Beeldrestauratie met behulp van inpainting

1. Inleiding

Inpainting is een beeldrestauratietechniek waarbij ontbrekende (of beschadigde) delen van een beeld opgevuld worden, bij voorkeur op basis van informatie in het beeld zelf. Bij het reconstrueren van de ontbrekende pixelwaarden, dient de structuur en textuurinformatie in het beeld zo goed mogelijk overeen te komen met omliggende gebieden, zodat visueel zo weinig mogelijk artefacten zichtbaar zijn.

De gebruiker duidt eerst ofwel zelf het gebied aan dat moet opgevuld worden, bijv. met een tekenprogramma, ofwel doet hij dat via gespecialiseerde algoritmes die automatisch beschadigde delen van een beeld herkennen (bijv. vlekken, krassen, paginavouwen enz.). Een voorbeeld wordt gegeven in Fig. 1.

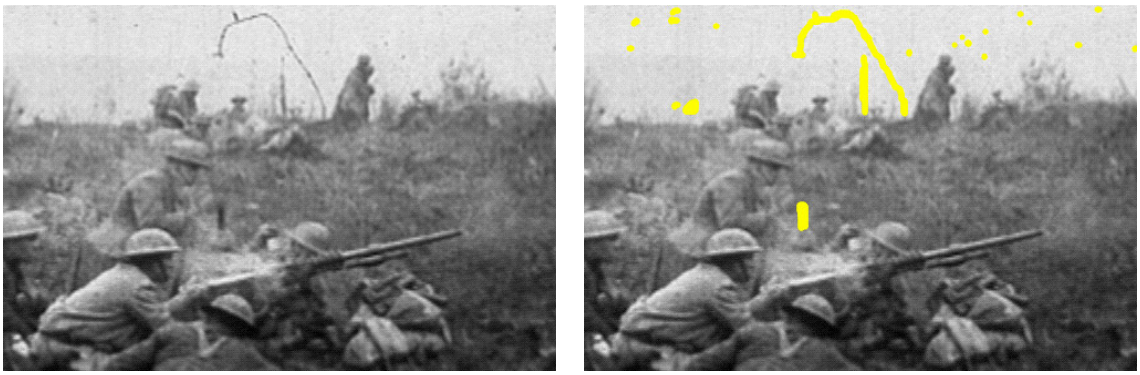

Fig. 1. (links) beschadigd beeld uit een opname tijdens WWI. (rechts) markering van de beschadigde gebieden in het geel.

In dit project zullen we een blokgebaseerde inpaintingstechniek ontwikkelen. In Fig. 2 wordt het basisprincipe hiervan geïllustreerd: veronderstel dat we blokken van grootte 5x5 gebruiken en we willen de pixel *p* in het midden van het blok in Fig. 2c reconstrueren. Dan stellen we eerst een 5x5 masker samen dat aangeeft welke pixelwaarden binnen het blok gekend zijn en welke niet (in rood aangeduid in de figuur; de centrale pixel *p* behoort uiteraard niet tot dit masker). Vervolgens zoeken we in het volledige beeld naar de blokken die het beste "gelijken" op het blok waarin we de ontbrekende pixelwaarde willen schatten, door gebruik te maken van het masker. Door de centrale pixels in de gevonden blokken te combineren kunnen we vervolgens de ontbrekende pixelwaarde schatten. Dit proces wordt herhaald totdat er geen ontbrekende pixels meer overblijven.

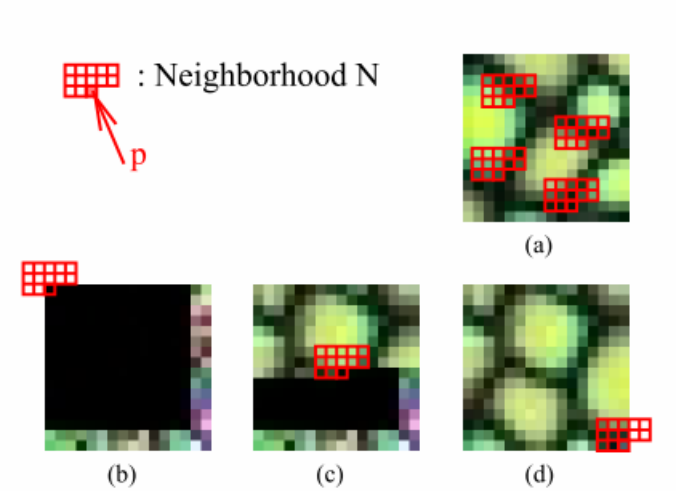

Fig. 2. Het basisprincipe van blokgebaseerde inpainting (figuur genomen uit [4]).

