A study of the 2D - SIFT algorithm

Dimitri Van Cauwelaert

HOGESCHOOL GENT
[LID VAN DE ASSOCIATIE UNIVERSITEIT GENT]

ASSOCIATIE UNIVERSITEIT GENT

Introduction

SIFT : Scale invariant feature transform

Method for extracting distinctive invariant features from images that can be used to perform reliable matching between different views of an object or scene

invented by David Lowe in 1999

Introduction

Feature: local property of an image

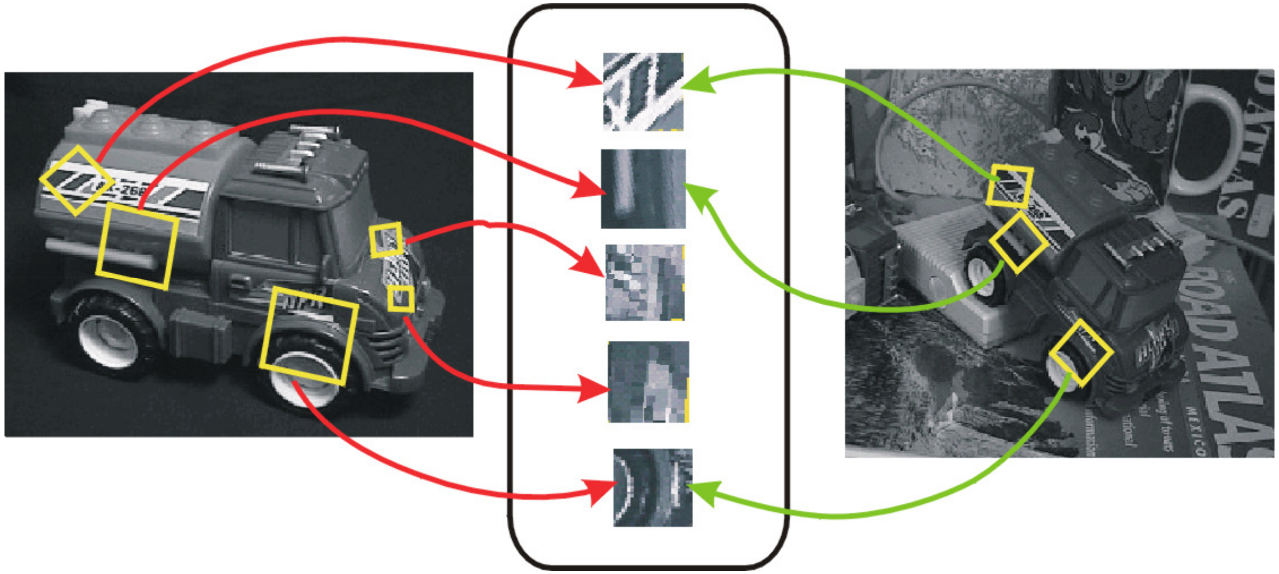
Invariant to:

- Image scaling
- Rotation

Robust matching across:

- Substantial range of affine distortion
- Change in 3D viewpoint
- Addition of noise
- Change in illumination

Introduction



Introduction

Based on a model of the behavior of complex cells in the cerebral cortex of mammalian vision

Recent research - Edelman, Intrator and Poggio – indicates that if feature position is allowed to shift over a small area while maintaining orientation and spatial frequency reliable matching increases significantly

The algorithm

For both the image an the training image,
feature extraction based on:

•Scale space extrema detection
•Keypoint localization
•Orientation assignment
•Keypoint descriptor

Large amounts of features are generated
⇒will provide more reliable matching
⇒Detection of small objects in cluttered backgrounds

Typically: 2000 stable features in an image of 500x500 pixels

The algoritm

Extraction, using a fast nearest neighbor algorithm, of candidate matching features based on the Euclidean distance between the descriptor vectors

Clustering of matched features that agree on object location and pose

These clusters are subject to further detailed verification

$\Rightarrow$Least squared estimate for an affine approximation to the object pose

$\Rightarrow$Outliers are discarded to improve the reliability of the matching

The algorithm

Cascade filtered approach

The more computationally challenging operations are applied to items that pass initial testing.

