixel replication creates "jaggies".

Figure 6.4: An interpolated raster graphic (4× magnified version of figure 6.3(a)).

In the field of digital libraries, interpolation is useful when a low-resolution electronic version of the document already exists. When the original paper document is no longer available, e.g., it has been destroyed or is lost, then interpolation is a solution to obtain a high resolution document. Similar problems arise in computer graphics where the source file of a line drawing is missing and only bitmaps are left.

6.1.2 Interpolation

Image interpolation or resampling has many applications [Lehmann et al., 1999, Meijering et al., 2001, Meijering, 2002]. Sometimes, the aspect ratio of the stored image is not the same as that of the desired output. The *pixel aspect ratio* describes the squareness of the pixels, i.e., the ratio between the height and the width of the pixel. For example, consider a photograph taken with a digital camera, which contains a CCD (charge-coupled device) with a certain pixel aspect ratio. If the pixel aspect ratio of the display device is different from the CCD's pixel aspect ratio, then interpolation is required to show the picture with the correct aspect ratio. Interpolation is also necessary for rotations or perspective projections. A more common feature would be the rescaling of existing images, i.e., changing the image resolution (as we mentioned in the

previous section). This rescaling is desired when we zoom in on a portion of the image (we magnify) or when we want to change the pixel density (we need more detail).

Several interpolation techniques exist, as we will discuss in the following sections. We can draw a distinction between *linear* and *non-linear* interpolation methods. Linear or *convolution-based* techniques are the most frequently used methods, because they are versatile and have a relatively low complexity. Some non-linear techniques are discussed in section 6.1.4. They often give better results than the linear methods, but their design is more complex.

Convolution-based image interpolation or resampling actually reconstructs a two-dimensional image on a continuous domain from the discrete samples. From this, the missing intermediate samples can be found by simple sampling. Optimal reconstruction is only possible when the *Nyquist criterion* (explained in section 6.1.2.1) was satisfied when the given input image was acquired. The ideal convolution-based interpolation faces practical problems (section 6.1.2.3), which force us to use other techniques, that approximate this ideal interpolation.

We now discuss the theoretical background of sampling and linear interpolation.

6.1.2.1 Sampling of a continuous image

Image sampling is the process of converting a continuous image $s(x, y)$, with $x, y \in \mathbb{R}$, into a discrete image $s(m, n)$, with $m, n \in \mathbb{Z}$. That is, the image function is spatially digitized.[3] The *sampled image* $s_s(x, y)$ is the discrete sample defined on a continuous domain:

$$s_s(x,y) = \sum_{m=-\infty}^{+\infty} \sum_{n=-\infty}^{+\infty} s(m\Delta x, n\Delta y)\delta(x - m\Delta x, y - n\Delta y) \ . \qquad (6.1)$$

We make use of the *Dirac comb* $\delta_T(t) = \sum_{k=-\infty}^{+\infty} \delta(t - kT)$ (or *Shah function* Ш), where T is the *sampling interval*. The sampled image is useful for theoretical derivations because it represents the discrete sample $s(m, n)$ in terms of a function of continuous variables.

A signal can be represented by its *Fourier transform*. Simply stated, the Fourier transform (FT) of a signal $s(t)$ is its representation $S(f)$ in the frequency domain and of a spatial signal its spatial frequency domain. Most often, the term Fourier transform refers to the continuous FT, but a discrete FT (operating on discrete images) can also be defined. The *discrete Fourier transform* (DFT)[4] of a discrete image $s(m, n)$, $S(u, v)$, is periodic with periods M and N. We

[3] Digitization of the signal values (for an image these are the grey values or colour values) is called *quantization*.

[4] For more information, see also chapter 4.

repeat equation (4.52):

$$S(u,v) = S(u + \alpha M, v + \beta N) \ , \tag{6.2}$$

with $\alpha, \beta \in \mathbb{Z}$. In other words, the Fourier transform of $s(m,n)$ repeats infinitely. The transformation of a continuous image $s(x,y)$ is not repeated. The periodicity of the Fourier image is a consequence of sampling.

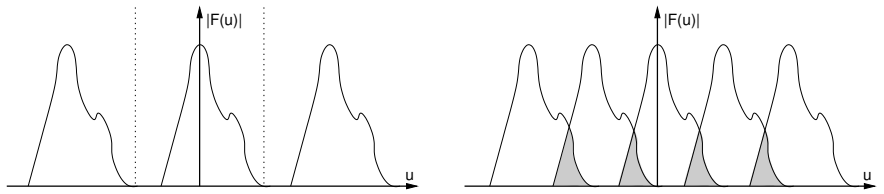
Reconstructing the original continuous image from its samples, is equivalent to removing the replicas (i.e., the copies of the FT around $(u,v) = (0,0)$) from the Fourier transform. For this to be possible, the replicas must not overlap, which is the case when the Nyquist criterion is satisfied. In turn, this requires the signal to be bandlimited. A function is *bandlimited* if in the Fourier domain above a specific absolute frequency the spectrum values are 0. In our two-dimensional case we have two such frequencies, u_{max} and v_{max}, corresponding to the maximum spatial frequencies in the image in the horizontal and vertical direction, respectively. The *Nyquist criterion* or *Whittaker-Shannon sampling theorem* states that the sampling intervals $(\Delta x, \Delta y)$ in the spatial domain must be chosen such that:

$$\begin{cases} \Delta x < 1/(2u_{max}) \\ \Delta y < 1/(2v_{max}) \end{cases} . \tag{6.3}$$

The sampling interval is the spatial distance between the sampling points in the image. This sampling criterion thus depends somewhat on the image content, because it is related to the frequencies (u_{max}, v_{max}). When the Nyquist criterion is satisfied, then all the image information is preserved. If not, the Fourier transform in the interval $([-u_{max}, u_{max}], [-v_{max}, v_{max}])$ is corrupted by frequency components from the replicas of the spectrum. This distortion of the image is called *aliasing*. High frequency components from one replica are mixed with low frequency components from the adjacent replica, which results in a loss of information in the sense that the original frequency information cannot be retrieved from the Fourier transform. Figure 6.5 shows, for a one-dimensional signal, the magnitude of the Fourier transform of a signal that has been sampled at intervals smaller than $1/2u_{max}$[5] and that of a signal that has been sampled at intervals greater than $1/2u_{max}$.

An example of aliasing is shown in figure 6.6. The left "Barbara" image is the original one. The right image is a downsampled version by a factor of 2, but rescaled again. Notice how the line pattern in the scarf is oriented differently. An effect of aliasing are moiré patterns.

[5]This is the same as to say that the sampling is performed at a frequency below $2u_{max}$. The frequency u_{max} is also called the *Nyquist frequency*, $2u_{max}$ is called the *Nyquist rate*.



(a) Nyquist criterion fulfilled (b) Aliasing

Figure 6.5: (a) The Fourier transform of a discrete signal is repeated infinitely. (b) When the sampling frequency is too low, then the signals overlap in the Fourier domain and aliasing occurs.

6.1.2.2 Convolution-based interpolation

When we have a digital image and we want to obtain its continuous counterpart or resample it, then image interpolation is necessary. The continuous image $s(x, y)$ (with $x, y \in \mathbb{R}$) is reconstructed from its sampled image $s_s(x, y)$ (with $s_s(x, y) = s(x\Delta x, y\Delta y)$ for $x, y \in \mathbb{Z}$ and $s_s(x, y) = 0$ otherwise), using a convolution operation:

$$s(x, y) = s_s(x, y) * h(x, y) \tag{6.4}$$

$$= \sum_{m=-\infty}^{+\infty} \sum_{n=-\infty}^{+\infty} s_s(m\Delta x, n\Delta y) h(x - m\Delta x, y - n\Delta y) \ . \tag{6.5}$$

The filter h is called the two-dimensional *impulse response*[6] of the two-dimensional reconstruction filter. In the following subsection we will define h.

The two-dimensional interpolation kernel h is usually separable into a horizontal and a vertical one-dimensional kernel, because this reduces the computational complexity:

$$h(x, y) = h_x(x) h_y(y) \ . \tag{6.6}$$

The calculation complexity is the following: suppose $\Omega(h_x)$, $\Omega(h_y)$ and $\Omega(h)$ are the *support* for h_x, h_y and h, respectively. The support is the set of pixels of the kernel that contribute to the convolution. In practice, only pixels in a small neighbourhood of the current pixel (x, y) contribute. For the one-dimensional kernels, we need $2\sharp[\Omega(h_x)] - 1$ and $2\sharp[\Omega(h_y)] - 1$ calculations per pixel.[7] On the other hand, a two-dimensional kernel needs $2\sharp[\Omega(h)] - 1 = 2\sharp[\Omega(h_x)]\sharp[\Omega(h_y)] - 1$ calculations per pixel. We can easily calculate that in all cases, except the trivial ones, the combination of two separate one-dimensional kernels is faster

[6] Other names are the *reconstruction kernel* or *interpolation kernel*.

[7] The symbol $\sharp$ denotes the number of pixels in the set.

(a) Nyquist criterion fulfilled (b) Aliasing

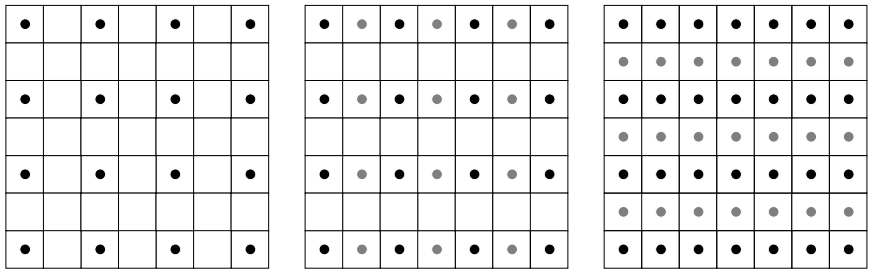
Figure 6.6: When the sampling frequency is too low, aliasing occurs. Compare the orientation of the line patterns in the encircled areas in (b) with the respective patterns in (a).

than one two-dimensional kernel. Not performing an interpolation technique is faster,[8] but the consequence is visual artefacts (jaggies).

Figure 6.7 schematically shows how a two-dimensional interpolation is performed, using two separate one-dimensional kernels. In the example, a magnification factor $M = 2$ is used. Figure (a) shows a grid of pixels, where the squares marked with a black dot represent the image pixels from the input image. The white squares are image locations that do not exist in the low-resolution image. The squares marked with a grey dot in figure (b) show the positions of the interpolation results after a horizontal interpolation. These results, together with the original input pixels, are used as input for the vertical interpolation. This can be seen in figure (c). For simplicity, we will further explain the interpolation principles with one-dimensional images.

The interpolation factor M can be some other value than 2. For example, when the magnification is 4, we must calculate three new pixel values between the defined pixels, instead of just one. An interpolation in multiple stages is also possible (e.g., interpolating twice using $M = 2$ instead of once with $M = 4$). Fractional magnifications, like $M = 3/2$ can be achieved by first interpolating by a factor of 3, and then downsampling by a factor of 2. In practice, this would be implemented in one step in the interpolation kernel.

[8] In that case, actually pixel replication is used.



(a) Discrete image (b) Horizontal interpolation (c) 2D interpolation

Figure 6.7: Two-dimensional image interpolation using separable interpolation kernels. The black dots indicate input pixels; the other pixels do not exist in the low-resolution image; the grey dotted pixels are calculated (interpolated) using the black dotted pixels.

6.1.2.3 Ideal interpolation

In section 6.1.2.1 we explained that the Fourier transform of a discrete image is periodic and consists of the superposition of the original spectrum and replicas. If the replicas do not overlap, the continuous image can be reconstructed by removing the replicas through lowpass filtering. This can be implemented by multiplication of the Fourier domain with a rectangular function. This *rectangular function* $\Pi(v)$ is defined as follows:[9]

$$\Pi(v) = \begin{cases} 0 & \text{for } |v| > 1/2 \\ 1/2 & \text{for } |v| = 1/2 \\ 1 & \text{for } |v| < 1/2 \end{cases} , \tag{6.7}$$

with v the (spatial) frequency.

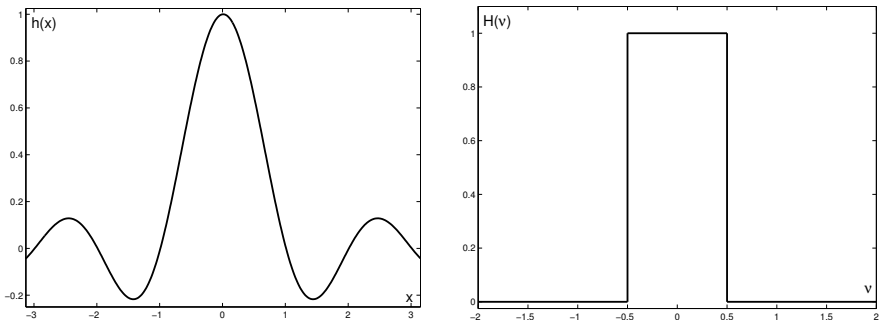
In the spatial domain, the product of the Fourier transformed function with the rectangular function is a convolution of this function with a (normalized) sinc function. This sinc is the *ideal interpolator*:

$$\begin{aligned} h_{ideal}(x) &= \operatorname{sinc}(x) \\ &= \sin(\pi x)/(\pi x) \ . \end{aligned} \tag{6.8}$$

The sinc function assumes the value 1 for $x = 0$, and the value 0 for $x \in \mathbb{Z}_0$.[10] The sinc function is displayed in figure 6.8 for the interval $-\pi < x < \pi$. The Fourier transform of this interpolation kernel is the rectangular function,

[9] A common alternative definition is to give for all $|v| < 1/2$ the value 1, and for every other v value 0.

[10] With the notation $\mathbb{Z}_0$ we denote the set of all integers, both positive and negative, with the exception of 0. In other words: $\mathbb{Z}_0 = \{\ldots, -3, -2, -1, +1, +2, +3, \ldots\}$.



(a) Sinc kernel (b) The Fourier spectrum

Figure 6.8: The ideal interpolator and its Fourier spectrum.

as mentioned before. This transfer function $\Pi(v)$ in the Fourier domain is constant and has value 1 in the interval $-\pi < \omega < \pi$ (or $-1/2 < v < 1/2$), with ω the *angular frequency* and v the *frequency*, which is $v = \omega/2\pi$. This interval is called the *passband*, and the value $\omega = \pi$ or $v = 1/2$ is the *cutoff point*. The values $|\omega| > \pi$ (or $|v| > 1/2$) are part of the so-called *stopband*.

The sinc function is called an *interpolator*. In order for a function to be called "interpolator", it must satisfy the following conditions:

$$\begin{cases} h(0) = 1 \\ h(x) = 0, |x| = 1, 2, \dots \ . \end{cases} \tag{6.9}$$

Interpolation kernels that do not satisfy these properties, are called *approximators*.

The sinc function is capable of reconstructing the original continuous image from the discrete image, using a convolution operation. Its Fourier spectrum, a rectangular shaped spectrum, is multiplied with the discrete image, which separates the frequency components of the image from frequency components of the replicas. Perfect reconstruction is only possible when the Nyquist criterion is fulfilled. We remark that in practice it is not possible to satisfy the Nyquist criterion, because images have a finite spatial support and objects with a finite spatial support have an infinite frequency support. Another problem is that the sinc function is spatially unlimited. This makes the exact computation of the sinc interpolation impossible, since all sampling points are needed for reconstruction. Therefore, the ideal interpolator is replaced by some approximation of this ideal, which results in artefacts. We will now discuss some existing interpolators.

6.1.3 Standard linear interpolation techniques

Linear (or *non-adaptive*) interpolation techniques are relatively simple to implement and are therefore popular interpolators. Because the exact computation of the spatially unlimited sinc interpolator is not possible, alternative interpolators are available and used. These interpolators are similar to the sinc, but spatially limited. The following interpolators are frequently used basic alternatives for the sinc function.

6.1.3.1 Pixel replication

Pixel replication (also called *nearest neighbour interpolation*), introduced in section 6.1.1, is the most straightforward interpolation technique. We replace the sinc function as interpolation kernel by the rectangular function $\Pi(x)$. The value is 1 inside of the interval $-\Delta/2 < x < +\Delta/2$, and 0 outside of this interval. Δ is the length of the sampling interval. The Fourier transform of $\Pi(x)$ is a sinc function. This has two implications: due to the sidelobes in the FT of $\Pi(x)$, the frequency components of the replicas are not removed (i.e., not all frequencies in the stopband are attenuated), which results in staircase patterns, *jaggies*.[11] Pixel replication also introduces blur, because the sinc function slightly attenuates the highest frequencies of the passband (the sinc already descends before it reaches the cutoff point). An example of jaggies in a greyscale image is shown in figure 6.10(a). It is a cut-out of the "boat"-image shown in figure 6.9, after magnification with factor $M = 4$.

6.1.3.2 Bilinear interpolation

Other simple techniques are *bilinear* and *bicubic interpolation*. Here, the new pixel value is the (weighted) mean of respectively the 4 and 16 closest neighbours. The one-dimensional interpolation kernel for bilinear interpolation is a *triangular function*:

$$h(x) = \begin{cases} 1 - |x|, 0 \leq |x| < \Delta \\ 0, \qquad \text{elsewhere.} \end{cases} \tag{6.10}$$

The impulse response for the bilinear interpolation (i.e., the triangular function) and its Fourier spectrum are shown in figure 6.11. The jagginess is less prominent than with pixel replication, but bilinear interpolation blurs the image. This is due to the loss of some high frequency information when using this interpolation kernel. This area is marked with a rectangle in the figure. An example of this blur is shown in figure 6.10(b).

[11] Often, the terms *aliasing* and *jaggies* are interchanged, while they actually describe different phenomena. Aliasing occurs when the image is downsampled and the sampling rate is too low, resulting in strange results like moiré patterns. Jaggies occur when the image is upsampled, resulting in staircase patterns.

Figure 6.9: An example image. Figures 6.10 and 6.16 show 4× magnified versions of the box selection.

6.1.3.3 Truncated and windowed sinc interpolation

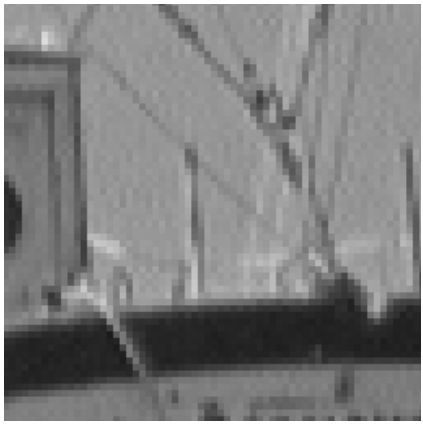
The *truncated sinc function* and *windowed sinc functions* are (altered) sinc functions with a finite number of lobes, and are thus spatially limited. This is equal to a multiplication of the sinc with a window function. The truncated/windowed sinc function is then:

$$h(x) = \begin{cases} \text{sinc}(x)w(x), 0 \leq |x| < \frac{N}{2}\Delta \\ 0, \qquad\qquad \text{elsewhere,} \end{cases} \tag{6.11}$$

with $N \in \mathbb{N}_0$. N is a strictly positive integer that defines the length of the support of the truncated or windowed sinc.

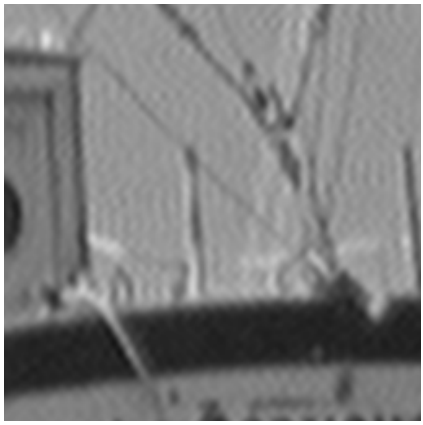
For the *truncated sinc function*, a rectangular function is used, i.e., $w(x) = 1$ for $0 \leq |x| < \frac{N}{2}\Delta$. Figure 6.12 shows the impulse response for the truncated sinc interpolation with $N = 5$. Because of this truncation, the Fourier transform is no longer a block function. Overshoots appear near the transition from the passband to the stopband and oscillations are visible in the stopband region. The ringing artefacts that are visible in the interpolation result at the locations of discontinuities (sharply defined edges) when the higher frequencies are cut off (see figure 6.10(c)), are known as the *Gibbs phenomenon*.

For a general *windowed sinc function*, an equation similar to (6.11) is used, but



(a) Jagginess

(b) Blurring



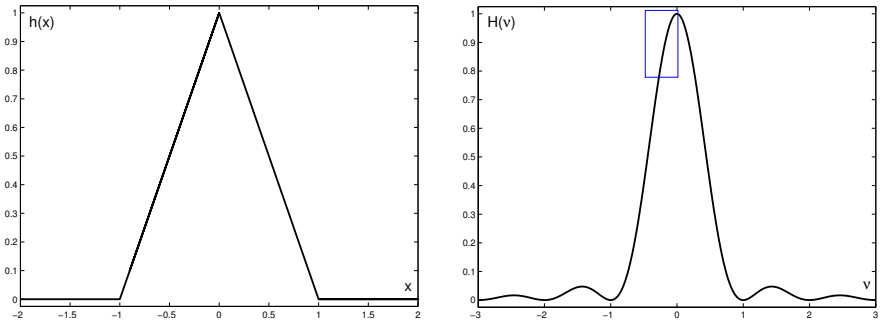
(c) Ringing

Figure 6.10: Possible artefacts that appear when linear interpolation techniques are used.

with $w(x)$ not a rectangular function. A wide variety of window functions exist [Lehmann et al., 1999, Meijering et al., 2001]. The *Blackman-Harris* window [Harris, 1978] is an example of such a window that gives a quite good approximation of the ideal interpolator. A windowed sinc function can avoid the Gibbs phenomenon mentioned before that is obtained when using a truncated sinc function as interpolator.

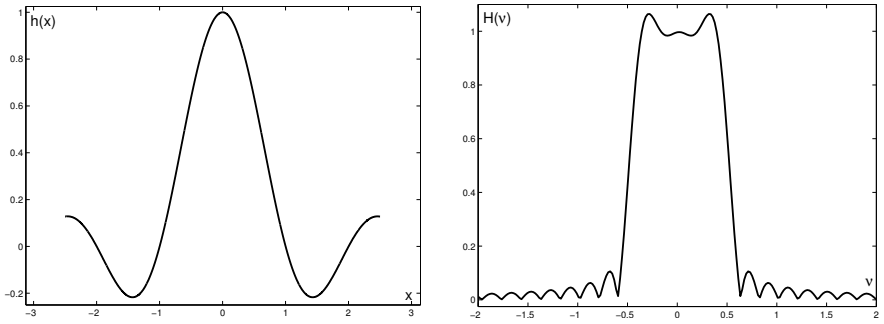
6.1.3.4 B-spline interpolation

Other linear methods use higher order (piecewise) polynomials [Lehmann et al., 1999, Meijering et al., 2001] or B-splines [Unser et al., 1993a,



(a) Bilinear kernel (b) The Fourier spectrum

Figure 6.11: The bilinear interpolator and its Fourier spectrum.



(a) Truncated sinc kernel ($N = 5$) (b) The Fourier spectrum

Figure 6.12: A non-ideal interpolator and its Fourier spectrum.

Unser et al., 1993b]. *B-splines*, or basis splines, are a special type of piecewise polynomials and are one of the most commonly used family of spline functions. The basis function is the rectangular function $\Pi(x)$ (equation (6.7)). This is the zeroth-degree B-spline, and is denoted $\beta^0(x)$. Higher degree B-splines are constructed using self-convolution [Unser, 1999]:

$$\beta^n(x) = \underbrace{\beta^0 * \beta^0 * \cdots * \beta^0(x)}_{n+1 \text{ times}} \quad . \tag{6.12}$$

We can explicitly write down the definition of a B-spline of degree n:

$$\beta^n(x) = \frac{1}{n!} \sum_{k=0}^{n+1} \binom{n+1}{k} (-1)^k \left(x - k + \frac{n+1}{2} \right)_+^n \quad , \tag{6.13}$$

where $\binom{a}{b}$ is a *binomial coefficient*[12] and the *unit step function* $(x)_+^n$ is defined as:

$$(x)_+^n = \begin{cases} x^n, x \geq 0 \\ 0, \quad x < 0 \end{cases} \quad . \tag{6.14}$$

It is also possible to construct the B-splines recursively [Unser et al., 1993a]:

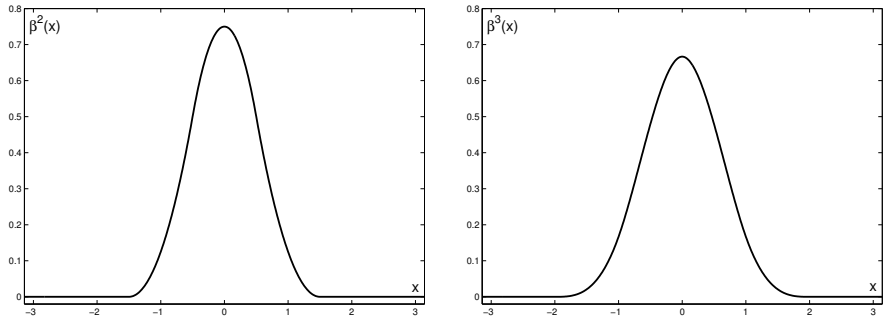
$$\beta^n(x) = \frac{\left[\left(\frac{n+1}{2} + x \right) \beta^{n-1} \left(x + \frac{1}{2} \right) + \left(\frac{n+1}{2} - x \right) \beta^{n-1} \left(x - \frac{1}{2} \right) \right]}{n} \quad . \tag{6.15}$$

So, β^0 is the reconstruction kernel used for pixel replication, but β^1 is also a known function, namely the triangular function (equation (6.10)) used for (bi)linear interpolation. The B-splines for $n = 2$ and $n = 3$ are shown in figure 6.13. They are called the *quadratic* and *cubic* B-spline, respectively. Note that they are not interpolators but approximators, since they do not satisfy condition (6.9). The cubic B-spline could be the (bi)cubic interpolation we mentioned before (subsection 6.1.3.2).

The Fourier transform of the B-spline $\beta^n(x)$ is of course $\text{sinc}^{n+1}(v)$. As a result, the interpolation with such a B-spline (approximator) displays stronger blurring effects and more attenuation of unwanted high-frequency components for a higher value of n.