Materiaal dat ter beschikking staat:

Artikels:

[1] Z. Tauber, Z.N. Li, and M.S. Drew, " Review and preview: Disocclusion by impainting for image-based rendering", IEEE Trans. on Systems, Man, and Cybernetics, Part C: Applications and Reviews, 37(4), 2007, 527-540.

[2] Steven L. Kilthau, Mark S. Drew, Torsten Möller, "Full Search Content Independent Block Matching based on the Fast Fourier Transform", Proceedings of IEEE International Conference on Image Processing, pp. 669-672, September 2002 Zie ook de bijhorende poster op http://www.cs.sfu.ca/~mark/ftp/Icip02/images/icip_2002.jpg

[3] W. Philips, "Niet-lineaire filtertechnieken: morfologische filters, " lesnota's beeldverwerking

[4] Li-Yi Wei and Marc Levoy, "Fast Texture Synthesis using Tree-structured Vector Quantization," SIGGRAPH 2000

Software: matlab programma

2. Opgave

2.1. Analyse van het programma

Bestudeer eerst het programma inpaint.m, dat het algoritme beschreven in de inleiding reeds gedeeltelijk implementeert. Om gemakkelijk met kleurenbeelden te kunnen werken, wordt er eerst een luminantiebeeld berekend. Dit luminantiebeeld wordt vervolgens gebruikt om blokken met elkaar te vergelijken.

- Ondanks het feit dat blokken op deze manier sneller kunnen vergeleken worden (er dient slechts rekening gehouden te worden met één kanaal), heeft het gebruik van een luminantiebeeld hier een belangrijk nadeel. Leg uit.

- Hoe wordt er omgegaan met de randen van het beeld, wanneer blokken zich aan de rand bevinden? Is dit een goede keuze? Welke alternatieve technieken zou je hiervoor kunnen bedenken die beter zijn voor deze taak? (Je hoeft dit niet te implementeren)

- Indien verschillende kandidaat-blokken gevonden worden, die goed op het blok met de ontbrekende pixelwaarde gelijken, welk blok wordt er dan door het programma gekozen? Wat zijn de voor- en nadelen van deze aanpak (denk aan beeldruis, …)?

2.2. Herkenning van het opvulgebied

Eerst dienen de gele markeringen van de gebruiker gedetecteerd te worden door het programma. Schrijf hiervoor code, die na het inlezen van het beeld een binair masker `Mask` aanmaakt, die aangeeft waar de gele markeringen zich bevinden (de matrix `Mask` bevat een 0 in geval van een gele markering op die positie; een 1 anders). Gebruik hiervoor het beeld *lincoln_masked.png.*

- Leg uit hoe je deze gele gebieden op een correcte manier kan detecteren. Hou er rekening mee dat de kleurwaarde voor geel die gebruikt wordt niet exact op voorhand gekend is (dit kan bijv. R=255, G=230, B=40 zijn).
- In een volgende stap zet het programma alle pixelwaarden op nul, op alle posities waar het masker een nul bevat. Bekijk dit beeld met behulp van de functie `imshow`. Welk probleem stel je vast? Welke oplossing kan je hiervoor bedenken? Implementeer de gekozen oplossing.