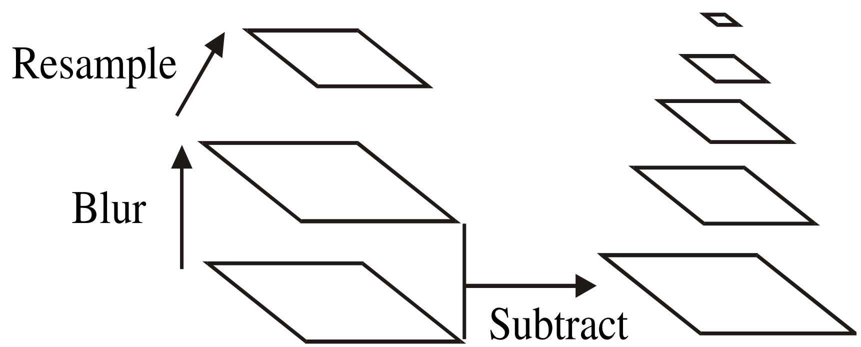
For small images images near real-time computation

The algorithm – detection of scale space extrema

Building a scale space pyramid:

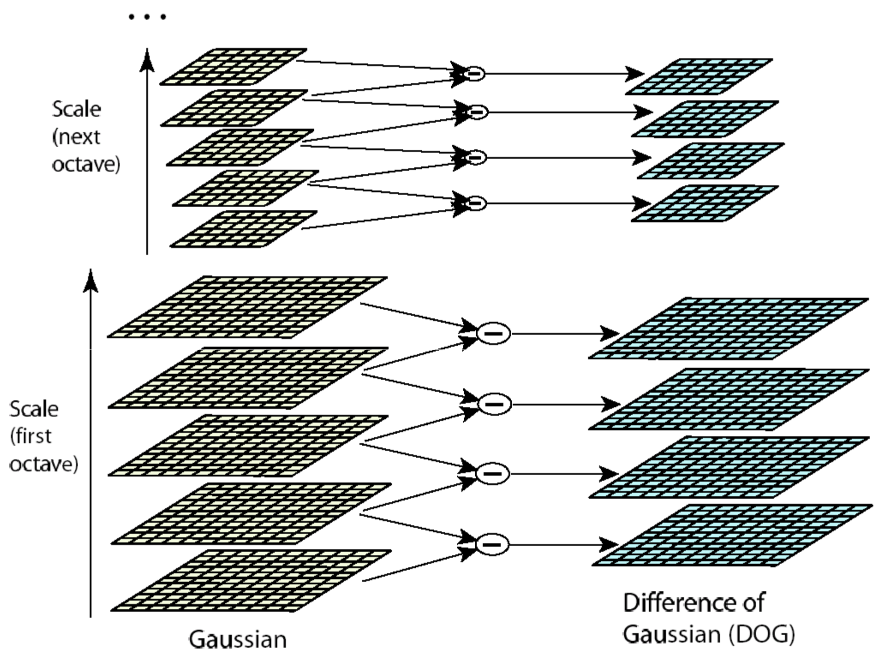
All scales must be examined to identify scale-invariant features

An efficient function is to compute the Difference of Gaussian (DOG) pyramid (Burt & Adelson, 1983)