It is possible to transform the B-splines to become interpolators instead of approximators (i.e., they satisfy condition (6.9)). These B-splines are called

[12] A binomial coefficient is: $\binom{a}{b} = \frac{a!}{(a-b)!b!}$.



(a) $\beta^2(x)$ (b) $\beta^3(x)$

Figure 6.13: The quadratic and cubic B-spline.

cardinal B-splines. The interpolation kernel is [Unser et al., 1993a]:

$$h(x) = \sum_{k=-\infty}^{+\infty} (b^n)^{-1}(k)\beta^n(x-k) \ . \tag{6.16}$$

$(b^n)^{-1}$ is a recursive pre-filter (or direct B-spline filter) [Unser et al., 1993b]. What is actually done is the following: the input image I_s is pre-processed with the direct B-spline filter, in order to get a new set of sample points. The next step is to apply the B-spline kernel, giving us the interpolated image:

$$I = \sum_{m=-\infty}^{+\infty} ((b^n)^{-1} * I_s)(m)\beta^n(x-m) \ . \tag{6.17}$$

The B-spline is convolved with an image with repositioned samples. The B-spline still behaves as an approximator, but on $(b^n)^{-1} * I_s$, not on I_s. On I_s, the basis spline is an interpolator.

These are very good interpolators: the shape of a cardinal B-spline is similar to that of the sinc. When $n \to \infty$, the cardinal spline converges to the sinc function [Aldroubi et al., 1992]. As a consequence, the rectangular shape of the Fourier transform is obtained for $n \to \infty$. The support of the cardinal B-spline is infinite, so we would need to truncate this function as with the sinc (see subsection 6.1.3.3), which leads to the Gibbs phenomenon. On the other hand, the B-splines (not the cardinal B-splines) β^n decay exponentially and their support is finite, as can be seen in figure 6.13. It is a major advantage that we can use these functions with finite support for interpolation, using equation (6.17).

Most of the linear techniques create a greyscale image with artefacts, like jaggies, blurring and/or ringing. The extra lobes in the stopband region cause

jagginess, as is for example the case with pixel replication, and in a lesser degree with bilinear interpolation. Blurring is a consequence of the attenuation of the high frequency information. Ringing artefacts in the Fourier domain are caused by the truncation of the sinc, and in the real domain by the frequency cut-off.

6.1.4 Non-linear interpolation techniques

Adaptive or *non-linear* interpolation methods incorporate prior knowledge about images to achieve better interpolation results. They are not as straightforward to implement as the convolution-based techniques, but they are able to avoid the previously mentioned types of artefacts (jaggies, blurring, ringing), although new kinds of artefacts might be introduced. We can define different classes, based on the method used.

6.1.4.1 Edge-based interpolation

Edge-based techniques follow the principle that no interpolation across the edges in the image is allowed or that interpolation has to be performed along the edges. Examples of these techniques are EDI (*Edge-Directed Interpolation*) [Allebach and Wong, 1996], NEDI (*New Edge-Directed Interpolation*) [Li and Orchard, 2001] and AQua (*Adaptively Quadratic interpolation*) [Muresan and Parks, 2004].

The EDI technique, for example, consists of two phases: rendering and data correction. The rendering is based on bilinear interpolation of the low-resolution image data. The quality is then improved by taking edge information into account: the bilinear interpolation is modified to prevent interpolation across edges.

6.1.4.2 Restoration-based interpolation

The so-called *restoration* methods use regularization methods or smoothing to limit interpolation artefacts. Some restoration methods use partial differential equations based (or PDE-based) regularization [Tschumperlé, 2002], isophote smoothing [Morse and Schwartzwald, 1998] and level curve mapping [Luong and Philips, 2005, Luong et al., 2005]. The goal of these techniques is to remove the artefacts obtained by interpolation with a linear technique.

Isophote smoothing, for example, is a geometry-based approach: the interpolation is performed by reconstructing geometric properties of the original image. As a first approximation, some standard linear interpolation technique is used. When examining the *level curves* of this interpolated image, we notice that their local spatial curvature is quite high. A level curve or isophote is a curve of pixels with the same intensity. The staircase patterns in a pixel-replicated image clearly show such high curvatures (see figure 6.14). Isophote smoothing

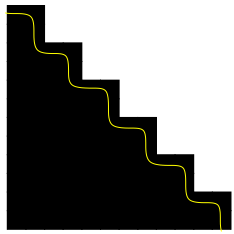


Figure 6.14: The isophotes (yellow line) of a pixel replicated image have a high spatial curvature.

removes the jagginess from the level curves, which results in a better magnified image (i.e., with less jaggies and blur).

The manipulation of the level curves can be done locally by using a curve evolution method [Morse and Schwartzwald, 2001].[13] The relation between the change of the image function I and the movement of the curve in the direction of its normal is:

$$I_t = -\kappa(\nabla I)||\nabla I|| \ , \tag{6.18}$$

where the isophote curvature κ is:

$$\kappa(\nabla I) = \mathrm{div}(\nabla I/||\nabla I||) = -\frac{I_x^2 I_{xx} - 2I_x I_y I_{xy} + I_y^2 I_{yy}}{(I_x^2 + I_y^2)^{3/2}} \ . \tag{6.19}$$

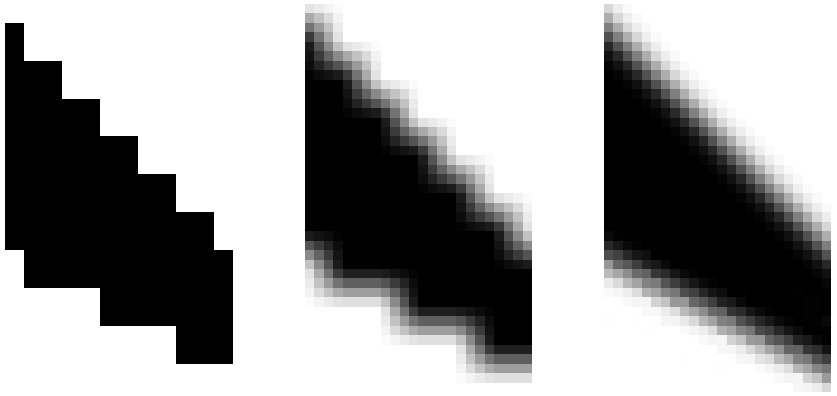
I_x is the derivative w.r.t. x, and $||\nabla I|| = \sqrt{I_x^2 + I_y^2}$. The curvature determines how much the pixel value must change. In practice, isophote smoothing is performed iteratively, until the level curve is completely smooth.

Figure 6.15 shows an example with a jagged line. The bilinear interpolation blurs the edge, but the jaggies are not completely removed. Isophote smoothing removes the staircase pattern.

6.1.4.3 Example-based interpolation

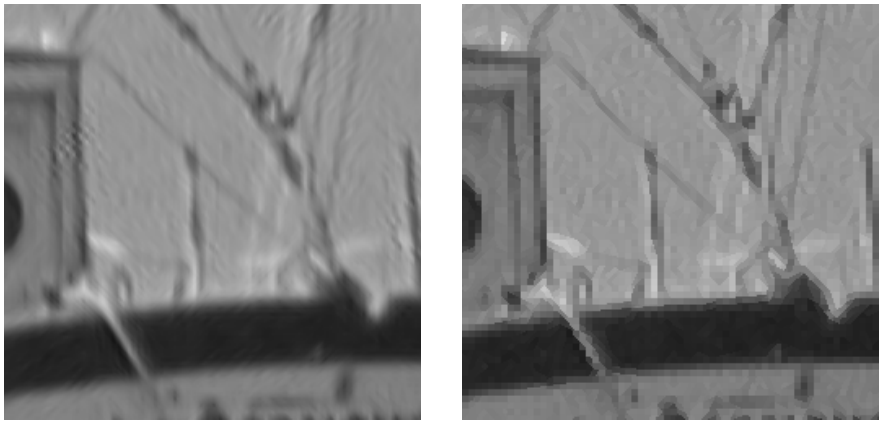
Example-based approaches are yet another class of adaptive interpolation methods. They map blocks of the low-resolution image into pre-defined interpolated patches [Stepin, 2003, Freeman et al., 2002]. The HQ technique [Stepin, 2003], for example, is a fast, high quality magnification filter. For every image pixel, the colour difference (using a lexicographical ordering in the YUV colour space)

[13] I.e., we do not need to know the exact path of the level curve, we can calculate how large the local curvature is.



(a) Pixel replication (b) Bilinear interpolation (c) Isophote smoothing

Figure 6.15: A line is 4× magnified and interpolated using (a) pixel replication, (b) bilinear interpolation and (c) isophote smoothing.



(a) Painting effect (b) Segmentation effect

Figure 6.16: Possible artefacts that appear when non-linear interpolation techniques are used.

of this pixel with its 8 neighbouring pixels is measured. The differences are compared to a pre-defined threshold value, and a classification is made. Pixels fall into the "close" or "distant" category. Because there are 8 nearest neighbours, 256 different classification combinations are possible. For each combination, a description how to change the input pixels is stored in a lookup table. HQ is implemented in different games emulators.

Some other adaptive methods exploit the *self-similarity* property of an image, e.g., methods based on iterated function systems (or fractal inter-

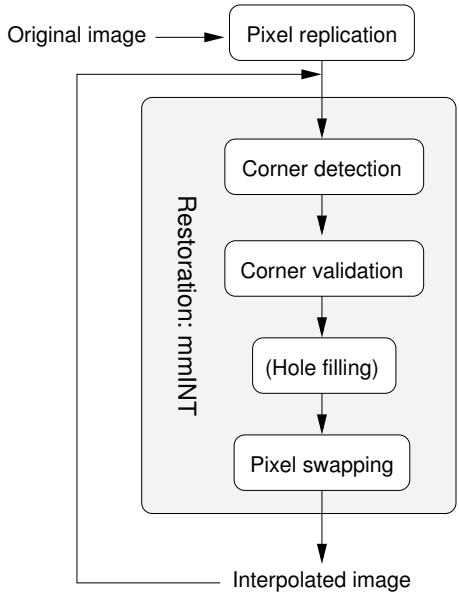


Figure 6.17: Schematic representation of the algorithm of mmINT.

polation) [Honda et al., 1999] or non-local interpolation [Luong et al., 2006b, Luong et al., 2006a].

While artefacts like jaggies, blurring and ringing can be avoided with adaptive techniques, new artefacts might be introduced. Figure 6.16(a) shows stripes or sweeps in the image, a painting effect that occurs when NEDI is used for the interpolation of the image. Also, random pixels are created in smooth areas. Sometimes the result looks segmented (figure 6.16(b)). This is the case when local binarization is used. Local binarization is a binarization procedure where the threshold is defined for every pixel, using the values of neighbouring pixels instead of all image pixels. Results can also suffer from important visual degradation in finely textured areas.

6.2 Morphological interpolation: mmINT

In this section we present our own original non-linear interpolation technique for black-and-white images. As we will see in section 6.4, it performs better than several other techniques, both linear and non-linear.[14] Since mathematical morphology concepts are used in various steps of the algorithm, we will refer to our new method as *mmINT* (*Mathematical Morphological INTerpolation*).

The purpose of mmINT is to remove the jagged edges from a pixel-replicated

[14]Some of the results have been published in [Ledda et al., 2005] and [Ledda et al., 2006b].

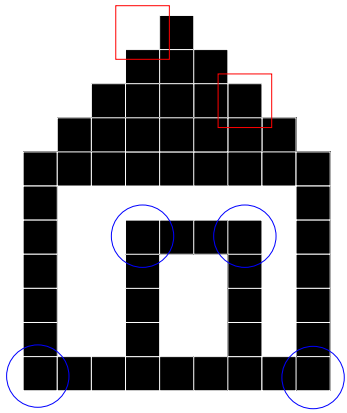


Figure 6.18: The difference between "jagged corners" (boxed) and "real corners" (encircled).

image, by changing specific pixels from the background colour to the foreground colour and vice versa. This will improve the visual result when we magnify a binary input image.

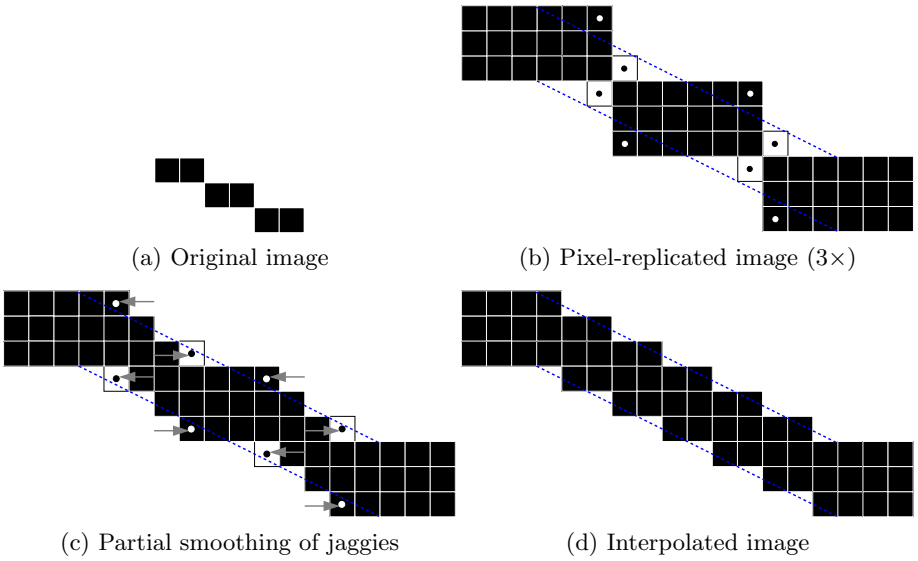
The method uses the concept of "background pixel", which we define as the most frequent colour in the image. A schematic representation of the algorithm is shown in figure 6.17. We summarize the different steps:

1. **Pixel replication** (section 6.2.1): First the image is pixel-replicated by an integer factor M. The resulting image contains strong staircase patterns because of the pixel replication (see the object in figure 6.3 for example).

2. **Corner detection** (section 6.2.2): Using a combination of hit-miss transforms, the algorithm determines the positions of corners, both real ones and false ones (due to jaggies) in the image.

3. **Corner validation** (section 6.2.3): Some corners found in the preceding step are *real corners*, which have to be retained in the interpolated image. For example, the corners of the door and walls of the house illustrated in figure 6.18 are real corners, because they belong there. The corners detected on the roof are *jagged corners*, because the ideal roof is a diagonal line which should not have jaggies. In a discrete image it is not possible to construct a perfect diagonal line, it will always be a staircase pattern, but we want to reduce the jagginess so that it becomes invisible (or at least undisturbing). The aim of corner validation is to distinguish false corners from real ones.

4. **Hole filling** (section 6.2.4): Some of the corners detected as jagged corners cause artefacts after interpolation. We will explain this type



(a) Original image (b) Pixel-replicated image (3×)

(c) Partial smoothing of jaggies (d) Interpolated image

Figure 6.19: The jagged edges have to be removed, by replacing the values of specific pixels. The dotted lines show the orientation of the original line (a). The dots show the pixels that will change value after interpolation.

of artefacts in section 6.2.4 and we will present a solution that removes them. This procedure is performed once, during the first iteration step.

5. **Pixel swapping** (interpolation) (section 6.2.5): We swap the colour of pixels classified as false corners, and the colour of some of their neighbours.

The above operations (except the 1st and the 4th one) are repeated iteratively (section 6.2.6), each iteration operating on the output image of the preceding iteration.

To illustrate our method, we consider the case of enlarging an object consisting of a thin line of foreground pixels (see figure 6.19). The result after pixel replication is shown in figure 6.19(b) and is clearly jagged. The dotted lines show the ideal boundaries of the magnified line.[15] The ideal solution would be to replace all (white) background pixels between the dotted lines with (black) foreground pixels and to replace all foreground pixels outside the dotted lines with background pixels. The iterative procedure aims at doing just that.

As we can see when we compare figure 6.19(b) to figure 6.19(c), jaggies can be removed by altering the pixel values at the locations of object and background corners. The positions of these corners can be located with the morphological

[15]One might note that we can also use the boundaries of the convex hull of the line. If we do that, the lines will look thicker than originally intended.

hit-miss transform, introduced in chapter 2. After the corner detection, which will be fully discussed in section 6.2.2, corners that are a geometrical feature of the image content (the so-called *real corners*) are detected, so they can be retained after interpolation. To know if a corner is a real or a *jagged corner*, we will introduce the concept of *complementary corners* in section 6.2.3. A complementary corner of an object corner c_o[16] is a corner in the background (or vice versa) that lies at specific relative coordinates with respect to c_o. We also take care of hole artefacts, which will be discussed in section 6.2.4.

In the pixel swapping step (see section 6.2.5), we change the value (0 or 1) of the pixels (and surrounding neighbours) detected as corners of a jagged edge. Which pixels exactly need to be changed, depends on the magnification factor, the iteration step and the *corner orientation* (i.e., upper-left (ul), upper-right (ur), lower-right (lr) or lower-left corner (ll)).

At this point, lines with an orientation other than $0°$, $\pm 45°$ or $90°$ are not yet completely smooth.[17]

In our example, figure 6.19(c) shows the result of the first iteration step of our method. The small arrows point to the next set of pixels that need to be changed. The orientations of the arrows also show how the pixel swapping evolves with respect to the originally detected corners: we start from a location of a (false) corner, but in every iteration step the corner moves away from its original position and the values of its surrounding pixels change. Therefore the algorithm is repeated until all appropriate changes have been made, using different (and larger) structuring elements in successive iteration steps for the hit-miss transform (i.e., the corner detection) and pixel swapping step. The final interpolation result of our example is shown in figure 6.19(d).

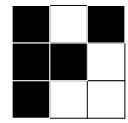
6.2.1 Interpolation by pixel replication

First, we magnify the image by a scaling factor (or magnification) M using *pixel replication* (also known as *nearest neighbour interpolation*). The only restriction on the scaling factor is that it has to be an integer value, the magnification can be as high as we want.

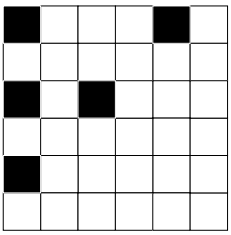
Pixel replication is a simple interpolation technique: every pixel of the low-resolution image is replaced by a square of $M \times M$ pixels with the value of the original pixel. If two pixels a and b are located at a distance of Δ pixels from each other, then they will be $M\Delta$ pixels away from each other, after pixel replication. As a result, a blocked pattern (jagged edges or *jaggies*) becomes visible (see the object in figure 6.3 for an example). Next, we will remove those disturbing jaggies and let the lines appear less jaggy, as schematized in figure 6.19.

[16] The corner c_o is defined by the pixel coordinates (x_{c_o}, y_{c_o}). This pixel is obtained in the corner detection step using the hit-miss transform.

[17] We consider a horizontal line in the image oriented $0°$.

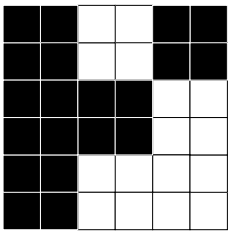


(a) Original image



(b) Pixels moved: $(x, y) \rightarrow (Mx, My)$

(c) Structuring element



(d) Dilation of (b) by (c)

Figure 6.20: Pixel replication (with magnification $M = 2$) using mathematical morphology.

Pixel replication can be defined in terms of a mathematical morphological operation (see figure 6.20): when the magnification is M, every object pixel (x, y) (figure (a)) is first moved to a new location (Mx, My) (figure (b)), while all other pixels are set to background. A binary dilation by a square structuring element (figure (c)) then results in the pixel-replicated image (figure (d)). We assume that the origin of the coordinate system of the image is at one of the corners of the image. This convention also applies to the structuring element. The side of the structuring element is M pixels.

6.2.2 Corner detection

If we examine figure 6.19(b), we notice that the jaggies can be removed by removing or adding some pixels from or to the image set. These pixels are respectively at the locations of object and background corners. In the figure they are marked with a dot. We need the positions of these corners in order to remove the jaggies.

The morphological *hit-miss transform* is a very useful tool for corner detection. We have introduced this operator in chapter 2, section 2.4.4. The hit-miss result is a set containing the positions where the desired shape is present. Since we will look for corners, we will further refer to this set as a *corner map*. The hit-miss transform needs two structuring elements, one to erode the set of object pixels and one to erode the set of background pixels.

(a) B (foreground)

(b) C (background)

Figure 6.21: Upper-left corner detection with the hit-miss transform. Specific structuring elements are used. The black squares are pixels of the structuring element; the cross is the origin of the structuring element.

Figure 6.21 shows the structuring elements B and C used for the detection of an upper-left corner. Other variations are also possible: B (figure 6.21(a)) could be replaced by a square element; C (figure 6.21(b)) could be substituted by a similar but more strict element with an extra pixel at the upper-left position; etc. We have chosen the structuring elements B and C as shown in figure 6.21, since they are compact, they produce the desired result and are not too restrictive, i.e., they detect all corners that are possible jagged corners. In section 6.2.4 we state some extra remarks about the choice of structuring elements.

The other three corners (upper-right, lower-right and lower-left) are detected using rotated versions of these elements.

We not only look for corners of the objects, but also for corners in the background. For the detection of object corners, we use the structuring element B for the erosion of the foreground and structuring element C for the erosion of the background. For the detection of background corners, the hit-miss transform is performed on the complement of the image (i.e., we use structuring element B for the erosion of the background and structuring element C for the erosion of the foreground). This way we will have a total of 8 corner maps (4 *object* corner maps and 4 *background* corner maps).

6.2.3 Corner validation

Not all corners detected by the method described in section 6.2.2 need to be changed. Some corners are *real corners*, which have to be retained in the interpolated image. We have already explained that the corners of the door and walls in figure 6.18 are real corners, but the corners detected on the roof are *jagged corners*, because the ideal roof is a diagonal line without jaggies.

We have to determine which corner pixels should be transformed by the interpolation part (section 6.2.5) (the jagged corners) and which should be left intact (the real corners). To distinguish between both, we try to find for every detected corner pixel one or more *complementary corner pixels*.

A complementary corner of a corner is a (nearby)[18] corner of the opposite

[18]Initially, the complementary corners lie in a close neighbourhood of the corner pixel. As we will see in section 6.2.6, the distance between the corner and its complement can increase.

colour. The relative coordinates of this complementary corner are specifically chosen. The existence of a complementary corner indicates the presence of a jagged edge. The argumentation for this is the following: if a background corner pixel is changed to an object pixel (which is done in the pixel swapping step, section 6.2.5), then an object corner pixel must also change to background, in order to keep the total number of foreground pixels, i.e., the global image intensity, (quasi) constant. This global intensity is not exactly constant, since in some cases more than one complementary corner is found for a corner. Also, as we will discuss in section 6.2.5, when the magnification is even, the pixel swapping of object pixels is not the same as for background pixels.

We look in the direction of the jagged edge for a complementary corner, and/or we look in the direction across the edge for such complement, and/or we look inside a small window around the considered corner pixel. We will discuss this more thoroughly in the following subsections. In figure 6.19(b), for example, the corner pixels, both from the background and the foreground, are marked with a dot. These corners are each other's complement, at least the ones that are situated at a distance of no more than 3 pixels away from each other.

Complementary corners can be classified into two classes: complements that lie across the edge and complements that lie along the edge. If we find at least one of such complementary corners, then we state that the considered corner pixel is a jagged corner. If no complementary corner can be detected, the corner is deleted from its corner map. We will now discuss both classes in more detail.

6.2.3.1 CC I: Search for complements across the edges

Complementary pixels lie across or along a jagged edge. In this subsection we look for pixels that are positioned at the opposite side of a line. If we detect a complementary corner of this type, we label it as CC I (*Complementary Corner of type I*).

Specific neighbour based approach Suppose we have a jagged line such as in figure 6.19. In the original low-resolution image, this line has a thickness of 1 pixel. After pixel replication with magnification M this thickness is M pixels (3 in the example figure). With the corner detection step (section 6.2.2) we find several corner pixels, both from the background and the foreground. When the line is indeed thin and jagged, complementary (background) corners are detected for every object corner at a distance of M pixels at the other side of the line. This distance is measured either in a straight vertical or straight horizontal direction.

Whether the possible complementary corner is positioned at a vertical or horizontal distance depends on the orientation of the line. When the orientation β lies in the interval $45° < \beta < 135°$ (or $-135° < \beta < -45°$), the possible CC I pixel is located at M pixels in the horizontal direction. If $135° < \beta < 225°$

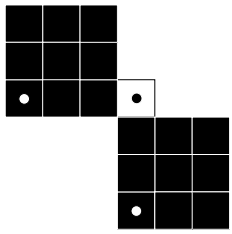


Figure 6.22: Complementary corners: the background corner (black dot) has two complementary corners (white dots) at specific relative coordinates.

(or $-45° < \beta < +45°$), the possible CC I pixel is located at M pixels in the vertical direction.

For lines with orientation $\pm 45°$, both directions are possible. Figure 6.22 shows part of a pixel-replicated line ($M = 3$) with orientation $-45°$. For such a line a background corner has two complementary corners (and vice versa), each across an edge. Note that the complementary corners are of the same type: in the figure they are all three lower-left corners.

To summarize: a CC I pixel has the same corner orientation as the corner under investigation. It is located M pixels away, either in the horizontal or vertical direction, at the other side of the jagged line. If at least one complementary pixel is found, then the corner pixel (and also its complement(s)) is classified as a jagged corner. If not, then this corner pixel is removed from its corner map.

Window based approach The above solution (i.e., the *specific neighbour based approach*) is elegant and efficient, because it only looks at two specific relative locations. An alternative (i.e., the *window based approach*), which uses the morphological dilation, is also possible.[19]

Again we suppose a pixel-replicated line with a thickness of 1 pixel in the low-resolution image that needs to be deprived from its jaggies. Figure 6.23 shows part of such a line ($\beta = -45°$, and $3\times$ magnified), with a background corner pixel (black dot) and the union of all the object corner maps (white dots) marked. We take the intersection of this union map and the dilation of the aforementioned background corner by a square structuring element B (the window in the figure). If this intersection is not empty, then the corner is a jagged corner. Otherwise, the background corner is removed from its corner map. In other words, if we find at least one object corner inside an area around the considered background corner pixel, then this background corner is considered to be a jagged corner. The same approach is taken for the object corners.

[19]This search technique is less efficient than the previously explained one, since it checks more neighbouring pixels. We mention it here, because our first implementations of MMINT search for CC I pixels this way.