2.3. De berekening van de gelijkenis tussen twee blokken

Om de gelijkenis tussen twee blokken te berekenen, zullen we in dit project het gemiddelde kwadratische verschil (MSD=mean square difference) gebruiken. Hou er rekening mee dat tijdens deze berekening, enkel pixelwaarden mogen gebruikt worden die gekend zijn. Bekijk nu de binnenste lus in het programma. In deze lus wordt de pixelwaarde op positie (x,y) geschat, door het referentieblok gecentreerd rond positie (x,y) te vergelijken met alle andere blokken in het beeld. Omdat deze taak echter zeer rekenintensief is en vereist om exhaustief voor elke ontbrekende pixel het volledige beeld af te lopen, zullen we een techniek gebruiken die sneller is maar die hetzelfde eindresultaat geeft. Deze versnellingstechniek, genaamd "FFT-block matching", is beschreven in Sectie 2 van [2].

- Implementeer de techniek die in [2] voorgesteld wordt. Maak hierbij gebruik van `submask`, die aangeeft welke pixelwaarden in het blok gekend zijn (dit is niet beschreven in [2], maar kan gemakkelijk opgelost worden door de template matrix met `submask` puntsgewijs te vermenigvuldigen, waarom?). *Opmerking*: de "Windowed Sum Squared Table" in [2] hoeft niet gebruikt te worden, en mag vervangen worden door één enkele convolutie (eventueel via FFT) op de kwadratenbeelden.
- Het resultaat van de berekening dient de matrix `msd` te zijn, die voor elke positie in het beeld aangeeft wat het gemiddelde kwadratische verschil is van het blok op deze positie met het referentieblok. Controleer best ook of de berekende `msd` waarden correct zijn (door bijv. te vergelijken met een brute-force implementatie).
- Test nu het programma op een aantal meegegeven beelden. Hoewel de MSD berekening op het eerste zicht correct verloopt, treedt er toch nog belangrijk probleem op. Hoe kan je dit oplossen?

2.4. De volgorde om het beeld te doorlopen

Bemerk dat het programma de ontbrekende pixels opvult van boven naar onder. Dit heeft als nadeel dat "foutieve" schattingen gemakkelijk verder propageren en uiteindelijk een waardeloos eindresultaat opleveren.

- Welke betere oplossing, gebaseerd op mathematische morfologie, kan je hiervoor bedenken? Pas hiervoor het programma aan. (tip: het kan nuttig zijn om in meerdere iteraties te werken)
- Hoe zou je het eindresultaat onafhankelijk kunnen maken van de volgorde waarin het beeld doorlopen wordt? Test of deze oplossing ook kwantitatief beter is door de piek-signaal-ruisverhouding (PSNR) te vergelijken met een origineel beeld. Doe dit voor een drietal beelden. Wat zijn je bevindingen?
- Experimenteer nu met verschillende blokgroottes. Welke afwegingen kan je hier maken (beeldruis, nauwkeurigheid)? Hoe kies je de blokgrootte best?

2.5. Toepassingen

In dit laatste onderdeel van het project zullen we het inpainting-programma toepassen in enkele realistische situaties:

1. ***Verwijderen van vlekken en krassen in oude films*.** Gebruik hiervoor het beeld *ww1_original.png.* Hoe zou je de puntvlekken of lijnkrassen op automatische wijze kunnen detecteren? Werk deze oplossing uit, en restaureer het beeld.

2. ***Verbergen van huizen in luchtfoto's.*** Pas het programma toe op het beeld *houses_erased.png.* Welk praktisch probleem stel je vast? Wat zou je hieraan kunnen doen (je hoeft dit niet te implementeren)

3. ***Verwijderen van tekst in een beeld.*** Test nu het programma op het beeld *houses_text.png* en beschrijf je bevindingen.

4. ***Pakket loss concealment (PLC).*** PLC is een techniek om pakketten, die verloren zijn gegaan bij transmissie over een netwerk, te maskeren. Voor de eenvoud veronderstellen we dat het beeld in JPEG formaat verstuurd wordt en dat elk beeldblok verstuurd wordt in één pakket. Wanneer er willekeurig pakketverlies optreedt, krijgen we een beeld zoals in *lena_packetloss_10%.png*. Ga na of de inpainting-techniek hier ook bruikbaar is.

Fig. 3. Enkele testbeelden gebruikt voor opgave 2.5

Nuttige Matlab commando's: fft2, fftshift, ifft2, imdilate, imerode, ones, strel, x(end:-1:1)