The algorithm – detection of scale space extrema

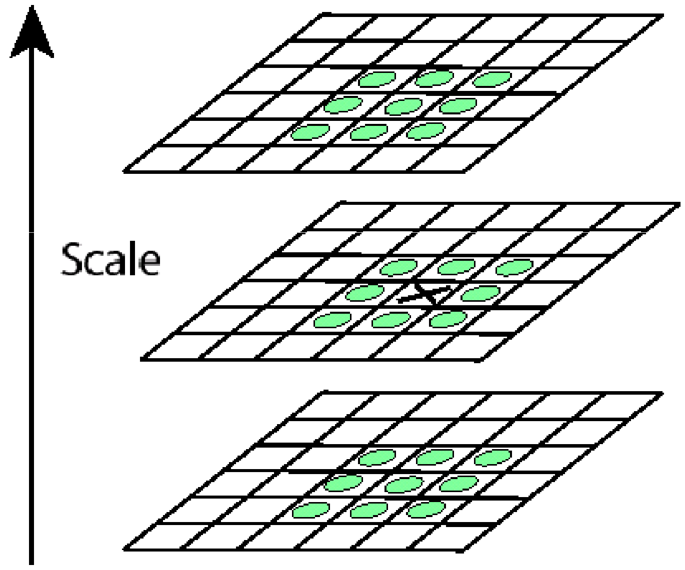
Scale space processed one octave at the time



Resamping to limit computations, we can do this without aliasing problems because the blurring is limiting the higher spatial frequencies

The algorithm – detection of scale space extrema

Within one DOG scale look for minima and maxima considering the current scale, the scale above and the scale below



The algorithm – orientation assignment

Goal: expressing the feature descriptor relatively to this orientation and thus achieving rotational invariance

A circular Gaussian weighted window (radius depending on the scale of the keypoint) is taken around the keypoint

For each pixel within this window the magnitude and the orientation of the gradient is determined.

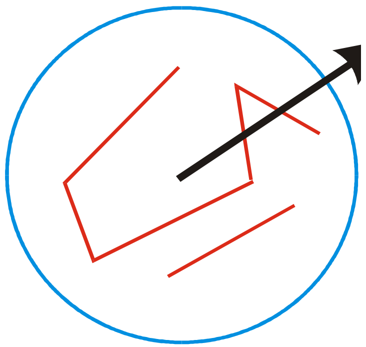
A 36 bins (covering 360 degrees) orientation histogram is filled using the Gaussian window and gradient magnitude as weights.

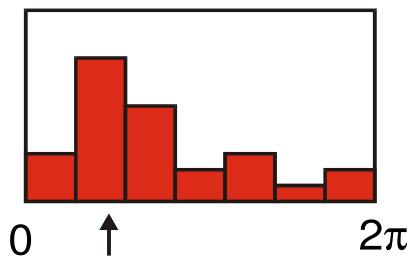
The algorithm – orientation assignment

Highest peak in the smoothed histogram is the assigned orientation

Peaks having more than 80 % of the value of this highest peaks are also assigned as possible orientations

A parabola is fit to the 3 histogram values closest to the peak to interpolate the peak position for better accuracy





The algorithm – the local image descriptor

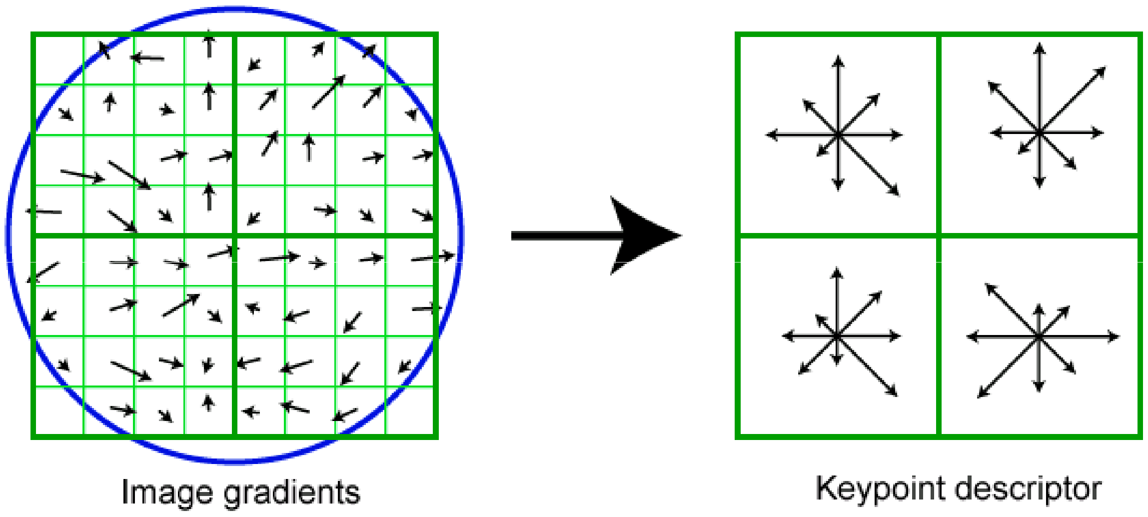
Again consider a Gaussian weighting function around the keypoint location