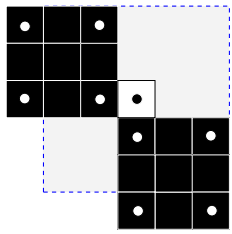


Figure 6.23: Complementary corners in a window with size $2M-1$ (the area inside the dotted box). Black dot: background corner; white dots: object corners.

We can formulate this mathematically for every pixel $\mathbf{r}$. Here, we do this for the search of complements of a lower-left background corner pixel:

$$\begin{aligned} A_{ll}^{bg}(\mathbf{r}) \equiv\ & (C_{ll}^{bg}(\mathbf{r}) \oplus B) \cap (C_{ul}^{fg} \cup C_{ur}^{fg} \cup C_{lr}^{fg} \cup C_{ll}^{fg}) \\ \Rightarrow & \begin{cases} C_{ll}^{bg}(\mathbf{r}) = 0, \text{if } A_{ll}^{bg}(\mathbf{r}) = \emptyset \\ C_{ll}^{bg}(\mathbf{r}) = 1, \text{if } A_{ll}^{bg}(\mathbf{r}) \neq \emptyset\ . \end{cases} \end{aligned} \tag{6.20}$$

The sets C_p^q are the corner maps, with p the type of corner (i.e., its orientation), and q foreground or background. Similar equations can be formulated for the other corner maps.

The size of the structuring element B is $(2M-1) \times (2M-1)$, with M the desired magnification, and the origin in the centre of the square. Figure 6.23 illustrates this. The size of the window (i.e., the square structuring element) is chosen such that the upper-left pixel and the lower-right pixel in the figure are not covered by this window. This ensures that, if one of these pixels indicates a real corner, this pixel will indeed be detected as a real corner.

The two discussed search methods look for CC I pixels and classify the considered pixel as a jagged corner if at least one CC I pixel is found. It works only for thin line drawings, since the algorithm looks for complementary corners in the direct neighbourhood of the pixel. For solid objects, a different method, discussed in subsection 6.2.3.2, has to be used.

The specific neighbour based approach and the window based approach do not find the same complementary corners. Indeed, when we compare figure 6.22 to figure 6.23, we see that both methods detect different complements. With the former method we detect (per corner pixel) two complements at most, with the latter we can detect up to four complements. This is only true in the first iteration step.

6.2.3.2 CC II: Search for complements along the edges

Complementary corners can also be found along the edge of an object (instead of across). If at least one complementary corner is found, the corner under

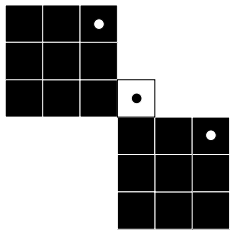


Figure 6.24: Complementary corners: the background corner (black dot) has two complementary corners (white dots) at specific relative coordinates.

investigation is classified as a jagged corner. If not, then the corner is removed from its corner map.

Suppose we have a jagged edge, which may be the border of a solid object.[20] Figure 6.24 shows a part of a diagonal line (orientation $\beta = -45°$), 3× magnified. A detected lower-left background corner is marked with a black dot. The search for *Complementary Corners of type II*, CC II, is performed along the edge. In the figure these complements are marked with a white dot. These corners are upper-right object corners. This can be generalized:

$$\begin{aligned}
&\text{Corner: upper-left} &&\Rightarrow \text{CC II: lower-right,}\\
&\text{Corner: upper-right} &&\Rightarrow \text{CC II: lower-left,}\\
&\text{Corner: lower-right} &&\Rightarrow \text{CC II: upper-left,}\\
&\text{Corner: lower-left} &&\Rightarrow \text{CC II: upper-right.}
\end{aligned} \tag{6.21}$$

We can specify the location of possible complements by relative coordinates to the examined pixel:

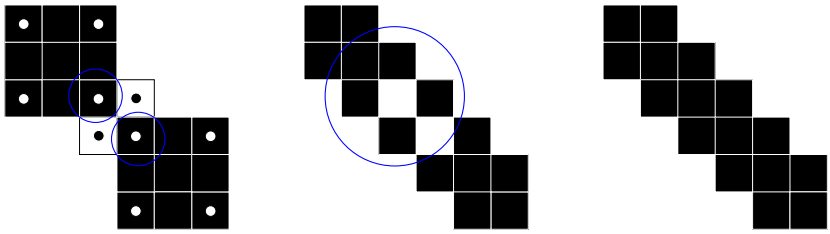
$$\begin{aligned}
M - 1 &\text{ pixels in one direction,}\\
1 &\text{ pixel in the other direction,}
\end{aligned} \tag{6.22}$$

where M is the desired (integer) magnification. The direction and sense depend on the corner orientation. In section 6.2.6.2 we generalize this equation for the higher iteration steps.

6.2.3.3 Combination of CC I and CC II searches

We can combine the search for CC I and CC II. This way we smooth edges and lines by looking for complementary corners across a thin line, as well as

[20] When we search for CC I pixels, we will only find them if the edge belongs to a thin line. If we look for CC II pixels, the thickness of the line plays no role, since we search along the edge.



(a) Before interpolation (b) Without hole filling (c) With hole filling

Figure 6.25: Barely touching pixels (encircled in figure (a)) can give artefacts after interpolation. Hole filling prevents this.

along an edge. The combination is straightforward: if we find at least one complementary corner, CC I or CC II, then the considered corner pixel indicates a jagged edge or line and is retained in its corner map.

This combination has the only disadvantage that (at most) twice as much pixels must be investigated to verify the existence of a complementary corner.

6.2.4 Hole filling

After the previous algorithm steps, some of the corners detected as jagged corners are not wanted, because they cause artefacts. We will now explain this type of artefacts and we will show how to remove them. This procedure is performed once, during the first iteration step.

When we have a line where some pixels are only point-connected, i.e., a line with pixels connected only in 8-connectivity (figure 6.25(a)), then some of the detected corners are unwanted. The encircled pixels in the image are detected as object corners, but their neighbours are also detected as background corners. If all these corners are validated as jagged corners, then hole artefacts are introduced after the pixel swapping (figure 6.25(b)). This hole filling step removes certain corner pixels from their respective corner map, so that the interpolation result will look good (figure 6.25(c)), i.e., without unwanted holes.

The circumstances in which these artefacts appear, depend on the search techniques used in section 6.2.3. If we use the window based approach, then all situations with 8-connectivity lines, as shown in figure 6.25(a), will introduce artefacts. When we use the specific neighbour based approach (CC I or CC II), then not all the corner pixels adjacent to the considered corner pixel are detected as complement, but only corners at specific relative locations (see the previous section). Hole artefacts then only appear at V-shaped lines. Figure 6.26 shows such a shape. The pixel with the white dot is a validated object corner pixel, because there is a CC I pixel (the most left pixel marked with a black dot), as well as a CC II pixel (the other pixel with a black dot). If this

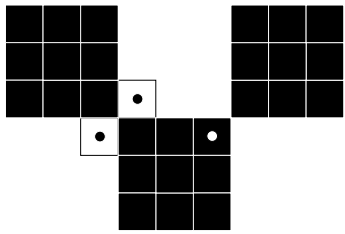


Figure 6.26: A V-shaped line. The pixels marked with a black dot are complementary corners of the pixel marked with a white dot (the left most one is CC I, the uppermost one is CC II).

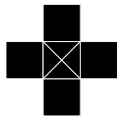


Figure 6.27: The structuring element for the hole filling.

object pixel is removed in the interpolation step (section 6.2.5), then a hole appears. This is illustrated in figure 6.25(b).

We have to remove these (object) corner pixels from their corner map, obtained in section 6.2.3. Suppose we want to filter out the upper-left corners that cause artefacts. We take the union of the pixel-replicated image A with all the background corner maps C_p^{bg} (with $p = \{ul, ur, lr, ll\}$). In our example, figure 6.25(a), this is the set of the black pixels and the pixels with the black dots. Within this union, we look for set elements that form a cross shape, like in figure 6.27. The choice for this cross-shaped element comes from the shape of the artefact: in figure 6.25(b) we encircled a hole and its surrounding pixels. The shape is a cross. If we find such cross-like elements, then the pixels at these positions are not allowed to change during the interpolation sequence. We can define the updated upper-left (object) corner map as:

$$C_{ul}'^{fg} = C_{ul}^{fg} \setminus \left(\left(\bigcup_p C_p^{bg} \cup A \right) \ominus B \right) . \tag{6.23}$$

The structuring element B is shown in figure 6.27. Similar equations can be formulated for the other foreground corner maps.

This part of the algorithm introduces an asymmetry between the black and white pixels: the object and background pixels are not treated in the same way. The hole filling explained with equation (6.23) is only performed on the object corner maps.

It is therefore important to know which pixels represent the background and which ones the foreground. If we define the wrong colour as background, the

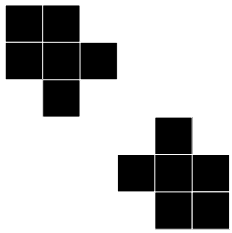


Figure 6.28: Hole filling produces wrong results when the wrong colour is chosen as background.

interpolation result looks quite different. This is illustrated in figure 6.28, which is the interpolation result of figure 6.25(a) when black is considered background instead of white (figure 6.25(c)).

We define the background colour as follows: we count the white and the black pixels in the image and the colour that is most present is considered background, which indeed is mostly the case.[21] The hole filling step adds more foreground pixels than background pixels to the magnified image, since now some object corner pixels will not be swapped in the next step of the algorithm.

6.2.4.1 Alternative hole filling

The removal of the hole artefacts can be accomplished in an alternative way, without the inclusion of the hole filling step in the algorithm. The solution is to use other (more strict) structuring elements for the corner detection in section 6.2.2. The asymmetry issue is still present, but now in the form of the used structuring elements for the hit-miss transform.

We still use the structuring elements shown in figure 6.21 for the detection of the background corners, so the same background corners in our example figure 6.25(a) will be detected (the pixels marked with a black dot). For the alternative hole filling, we use the structuring elements shown in figure 6.29 for the object corner detection. This way we exclude the encircled object pixels in figure 6.25(a) as object corners during the corner detection step itself, but still detect the other object corners (the pixels marked with a white dot). As a result, the hole artefacts will not become present at the 8-connectivity locations.

We have to mention that in some rare situations the interpolation results between the two hole filling approaches can differ. For example, figure 6.30(a)

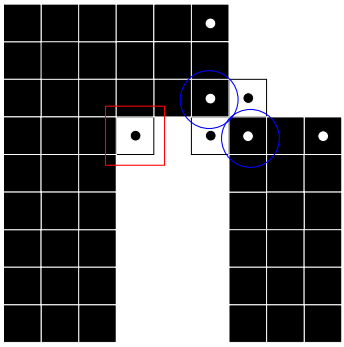
[21] Most binary images contain some objects on a background that occupies more pixels than the objects. Exceptions are: large solid objects (relative to the image resolution) or large objects with a lot of texture, a symmetrical texture such as a checkerboard pattern, two identical image parts that are each other's complement, etc.

In these situations, the definition of foreground and background could be done arbitrarily or locally. The latter principle is used in section 6.5 where we discuss a greyscale extension of mmINT.

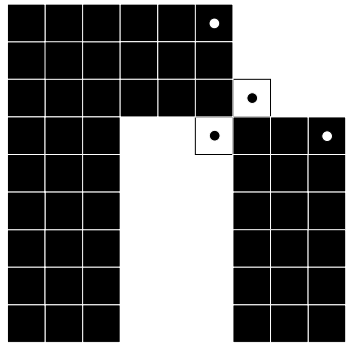
(a) B (foreground)

(b) C (background)

Figure 6.29: Alternative structuring elements for the hit-miss transform for an upper-left corner detection.



(a) Hole filling approach



(b) Different structuring elements

Figure 6.30: On some rare occasions, different results are obtained if the hole filling step is replaced by a corner detection using different structuring elements for foreground and background.

shows a pixel-replicated image where the detected and validated corner pixels are marked with a dot. In figure (b), the alternative approach is used, i.e., another set of structuring elements is used for the detection of the object corners as for the background corners. After the hole filling step, the blue encircled corner pixels in figure (a) are removed from their corner map, but they make it possible to validate the pixel in the red rectangle as a jagged corner. This is not the case with the alternative approach. These subtle differences can improve the interpolation of text.

6.2.5 Pixel swapping

We have first magnified an image using pixel replication (section 6.2.1). Then we have detected jagged edges and took care of hole artefacts (sections 6.2.2, 6.2.3 and 6.2.4). We now replace some specific background pixels with foreground pixels, and vice versa.[22] The pixel swapping is the actual interpolation step, since the previous steps merely perform magnification, corner detection and validation, while this step actually alters image pixels so the image would

[22]In other words, if a pixel $\mathbf{r}$ has value $v(\mathbf{r})$ (which is 0 or 1), its swapped value is $v'(\mathbf{r}) = 1 - v(\mathbf{r})$.

(a) Before interpolation (b) After interpolation

Figure 6.31: Pixel swapping for $M = 3$. The detected corner pixels (marked with a dot) are changed from foreground to background, and vice versa.

look not jagged. Not only the detected corner pixels are affected, but also some of their neighbouring pixels, depending on the used magnification.

Figure 6.31 shows a diagonal line, pixel-replicated with magnification $M = 3$. In figure (a), the jagged corners are marked with a dot (white for object corner pixels and black for background corner pixels). The pixel swapping (figure (b)) changes the value of these pixels to their opposite colour, i.e., black becomes white and white becomes black.

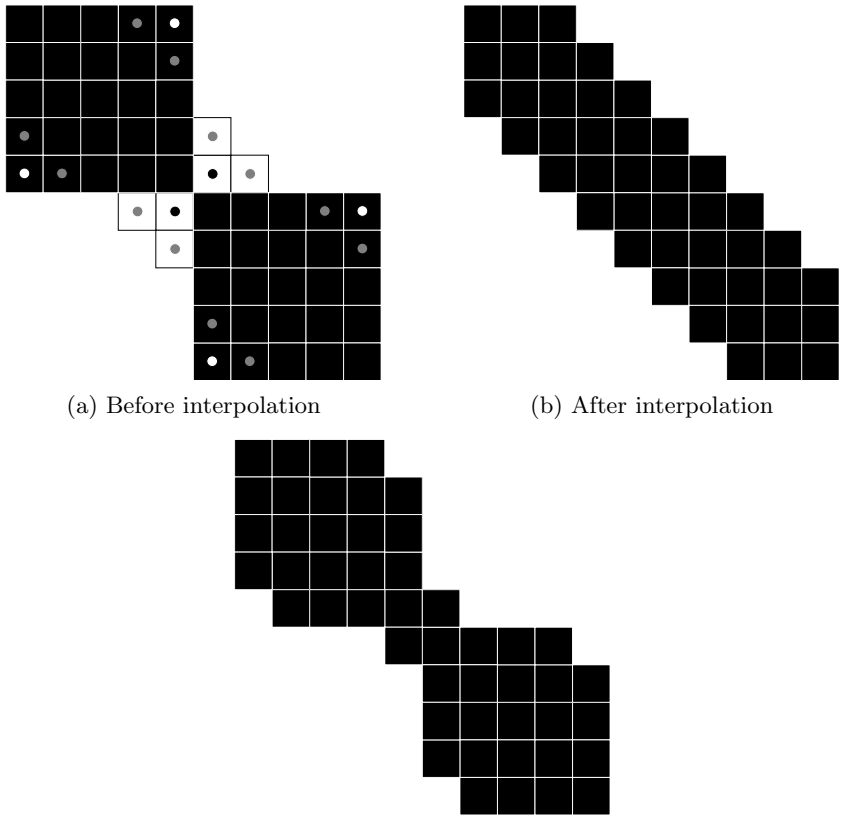
Figure 6.32 shows the same diagonal line, but now pixel-replicated with $M = 5$. To get rid of the jaggies, the change of only the corner pixels is not enough (figure 6.32(c)). Also some of their neighbouring pixels must be swapped. These extra pixels are marked with a grey dot in figure (a). The desired interpolation result is shown in figure (b).

The principle of the interpolation step is to perform a morphological dilation by a specific structuring element on the detected (and validated) corner pixels, treating each corner map separately. The result is used as a mask. The pixels in the mask with value 1 will be swapped from black to white, or from white to black. The mask A_p^q is thus:

$$A_p^q = C_p^q \oplus nB \ , \tag{6.24}$$

with B a base structuring element that resembles a corner (figure 6.33(b)). nB is defined in chapter 2, equation (2.58). q is either fg or bg, and $p = \{ul, ur, lr, ll\}$. The structuring element shown in figure 6.33(b) is used for an upper-left corner. Rotated versions are used for the other corner orientations. The value of n is discussed in the following subsections.

All interpolation structuring elements are designed to transform the jagged edges or lines into a staircase pattern with a step size of one pixel. When we look at figure 6.19(b), we notice a staircase pattern with steps or *plateaus* of 6 pixels long and 3 pixels high. Our goal is to transform these plateaus into a staircase pattern as shown in figure 6.19(d), namely 2 pixels long and 1 pixel



(a) Before interpolation

(b) After interpolation

(c) After interpolation, only corners

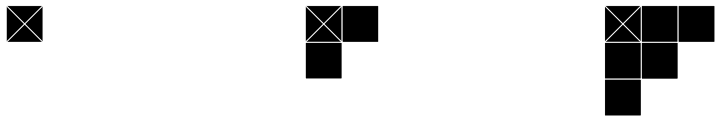
Figure 6.32: Pixel swapping for $M = 5$. The detected corner pixels (marked with a black or white dot) and surrounding pixels (marked with a grey dot) are changed from foreground to background, and vice versa. Figure (c) illustrates the result if only the corner pixels are swapped.

high. There are differences between each iteration step and also between odd and even magnifications.

6.2.5.1 Magnification by an odd factor

We dilate every corner map n times with its respective base structuring element (figure 6.33(b) is the structuring element for the upper-left corners). In the case of an odd magnification $M > 1$ we get:

$$n = \left\lfloor \frac{M}{2} \right\rfloor - 1 = \frac{M-3}{2} \; . \tag{6.25}$$



(a) $M = 3$ (b) $M = 5$: the struct. elem. (c) $M = 7$

Figure 6.33: Not only the corner pixel value (cross) will change its value in the interpolation part. Depending on the magnification, also neighbouring pixels change.

This value is experimentally determined, by investigating the interpolation behaviour. We pursue the smallest possible (and thus least visible) staircase pattern. The resulting structuring element nB is shown in figure 6.33 for $M = 3$, $M = 5$ and $M = 7$.

The result is an expanded corner map A_p^q (equation (6.24)) that is used as a mask that specifies which pixels must change value. For an object corner map, the mask defines the pixels that become background. For a background corner map the opposite is true.

So, at magnification 3 the created mask will be identical to the corner map, because the corner map is not changed by the dilation operation.[23] At magnification 5 also two neighbours (the shape of the structuring element) will be added to the mask. The values of the pixels in the obtained mask are swapped. We have already shown interpolation examples for $M = 3$ and $M = 5$ in figures 6.31 and 6.32, respectively.

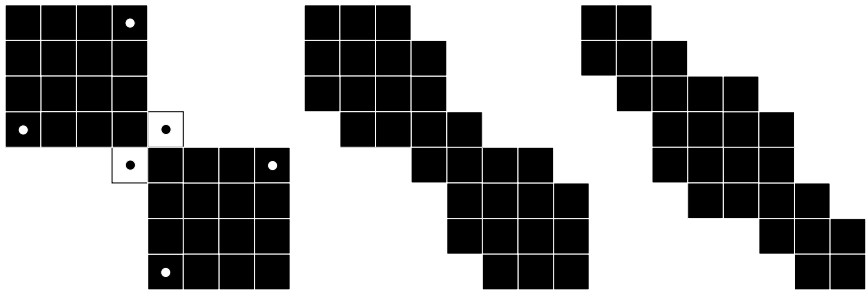
6.2.5.2 Magnification by an even factor

For magnification by an even factor the principle is the same, but the object corner maps are treated differently from the background corner maps. Consider the $4\times$ magnified line in figure 6.34(a). The detected jagged corners are marked with a dot. If we use $n = 0$ for background and foreground, we get figure 6.34(b) as a result. For $n = 1$ we get figure 6.34(c). The interpolated line is in both cases an oscillating one. Either we removed too few background pixels, or too many. A similar remark can be made for the object pixels. The correct interpolation is shown in figure 6.35, where more background pixels are swapped to object than vice versa.

We must use different structuring elements (i.e., n values) for the pixel swapping of the foreground as for the background. As a consequence, there will be more or fewer object pixels in the image after interpolation. We choose to add more foreground pixels in the first iteration step.

The value n that defines the size of the structuring element, and thus the mask

[23] Remember that $0B = \mathbf{0}$, so $A \oplus 0B = A$.



(a) Before interpolation (b) After, with $n = 0$ (c) After, with $n = 1$

Figure 6.34: Pixel swapping for $M = 4$. If both the background and foreground corners use the same n value, undesired results are obtained.

A_p^q, is:

$$n = \left\lfloor \frac{M}{2} \right\rfloor - 1 = \frac{M-2}{2} \text{ for the background,} \tag{6.26}$$

$$n = \left\lfloor \frac{M}{2} \right\rfloor - 2 = \frac{M-4}{2} \text{ for the objects.} \tag{6.27}$$

Magnification 2 is a special case, because n, as defined in equation (6.27), can have value -1. When this is the case, the (object) corner pixels will not change, i.e., we do not apply the interpolation as stated in equation (6.24).[24]

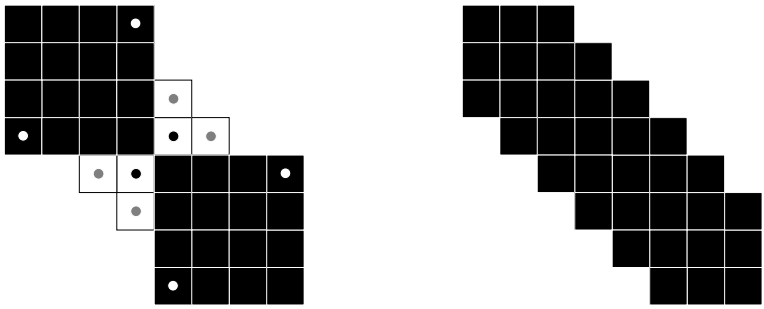
6.2.6 Higher orders

The former steps will interpolate lines that are tilted $\pm 45°$ (and of course $0°$ and $90°$) correctly, but lines at other orientations are only partly interpolated. This can be seen in figure 6.19(c). The jaggies are only removed in the direct neighbourhood of the original corners. Therefore, in order to obtain better results (like figure 6.19(d)), we repeat the procedure from step 6.2.2 on (the corner detection), until all corner maps are empty.

For illustration purposes we always used straight lines, but as we will see in section 6.4 with the experimental results, mmINT interpolates curved lines as well. This is an intrinsic property of mmINT since the interpolation occurs locally at jagged corners, without any knowledge about the curvature of the line.

The structuring elements used in the corner detection and the pixel swapping sequence will be different in each iteration. We will now look at the differences with the first iteration.

[24] This is also true for odd magnifications: when $M = 1$, equation (6.25) results in $n = -1$, so we do not interpolate the image. This is logical, since for a magnification of $1\times$ no interpolation is needed.



(a) Before interpolation (b) After interpolation

Figure 6.35: Pixel swapping for $M = 4$. While the detected background corner pixels (marked with a black dot) and surrounding pixels (marked with a grey dot) are swapped, this is not the case for the surrounding pixels of the object corner pixels (marked with a white dot).

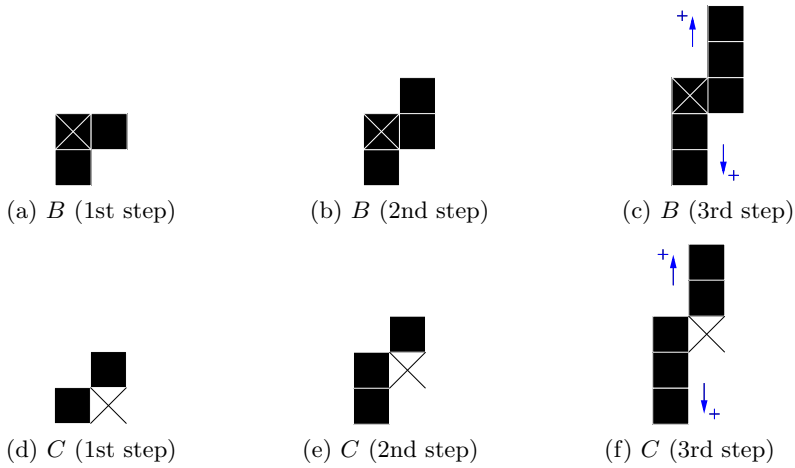
6.2.6.1 Corner detection

As we can see in figure 6.19, the corners are not only shifted, but the shape of the corners has also changed. In the first iteration step, the shape of the corners is defined by the structuring elements B and C in figure 6.21. After the first iteration, corner pixels (and neighbouring pixels) are swapped, and thus new corners are created. Most of these new corners must be kept, and we cannot re-use the same structuring elements from the first step, because this would remove them.

Therefore, the structuring elements for the corner detection part (section 6.2.2) have to be altered (see figure 6.36). These elements are chosen such that they gradually, with every iteration step, improve the smoothness of the jagged line, starting from the original jagged corners and then moving further away (as shown in figure 6.19). In the first iteration step there are 4 corner orientations, but from step 2 onwards the structuring elements are less symmetric (as they are more elongated), which implies now 8 different corner orientations (three rotational variants and mirrored versions). The total number of corner maps (foreground and background) hereby increases from 8 to 16.

6.2.6.2 Corner validation

CC I If we use the window based approach to search for complementary corners, then we need to define a structuring element. In the first iteration step, as we explained in section 6.2.3.1, this is a square with side $2M - 1$ pixels, with M the magnification. In the iteration step $\tau + 1$, the detected corners are at another location than in step τ. This can be seen in figure 6.19. If we use the same structuring element for the higher orders ($\tau > 1$), we will not



(a) B (1st step) (b) B (2nd step) (c) B (3rd step)

(d) C (1st step) (e) C (2nd step) (f) C (3rd step)

Figure 6.36: The hit-miss structuring elements are different for every iteration step. The arrows and plus signs indicate that every iteration step an extra pixel is added to the structuring element at those locations.

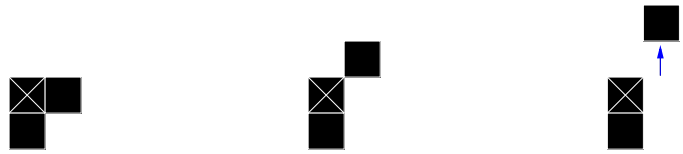
find any complementary pixel. Therefore, we change the size of the structuring element (the window) to $(2M+1) \times (2M+1)$ pixels. In figure 6.19(c), for example, where $M = 3$, we now find CC I pixels, which was not possible with the smaller window size.

This window size of $2M+1$ pixels is used for all $\tau > 1$, so it does not increase with τ. Complementary corners of this type (CC I), i.e., complements that lie at the other side of a thin line, move along every iteration. The distance between a corner and its complement therefore remains constant, namely M pixels horizontally or vertically. The reason why we use a window size of $2M-1$ pixels in the first iteration is to ensure that real corners are not considered as jagged corners (see section 6.2.3.1). We do not encounter this problem for $\tau > 1$.

Equation (6.20) needs a small update for $\tau > 1$: in the previous subsection we mentioned the increase in number of corner maps from 8 to 16. Equation (6.20) therefore becomes:

$$\begin{aligned} A^{bg}_{ll,1}(\mathbf{r}) \equiv\ & (C^{bg}_{ll,1}(\mathbf{r}) \oplus B) \cap (\textstyle\bigcup_{p,s} C^{fg}_{p,s}) \\ \Rightarrow & \begin{cases} C^{bg}_{ll,1}(\mathbf{r}) = 0, \text{if } A^{bg}_{ll,1}(\mathbf{r}) = \emptyset \\ C^{bg}_{ll,1}(\mathbf{r}) = 1, \text{if } A^{bg}_{ll,1}(\mathbf{r}) \neq \emptyset\ . \end{cases} \end{aligned} \tag{6.28}$$

The sets $C^{q}_{p,s}$ are the corner maps, with p the type of corner, s one of the two mirror versions of that type of corner, and q foreground or background. Similar equations can be formulated for the other corner maps.



(a) 1st step (b) 2nd step (c) 3rd step

Figure 6.37: The interpolation structuring element is different for every iteration step. The arrow indicates that every iteration step the pixel moves up one place at that location.

CC II In iteration step $\tau + 1$, the detected corners are at another location than in step τ. This can be seen in the example figure 6.19. This is true for CC I pixels, but also for CC II. The latter type of corner pixels move away from each other. By examining the motion of those pixels, we can generalize equation (6.22):

$$\begin{aligned} &M-1 \text{ pixels in one direction,} \\ &1+(\tau-1)(M-1) \text{ pixels in the other direction.} \end{aligned} \tag{6.29}$$

$\tau = 1, 2, \ldots$ is the iteration step. The direction and sense depend on the corner orientation.

A disadvantage is that in the result, some corners that are part of a jagged edge are not removed because the search rule is too strict. Indeed, while removing jaggies along an edge, the complementary pixels move away from each other. It is possible that one of these pixels reaches the end of the edge before the other one does. If at the next iteration step the latter corner is detected, no complementary pixel will be found anymore and the pixel will be considered a real corner, although jaggies might still be present. Therefore, a combination of CC I with CC II gives the best results, because CC I then might be able to validate corners as jagged ones when CC II cannot (or vice versa).

6.2.6.3 Pixel swapping

Also for the interpolation part other structuring elements are needed in steps $\tau > 1$ compared to step 1. They are shown in figure 6.37. There too the loss in symmetry implies 8 different orientations.

As we have treated magnifications by an odd factor differently from magnifications by an even factor in section 6.2.5, there is also a difference between odd and even iteration steps.

Magnification by an odd factor Equation (6.25) is used to define the value n (for nB in equation (6.24)) for odd magnifications. We can also use

this equation for every $\tau > 1$:

$$n = \left\lfloor \frac{M}{2} \right\rfloor - 1 = \frac{M-3}{2} \ . \tag{6.30}$$

Magnification by an even factor For even magnifications, on the other hand, there is a difference in treatment between the object and background corners: equations (6.26) and (6.27) are used for the pixel swapping in the first iteration step. In this first step, more background pixels are changed to foreground than the opposite. Therefore we revert the situation for the second iteration step: more object pixels are changed to background than the opposite. The third iteration step we revert again, etc. This way we can keep the global intensity quasi constant.

For the *odd iterations* (for an *even magnification*) we have:

$$n = \left\lfloor \frac{M}{2} \right\rfloor - 1 = \frac{M-2}{2} \text{ for the background,} \tag{6.31}$$

$$n = \left\lfloor \frac{M}{2} \right\rfloor - 2 = \frac{M-4}{2} \text{ for the objects.} \tag{6.32}$$

For the *even iterations* (for an *even magnification*) the situation is reversed:

$$n = \left\lfloor \frac{M}{2} \right\rfloor - 2 = \frac{M-4}{2} \text{ for the background,} \tag{6.33}$$

$$n = \left\lfloor \frac{M}{2} \right\rfloor - 1 = \frac{M-2}{2} \text{ for the objects.} \tag{6.34}$$

Magnification 2 is a special case, because n can have value -1. When $n = -1$, no pixel swapping is done on those corner pixels (either object or background corner pixels, depending whether equation (6.31) or (6.34) results in $n = -1$).

6.2.7 Optimizations

Several options are available to improve the speed of the algorithm and the quality of the interpolation results.

- The first optimization is the localization of the possible corners in the next iteration. These locations can be calculated from the locations in the previous step using the magnification and iteration step, and knowing what kind of corner (which orientation and whether or not mirrored) the pixel is. Corner pixels (and neighbouring pixels) are swapped. Possible new corner pixels will emerge along the edge, but we know in which direction to search.

Because the odd and even magnifications are treated differently in the pixel swapping step, the treatment here is also different for odd and even magnifications.

For *odd magnifications* the pixel displacement is:

$$\left\lfloor \frac{M}{2} \right\rfloor = \frac{M-1}{2} \ . \tag{6.35}$$

The direction of the displacement depends on the kind of corner. We have the following possibilities:

$$\begin{aligned} &\text{Corner: upper-left} &&\Rightarrow \text{New: right or down,} \\ &\text{Corner: upper-right} &&\Rightarrow \text{New: left or down,} \\ &\text{Corner: lower-right} &&\Rightarrow \text{New: left or up,} \\ &\text{Corner: lower-left} &&\Rightarrow \text{New: right or up.} \end{aligned} \tag{6.36}$$

If the *magnification* is *even*, then the displacement is, for *odd iterations*:

$$\left\lfloor \frac{M}{2} \right\rfloor = \frac{M-0}{2} \text{ for the background,} \tag{6.37}$$

$$\left\lfloor \frac{M}{2} \right\rfloor - 1 = \frac{M-2}{2} \text{ for the objects.} \tag{6.38}$$

If the *magnification* is *even*, then the displacement is, for *even iterations*:

$$\left\lfloor \frac{M}{2} \right\rfloor - 1 = \frac{M-2}{2} \text{ for the background,} \tag{6.39}$$

$$\left\lfloor \frac{M}{2} \right\rfloor = \frac{M-0}{2} \text{ for the objects.} \tag{6.40}$$

The direction of the displacement is as stated in equation (6.36).

In the next iteration step, we only perform a hit-miss transform on these pixels for the corner detection. This reduces the calculation time, because without this knowledge we would have to scan the entire image again.[25] This also takes care of possible artefacts: two regions that are interpolated independently (i.e., pixels are swapped and corners move every iteration step) and that meet each other can create pixels that accidentally satisfy the hit-miss condition. These pixels are now excluded, and only pixels that are expected to be possible corners will be investigated. This procedure also ensures that the algorithm converges.

- In the non-optimized version of the algorithm, there are 16 different corner maps with each the size of the (magnified) image. These maps

[25] The optimization suggestion in the next paragraph is closely related to this suggestion.

are stored as binary images, and only at pixels with value 1 some operation[26] is performed. If only these 1-pixels are stored in lookup tables that contain their coordinates, the image does not need to be scanned completely, which means a further reduction of the calculation time (and also of computer memory).

- Another optimization option has already been mentioned: the hole filling solution is incorporated in the corner detection part, using different structuring elements for the creation of the foreground corner maps as for the background corner maps. This also improves the interpolation of some characters in text.
- Of course it is also possible to stop repeating the algorithm after a certain number of iteration steps, but this implies a loss in interpolation quality. This option gives much freedom to the user: quick results can be obtained when only calculating one or two iterations. An "ideal" result is possible, but takes more time to calculate.

6.2.8 Further discussion

The interpolation method mmINT is a technique based on mathematical morphology that works very well on binary images, like logos, cartoons and maps. A visual study and comparison with other interpolation techniques is presented in section 6.4.

mmINT can be classified as a restoration method (see section 6.1.4), because it improves a pixel-replicated image by removing the jagged edges. From another point of view, it is also an edge-based interpolation technique, because it specifically looks for corners of jagged edges.

We emphasize that errors in the original image, e.g. noise, will appear in the magnified image, and will be interpolated by mmINT. Isolated pixels that are not removed before using mmINT will show up as magnified blocks after interpolation. If this is undesirable, then first some pre-processing with a noise filter must be performed. If the original image does not look good, e.g., text that has been rasterized at a very low resolution, then the interpolated result will not be visually attractive, although the jaggies have been removed. In this respect, mmINT is not a restoration method, only regarding interpolation (i.e., removing jaggies from a pixel-replicated image).

6.3 Alternative implementation: mmINTone

In section 6.2 we described a new interpolation algorithm, mmINT, based on mathematical morphology. mmINT is an iterative procedure, where every iter-

[26] By some operation we mean the hit-miss transform in the corner detection step, the corner validation, and the swapping of pixel values (validated corner pixels and possibly surrounding pixels).

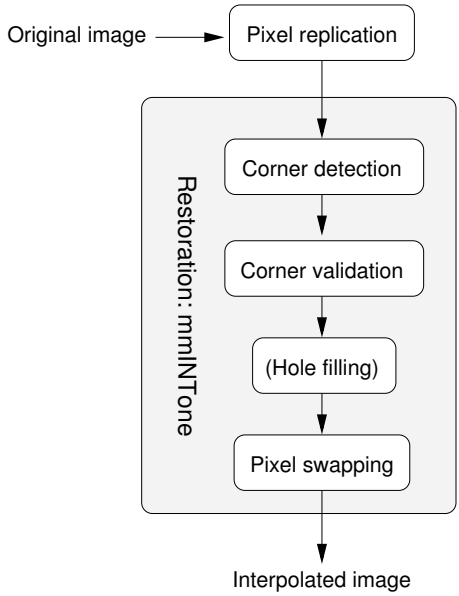


Figure 6.38: Schematic representation of the algorithm of mmINTone.

ation step more and more jaggies are removed. This can be a time consuming operation when the input image is large and/or when a lot of lines are present in the image with "unfavourable" orientations. By unfavourable we mean lines with orientations near (but not equal to) 0° or 90°, because these are jagged lines with long plateaus which need many interpolations before completely smoothened.

This section contains the description of a different approach to the same problem. The same results as with mmINT are pursued. The main difference is the fact that only one iteration step is needed: after the jagged corners are detected, all the appropriate pixels are swapped in one treatment, instead of several iteration steps. This algorithm is called *mmINTone* and is schematized in figure 6.38.

The algorithmic structure of mmINTone is almost identical to the one of mmINT. We distinguish the following steps:

1. **Pixel replication**: Section 6.2.1: First the image is pixel-replicated by an integer factor M. The resulting image contains strong staircase patterns because of the pixel replication.

2. **Corner detection**: Section 6.2.2: Using a combination of hit-miss transforms, the algorithm determines the positions of corners, both real and false (due to jaggies) in the image.

3. **Corner validation**: Section 6.2.3: Some corners found in the preceding

step are *real corners*, which have to be retained in the interpolated image. The aim of corner validation is to distinguish false corners from real ones.

4. **Hole filling**: Section 6.2.4: Some of the corners detected as jagged corners are not wanted, because they cause artefacts. We remove these artefacts. As we have discussed, the hole filling can be combined with the corner detection, using other structuring elements for the hit-miss transform.

5. ***New* pixel swapping** (*new* interpolation): Section 6.3.1: We swap the colour of pixels classified as false corners, and the colour of some of their neighbours. This step is different from the one in section 6.2.5. We discuss this in the next section.

Because we introduce a new pixel swapping step, the above operations do not need to be repeated iteratively.

The new pixel swapping part combines the different iterations (corner detection, corner validation and pixel swapping). We study the behaviour of the iterative mmINT and mimic it by changing all relevant pixels at once. For this, we (re-)introduce the concept of *plateaus*, i.e., the steps in the staircase pattern of the edges.

6.3.1 New pixel swapping for step 5

The procedure for pixel swapping is completely different from the procedure explained in section 6.2.5. The approach in mmINT is to change pixel values at the location of the detected jag corners. After that, the different steps for corner detection and corner validation are repeated in order to locate new jag corners. These corners will then be swapped. This iteration is repeated until all jaggies have been removed form the image.

With mmINTone, we swap the value of the appropriate pixels at once. The length of the *plateau* of a jagged line determines the number of pixels that need to change at that location. A plateau is a step in the staircase pattern. Its length is always longer (or equally long) than its height, which is always M pixels[27] for the pixel-replicated image, with M the magnification. In figure 6.19 for example, the plateaus in the pixel-replicated image are $2M$ pixels long. The height is indeed M pixels. Two object pixels are changed to background and two background pixels are changed to object, which results in M smaller

[27]The reason why the height of a plateau is always M pixels in the pixel-replicated image is the following: in the corner validation step we search for complementary corners at specific locations. These locations must be close enough to the considered corner, otherwise the corner is classified as a real corner. Real corners are not interpolated (pixel swapped), and thus plateaus with heights greater than M pixels will not be considered in the (new) pixel swapping step.

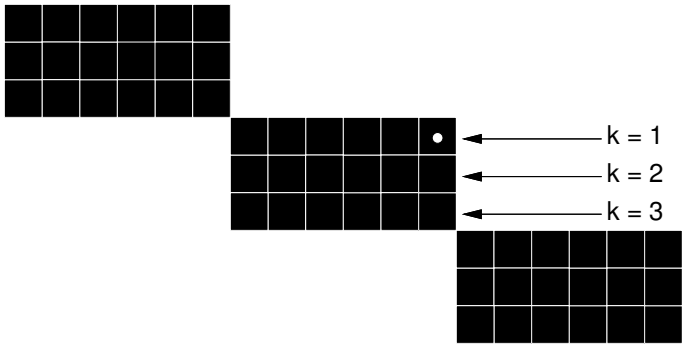


Figure 6.39: A plateau consists of several layers. The figure shows an $M = 3$ magnified image. The plateau is $2M$ pixels long.

plateaus with length $\frac{2M}{M} = 2$. This is generally true:

$$\begin{aligned} \text{length}(P) = nM &\Rightarrow \text{length}(P_i) = n \\ \text{height}(P) = M \ &\Rightarrow \text{height}(P_i) = 1 \end{aligned} , \tag{6.41}$$

with P the initial plateau (in the pixel-replicated image), n a strictly positive integer value, and P_i the plateaus that replace P after interpolation.

A plateau consists of several *layers* (see figure 6.39). The first layer is the one of the detected corner pixel (the white dot in the figure). The next layer lies below the first layer.[28] A plateau (and the layers) can be vertically oriented. Notice that a corner pixel can be part of two plateaus, a vertical one and a horizontal one. Since the height of the plateau is less or equal to the length of the plateau, this only occurs at edges with orientation $\pm 45°$.

How the pixels are swapped, is defined by the magnification, the length of the plateau, and a distinction is made between odd and even magnifications.

6.3.1.1 Magnification by an odd factor

Taking equation (6.41) into consideration, we can derive for each layer how many pixels need to change value. For an odd magnification M and k as the layer index, this number Δ_k will be:

$$\Delta_k = \frac{n}{2}(M - 2k + 1) \ . \tag{6.42}$$

n is a strictly positive integer value that defines the length of the plateau, nM. Index k is a strictly positive integer within a range that depends on the

[28] In figure 6.39, the following layer lies indeed *below* the current layer. More generally, we interpret "below" as a position *away* from the opposite colour of the plateau.

magnification:

$$\begin{aligned} 0 < k &\leq \tfrac{M-1}{2} \\ &\leq \left\lfloor \tfrac{M}{2} \right\rfloor . \end{aligned} \tag{6.43}$$

In the example illustrated in figure 6.39, $k = 1$ is the only layer where the pixel values change. With values $M = 3$ and $n = 2$ we can calculate that two pixels will change value, namely the detected corner pixel and its neighbour (with the same pixel value) on the same layer.

6.3.1.2 Magnification by an even factor

For an even magnification, the situation is a little more complicated. Just as with mmINT (section 6.2.5.2), object pixels are treated differently from background pixels. For the change of *object* pixels into background pixels we have found that the number of pixels to change is:

$$\Delta_k = \frac{n}{2}(M - 2k + 1) - \frac{1}{2}(n \bmod 2) . \tag{6.44}$$

For *background* pixels the number of pixels to change is:

$$\Delta_k = \frac{n}{2}(M - 2k + 1) + \frac{1}{2}(n \bmod 2) . \tag{6.45}$$

Index k is a strictly positive integer within a range that depends of the magnification:

$$\begin{aligned} 0 < k &\leq \tfrac{M}{2} \\ &\leq \left\lfloor \tfrac{M}{2} \right\rfloor . \end{aligned} \tag{6.46}$$

The expression $n \bmod 2$ in equations (6.44) and (6.45) is the modulo of n to 2. If n is even, then $n \bmod 2 = 0$ and the equations reduce to equation (6.42) for odd magnifications.

6.3.1.3 Calculation of length of plateau

The new pixel swapping step we just described relies on the value n. This value can be derived from the length of the plateau, i.e., $\text{length}(P) = nM$. In other words, we first need to know $\text{length}(P)$ before we can perform the one-step interpolation.

Consider a pixel classified as a jagged corner. Figure 6.40 illustrates a few possible situations of lines. The pixel marked with a white dot is an upper-left corner pixel. The background pixel marked with a black dot is a complementary

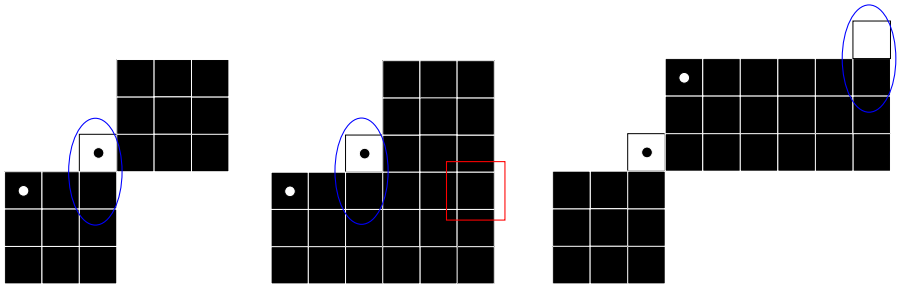


Figure 6.40: An upper-left corner (marked with a white dot) and its horizontal plateau. The blue encircled region marks the end of the plateau.

corner (CC II). The plateau this upper-left corner pixel is part of, extends to the right or downwards. Similar conclusions can be drawn for the other corner orientations.

Detecting the end of the plateau, and thus its length, is simple: scan the plateau, starting from the jagged corner $c_0 = (i, j)$, and check the following conditions:

- Has the pixel under investigation the same value as the corner pixel (the starting point c_0)?
- If so, has the pixel that lies "above"[29] the plateau the opposite colour as the corner pixel?

The pixel that came right before the first pixel that does not satisfy these two conditions, marks the end of the plateau.

In our example, we first scan to the right. Afterwards we must also do this downward, but this is not shown in the figures. The first condition is checked by investigating the pixels $(i, j+p)$ for $p > 0$. In figure 6.40(a), the final value for p (before the condition is not satisfied) is $p_e = 2$. For figures (b) and (c), $p_e = 5$.

We notice that the value for p_e, that marks the position of the end of the plateau, is incorrect in figure 6.40(b) (the boxed pixel), because above the layer there are object pixels present that are part of another plateau of the staircase. The end of the plateau should be marked at $p = 2$. Therefore, we also check the second condition by looking at pixel $(i-1, j+p)$ (for $p > 0$) to see if this is background. If both conditions are true, then the pixel $(i, j+p)$ is still part of the plateau. This way we can find the end of the plateau: $p_e = 2$ for figures (a) and (b), and $p_e = 5$ for figure (c). The following pixel, i.e., $(i, j+p_e+1)$, is the first pixel that does not satisfy both conditions.

[29] If we consider figure 6.39, we would look at layer 0.

The length of the plateau, length$(P) = nM$, equals $p_e + 1$. Thus:

$$n = \frac{p_e + 1}{M} \ . \tag{6.47}$$

Because of the pixel replication, several pixels in the magnified image are just copies of a pixel from the low-resolution image. We therefore can restrict the scan to pixels at locations $(i, j + p)$ (and $(i - 1, j + p)$), with p:

$$p = qM - 1 \ , \tag{6.48}$$

with $q > 0$ (a strictly positive integer).

6.4 Experimental results

In the previous sections we presented the theoretical background and implementation of our morphological interpolation techniques mmINT and mmINTone. Now, we discuss the results of our techniques.

First, in sections 6.4.1 and 6.4.2, we show the differences between our two methods, mmINT and mmINTone. Next, we compare our techniques with existing ones, like the example-based method of [Stepin, 2003] and classical linear interpolation methods [Lehmann et al., 1999].

In order to determine the statistical significance of our premise that mmINT (and mmINTone) outperforms other techniques, we performed a traditional PSNR measurement (section 6.4.5) and a psychovisual experiment (sections 6.4.6 and 6.4.7).

6.4.1 Visual comparison of mmINT with mmINTone

mmINT and mmINTone are based on the same principles. A jagged edge has a staircase pattern, where each step, the *plateau*, has a certain length. The plateau is divided into smaller plateaus, and the height is reduced to one pixel. The two methods try to achieve this goal in a different way. They are very similar, but the results will not be exactly the same.

Figure 6.41 shows a line that has been magnified 3 times (using pixel replication). The result of mmINT interpolation can be seen in figure 6.42, the output of mmINTone is shown in figure 6.43. The differences can be ascribed to the corner validation of CC I pixels: while mmINTone only passes this validation step once, with mmINT however, a line can be affected multiple times, leading to different results. A clear example is the top plateau (of the object) in the figure: mmINTone calculates that this plateau is $3M$ pixels long, and thus 3 pixels will be swapped (using equation (6.42)). With mmINT, the search for

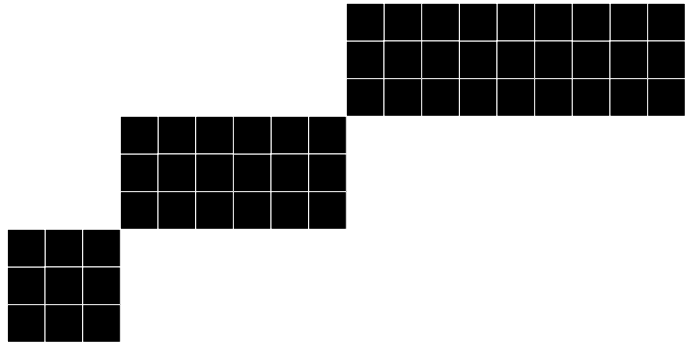


Figure 6.41: A sample image, pixel-replicated ($M = 3$).

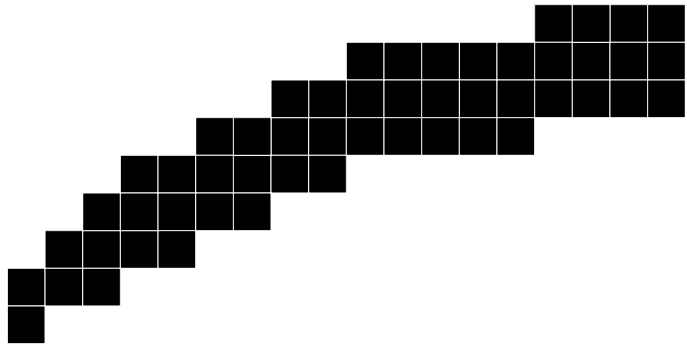


Figure 6.42: Figure 6.41 interpolated with mmINT.

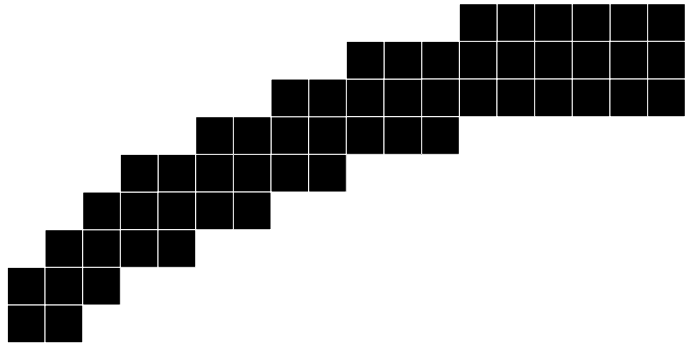
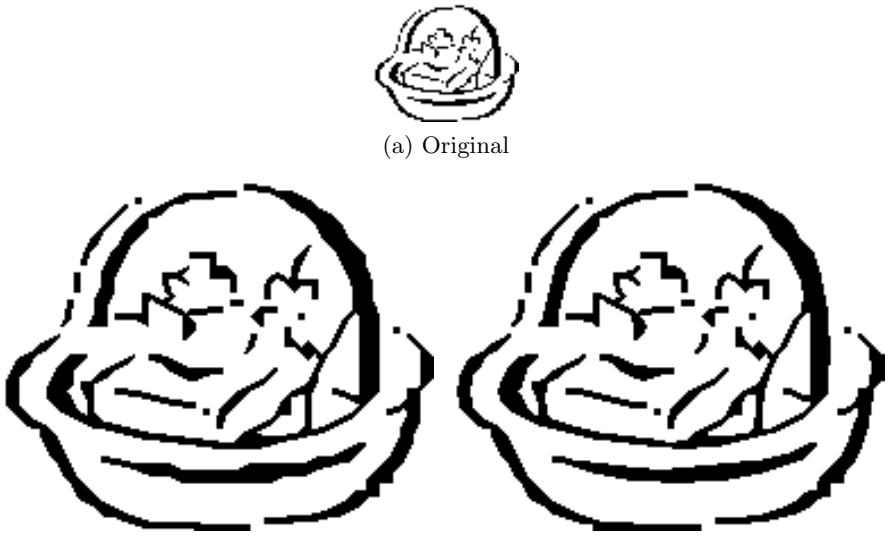


Figure 6.43: Figure 6.41 interpolated with mmINTone.