In this window gradient magnitudes and orientation are rotated according to the assigned keypoint orientation

The 16x16 samples around the keypoint are grouped in a 4x4 array.

In each array the samples are added to orientation bins (here 8) using again the Gaussian window as well as the gradient magnitude as weighting functions

The algorithm – the local image descrciptor



The algorithm – the local image descriptor

To avoid significant changes in the descriptor vector as one pixel would shift from one pixel group to another. Shifting pixels in and out of a group is done using an additional linear weighting function

Dimensionality:

Using r orientation bins for each pixel group
Using and n x n pixel group array

The resulting vector describing the feature has r x n x n dimensions

The algorithm – matching to large databases

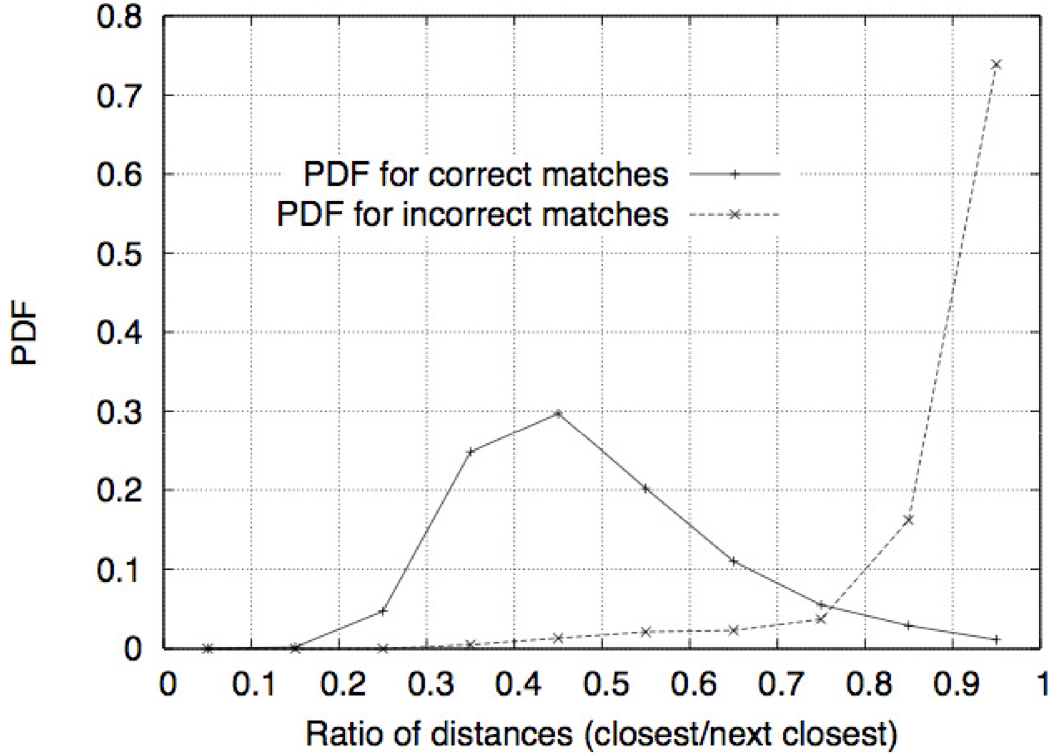
Matching features in two images:

Using the Euclidean distance between the two descriptor vector and then treshholding them would be intuitive, but appears not to give reliable results

A more effective measure is obtained by comparing the distance of the closest neighbor to that of the second closest neighbor

Distance of correct match must be significantly greater than the distance of the second closest neighbor in order to avoid ambiguity

The algorithm – matching to large databases



Threshold of 0.8 provided excellent separation

The algorithm – matching to large databases

No algorithms are known that can identify the exact nearest neighbor of points in high dimensional spaces that are more efficient than exhaustive search

Algorithms such as K-d tree provide no speedup

Approximate algorithm called best bin first (BBF)

$\Rightarrow$Bins in feature space are searched in order of their closest distance from the query location (priority queue)

$\Rightarrow$Only the first x bins are tested

$\Rightarrow$Returns the closest neighbor with high probability

$\Rightarrow$Drastic increase in speed

The algorithm – matching to large databases

The Hough transform identifies clusters of features with a consistent interpretation by using each feature to vote for all object poses that are consistent with the feature.

The affine transformation has 6 degrees of freedom, thus using a minimum of 3 points from a cluster we can make an estimate for the affine transformation between the image and the training image

$\Rightarrow$Clusters of less then 3 features are discarded

$\Rightarrow$Using all the features within a cluster, a least-squared solution in determined for the fitted affine transformation

The algorithm – matching to large databases

Each feature in the cluster is now checked not to deviate to much from the least square solution. If it does the feature is discarded and the least square solution is recalculated

=> After several iterations (providing the number of remaining features in the cluster does not fall below 3) a reliable affine transformation is determined.