(a) Original

(b) mmINT (c) mmINTone

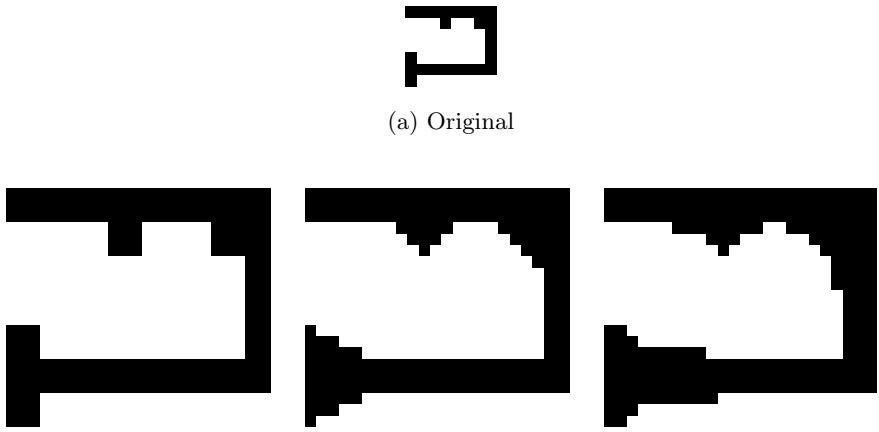
Figure 6.44: Differences between mmINT and mmINTone. Magnification $M = 3$.

CC I makes it possible to perform five iteration steps, which results in an eroded plateau. Here, 5 pixels are swapped.

Figure 6.44 shows the difference between interpolation with mmINT and interpolation with mmINTone on line art image. mmINTone does a slightly better job at rounding the curved lines. Sometimes, mmINTone is too ambitious (see figure 6.45): small protrusions can cause unwanted artefacts, because mmINTone looks at the length of the plateaus. If the lines are very long, and the protrusions very small (compared to the lines), then the region of pixels that change values will expand more compared to mmINT. This feature of mmINTone can be inconvenient for the interpolation of text. An example of text interpolation is shown in figure 6.46. Straight lines like those in the letter "u" are overinterpolated.

6.4.2 Speed comparison of mmINT with mmINTone

Based on the lower number of iterations, we expect that mmINTone (one iteration) will be significantly faster than mmINT (several iterations). To demonstrate that this is indeed the case, we interpolate artificial as well as real binary images with mmINT and mmINTone. The results are discussed in the following subsections.



(a) Original

(b) Pixel replication (c) mmINT (d) mmINTone

Figure 6.45: Protrusions introduce artefacts. Magnification $M = 3$.

(a) Pixel replication (b) mmINT (c) mmINTone

Figure 6.46: Interpolation of a text sample. Differences between mmINT and mmINTone are visible.

6.4.2.1 Artificial images

We take 16 artificial images (size 50×50) and interpolate them for different magnifications. Each image I_m ($m = \{1, 2, \ldots, 16\}$) contains lines under a specific angle. The angle of the lines in I_m is chosen in such way that mmINT would need m iterations for achieving optimal interpolation of I_m. The number of jagged corners in the images is kept constant (188).[30]

The timing experiments are performed on an AMD Athlon XP 2200+ (1.8 GHz, 1.5 GB RAM) running Linux, kernel v2.6.3. We used the `time` command in Linux to perform the measurements.[31]

We can draw the following conclusions:

[30] We only have 16 different images, because we want to keep the number of jagged corners constant, with the image size restricted to 50×50 pixels.

[31] The calculation time (using `time`) is the addition of the measured CPU-seconds used by the system on behalf of the process (in kernel mode) and the CPU-seconds used directly by the process (in user mode).

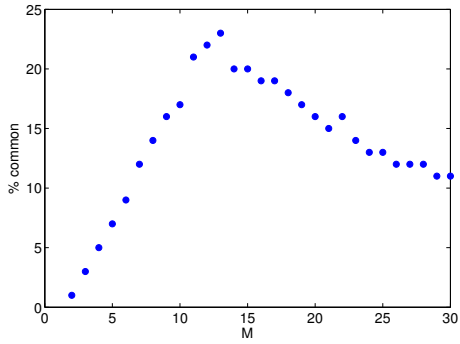


Figure 6.47: The percental contribution of the common parts of the algorithm to mmINT.

- The common parts of mmINT and mmINTone, i.e., image read-in, pixel replication, corner detection (including hole-filling) (first iteration step), corner validation (first iteration step) and image write-out, occupy a certain percentage of the total calculation time. For the artificial images and magnifications $M = \{2, 3, \ldots, 30\}$, this is on average about 18 % for mmINT and about 98 % for mmINTone. For mmINTone, the percentage of contribution of the common parts is by approximation constant with the magnification. For mmINT, however, the contribution depends on the magnification used (see figure 6.47). We notice that for a magnification factor $M = 13$ and higher the interpolation part (i.e., the iterative procedure of mmINT) requires more and more time compared to the common parts. Only for lower M, the magnification factor implies an increase of the percental contribution of the common parts.[32]

- The calculation time t (in seconds) of the common parts at magnification M has the following (experimentally obtained) quadratic relationship:

$$t(M) = 0.0055M^2 + 0.0062M - 0.013 \; . \tag{6.49}$$

  The norm of the residuals is 0.35. The *residuals* are the differences between the experimental data and the fitted data. The norm of the residuals is the square root of the sum of squares of the residuals. The lower this value, the better the fit. This quadratic relationship seems logical, because a magnification M implies an increase of the image pixels by factor M^2. A similar relationship can be found for the processing time in function of the width and height of the input image.

- The processing time for mmINTone interpolation is independent of the image content, for a given magnification. This is logical, since all the

[32] More pixels are created, more pixels must be checked, a larger output image must be written to file.

Table 6.1: The average calculation time for mmINTone on the artificial images.

Magnification	**Speed** (s)
4×	0.09849 ± 0.00045
11×	0.8119 ± 0.0020
30×	5.1319 ± 0.0081

necessary changes are made in one single step. On the other hand, the processing time depends on the magnification M. The average total calculation times for 4×, 11× and 30× magnification is shown in table 6.1.[33] The read-in and write-out operations of the image files are included in the timings. So, the interpolation of a 50×50 binary image, with a magnification factor $M = 30$, takes about 5 s with mmINTone, no matter what the content of the image is.

- The processing time for mmINT interpolation increases polynomially with the number of iterations needed (this number depends on the image content). This increase is not only due to the number of iterations, but also due to the higher calculation cost for using larger structuring elements in the higher iteration steps. A plot is shown in figure 6.48(a) for $M = 4$, $M = 11$ and $M = 30$. τ is the number of iterations needed. Notice the logarithmic scale of the ordinate. The dependency on τ is approximately cubic, i.e., $t \propto \tau^3$.

- The processing time ratio mmINT/mmINTone ranges from 1.07 to 52.95 for magnification 4×, from 1.08 to 12.45 for magnification 11×, and from 1.37 to 31.39 for magnification 30×. The gain in speed when using mmINTone instead of mmINT can thus be one or two orders of magnitude. The progress of the increases is similar to those shown in figure 6.48(a).

- Figure 6.48(b) shows a plot of the average calculation time for the interpolation of the 16 artificial images with mmINT and mmINTone, for magnifications 2 to 30.

  By fitting a polynomial to the time measurements, we obtain the following functions, just as in equation (6.49) but now including the interpolation-specific parts:

$$\text{mmINT: } t(M) = 0.0018M^3 - 0.02M^2 + 0.18M - 0.8 \ , \quad (6.50)$$

$$\text{mmINTone: } \quad t(M) = 0.006M^2 - 0.0034M + 0.032 \ . \quad (6.51)$$

  The norm of the residuals is 0.98 and 0.49, respectively. We indeed expect an increase in calculation time with the magnification factor, since this implies more pixels need to change value. In the case of mmINT the

[33] The average is taken over respectively 100, 15 and 2 timings of each of the 16 images.

structuring elements in the pixel swapping step are bigger, in the case of mmINTone more layers need to be processed. When more iterations are needed, the calculation time increases non-linearly for mmINT, since bigger structuring elements are used in the higher orders. We expect the number of jagged corners to decrease with each iteration.

The computational cost (in number of operations) in the corner detection part of the algorithm (i.e., the hit-miss operation) for a pixel that is detected as a jagged corner for τ iterations is:

$$9 + \sum_{i=2}^{\tau}(8i-3) = 9 - 3(\tau-1) + 8\sum_{i=2}^{\tau} i = 4\tau^2 + \tau + 4 \ . \qquad (6.52)$$

This formula can be deduced from the structuring elements in figure 6.36: at iteration step $i > 1$, two erosions are performed, which have a cost of $4(i-1)+3$ (foreground of the hit-miss) and $4(i-1)+1$ (background of the hit-miss) operations.[34] An extra operation for the intersection of the two erosions results in a total of $8i-3$ operations, which appears in equation (6.52). For the first iteration step, there are only 9 operations needed. This value is also present in the equation.

The computational cost (in number of operations) for swapping a jagged corner (and its neighbouring pixels), in function of the magnification M, is:

$$\sum_{j=1}^{\lfloor M/2 \rfloor} \left(j - \frac{1-(M \bmod 2)}{2} \right) \ . \qquad (6.53)$$

This result is obtained from the examination of the structuring elements in figure 6.37. The equation holds for both odd and even magnifications. Because of the different treatment of objects and background when M is even, the average number of operations can be a fractional value.

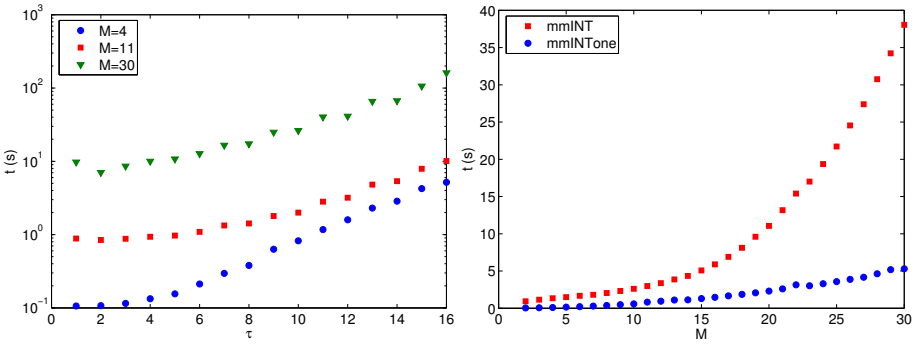
6.4.2.2 Realistic images

We also compare the speed of our two methods on realistic binary images. 33 random binary images, ranging from a size of 14×42 to 504×300, are interpolated with $M = 4$, $M = 11$ and $M = 30$ using mmINT and mmINTone. The number of iterations goes from 4 to 38.

The timing experiments are performed on an AMD Athlon XP 4000+ (2.41 GHz, 2 GB RAM, 64 bit) running Linux, kernel v2.6.11.

We can draw the following conclusions:

[34] The pixels are compared to their corresponding structuring element pixel, and the logical results are then compared using the intersection operation.



(a) mmINT for $M = \{4, 11, 30\}$ (b) mmINT vs mmINTone

Figure 6.48: Calculation time needed for interpolating artificial images with different content.

Table 6.2: The ratio mmINT/mmINTone for the realistic images.

Magnification	Average ratio	Min. ratio	Max. ratio
4×	1.54 ± 0.68	1.00	4.38
11×	2.6 ± 3.0	1.1	16.8
30×	12 ± 23	1	128

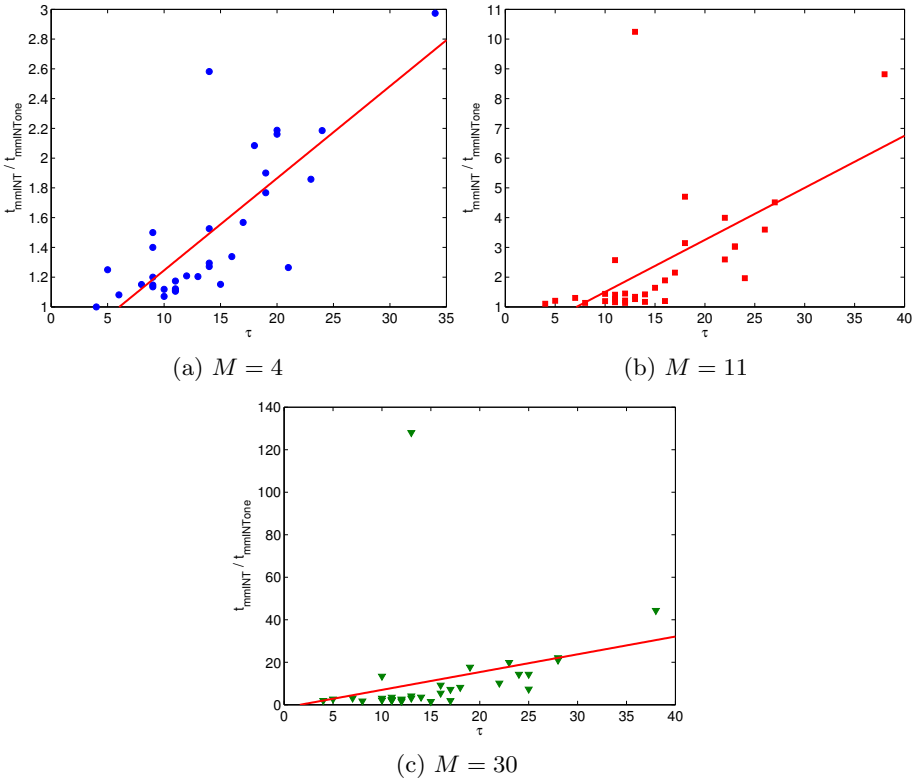
- The average processing time ratios mmINT/mmINTone increase with the desired magnification. This is shown in table 6.2, which also shows the minimum and maximum ratios obtained.

  The test images are quite diverse: some images are small while others are large, some images have more jagged corners than others, and some images need many iterations while others only need a few. This diversity is noticeable in the standard deviation in table 6.2, which can be larger than the average ratio. Sometimes mmINT needs much calculation time and mmINTone does not, in other situations they both process the image relatively quickly.

- Figure 6.49 shows a plot of the number of iterations needed (for mmINT) against the ratio mmINT/mmINTone. A linear fit of these data is possible, although the norms of the residuals are rather high, namely 1.7, 9.6 and 120 for $M = 4$, $M = 11$ and $M = 30$, respectively.[35]

The calculation time for mmINTone is on average smaller than that of mmINT. With the latter algorithm, we need several iterations in order to obtain the

[35] By removing the outliers at $\tau = 13$, we can reduce the norms of the residuals to 1.3, 4.8 and 27 for $M = 4$, $M = 11$ and $M = 30$, respectively.



(a) $M = 4$ (b) $M = 11$

(c) $M = 30$

Figure 6.49: Ratio mmINT/mmINTone for interpolating realistic images with different content. The red line is the linear fit of the data.

final interpolation result. With every iteration step, the sizes of the structuring elements increase, and so does the computational cost.

6.4.3 Binarization of greyscale images

Most interpolation techniques for binary and greyscale images produce a greyscale result, even if the input image is a binary image. This is the case when the values for the new pixels are calculated using some known pixel values (e.g., an average of neighbouring pixels for the bilinear interpolation), which introduces new pixel values. This is not the case with mmINT(one), since it only works with binary images where pixel values are swapped from black to white, and vice versa. In other words, only existing grey values (black and white) are used.

This greyscale result for a binary image is not always desired, as binary logos or cartoons often should remain binary. Binarization of the output images of the other techniques is thus needed in order to compare. This binarization can be advantageous, since the greyscale interpolation of a binary image often looks

(a) Greyscale (b) Binary

Figure 6.50: A bilinearly interpolated image. Figure (b) is the binary version of figure (a) after Otsu thresholding.

blurry. The difference between a bilinearly interpolated image in grey values and after thresholding is shown in figure 6.50. We use the well known *Otsu threshold* method [Otsu, 1979] to define the ideal threshold for the grey values. We now explain how this thresholding technique works.

Otsu thresholding

The image pixels are categorized into two clusters: the foreground and the background [Liao et al., 2001]. The threshold grey value that divides these clusters is chosen in such way that the clusters' overlap is minimized. We therefore need to minimize the *within-class variance*:

$$\sigma_W^2(T) = \omega_{bg}(T)\sigma_{bg}^2(T) + \omega_{fg}(T)\sigma_{fg}^2(T) \ . \tag{6.54}$$

The threshold value T that minimizes this equation, is called the *Otsu threshold*. We use the following definitions:

$$p_i = \frac{n_i}{N} \text{ probability of grey level } i, \tag{6.55}$$

$$\omega_{bg}(T) = \sum_{i=0}^{T-1} p_i \text{ probability of the background,} \tag{6.56}$$

$$\omega_{fg}(T) = \sum_{i=T}^{N-1} p_i \text{ probability of the foreground,} \tag{6.57}$$

$$\mu_{bg}(T) = \frac{\sum_{i=0}^{T-1} i\ p_i}{\omega_{bg}} \text{ mean for the background,} \tag{6.58}$$

$$\mu_{fg}(T) = \frac{\sum_{i=T}^{N-1} i\ p_i}{\omega_{fg}} \text{ mean for the foreground,} \tag{6.59}$$

$$\mu = \sum_{i=0}^{N-1} i\ p_i \text{ total mean,} \tag{6.60}$$

$$\sigma_{bg}^2(T) = \sum_{i=0}^{T-1} (i - \mu_{bg}(T))^2 p_i \text{ variance for the background,} \tag{6.61}$$

$$\sigma_{fg}^2(T) = \sum_{i=T}^{N-1} (i - \mu_{fg}(T))^2 p_i \text{ variance for the foreground,} \tag{6.62}$$

$$\sigma^2 = \sum_{i=0}^{N-1} (i - \mu)^2 p_i \text{ total variance.} \tag{6.63}$$

The value N is the total number of pixels in the image. n_i is the number of pixels with grey value i (usually $i = \{0, \ldots, 255\}$). Equivalent to the minimization of equation (6.54) is the maximization of the *between-class variance*:

$$\begin{aligned} \sigma_B^2(T) &= \sigma^2 - \sigma_W^2 \\ &= \omega_{bg}(T)\omega_{fg}(T)[\mu_{bg}(T) - \mu_{fg}(T)]^2 \ . \end{aligned} \tag{6.64}$$

Finding the Otsu threshold is a discriminant analysis problem: it is sufficient to maximize equation (6.64), but the actual goal is to maximize the value of the *discriminant criterion*, which is:

$$\eta(T) = \frac{\sigma_B^2(T)}{\sigma^2} \ . \tag{6.65}$$

6.4.4 Visual comparison

Many interpolation techniques exist. Our techniques, mmINT and mmINTone, have been compared to the ones listed in table 6.3. There are only few that are strictly binary. These techniques are marked in the table. In this section, we show the results and differences for some of these methods. The other methods pose similar results as or worse results than the interpolation methods we discuss.

In figure 6.51 we show a cartoon that serves as the low-resolution input image. In figure 6.52 we show several magnified versions ($M = 3$) of the cartoon.

Magnification using pixel replication introduces jaggies (figure (a)). Figure (b) shows the bilinear interpolation result, after binarization. We still see some jagginess in this type of images. Similar results are obtained in figure (c): the windowed sinc kernel with a 4-term Blackman-Harris window is not able to interpolate binary images well. HQ (figure (d)) shows a nice interpolation result: the binarization does not pose any problems and the lines appear smooth. Our morphological techniques (figures (e) and (f)) are very similar. Compared to HQ they are not that different, until we look at the details. The contours

Table 6.3: A list of interpolation techniques that we have examined. Techniques that produce a binary result for a binary input image are marked with *.

Technique	Reference
Pixel Replication*	[Lehmann et al., 1999]
Bilinear	[Lehmann et al., 1999]
Bicubic	[Lehmann et al., 1999]
Catmull-Rom	[Catmull and Rom, 1974]
Sinc (4-term Blackman-Harris)	[Harris, 1978]
Sinc (Lanczos)	[Meijering et al., 2001]
B-Spline (Bicubic)	[Unser, 1999]
Isophote (B-Spline Bicubic)	[Morse and Schwartzwald, 1998]
NEDI	[Li and Orchard, 2001]
Hierarchical Filling Strategy*	[Askar et al., 2002]
2xSaI	[Liauw Kie Fa, 2001]
Super2xSaI	[Liauw Kie Fa, 2001]
SuperEagle	[Liauw Kie Fa, 2001]
Scale2x*	[Mazzoleni, 2001]
HQ	[Stepin, 2003]
mmINT*	Section 6.2
mmINTone*	Section 6.3

Figure 6.51: A sample image.

in the figures interpolated with mmINT(one) are smoother. For example, the man's chin, neck, mouth and shirt collar are better interpolated. Therefore we state that our techniques are visually better than other interpolation methods.

6.4.5 PSNR calculation

Most quality tests use a reference image as ground truth. The altered images are compared to the reference image and some value is calculated, mostly the *Peak Signal-to-Noise Ratio* (PSNR) or the related *Mean Squared Error* (MSE).

The mean squared error (MSE) [Pratt, 2001] is the expected value of the square of the "error", which is the difference between the reference data and the

(a) Pixel replication

(b) Bilinear

(c) Sinc (Blackman-Harris)

(d) HQ

(e) mmINT

(f) mmINTone

Figure 6.52: Interpolation results of figure 6.51 for 3× magnification.

measured data. For a discrete image $I(x, y)$, with $x = 0, 1, \ldots, X-1$ and $y = 0, 1, \ldots, Y-1$, the mean squared error in image processing is defined as:

$$MSE = \frac{1}{XY} \sum_{x=0}^{X-1} \sum_{y=0}^{Y-1} ||I(x,y) - I_{ref}(x,y)||^2 \ , \tag{6.66}$$

where $I_{ref}(x, y)$ is the ground truth reference image. The root of this result is called the *Root Mean Squared Error* (RMSE). When we define N as the maximal possible grey value,[36] then the peak signal-to-noise ratio (PSNR) can be computed from the MSE:

$$\begin{aligned} PSNR &= -10 \log \left(\frac{MSE}{N^2} \right) \\ &= 20 \log \left(\frac{N}{\sqrt{MSE}} \right) . \end{aligned} \tag{6.67}$$

The unit of the PSNR is *decibel* (dB). Typical values are between 30 dB and 40 dB, when measuring the error of good quality image compression techniques or the quality of noise filtering algorithms.

In the case of interpolation, no reference image exists: we start with a small image, but the interpolated images are magnified versions of the original. This problem can be solved by taking an image, scaling it down by subsampling it, and comparing the interpolation result of this small image with the original figure.

Table 6.4 shows the PSNR values for 7 interpolation techniques (mmINT, mmINTone, HQ, pixel replication, a bilinear and a bicubic algorithm, and a Blackman-Harris windowed sinc interpolator). We took 57 different binary images, scaled down 2, 3 and/or 4 times, resulting in a set of 152 downsampled images. From the table, which shows the average PSNR, calculated in the logarithmic domain, we can conclude that the standard deviation is too high and the difference between the PSNR values is too low to draw a clear conclusion. For example, the PSNR difference between mmINTone and HQ is only about 0.45 %. The behaviour of the PSNR is correlated with the visual quality, though.

A partial explanation for the low PSNR values[37] is the procedure of downsampling: lines that are too thin can disappear after subsampling. Also, a line can have different thicknesses after subsampling, depending on the position of that line in the image. This affects the interpolation results. Figure 6.53 shows two simple examples of images where the downsampling can generate different results. If the resolution of those images is decreased by a factor of 2, then there are two possibilities: the black lines or dots still remain visible in the subsampled image, resulting in a totally black image; or the white pixels are preserved, resulting in a totally white image.

[36] For binary images, $N = 1$, for greyscale images, we usually have $N = 255$.

[37] We would expect values around 30 dB for a good technique.

Table 6.4: PSNR calculation (in dB) for 7 interpolation techniques (152 images). The higher the PSNR value, the better. The standard deviation is indicated for each result.

Technique	**Average PSNR** (dB)
mmINT	19.2 ± 5.2
mmINTone	19.2 ± 5.2
HQ	19.1 ± 5.1
Bicubic	18.7 ± 5.1
Bilinear	18.6 ± 5.0
Sinc (B-H)	18.6 ± 5.0
Pixel Replication	18.2 ± 4.9

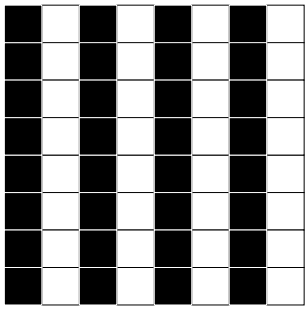
Another reason for the low PSNR values is the binary input: the MSE is a difference in intensity between two images (equation (6.66)). When dealing with greyscale images, there are usually 256 different grey values, and images that are similar, will most probably have small differences in grey values. In the binary case though, a difference in value is a sudden difference: the pixel values change from one extreme to the other, because only two grey values are available.

We can conclude that the peak signal-to-noise ratio (or mean squared error) gives us an idea of the quality of the different interpolation techniques, but it is not a very useful quality measure for the comparison of interpolation techniques on binary images. The PSNR differences are too small, the PSNR values are low and the standard deviations are too high. We should remark that the biggest PSNR value does not necessarily belong to the visually best result. While this error value is often used as a quantitative comparison between different techniques, and can be a good comparison measure, it does not always correlate well with human subjective scoring. In [Vansteenkiste et al., 2006b, Vansteenkiste et al., 2006c], more information can be found on several similarity measures, other than PSNR or MSE.

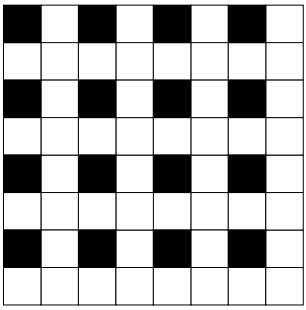
6.4.6 Ranking experiment

We showed 8 different images, interpolated with 4 different techniques (mmINT, HQ, pixel replication and a bicubic algorithm), in random order to 37 (non-expert) persons. The magnification was 2, 3 or 4 times the original binary image. The test images are cartoons (both line drawings and filled drawings), text, and maps (containing both line drawings and text). They are shown in figure 6.54. Because this survey was conducted before the implementation of mmINTone, only mmINT is compared with existing techniques.

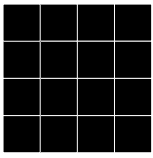
For this test, we asked our test panel to rank the 4 different interpolation



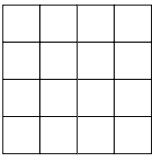
(a) Original image



(b) Original image



(c) Downsampling of (a) and (b)



(d) Downsampling of (a) and (b)

Figure 6.53: Downsampling images is not without problems. Different results are possible.

techniques in order of preference, this for each of the 8 images. A tie was not allowed. We then calculated the average ranking, as can be seen in table 6.5.

The results from this experiment are consistent: mmINT is always ranked first, followed by HQ, except for text images. Bicubic interpolation and pixel replication follow at a rather big distance. In the case of the map images (lines and text combined), the pixel replication is preferred to the bicubic interpolation.

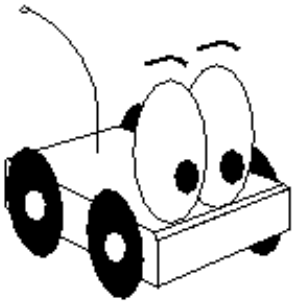
An explanation for the less favourable ranking of mmINT for text samples, is the following: when we look at figure 6.55, we notice that the roundings of the letters "q" and "a" are not that well interpolated by mmINT. This is because different regions, that are interpolated independently, touch each other. This visually less attractive result explains the outcome of the psychovisual experiment. If the original text is rasterized with a higher resolution, then this problem does not occur.

6.4.7 Multi-dimensional scaling experiment

We showed 8 different images, interpolated with 4 different techniques (mmINT, HQ, pixel replication and a bicubic algorithm), in random order to 35 (non-expert) persons. The magnification was 2, 3 or 4 times. The test images are the same as for the ranking experiment (figure 6.54). Because this survey was conducted before the implementation of mmINTone, only mmINT is compared with existing techniques.

(a) Filled 1

(b) Filled 2



(c) Lines 1

(d) Lines 2

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. **Suspendisse blandit**, nulla quis lobortis pretium, quam elit lobortis felis, vel tempor erat lacus ac nibh. ***Nunc scelerisque***, urna eu fringilla fringilla, erat lectus elementum sem, eu sollicitudin diam lorem eget quam.

(e) Text 1

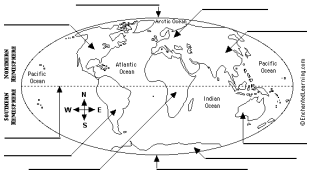
be terse.

nulas may also be displayed. A
rmulas require special formattir

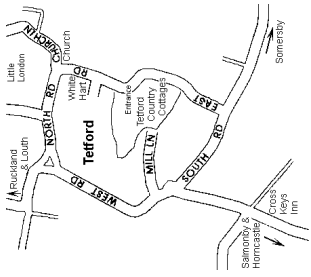
$$(\Gamma, \psi') = x'' + y^2 + z_i^n$$

ph with a displayed equation, 1

(f) Text 2



(g) Map 1



(h) Map 2

Figure 6.54: A set of 8 test images. Their resolution (DPI) does not correspond to their printed size.

Table 6.5: Ranking of the 4 interpolation techniques. A lower number means a more preferred technique. The standard deviation is indicated for each result. Notations: B=bicubic; H=HQ; M=mmINT; P=pixel replication.

	Filled 1	**Filled 2**	**Lines 1**	**Lines 2**
1st	M: 1.76 ± 0.68	M: 1.27 ± 0.45	M: 1.00 ± 0.00	M: 1.24 ± 0.43
2nd	H: 1.84 ± 0.80	H: 1.78 ± 0.53	H: 2.00 ± 0.00	H: 1.84 ± 0.50
3rd	B: 2.43 ± 0.87	B: 2.97 ± 0.29	B: 3.08 ± 0.28	B: 3.11 ± 0.57
4th	P: 3.97 ± 0.16	P: 3.97 ± 0.16	P: 3.92 ± 0.28	P: 3.81 ± 0.40
	Text 1	**Text 2**	**Map 1**	**Map 2**
1st	H: 1.19 ± 0.40	H: 1.41 ± 0.69	M: 1.32 ± 0.53	M: 1.49 ± 0.61
2nd	M: 2.03 ± 0.44	M: 1.78 ± 0.53	H: 1.76 ± 0.49	H: 1.59 ± 0.50
3rd	B: 3.14 ± 0.89	B: 3.08 ± 0.72	P: 3.16 ± 0.60	P: 3.41 ± 0.64
4th	P: 3.65 ± 0.48	P: 3.73 ± 0.45	B: 3.76 ± 0.43	B: 3.51 ± 0.56

(a) Pixel replication (b) mmINT (c) HQ

Figure 6.55: A text sample interpolated. Problems occur at roundings in letters.

This second psychovisual experiment utilizes a *multi-dimensional scaling* technique (MDS) [Kayargadde and Martens, 1996a, Martens and Meesters, 1998, Kayargadde and Martens, 1996b, Martens, 2003] to judge the results.[38] We first define two terms used: the *stimuli* and the *subjects*. The *stimuli* are the input images, in our case the results of the different tested interpolation techniques. The *subjects* are the persons that perform the psychovisual experiment and score the stimuli.

We represent the stimuli in a geometrical *perceptual space*. Usually, this is a multi-dimensional vector space. The positions of the stimuli in such a vector space are directly related to their mutual relationship: the distance between the stimuli states how similar or dissimilar these images are perceptually, based on the relevant image *attributes* (e.g., blur, noise, contrast, overall quality, ...) The scoring of these attributes should correlate strongly with the stimulus positions in this perceptual space.

While the subjects can score the stimulus pairs on a different (subjective) scale,

[38] I hereby would like to thank Ewout Vansteenkiste for parsing the data and helping me with the theoretical concepts of the technique. I would also like to thank Stefaan Lippens for kindly providing me his PHP/MySQL framework for the test set-up.

we assume the *principle of homogeneity of perception*: the stimulus positions in the perceptual space are the same for all subjects, i.e., they share the same *stimulus configuration*.

Stimuli can be compared in several ways. In a *single-stimulus scaling* experiment, every image is shown to the subject by itself and is rated by a certain attribute.

In *double-stimulus scaling*, two input images are judged together, but scored separately. The difference between the responses for the individual stimuli is retained as the final response.

In *dissimilarity scaling*, the subject has to indicate how different two images are perceived. The global difference is judged, without focusing on one attribute.[39]

In *difference scaling* (or *preference scaling*), two images are shown to the subject, which has to give a preference score by a certain attribute. This is the scaling technique used in our experiment.

For each image, we showed the subjects 6 pairs of images of 2 different interpolation techniques and the subjects had to give a preference score for the perceived quality for that pair. In the preference interval it is possible to assign an integer value from -3 (the left image is most preferred), via 0 (both images are equally good), to $+3$ (the right image is most preferred). From these preference scores, we calculate a ranking, using MDS, for the global image quality.

We now briefly explain the principle of MDS for preference scoring. This is schematized in figure 6.56. A more in-depth discussion is available in [Martens, 2003].

Every subject $k = 1, \ldots, K_p$ defines a preference score $P_{k,i,j}$ for a stimulus pair (i, j). In our experiment, there are $K_p = 35$ observers and 6 different stimulus pairs to score for each of the 8 input images. In the MDS model in figure 6.56, the preference scores are on the right, and every line represents a subject.

The goal is to optimize the positions of the stimuli $\mathbf{x}_1, \ldots, \mathbf{x}_N$ ($N = 4$ in our experiment, the number of compared interpolation techniques) in a geometrical space (the perceptual space (one-dimensional in our case, because we only have one attribute, *quality*)). We assume the principle of homogeneity of perception, as mentioned before, thus we pursue one stimulus configuration, shared by all subjects.

The conceptual idea of a stimulus configuration is shown in figure 6.56 for the stimuli $\mathbf{x}_i$ and $\mathbf{x}_j$. The vertical line on the left illustrates that the relationship between $\mathbf{x}_i$ and $\mathbf{x}_j$ in the stimulus configuration is shared by all subjects $k = 1, 2, \ldots$. The horizontal lines illustrate how the shared stimuli $\mathbf{x}_i$ and $\mathbf{x}_j$ are converted to the individual preference scores $P_{1,i,j}$, $P_{2,i,j}$, ...

[39] Thus, the subject scores the stimulus pair for the dissimilarity in noise, *and* blur, *and* overall quality, *and* ....

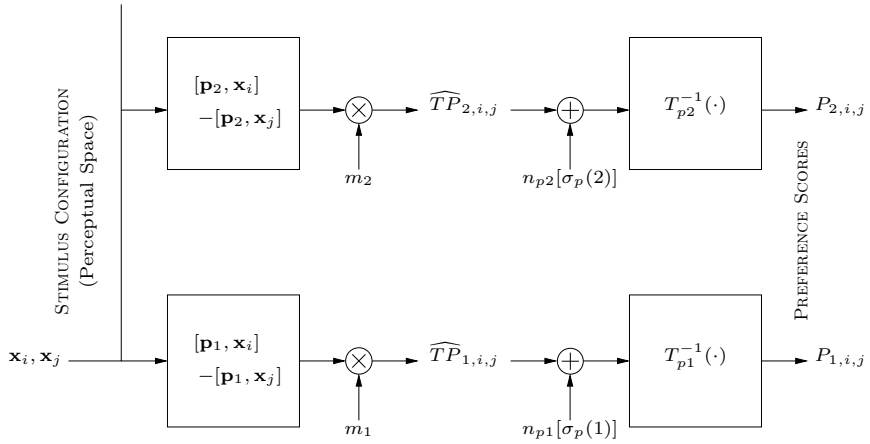


Figure 6.56: The MDS model relating the stimulus positions $(\mathbf{x}_i, \mathbf{x}_j)$ to the preference ratings $P_{k,i,j}$ for the stimulus pair (i, j) and the subject k.

In practice, we optimize the stimulus positions using the program *XGms* [Martens, 2003]. The stimuli should be positioned in such way that the experimentally observed preference scores $P_{k,i,j}$ for the stimulus pair (i, j) and subject k are linearly related to the interstimulus distances d_{ij}. In order to obtain such linear relationship, we must transform $P_{k,i,j}$ into metric data with a transformation T_{pk}, thus $T_{pk}P_{k,i,j} = T(P_{k,i,j})$, i.e., $P_{k,i,j}$ must have ratio properties since they are compared to metric distances.

So, we want to obtain a linear relationship between the transformed preference scores and the interstimulus distances, i.e.:

$$T_{pk}P_{k,i,j} \approx m_k d_{ij} \ . \tag{6.68}$$

In XGms, it is possible to choose between no transformation (i.e., the data are already metric), a generalized optimum power-like transformation, a generalized Kruskal transformation and an optimum spline transformation.[40] We use a spline transformation in our experiment, but there is little difference in the results when using an identity matrix.

The *interstimulus distance* d_{ij} between the stimuli $\mathbf{x}_i$ and $\mathbf{x}_j$ is defined by:

$$d_{ij} = ||\mathbf{x}_i - \mathbf{x}_j||_l = \Big(\sum_{m=1}^{n} |x_{im} - x_{jm}|^l \Big)^{1/l} \ . \tag{6.69}$$

This is the *Minkowski norm*, but we will use the specific case of the *Euclidean*

[40] For details we refer to [Martens, 2003].

norm, i.e., take $l = 2$, and this for one dimension, i.e., $n = 1$:

$$d_{ij} = ||\mathbf{x}_i - \mathbf{x}_y||_2 = |x_i - x_j| \ . \tag{6.70}$$

The regression factors m_k and the transformations T_{pk} need to be derived from the data. They are subject-dependent, because each subject scores in its own individual way so we cannot generalize these variables.

From a certain stimulus configuration we can derive the expected values for the transformed preference scores, defined as:

$$\widehat{TP}_{k,i,j} = m_k d_{ij} \ . \tag{6.71}$$

In order to know how well these estimates compare to the real transformed preferences, we define a *probability density function* (PDF) or *link function* $\phi(x;\sigma)$, for every subject k. This function is used to define the probability $p(P_{k,i,j})$ that the preference score $P_{k,i,j}$ is given by subject k for stimulus pair (i,j), based on the estimate $\widehat{TP}_{k,i,j}$:

$$p(P_{k,i,j}) = \phi[TP_{k,i,j} - \widehat{TP}_{k,i,j}; \sigma_p(k)] \cdot [T'_{pk}(P_{k,i,j})] \ . \tag{6.72}$$

T'_{pk} denotes the derivative of T_{pk}. In different experiments [Martens, 2003, Vansteenkiste et al., 2006c], the *generalized Gaussian* PDF is used. For our experiment it also gives good results. Its definition is:

$$\phi(x;\sigma) = \frac{r}{2\rho\Gamma(1/r)} \exp\left(- \left| \frac{x}{\rho} \right|^r \right) \ , \tag{6.73}$$

with $r = 2$, $\rho = \sigma\sqrt{\frac{\Gamma(1/r)}{\Gamma(3/r)}}$, and the gamma function

$$\Gamma(a) = \int_0^\infty z^{a-1} e^{-z} dz \ . \tag{6.74}$$

A good stimulus configuration implies high probabilities $p(P_{k,i,j})$, i.e., the stimulus configuration should predict well how a subject k will score the stimulus pairs (i,j).[41]

Since we want an optimal configuration for all stimuli and all subjects, we pursue the minimization of the following *log-likelihood function*:

$$L_p = \sum_{k=1}^{K_p} L_p(k) = -\sum_{k=1}^{K_p} \sum_{(i,j)\in I_p(k)} \log p(P_{k,i,j}) \ , \tag{6.75}$$

[41] This implies that the difference between $\widehat{TP}_{k,i,j}$ and $TP_{k,i,j}$ must be as small as possible.

with $I_p(k)$ the set of pairs judged by subject k. XGms calculates this *maximum-likelihood* (ML) criterion and recursively minimizes this criterion, by repositioning the stimuli in the perceptual space.

We will now discuss our results. Figure 6.57 shows the stimulus positions (quality values) for the different techniques for all images together. Since we only scored the attribute "quality", we obtain a one-dimensional ordering (the ordinate in the figure). The abscissa shows the different interpolation methods, the values on the ordinate are the quality values. The projection on the vertical axis gives the actual stimulus configuration. The distances between the stimulus positions represent the differences in quality between the corresponding stimuli. A small (relative) distance means that the subjects perceived the techniques to be similar in quality. When the (relative) distance is large, the techniques are scored quite differently.

mmINT is clearly superior to the other methods, because its quality value is higher than that of the other interpolation techniques. HQ is the second best technique, and is competitive to mmINT. The bicubic and pixel replication method provide much worse results. If we look at the stimulus graphs for the images separately, then we can draw the same conclusion as from the ranking experiment (section 6.4.6): HQ seems to work better for text (figure 6.58(b)); mmINT interpolates better cartoons, logos and maps than the other techniques (figure 6.58(a)).

6.4.8 One-step magnification compared to multi-step magnification

Our techniques mmINT and mmINTone allow the magnification by an arbitrary integer factor M. If we can write this magnification factor as $M = \prod_{i=1}^{n} M_i$, then we could also iteratively magnify our image using smaller magnifications M_i.

The question we then ask ourselves is whether the following equation is valid:

$$\varphi_M(I) = \varphi_{M_n}(\cdots(\varphi_{M_2}(\varphi_{M_1}(I)))) \ , \tag{6.76}$$

where φ_M stands for the mmINT(one) interpolation by magnification factor M.

In general this is not the case. Consider figure 6.59: figure (a) shows the result of mmINT for $M = 9$ for figure 6.51. In figure (b) we applied mmINT with $M = 3$ twice. As we can see in these figures and in the difference map (figure (c)), the images are not completely the same.

Visually, we do not see much difference between the two approaches (about 0.32 % for mmINT and 0.20 % for mmINTone).[42] Figure 6.60(a) shows a sample

[42] We performed a small experiment on the images from figure 6.54. We compared $M = 4$ to $M_1 = 2$ and $M_2 = 2$; $M = 6$ to $M_1 = 2$ and $M_2 = 3$, and to $M_1 = 3$ and $M_2 = 2$; $M = 8$ to $M_1 = 2$ and $M_2 = 4$, and to $M_1 = 4$ and $M_2 = 2$, and to $M_1 = 2$ and $M_2 = 2$ and $M_3 = 2$; $M = 9$ to $M_1 = 3$ and $M_2 = 3$.

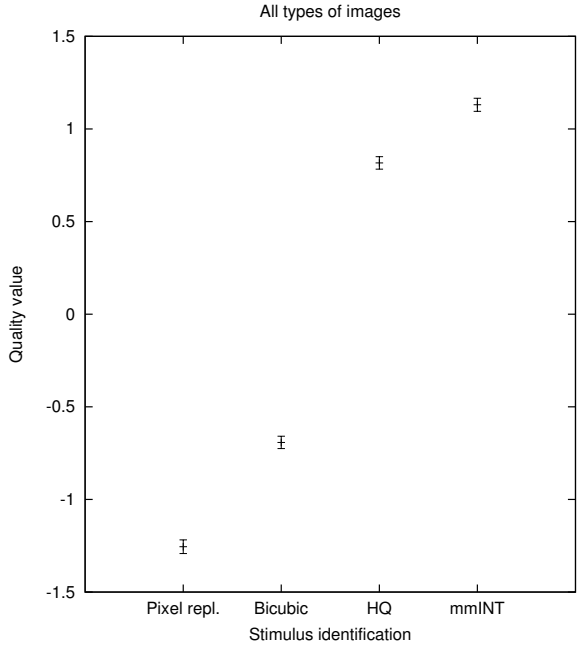
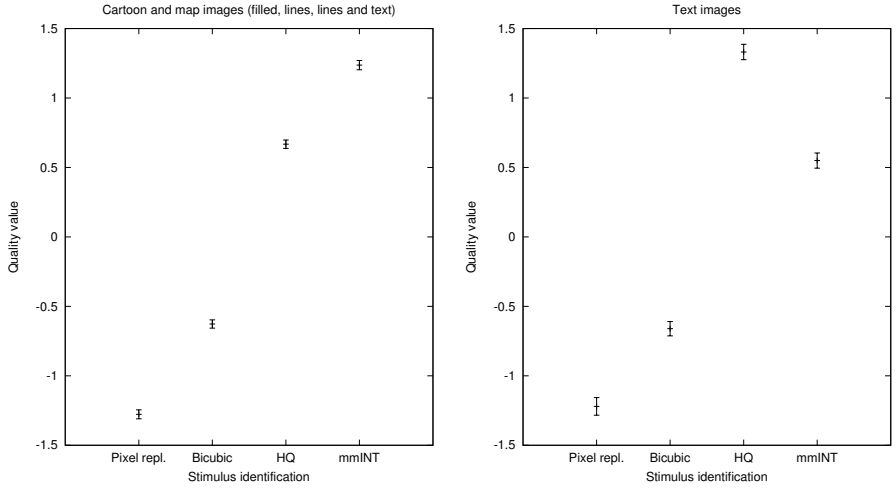
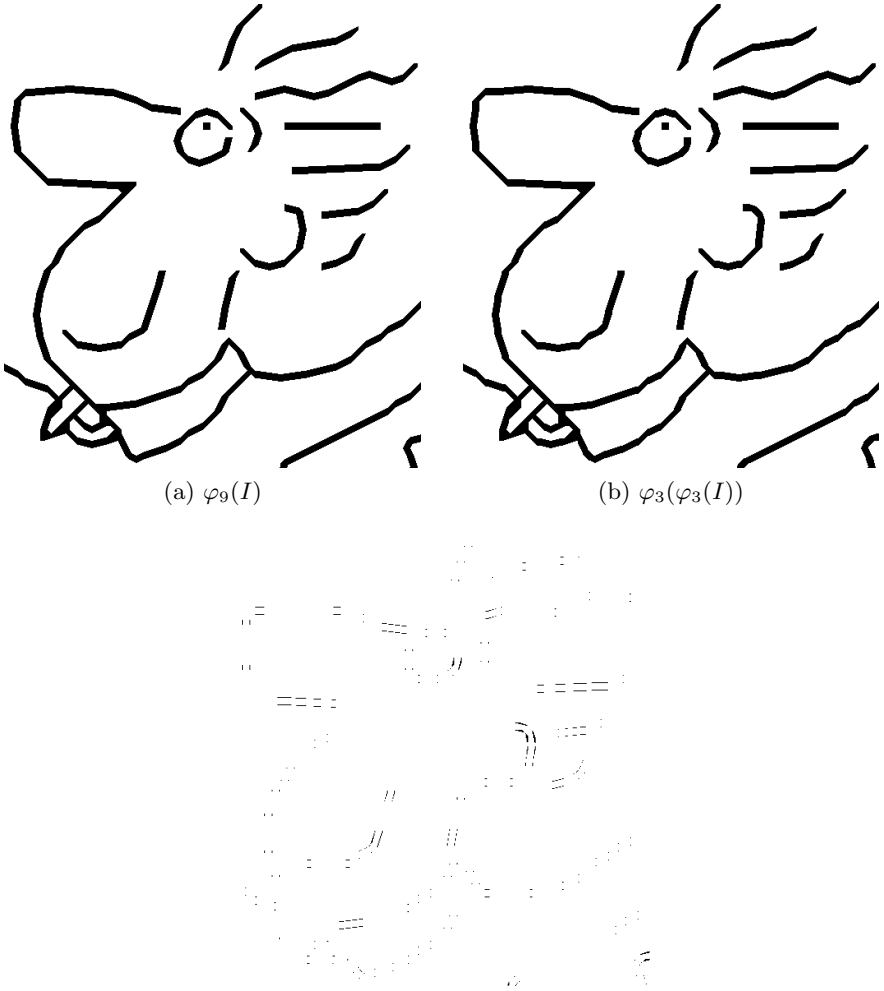


Figure 6.57: Stimulus positions for the 4 interpolation techniques, for all 8 test images together.



(a) Cartoons and maps (b) Text

Figure 6.58: Stimulus positions.



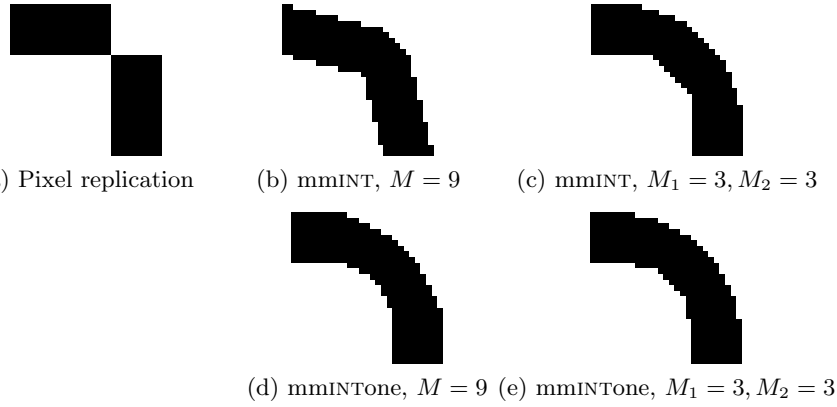
(a) $\varphi_9(I)$ (b) $\varphi_3(\varphi_3(I))$

(c) Difference between (a) and (b)

Figure 6.59: Interpolation with mmINT of figure 6.51 with (a) $M = 9$ and (b) twice $M = 3$.

image (pixel-replicated with $M = 9$) that generates different results for $M = 9$ and $M_1 = M_2 = 3$, both for mmINT and mmINTone. Just like mmINTone (see section 6.4.1), the multi-step approach produces lines that are bent more. The visual difference between the one-step and multi-step approach is very subtle for mmINTone.

Computationally, it is more efficient — on average about 14 % — to interpolate with mmINT two or more times using a small magnification factor, than once with a large value for M. This can be explained by the structuring elements



(a) Pixel replication (b) mmINT, $M = 9$ (c) mmINT, $M_1 = 3, M_2 = 3$

(d) mmINTone, $M = 9$ (e) mmINTone, $M_1 = 3, M_2 = 3$

Figure 6.60: Interpolation with mmINT and mmINTone with $M = 9$ and twice $M = 3$.

used in the pixel swapping step: for a magnification $M = 3$, only the corner pixels must change value, but for $M = 9$ we need to change 10 pixels per corner in the first iteration step (see section 6.2.5). The size of the structuring element is thus bigger for $M = 9$, which increases the calculation time (as we showed in chapter 2, section 2.2.4.4). For mmINTone, a one-step interpolation is more efficient (about 21 %).

For M even, we have to remember that the interpolation of foreground and background pixels is handled differently (see section 6.3.1). For example, a line drawing interpolated with $M_1 = 2, M_2 = 2, M_3 = 2$ generally looks thicker than an interpolation of this line with $M = 8$. We can distinguish the following situations:

Magnification M is an odd factor: In this situation we do not encounter a problem, because foreground and background are treated the same.

Magnification M is an even factor, and the number of iterations τ is even: We explained in section 6.2.6.3 that the equations for foreground and background are interchanged every iteration step. This way, after every two iterations, the global intensity is kept quasi constant. The thickness of a line is therefore the same when interpolated in multiple steps instead of in one step.

Magnification M is an even factor, and the number of iterations τ is odd: If we magnify a line with a thickness of p pixels with magnification M, then we expect a line with a thickness of pM pixels. In the multi-step approach we expect the same behaviour, i.e., a line of thickness p pixels should become a line of $pM_1M_2 \ldots M_n$ pixels, with $M = \prod_{i=1}^{n} M_i$.

In the previous situations, this is indeed the case. However, for an even magnification and an odd number of iterations the situation is different. When we look at figure 6.35, we notice that the thickness of the line ($p = 1$) has become 5 pixels for $M = 4$ after one iteration, instead of 4 pixels. In general, the newly obtained thickness for M even and τ odd is $pM + 1$.

If we use the multi-step approach, e.g., $M_1 = 2, M_2 = 2, M_3 = 2$ (thus, $M = 8$), we would obtain the following thicknesses for each step:

$$p \rightarrow pM_1 + 1 \rightarrow (pM_1 + 1)M_2 + 1 \rightarrow ((pM_1 + 1)M_2 + 1)M_3 + 1 \ . \quad (6.77)$$

Thus, after the first magnification step (M_1) we have one extra pixel (as is the case in the one-step approach). After two magnification steps (M_1M_2), we have $M_2 + 1$ extra pixels (i.e., 3 extra pixels), and finally we obtain a thickness with $M_2M_3 + M_3 + 1$ extra pixels (i.e., 7 extra pixels).