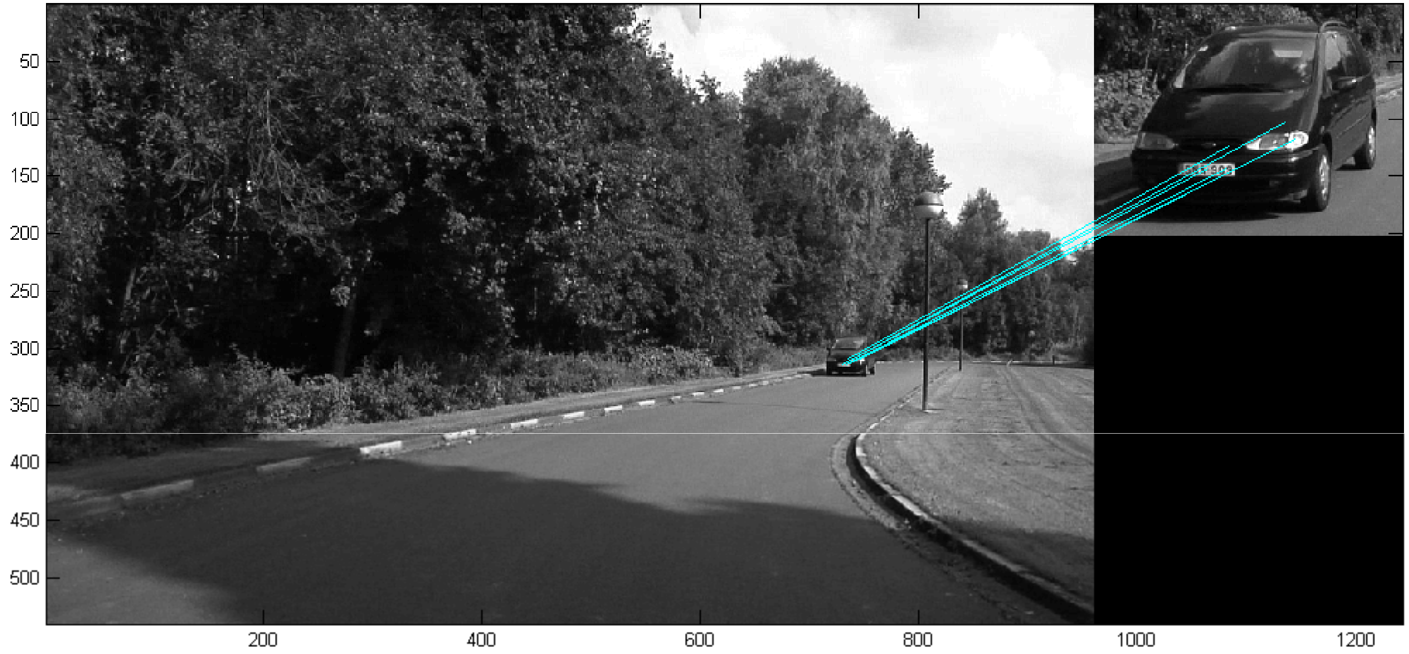
Demo – recognition of a car

We will use a template of a car and try to match it against a scene in which this car is present