In order to obtain the same thickness for the result of the multi-step approach as that for the one-step approach, we interchange equations (6.26) and (6.27) in mmINT for M_i with $i > 1$. That is, equation (6.26) is then used for the objects, while equation (6.27) is used for the background.[43] The thicknesses then become:

$$p \rightarrow pM_1 + 1 \rightarrow (pM_1 + 1)M_2 - 1 \rightarrow ((pM_1 + 1)M_2 - 1)M_3 - 1 \ . \quad (6.78)$$

Thus, after the first magnification step (M_1) we have again one extra pixel. After the second magnification step (M_1M_2), we have $M_2 - 1$ extra pixels (i.e., 1 extra pixel), and finally we obtain a thickness with $M_2M_3 - M_3 - 1$ extra pixels (i.e., 1 extra pixel).

This extra number of pixels for a multi-step interpolation with $M = \prod_{i=1}^{n} M_i$ can thus generally be defined as (for a minimum of 2 steps):

$$\prod_{i=2}^{n} M_i - \sum_{j=3}^{n} \left(\prod_{i=j}^{n} M_i \right) - 1 \ . \quad (6.79)$$

We immediately notice that this solution is not perfect, since it only works as intended for $M_i = 2$. For higher even magnifications we still have an increase in thickness, but not as much as before.[44] Although not a perfect solution for the thickness-increase problem, we can suggest the following general methodology:

- If possible, decompose even magnifications $M_i > 2$ into a product of smaller magnifications of value $2\times$ and possible odd values;

[43] Regarding mmINTone, we interchange equations (6.44) and (6.45).

[44] For example, $M = 16$ in multi-step as $M_1 = 4$ and $M_2 = 4$ results in an increase in thickness of 5 pixels. With the correction suggested, the increase is 3 pixels.

- Firstly, interpolate using odd magnifications, since here the problem does not occur;
- If even magnifications M_i are used, it is best to retain a certain order when interpolating by the different magnifications: put the highest magnification first, since it will always only increase the thickness with 1 pixel. According to equation (6.79), it is best to rank the remaining magnifications from small to large: first, interpolate with the lowest M_i and end the interpolation with the highest M_i. This will minimize the increase in thickness.

6.5 Extension to greyscale images: mmINTg

The previous sections deal with morphological interpolation techniques that generate good results for binary images. In the binary case, only two possibilities exist: a pixel is part of the foreground or the background. Also, when we look for jagged edges, the result is a detection of a corner or no corner.

The greyscale case is more complicated [Ledda et al., 2006a]: the classification of a pixel into foreground or background is not straightforward (since more than two grey values are present), which makes the detection of corners using the binary hit-miss transform more difficult. We therefore adapt the corner detection step (explained in section 6.2.2). For this purpose the pixels are locally binarized, before applying the hit-miss operation. A majority ordering is used for the classification of a pixel as a foreground or a background corner. All this is covered in section 6.5.1.

In the interpolation step (see section 6.5.2), the values of the neighbouring pixels are taken into account to calculate the interpolated pixel value.

This algorithm for greyscale interpolation is called *mmINTg*. The algorithm is schematized in figure 6.61. We distinguish the following steps:

1. **Pixel replication**: Section 6.2.1: First, the image is pixel-replicated by an integer factor M. The resulting image contains strong staircase patterns because of the pixel replication.
2. ***New* corner detection**: Section 6.5.1: Using a combination of hit-miss transforms, the algorithm determines the positions of corners, both real and false (due to jaggies) in the image. For greyscale images, we must perform a local binarization. Hole artefacts are avoided by using different structuring elements for foreground and background corners (see section 6.2.4).
3. **Corner validation**: Section 6.2.3: Some corners found in the preceding step are *real corners*, which have to be retained in the interpolated image. The aim of corner validation is to distinguish false corners from real ones.

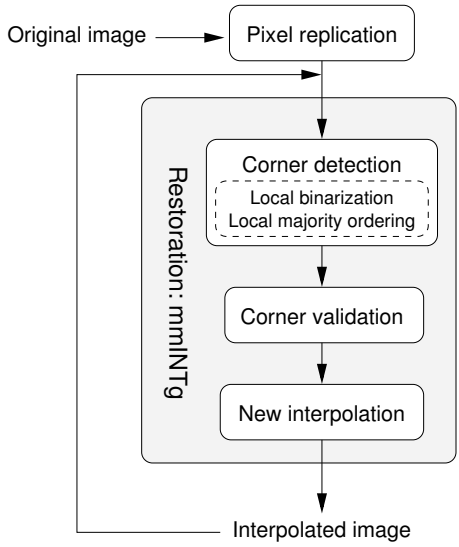


Figure 6.61: Schematic representation of the algorithm of mmINTg.

4. ***New* greyscale interpolation**: Section 6.5.2: We change the colour of pixels classified as false corners, and the colour of some of their neighbours. The new colour is defined by some neighbouring pixels.

The above operations (except the first one) are repeated iteratively, each iteration operating on the output image of the preceding iteration. Currently, a non-iterative procedure (like mmINTone) does not exist. We suppose this alternative implementation could be done as in the binary case, but we would have to take care of the use of local information that is taken into account, as we will see. We will now discuss the new corner detection and interpolation algorithm.

6.5.1 New corner detection method for step 2

6.5.1.1 Local binarization

In order to apply the binary hit-miss transform to detect corners, we binarize the pixel values. We use the binary hit-miss transform, since this gives us a hit or a miss for a corner, and nothing in between. A greyscale hit-miss[45] can give a gradation of correspondence to a corner, but we wish to know for sure if there is a corner present or not. The binarization is done locally and only the pixels that are then covered by the hit-miss structuring elements are taken

[45]Some greyscale hit-miss transforms are discussed in chapter 2, section 2.4.4.

into account (see figures 6.21 and 6.29 for the structuring elements used in the first iteration step). The threshold value $T(x, y)$ for binarization is defined by:

$$T(x,y) = \begin{cases} \frac{1}{2}(m+M) + \alpha, \text{ if } I(x,y) \geq \frac{1}{2}(m+M) \\ \frac{1}{2}(m+M) - \alpha, \text{ if } I(x,y) < \frac{1}{2}(m+M) \end{cases}, \tag{6.80}$$

with m and M respectively the minimum and maximum value of the set of pixels defined by the structuring elements, $I(x, y)$ is the grey value of the currently checked pixel, and α is a threshold for classifying more neighbouring pixels into a class different to the one of the current pixel. We have experimentally found that the following definition for α is a good choice:

$$\alpha = \frac{M-m}{10} . \tag{6.81}$$

The current pixel is always given the binary value 1, the values of the other considered pixels depend on their classification w.r.t. the current pixel.

6.5.1.2 Majority ordering

We include the hole filling step in the corner detection step. We have discussed in section 6.2.4 that we therefore must use different structuring elements for foreground corner detection as for background corner detection. Also, the corner validation step looks at complementary corners, which are part of the opposite class of the pixel under investigation. This means that we need to classify the pixels as either possible object corner or background corner.

We utilize the *majority sorting scheme* (MSS) to order the grey values in function of their frequency in a local window. A full discussion of the MSS is available in chapter 3, section 3.3. We assume that grey values that appear often can be considered background in the local window, while grey values that appear rarely can be considered to be object. We compare the index of the investigated pixel in the ordering map with the index of two of its neighbouring pixels. These neighbours are the ones covered by the background hit-miss structuring element (see figure 6.21(b) for the upper-left corner detection). So, if we look for an upper-left corner, we compare the pixel's index with that of its left and upper neighbours. When the index is lower than that of one of the neighbours, then the pixel is classified as foreground. Otherwise, the pixel is classified as background.

We must properly choose the size of the local window in which we calculate an ordering map with the MSS. If we take the window too small, the pixel count per colour in the window will be almost the same for each colour. This makes the ordering less useful. If we take the window too large, then some pixels might not be interpolated because they do not pass the corner validation step. For example, figure 6.62 shows a white line on a black background. We

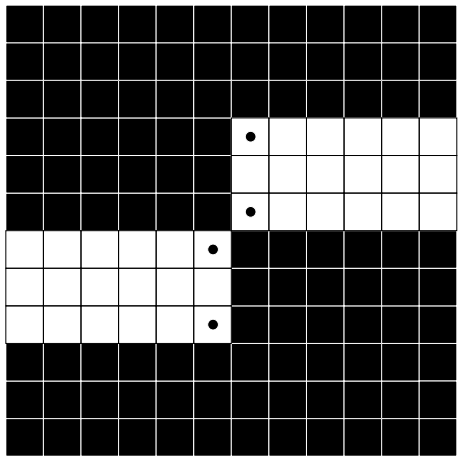


Figure 6.62: If white is considered background, then no black corners will be detected, which results in a "real" edge.

expect this line to be interpolated, but if in a larger window the white pixels are in the majority, then white will be taken as the background. In this case, no interpolation will occur, since different structuring elements are used for object and background corner detection. There will only be detection of white corners (the ones marked with a black dot in the figure), which will not pass the corner validation step, because there are no black corners detected in their neighbourhood. We therefore use a window size that is not too small nor too large. In practice, the area in which we calculate an ordering map with the MSS is an 11×11 window in the original image, since this size gives satisfying interpolation results.

A robust determination of the ideal window size is still an issue. A suggested solution would be to locally adapt the window size in function of the window content, like the number of different colours present. Situations like those in figure 6.62 could be avoided by changing the corner validation rules (making the rules less strict).

The majority ordering has to be performed only in the first iteration step. From then on we know whether the pixel belongs to the foreground or background.

6.5.2 New interpolation method for step 4

In the binary case, the values of the jagged corner pixels and surroundings are replaced by the opposite colour, i.e., black becomes white and white becomes black. With greyscale images, we cannot simply swap the pixel values to the opposite grey value. The grey values of the surrounding pixels must be taken into account, so that the transition between grey values does not occur

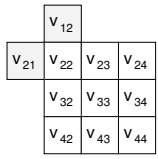


Figure 6.63: The new grey value for pixel v_{22} is the average of the values of its neighbouring pixels v_{12} and v_{21}.

abruptly.[46]

The new pixel value is the average grey value of the pixels that are covered by the background hit-miss structuring element when positioned at the corner pixel. So, suppose we have detected an upper-left corner and we must now change the value of this corner pixel. Figure 6.63 shows the upper-left background corner pixel v_{22}. The image is pixel-replicated with magnification $M = 3$. Its neighbouring pixels v_{23}, v_{24}, v_{32}, v_{33}, v_{34}, v_{42}, v_{43} and v_{44} are classified as background. Its neighbouring pixels v_{12} and v_{21} are classified as object.[47] These are the pixels covered by the background hit-miss structuring element when positioned at v_{22} (see figure 6.21(b) for the structuring element used in the first iteration step). The average grey value, i.e., the new value for pixel v_{22}, is thus $[I(v_{12}) + I(v_{21})]/2$.

The resulting value is thus defined in function of the surrounding values of the other class. For higher magnifications ($M > 3$), also neighbouring pixels of the corner pixels must be interpolated. Their value is replaced with the same value as the one for the corner pixel. For a binary image, the effect is the same as with mmINT, since the average is taken of pixels of the same colour, a colour that is opposite to that of the current pixel. As a consequence, no blurring occurs.

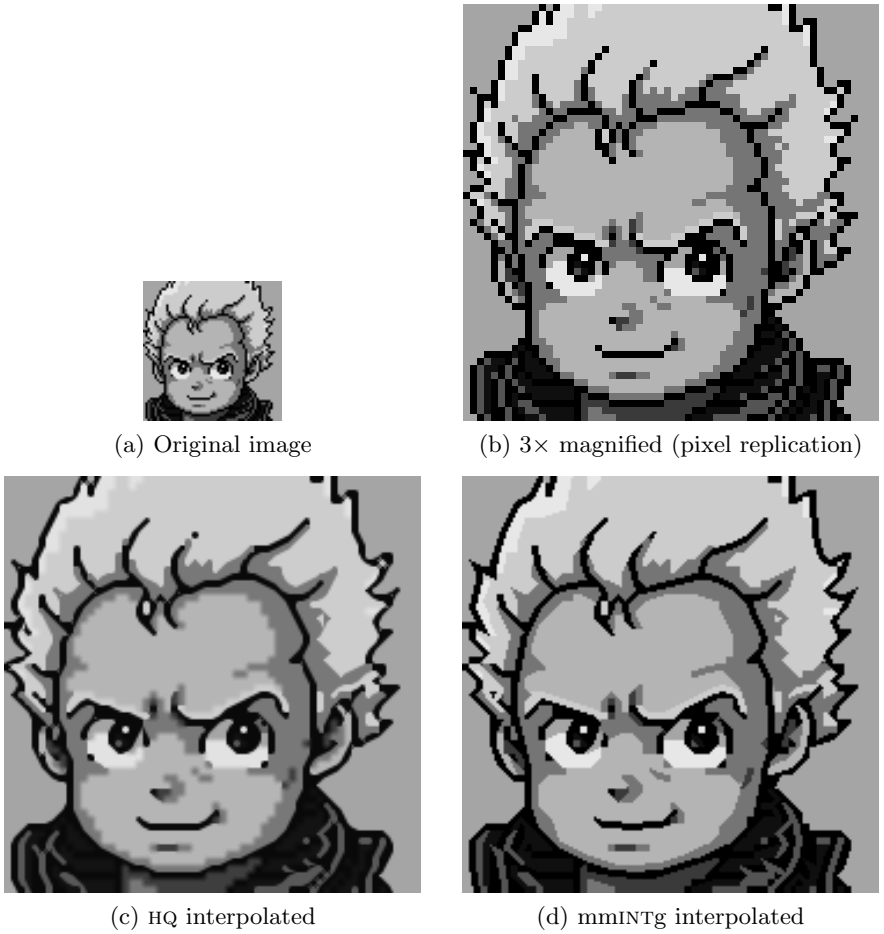
6.5.3 Visual results of mmINTg

mmINT is a technique that is very good at interpolating line art images, like logos, cartoons, maps, etc. Our greyscale extension, mmINTg, also performs very well on this kind of images. On binary images, we can expect mmINTg and mmINT to perform equally well. When the MSS locally produces the opposite

[46] When we described the algorithms of mmINT and mmINTone, we referred to the interpolation part as the "pixel swapping step". In these algorithms, this term is well-chosen, because in a binary image a pixel value can only change between black and white. The interpolation part changes some black pixels into white pixels, and vice versa.

For mmINTg, on the other hand, the term "swapping" is not correct: when a pixel must change value, we calculate this new value with its neighbouring pixels. Thus, the pixel value is not simply swapped to its greyscale complement or to the value of one of its neighbours. Hence, we discard the term "pixel swapping" and replace it by the more general "interpolation".

[47] This is of course true, otherwise the hit-miss transform would not have detected pixel v_{22} as a corner.



(a) Original image

(b) 3× magnified (pixel replication)

(c) HQ interpolated

(d) mmINTg interpolated

Figure 6.64: The interpolation of a greyscale cartoon.

ordering as on the entire image,[48] this will lead to slightly different results as when we process the image with mmINT. Situations like the ones in figure 6.62 will be tackled by mmINTg.

The figures 6.64 and 6.65 show images with clear sharp edges. We compare our greyscale method with HQ [Stepin, 2003], a technique that is very competitive with mmINT when interpolating binary images (see section 6.4). In the first figure, we notice that mmINTg interpolates the lines better and the result is less blurred. The results shown in figure 6.65 look very similar, except that the rotated squares look blurred with HQ. Also, HQ introduces too much un-

[48]This is actually done for mmINT to define the background colour: applying the MSS on a binary image simply reveals the colour most prevalent, which is chosen as background. The difference with mmINTg is that no *local* MSS is performed.

necessary colours in the arrows at the right, i.e., when 8-connectivity lines are present.

In some circumstances, one could argue that HQ performs better than mmINTg. For example, the smiling face and the curved lines in figure 6.65 are smoother (i.e., better anti-aliased) when HQ is used. In the algorithm of mmINTg, the interpolation step produces a binary output when the input image is binary. If we would change the interpolation step so that a binary input image produces a greyscale output, then we might be able to anti-alias the aforementioned objects. A simple post-processing step can also further smoothen the image: figures 6.66 and 6.67 show the interpolation results of the previous examples, using mmINTg, but filtered after interpolation, with a Gaussian 3×3 kernel, with $\sigma = 0.5$. Using an averaging kernel instead of a Gaussian kernel, or using a too large standard deviation σ results in a blurry image. If a binary output is desired, then mmINTg will (overall) produce better results than HQ (whose results have to be binarized), since its results are very similar to mmINT's and this method is superior to HQ, as we discussed in section 6.4.

Figure 6.68 shows a real life scene containing a lot of texture and grey value variation. mmINTg produces quite good results, but because the edges in the image are less sharply defined and more grey values are involved, the interpolation result looks segmented, i.e., the grey value change after interpolation is too abrupt. A suggested improvement is to use a more sophisticated interpolation step. When $M > 3$, the neighbouring pixels of a corner pixel that change too are given the same value as the corner pixel. A calculation of a separate new value for each pixel could increase the quality of the interpolated image.

6.6 Conclusion

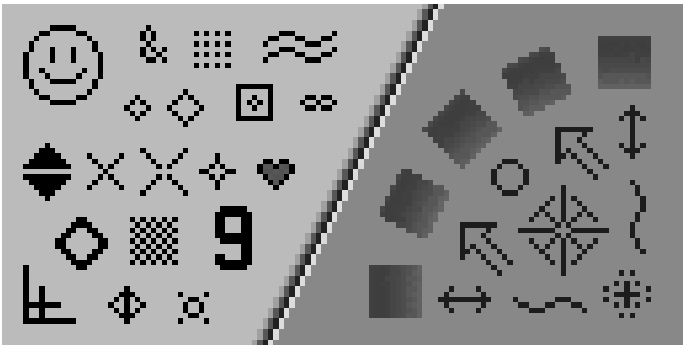
In this chapter, we gave an overview of the convolution-based interpolation. The ideal interpolation kernel, the normalized sinc, can be approximated in many ways, in the form of polynomials or B-splines. Adaptive methods exploit the information available in the low-resolution image, such as edges.

We developed a new interpolation technique for binary images, mmINT, based on mathematical morphology. The primary goal is to detect corners in a pixel-replicated image. The corners are validated, i.e., we check if they belong to a jagged edge, and eventually we swap pixel values.

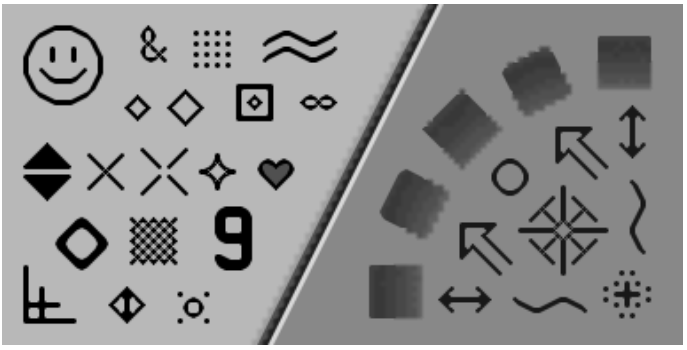
We examined some alternatives, such as mmINTone or the multi-step approach, and we compared our techniques to existing ones. Overall, mmINT produces better results than other methods.

Finally, we extended our interpolation method to greyscale images, using local binarization and a majority ordering.

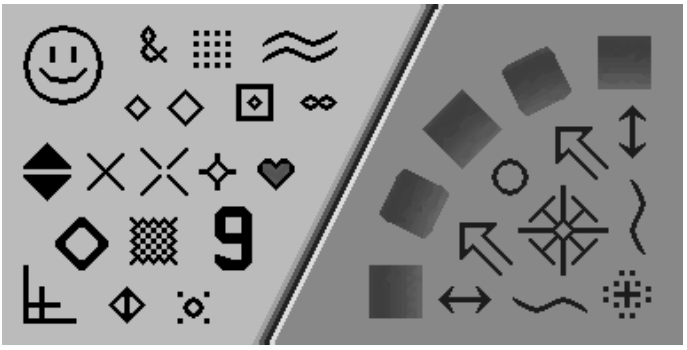
(a) Original image



(b) 3× magnified (pixel replication)



(c) HQ interpolated



(d) mmINTg interpolated

Figure 6.65: The interpolation of a greyscale line graphic.

Figure 6.66: Gaussian blur after interpolation of figure 6.64.

Figure 6.67: Gaussian blur after interpolation of figure 6.65.

(a) Pixel replication ($M = 4$)

(b) mmINTg interpolated

Figure 6.68: The interpolation with mmINTg of a real life scene (a cut-out of the "Lena" image).

Chapter 7

Conclusions

In this thesis, we considered various uses of *mathematical morphology* (MM), a theory based on lattice theory and random geometry. It is used for the investigation of geometric structures. We defined the basic operators, *dilation* and *erosion*, and the *structuring element* in chapter 2. These are the building blocks of mathematical morphology. With these ingredients, we can construct many other, more sophisticated operators.

We explained two greyscale extensions of the binary theory: the *threshold approach* and the *umbra approach*. The t-approach builds on the concept of cross sections, while the u-approach uses concepts like the top surface and umbra. With one of these extensions, we can process intensity images with mathematical morphology, a theory based on set theory and thus originally only applicable to binary images.

In chapter 3, we gave an introduction to the theory of colour; we defined colour, discussed how we perceive it and how we can reproduce it. Different colour spaces exist, such as XYZ, RGB, HSL or $L^*a^*b^*$. They all have their advantages, disadvantages and purposes.

Colours are not totally ordered in an obvious and unique way. Several ways of ordering exist, like marginal ordering or lexicographical ordering. We explained the latter for the HSL colour space. When the colours are ordered, the image can be treated as if it was a greyscale image. Greyscale mathematical morphology can now be applied to the colour image.

We proposed an original colour ordering, the *majority sorting scheme* (MSS). It is a content-dependent ordering that ranks the colours depending on the number of pixels with those colours in the image. The method assumes that background colours are highly present, while details and noise are pixels with rare colours. Morphological operations using the MSS perform well on such images.

We have to take care of several issues. First of all, when the number of colours present in the image is high, then the pixel count per colour will be low for

many colours. It is therefore likely that many colours will be assigned the same ranking, which is not desired because it is then difficult to discriminate between the colours, i.e., between detail and object or between noise and background. Colour quantization is a suggested solution to limit the amount of colours and to obtain a useful ordering.

This quantization step could be further optimized. Possibly there exist better suited colour reduction techniques than the peer group filtering (PGF), and we also do not know what the ideal number of colours is. We solved this problem by requiring each remaining colour to belong to a unique level in the MSS-map.

Another issue is the introduction of false colours after quantization, but we tackled this problem by storing the original colours and linking them to the MSS-map. Equally important colours with a different frequency should be merged. Equally frequent colours can be distinguished by comparing e.g. the grey value difference or distance. Generally, this extra comparison does not change the morphological result significantly.

The majority ordering has its advantage when the images to be processed do not contain natural colours. By this, we mean any colour image with uncorrelated colour combinations, such as logos, maps, post-coloured images, ... At the same time, the restriction to this kind of images is of course a drawback. The MSS can be applied to natural images, but most probably a colour quantization step is necessary to let the MSS work effectively.

Some advantages of the majority ordering are: when we change all colours in a new map of unique colours, i.e, the transformation is bijective, then there is no difference in the majority ordering. The technique does not depend on the colour value, but on the frequency of the colour in the image. We can also use the MSS for greyscale images or binary images.

In chapter 4, we introduced the morphological *pattern spectrum* (PS). It is a size distribution histogram of the objects in an image. The PS is defined for binary and greyscale images. We suggested the *colour pattern spectrum* and the MSS *pattern spectrum* for the spectrum of a colour image. The MSS-PS will first order the colours (or grey values) according to their frequency. Afterwards, a standard pattern spectrum is computed.

For a binary image, the interpretation is quite straightforward: the value $PS(A;B)(n)$ is the number of object pixels that disappear between an opening of image A by structuring element nB and an opening by $(n+1)B$. In the case of a greyscale image, the value denotes a decrease in grey value. Many pixels could have decreased only a little in grey value, or a few pixels a lot. For colour images, the interpretation of the spectrum is even more abstract.

Because of the high computational cost, we have examined several other spectra. The *area pattern spectrum* (APS) is similar to the regular pattern spectrum, but does not take the shape of the objects into account. This is useful if we do not look for a specific shape, but are interested in the size of the objects.

The principle of *opening trees* (OTs) allows us to calculate the pattern spectrum with one-dimensional structuring elements very quickly. It is the preferred method if we want to classify linear objects in function of their length and orientation. A granulometry by maxima of linear openings (OT_{max}) computes an oriented pattern spectrum (OPS). The oriented pattern spectrum can be used for the examination of anisotropic objects with unknown orientation. The pseudo-granulometry by minima of linear openings (OT_{min}) can be used as a rough approximation of the pattern spectrum with a square structuring element.

We proposed the *erosion pattern spectrum* (EPS). It is not a granulometry, but is fast compared to the pattern spectrum. It computes the greyscale difference between two successive erosions. We also suggested a one-dimensional histogram of the Fourier spectrum of the image, the so-called *Fourier pattern spectrum* (FPS).

The calculation time of the pattern spectrum increases quadratically with the maximal object size in the image, and linearly for the erosion pattern spectrum. The time needed for the calculation of the default pattern spectrum can easily increase to more than a few hours (for a 512×512 greyscale image), while the erosion spectrum only needs a few minutes at most. The other techniques are even faster, with the Fourier pattern spectrum the fastest, about one second, followed by the area pattern spectrum and the opening tree algorithm (tens of seconds at most).

We analysed debris particles from wear experiments in chapter 5. The materials tested were polymers that are used in sliding bearings. We used the pattern spectrum (and its alternative spectra) to analyse the correlation of the spectral parameters with the settings of the tribological experiment.

The biggest problem with our analysis of the pattern spectra is the amount of available data. For each of the experimental settings, we had access to only a few pictures of debris particles. In order to perform a more relevant statistical research, much more data would be needed. Also, the number of experimental settings is limited, which hinders the correlation accuracy.

Nevertheless, we have obtained some interesting results. We can confirm, using the pattern spectrum, that the size of the debris particles of the wear of polymer POMH increases with the contact pressure. The sintered polyimide SP-1 has a transition temperature above $T = 180°\text{C}$, when imidization starts to occur. We observe this through the behaviour of the spectral parameters with the temperature.

For the polymer SP-21, we can correlate the coefficient of friction with the average size and roughness obtained from the pattern spectrum. For the material TP, we notice a correlation between the coefficient of friction and the (normalized) average roughness, when the load is fixed.

Since the morphological pattern spectrum (PS) has a high computational cost, it would be good to be able to replace it by a much faster alternative spectrum. It

is not clear which one is best suited to replace the PS. Interesting to note is that the parameters of the pattern spectrum based on the majority sorting scheme, MSS-PS, correlate quite well with the experimental settings of the tribological experiments. The plots of the parameters from the MSS-PS against the load or temperature are similar to those from the regular pattern spectrum. This is a first prove that the MSS-PS has its use as an alternative pattern spectrum (and colour ordering).

In chapter 6, we talked about image interpolation. We gave an overview of the convolution-based interpolation. The ideal interpolation kernel, the normalized sinc, can be approximated in many ways, in the form of polynomials or B-splines. Adaptive methods exploit the information available in the low-resolution image, such as edges.

The interpolation of an image is not without errors. A simple pixel replication introduces a staircase pattern, jagged edges. Other non-adaptive techniques, such as bilinear of bicubic interpolation, blur the interpolation result. A ringing effect is yet another possible artefact (only in the case of greyscale interpolation). While the non-adaptive interpolation methods are better at avoiding these artefacts, other artefacts can be introduced. The interpolation results of adaptive techniques often look segmented, suffer from important visual degradation in finely textured areas or random pixels are created in smooth areas.

We developed a new interpolation technique for binary images, mmINT, based on mathematical morphology. Its primary goal is to detect corners in a pixel-replicated image. The corners are validated, i.e., we check if they belong to a jagged edge, and eventually we swap pixel values.

We examined some alternatives, such as mmINTone or the multi-step approach. They are both faster than mmINT. For example, interpolating a 337×174 binary text sample takes about 21s for mmINT, with magnification $M = 9$. Performing the same algorithm with the multi-step approach (twice $M = 3$) takes only about 14s. mmINTone is the fastest, as it interpolates the image in less than 11s. We believe there is further room for improvement of the speed of the algorithms.

We compared our techniques to existing ones. The comparison is done quantitatively, by means of the peak signal-to-noise ratio (PSNR), as well as qualitatively, by asking a test panel to score the quality of several interpolated images. Overall, mmINT received the highest scores. Only for text, the competing method HQ was appreciated more.

Finally, we extended our interpolation method to greyscale images (mmINTg), using local binarization and a majority ordering. A possible improvement is how we define the window size for the local binarization and for the majority ordering. This choice should be made in a more adaptive and more robust way. A better anti-aliasing step is also a desired feature. Nevertheless, mmINTg produces very nice results on images with sharply defined edges. The computational cost of this implementation is still quite high. A subject for future work could be to search for a way to implement the greyscale interpolation in

one step, like mmINTone. An extension of our greyscale interpolation method to colour is quite straightforward.

Bibliography

[Aldroubi et al., 1992] Aldroubi, A., Unser, M., and Eden, M. (1992). Cardinal Spline Filters: Stability and Convergence to the Ideal Sinc Interpolator. *Signal Processing*, 28(2):127–138.

[Allebach and Wong, 1996] Allebach, J. and Wong, P. (1996). Edge-directed interpolation. In *Proceedings of the IEEE International Conference on Image Processing ICIP'96*, volume 3, pages 707–710, Lausanne, Switzerland.

[Askar et al., 2002] Askar, S., Kauff, P., Brandenburg, N., and Schreer, O. (2002). Fast Adaptive Upscaling of Low Structured Images Using a Hierarchical Filling Strategy. In *Proceedings of VIPromCom 2002, 4th EURASIP IEEE International Symposium on Video Processing and Multimedia Communications*, pages 289–293, Zadar, Croatia.

[Barnett, 1976] Barnett, V. (1976). The Ordering of Multivariate Data. *Journal of the Royal Statistical Society. Series A*, 139(3):318–355.

[Breen and Jones, 1996] Breen, E. and Jones, R. (1996). Attribute Openings, Thinnings, and Granulometries. *Computer Vision and Image Understanding*, 64(3):377–389.

[Castleman, 1996] Castleman, K. (1996). *Digital Image Processing*. Prentice Hall.

[Catmull and Rom, 1974] Catmull, E. and Rom, R. (1974). Class Of Local Interpolating Splines. In *Computer Aided Geometric Design*, pages 317–326, Utah, USA. Academic Press.

[CIE, WWW] CIE (WWW). Commission Internationale de l'Eclairage. `http://www.cie.co.at/`.

[Dartnall et al., 1983] Dartnall, H., Bowmaker, J., and Mollon, J. (1983). Human visual pigments: microspectrophotometric results from the eyes of seven persons. *Proceedings of the Royal Society of London. Series B*, 220(1218):115–130.

[De Witte et al., 2006] De Witte, V., Schulte, S., Kerre, E., Ledda, A., and Philips, W. (2006). Morphological Image Interpolation to Magnify Images with Sharp Edges. In *Lecture Notes in Computer Science, ICIAR'06*, volume LNCS 4141, pages 381–393, Póvoa De Varzim, Portugal.

[De Witte et al., 2005] De Witte, V., Schulte, S., Nachtegael, M., Van der Weken, D., and Kerre, E. (2005). Vector Morphological Operators for Colour Images. In *Proceedings of ICIAR 2005 (International Conference on Image Analysis and Recognition)*, Toronto, Canada.

[Degrieck, 2002] Degrieck, J. (2002). Composieten I. Course.

[Deng et al., 1999] Deng, Y., Kenney, C., Moore, M., and Manjunath, B. (1999). Peer group filtering and perceptual color image quantization. In *Proceedings of IEEE International Symposium on Circuits and Systems*, volume 4, pages 21–24.

[Duda et al., 2000] Duda, R., Hart, P., and Stork, D. (2000). *Pattern Classification*. Wiley Interscience, 2nd edition.

[efg, WWW] efg (WWW). Chromaticity Diagrams Lab Report. `http://www.efg2.com/Lab/Graphics/Colors/Chromaticity.htm`.

[Expert, WWW] Expert, V. (WWW). SBFAQ Part 2: Color Discrimination. `http://www.visualexpert.com/FAQ/Part2/cfaqPart2.html`.

[Ford and Roberts, WWW] Ford, A. and Roberts, A. (WWW). Colour Space Conversions. `http://www.poynton.com/PDFs/coloureq.pdf`.

[Freeman et al., 2002] Freeman, W., Jones, T., and Pasztor, E. (2002). Example-Based Super-Resolution. *IEEE Computer Graphics and Applications*, 22(2):56–65.

[Gonzalez and Woods, 2002] Gonzalez, R. and Woods, R. (2002). *Digital Image Processing*. Prentice Hall, 2nd edition.

[Goutsias, 2001] Goutsias, J. (2001). Morphological Image Analysis. Course.

[Goutsias and Batman, 2000] Goutsias, J. and Batman, S. (2000). *Handbook of Medical Imaging, Volume 2. Medical Image Processing and Analysis*, volume PM80, chapter 4. SPIE Press Monograph.

[Halling, 1973] Halling, J. (1973). *Principles of Tribology*. Macmillan Publishing Ltd.

[Hanbury, 2001] Hanbury, A. (2001). Lexicographical Order in the HLS Colour Space. Technical Report N-04/01/MM, Centre de Morphologie Mathématique Ecole des Mines de Paris.

[Hanbury and Serra, 2001a] Hanbury, A. and Serra, J. (2001a). Mathematical Morphology in the HLS Colour Space. In *BMVC 2001*, pages 451–460, Manchester, UK.

[Hanbury and Serra, 2001b] Hanbury, A. and Serra, J. (2001b). Mathematical Morphology in the L*a*b* Colour Space. Technical Report N-36/01/MM, Centre de Morphologie Mathématique Ecole des Mines de Paris.

[Haralick et al., 1992] Haralick, R., Chen, S., and Kanungo, T. (1992). Recursive opening transform. In *Proceedings of IEEE Computer Vision and Pattern Recognition Conference (CVPR'92)*, pages 560–565, Champaign, Illinois, USA.

[Haralick and Shapiro, 1992] Haralick, R. and Shapiro, L. (1992). *Computer and Robot Vision*, volume 1, chapter 5. Addison-Wesley.

[Harris, 1978] Harris, F. (1978). On the use of windows for harmonic analysis with the discrete Fourier transform. *Proceedings of the IEEE*, 66(1):51–83.

[Heijmans, 1994] Heijmans, H. (1994). *Morphological Image Operators*. Academic Press.

[Honda et al., 1999] Honda, H., Haseyama, M., and Kitajima, H. (1999). Fractal Interpolation for Natural Images. In *Proceedings of the IEEE International Conference on Image Processing ICIP'99*, volume 3, pages 657–661, Kobe, Japan.

[HunterLab, WWW] HunterLab (WWW). `http://www.hunterlab.com/`.

[ICC, WWW] ICC (WWW). International Color Consortium. `http://www.color.org/`.

[Kayargadde and Martens, 1996a] Kayargadde, V. and Martens, J.-B. (1996a). Perceptual characterization of images degraded by blur and noise: experiments. *Journal of the Optical Society of America A*, 13(6):1166–1177.

[Kayargadde and Martens, 1996b] Kayargadde, V. and Martens, J.-B. (1996b). Perceptual characterization of images degraded by blur and noise: model. *Journal of the Optical Society of America A*, 13(6):1178–1188.

[Kerre and Nachtegael, 2000] Kerre, E. and Nachtegael, M., editors (2000). *Fuzzy Techniques in Image Processing*, volume 52 of *Studies in Fuzziness and Soft Computing*. Physica Verlag, Heidelberg.

[Khonsari and Booser, 2001] Khonsari, M. and Booser, E. (2001). *Applied Tribology: Bearing Design and Lubrication*. Wiley Interscience.

[Khosravi and Schafer, 1996] Khosravi, M. and Schafer, R. (1996). Template Matching Based on a Grayscale Hit-or-Miss Transform. *IEEE Transactions on Image Processing*, 5(6):1060–1066.

[Läy, 1987] Läy, B. (1987). Recursive algorithms in mathematical morphology. *Acta Stereologica*, 6(III):691–696.

[Ledda et al., 2005] Ledda, A., Luong, H., De Witte, V., Philips, W., and Kerre, E. (2005). Image Interpolation using Mathematical Morphology. In *Sixth FirW PhD Symposium, Ghent University*, page 094, Ghent, Belgium.

[Ledda et al., 2006a] Ledda, A., Luong, H., Philips, W., De Witte, V., and Kerre, E. (2006a). Greyscale Image Interpolation using Mathematical Morphology. *ACIVS, Proceedings Lecture Notes in Computer Science*, LNCS 4179:78–90.

[Ledda et al., 2006b] Ledda, A., Luong, H., Philips, W., De Witte, V., and Kerre, E. (2006b). Image Interpolation using Mathematical Morphology. In *Proceedings of 2nd IEEE International Conference on Document Image Analysis for Libraries (DIAL'06)*, pages 358–367, Lyon, France.

[Ledda and Philips, 2002] Ledda, A. and Philips, W. (2002). Quantitative Image Analysis with Mathematical Morphology. In *Third FTW PhD Symposium, Ghent University*, page 013, Ghent, Belgium.

[Ledda and Philips, 2005a] Ledda, A. and Philips, W. (2005a). Majority Ordering and the Morphological Pattern Spectrum. *ACIVS, Proceedings Lecture Notes in Computer Science*, LNCS 3708:356–363.

[Ledda and Philips, 2005b] Ledda, A. and Philips, W. (2005b). Majority Ordering for Colour Mathematical Morphology. In *Proceedings of the XIIIth European Signal Processing Conference*, Antalya, Turkey.

[Ledda et al., 2003] Ledda, A., Quintelier, J., Samyn, P., De Baets, P., and Philips, W. (2003). Quantitative Image Analysis with Mathematical Morphology. In *Proceedings of ProRISC 2003*, pages 399–406, Veldhoven, The Netherlands.

[Ledda et al., 2004] Ledda, A., Samyn, P., Quintelier, J., De Baets, P., and Philips, W. (2004). Polymer Analysis with Mathematical Morphology. In *Proceedings of SPS 2004 (the 2004 IEEE Benelux Signal Processing Symposium)*, Hilvarenbeek, The Netherlands.

[Lehmann et al., 1999] Lehmann, T., Gönner, C., and Spitzer, K. (1999). Survey: Interpolations Methods In Medical Image Processing. *IEEE Transactions on Medical Imaging*, 18(11):1049–1075.

[Leong, WWW] Leong, J. (WWW). Number of Colors Distinguishable by the Human Eye. `http://hypertextbook.com/facts/2006/JenniferLeong.shtml`.

[Li and Orchard, 2001] Li, X. and Orchard, M. (2001). New Edge-Directed Interpolation. *IEEE Transactions on Image Processing*, 10(10):1521–1527.

[Liao et al., 2001] Liao, P.-S., Chen, T.-S., and Chung, P.-C. (2001). A Fast Algorithm for Multilevel Thresholding. *Journal of Information Science and Engineering*, 17(5):713–727.

[Liauw Kie Fa, 2001] Liauw Kie Fa, D. (2001). 2xSaI, Super2xSaI, SuperEagle. `http://elektron.its.tudelft.nl/~dalikifa/`.

[Lloyd, 1982] Lloyd, S. (1982). Least Squares Quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137.

[Luong et al., 2005] Luong, H., De Smet, P., and Philips, W. (2005). Image Interpolation using Constrained Adaptive Contrast Enhancement Techniques. In *Proceedings of the IEEE International Conference on Image Processing ICIP'05*, pages 998–1001, Genova, Italy.

[Luong et al., 2006a] Luong, H., Ledda, A., and Philips, W. (2006a). An Image Interpolation Scheme for Repetitive Structures. In *Lecture Notes in Computer Science, ICIAR'06*, volume LNCS 4141, pages 104–115, Póvoa De Varzim, Portugal.

[Luong et al., 2006b] Luong, H., Ledda, A., and Philips, W. (2006b). Non-Local Interpolation. In *Proceedings of the IEEE International Conference on Image Processing ICIP'06*, pages 693–696, Atlanta, Georgia, USA.

[Luong and Philips, 2005] Luong, H. and Philips, W. (2005). Sharp Image Interpolation by Mapping Level Curves. In *Proceedings of Visual Communications and Image Processing 2005*, pages 2012–2022, Beijing, China.

[Maragos, 1989] Maragos, P. (1989). Pattern Spectrum and Multiscale Shape Representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(7):701–716.

[Martens, 2003] Martens, J.-B. (2003). *Image Technology Design*. Springer.

[Martens and Meesters, 1998] Martens, J.-B. and Meesters, L. (1998). Image dissimilarity. *Signal Processing*, 70(3):155–176.

[Matheron, 1975] Matheron, G. (1975). *Random Sets and Integral Geometry*. John Wiley & Sons, Inc., New York.

[Matthews and Rawlings, 1993] Matthews, F. and Rawlings, R. (1993). *Composite Materials: Engineering and Science*. Chapman & Hall.

[Mazzoleni, 2001] Mazzoleni, A. (2001). Scale2x, Scale3x, Scale4x. `http://scale2x.sourceforge.net/`.

[Meijering, 2002] Meijering, E. (2002). A Chronology of Interpolation: From Ancient Astronomy to Modern Signal and Image Processing. *Proceedings of the IEEE*, 90(3):319–342.

[Meijering et al., 2001] Meijering, E., Niessen, W., and Viergever, M. (2001). Quantitative Evaluation Of Convolution-Based Methods For Medical Image Interpolation. *Medical Image Analysis*, 5(2):111–126.

[Meijster and Wilkinson, 2001] Meijster, A. and Wilkinson, M. (2001). Fast Computation of Morphological Area Pattern Spectra. In *Proceedings of the International Conference on Image Processing*, pages 668–671, Thessaloniki, Greece.

[Meijster and Wilkinson, 2002] Meijster, A. and Wilkinson, M. (2002). A Comparison of Algorithms for Connected Set Openings and Closings. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(4):484–494.

[Morse and Schwartzwald, 1998] Morse, B. and Schwartzwald, D. (1998). Isophote-Based Interpolation. In *Proceedings of the IEEE International Conference on Image Processing ICIP'98*, pages 227–231, Chicago, Illinois, USA.

[Morse and Schwartzwald, 2001] Morse, B. and Schwartzwald, D. (2001). Image Magnification Using Level-Set Reconstruction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 333–340.

[Muresan and Parks, 2004] Muresan, D. and Parks, T. (2004). Adaptively quadratic (AQua) image interpolation. *IEEE Transactions on Image Processing*, 13(5):690–698.

[Nachtegael, 2002] Nachtegael, M. (2002). *Vaagmorfologische en vaaglogische filtertechnieken in beeldverwerking*. PhD thesis, Ghent University, Gent, Belgium.

[NIST, WWWa] NIST (WWWa). CODATA Value: electron volt. `http://physics.nist.gov/cgi-bin/cuu/Value?tevj`.

[NIST, WWWb] NIST (WWWb). CODATA Value: Planck constant. `http://physics.nist.gov/cgi-bin/cuu/Value?h`.

[NIST, WWWc] NIST (WWWc). CODATA Value: speed of light in vacuum. `http://physics.nist.gov/cgi-bin/cuu/Value?c`.

[Otsu, 1979] Otsu, N. (1979). A Threshold Selection Method from Gray-Level Histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1):62–66.

[Poynton, 1996] Poynton, C. (1996). *A Technical Introduction to Digital Video*, chapter 6. Gamma, pages 91–114. John Wiley & Sons, Inc.

[Poynton, 1997a] Poynton, C. (1997a). A Guided Tour of Color Space. `http://www.poynton.com/PDFs/Guided_tour.pdf`.

[Poynton, 1997b] Poynton, C. (1997b). Frequently Asked Questions about Color. `http://www.poynton.com/PDFs/ColorFAQ.pdf`.

[Poynton, WWW] Poynton, C. (WWW). `http://www.pynton.com/`.

[Pratt, 2001] Pratt, W. (2001). *Digital Image Processing*. John Wiley & Sons, Inc., 3rd edition.

[Quintelier et al., 2004] Quintelier, J., De Baets, P., Degrieck, J., De Geyter, M., Ledda, A., and Philips, W. (2004). New test-setup for online monitoring of wear mechanisms of polymer matrix composites. In *Proceedings of the 8th International Conference on Tribology*, volume 302, pages 36–42, Veszprem, Hungary.

[Quintelier et al., 2005] Quintelier, J., De Baets, P., Degrieck, J., Ledda, A., Philips, W., Sol, H., and Van Hemelrijck, D. (2005). On-Line Wear Monitoring of Polymer Matrix Composites. *Materials Science Forum*, 475–479:1083–1086.

[Ronse, 1996] Ronse, C. (1996). A Lattice-Theoretical Morphological View on Template Extraction in Images. *Journal of Visual Communication and Image Representation*, 7(3):273–295.

[Samyn, 2007] Samyn, P. (2007). *Tribophysical interpretation of scaling effects in friction and wear for polymers*. PhD thesis, Ghent University, Gent, Belgium. to appear.

[Samyn and De Baets, 2005] Samyn, P. and De Baets, P. (2005). Friction of polyoxymethylene homopolymer in highly loaded applications extrapolated from small-scale testing. *Tribology Letters*, 19(3):177–189.

[Samyn et al., 2004] Samyn, P., De Baets, P., Ledda, A., and Philips, W. (2004). Tribological Characterisation of Polyimides under Atmospheric Conditions at High Temperature. In *Proceedings of the 8th International Conference on Tribology*, volume 302, pages 43–50, Veszprem, Hungary.

[Samyn et al., 2003] Samyn, P., De Baets, P., Schoukens, G., and Hendrickx, B. (2003). Tribological behavior of pure and graphite-filled polyimides under atmospheric conditions. *Polymer Engineering and Science*, 43(8):1477–1487.

[Samyn et al., 2005] Samyn, P., Quintelier, J., De Baets, P., and Ledda, A. (2005). Image processing techniques for characterisation of polymer wear debris. In *Proceedings of the International Conference on Condition Monitoring*, pages 423–432, Cambridge, UK.

[Samyn et al., 2006] Samyn, P., Quintelier, J., Schoukens, G., and De Baets, P. (2006). The Sliding Behaviour of Sintered and Thermoplastic Polyimides Investigated by Thermal and Raman Spectroscopic Measurements. In *Proceedings of the 12th Nordic Symposium in Tribology (NORDTRIB'06)*, Helsingør, Denmark.

[Serra, 1982] Serra, J. (1982). *Image Analysis and Mathematical Morphology*, volume 1. Academic Press, New York.

[Serra, 1988] Serra, J. (1988). *Image Analysis and Mathematical Morphology: Theoretical Advances*, volume 2. Academic Press, New York.

[Shannon, 1948] Shannon, C. (1948). A Mathematical Theory of Communication. *The Bell System Technical Journal*, 27:379–423, 623–656.

[Sivakumar et al., 2000] Sivakumar, K., Patel, M., Kehtarnavaz, N., Balagurunathan, Y., and Dougherty, E. (2000). A Constant-time Algorithm for Erosions/Dilations with Applications to Morphological Texture Feature Computation. *Real-Time Imaging*, 6(3):223–239.

[Soille, 2003] Soille, P. (2003). *Morphological Image Analysis: Principles and Applications*. Springer-Verlag, 2nd edition.

[Stepin, 2003] Stepin, M. (2003). hq3x Magnification Filter. `http://www.hiend3d.com/hq3x.html`.

[Stiles and Burch, 1959] Stiles, W. and Burch, J. (1959). N.P.L. Colour-matching Investigation: Final Report (1958). *Journal of Modern Optics*, 6(1):1–26.

[Tarjan, 1975] Tarjan, R. (1975). Efficiency of a Good But Not Linear Set Union Algorithm. *Journal of the ACM*, 22(2):215–225.

[Tarjan, 1983] Tarjan, R. (1983). *Data Structures and Network Algorithms*. SIAM.

[Trussell et al., 2005] Trussell, H., Saber, E., and Vrhel, M. (2005). Color image processing [basics and special issue overview]. *IEEE Signal Processing Magazine*, 22(1).

[Tschumperlé, 2002] Tschumperlé, D. (2002). *PDE's Based Regularization of Multivalued Images and Applications*. PhD thesis, Université de Nice — Sophia Antipolis, Nice, France.

[Unser, 1999] Unser, M. (1999). Splines: A Perfect Fit for Signal and Image Processing. *IEEE Signal Processing Magazine*, 16(6):22–38.

[Unser et al., 1993a] Unser, M., Aldroubi, A., and Eden, M. (1993a). B-Spline Signal Processing: Part I — Theory. *IEEE Transactions on Image Processing*, 41(2):821–833.

[Unser et al., 1993b] Unser, M., Aldroubi, A., and Eden, M. (1993b). B-Spline Signal Processing: Part II — Efficient Design And Applications. *IEEE Transactions on Image Processing*, 41(2):834–848.

[van Beek, 2004] van Beek, A. (2004). *Machine lifetime performance and reliability*. TU Delft.

[Vansteenkiste et al., 2003] Vansteenkiste, E., Ledda, A., Stippel, G., Huysmans, B., Govaert, P., and Philips, W. (2003). Segmenting Leukomalacia using Textural Information and Mathematical Morphology. In *Proceedings of ProRISC 2003*, pages 441–446, Veldhoven, The Netherlands.

[Vansteenkiste et al., 2006a] Vansteenkiste, E., Philips, W., Conneman, N., Lequin, M., and Govaert, P. (2006a). Segmenting Periventricular Leukomalacia in Preterm Ultrasound Images. *Ultrasound in Medicine and Biology*. in review.

[Vansteenkiste et al., 2006b] Vansteenkiste, E., Van der Weken, D., Philips, W., and Kerre, E. (2006b). Evaluation of the Perceptual Performance of Fuzzy Image Quality Measures. In *LNCS: Knowledge-Based and Intelligent Information and Engineering Systems, KES*, volume 4251, pages 623–630, Bournemouth, UK.

[Vansteenkiste et al., 2006c] Vansteenkiste, E., Van der Weken, D., Philips, W., and Kerre, E. (2006c). Perceived Image Quality Measurement of state-of-the-art Noise Reduction Schemes. *ACIVS, Proceedings Lecture Notes in Computer Science*, LNCS 4179:114–126.

[Vincent, 1992] Vincent, L. (1992). Morphological Area Openings and Closings for Greyscale Images. In *Proceedings of the NATO Shape in Picture Workshop*, pages 197–208, Driebergen, The Netherlands.

[Vincent, 1994a] Vincent, L. (1994a). Fast Grayscale Granulometry Algorithms. In *Proceedings of the International Symposium on Mathematical Morphology (ISMM'94)*, pages 265–272, Fontainebleau, France.

[Vincent, 1994b] Vincent, L. (1994b). Fast Opening Functions and Morphological Granulometries. In *Proceedings of SPIE on Image Algebra and Morphological Image Processing V*, volume 2300, pages 253–267, San Diego, California, USA.

[Vincent, 1995] Vincent, L. (1995). Lecture Notes on Granulometries, Segmentation, and Morphological Algorithms. In *Summer School on Morphological Image and Signal Processing*, Zakopane, Poland.

[Vincent, 2000] Vincent, L. (2000). Granulometries and Opening Trees. *Fundamenta Informaticae*, 41(1–2):57–90.

[W3C, WWW] W3C (WWW). World Wide Web Consortium. `http://www.w3.org/`.

[Weisstein, WWW] Weisstein, E. (WWW). Totally Ordered Set. From MathWorld–A Wolfram Web Resource. `http://mathworld.wolfram.com/TotallyOrderedSet.html`.

[Wikipedia, WWWa] Wikipedia (WWWa). Jaggies. `http://en.wikipedia.org/wiki/Jaggies`.

[Wikipedia, WWWb] Wikipedia (WWWb). Vector graphics. `http://en.wikipedia.org/wiki/Vector_graphics`.

[Wilkinson and Roerdink, 2000] Wilkinson, M. and Roerdink, J. (2000). Fast morphological attribute operations using Tarjan's union-find algorithm. In *Proceedings of the International Symposium on Mathematical Morphology and its Applications to Image and Signal Processing*, pages 311–320, Palo Alto, California, USA.

[Yuan, 1991] Yuan, L.-P. (1991). Fast algorithm for size analysis of irregular pore areas. In *Proceedings of SPIE on Nonlinear Image Processing II*, volume 1451, pages 125–136, San Jose, California, USA.

[Zsidai et al., 2002] Zsidai, L., De Baets, P., Samyn, P., Kalacska, G., Van Peteghem, A., and Van Parys, F. (2002). The tribological behaviour of engineering plastics during sliding friction investigated with small-scale specimens. *Wear*, 253(5–6):673–688.

Index