template

Demo – recognition of a car

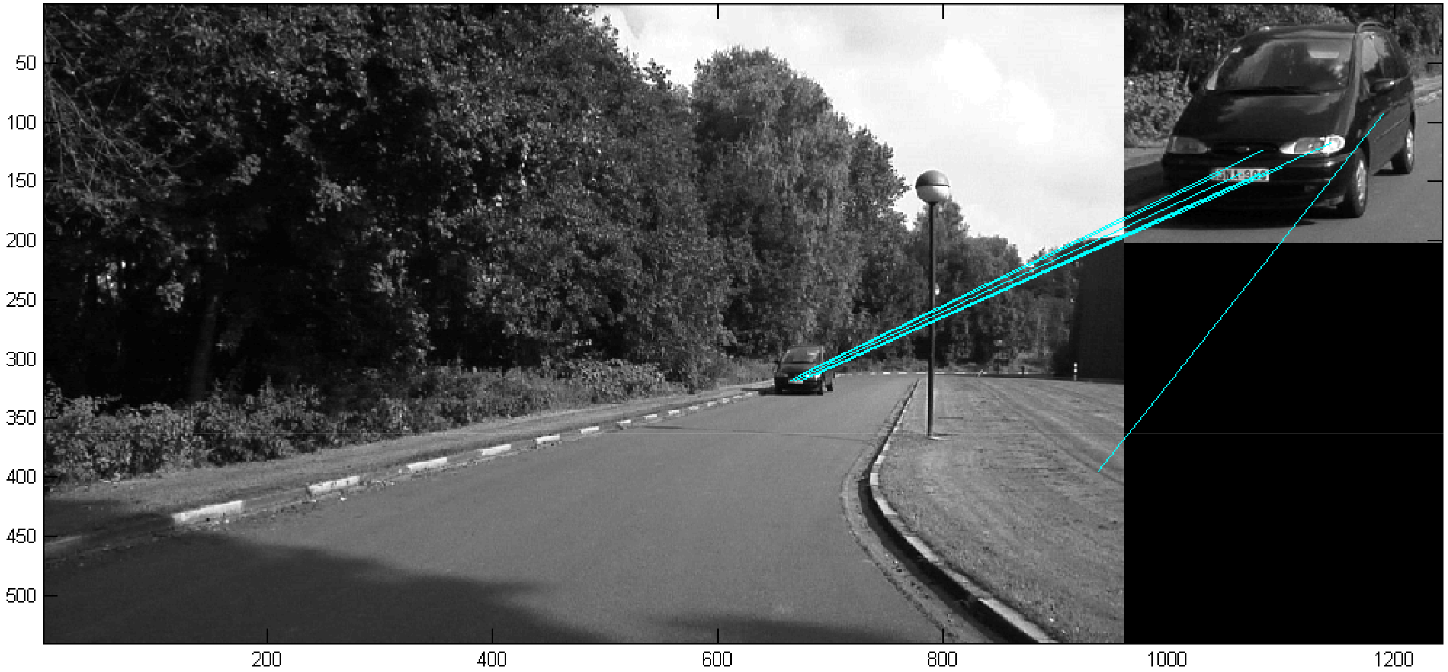
t = 0 ms



Five points from the template are correctly identified in the scene

Demo – recognition of a car

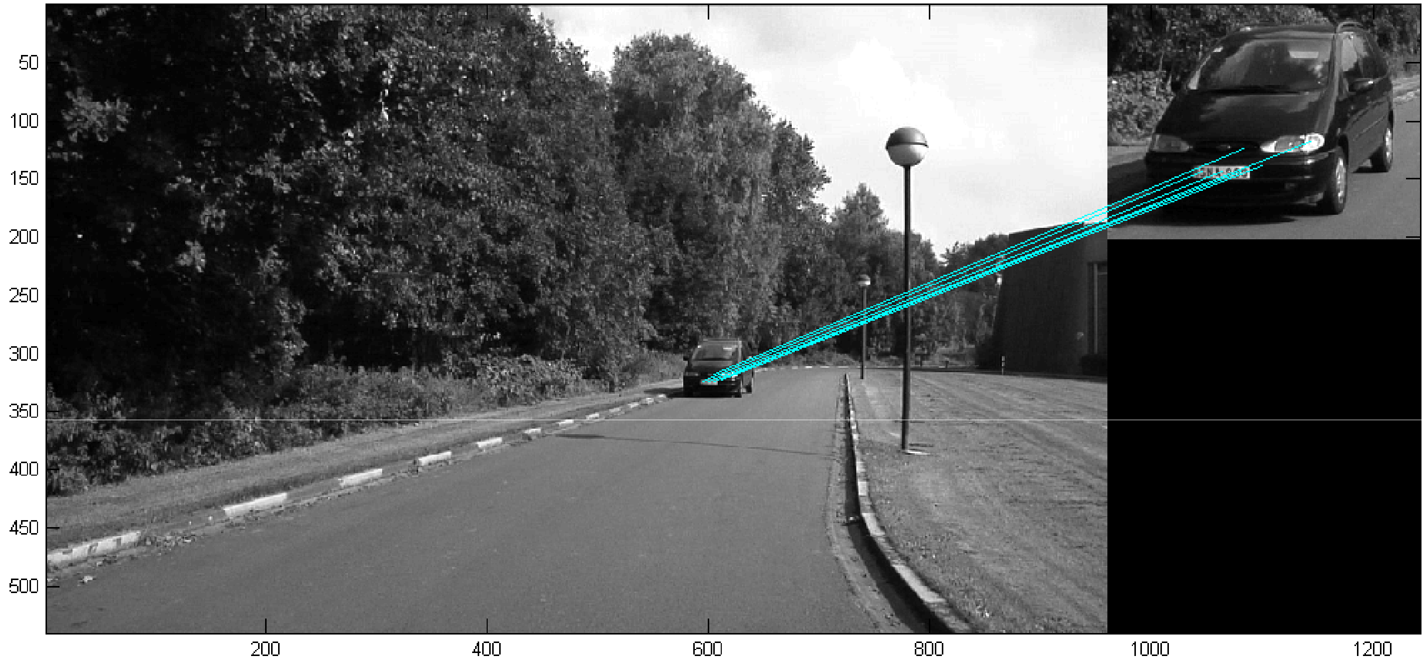
t = 400 ms



Six points from the template are correctly identified in the scene, however also note the incorrect match in the right of the image

Demo – recognition of a car

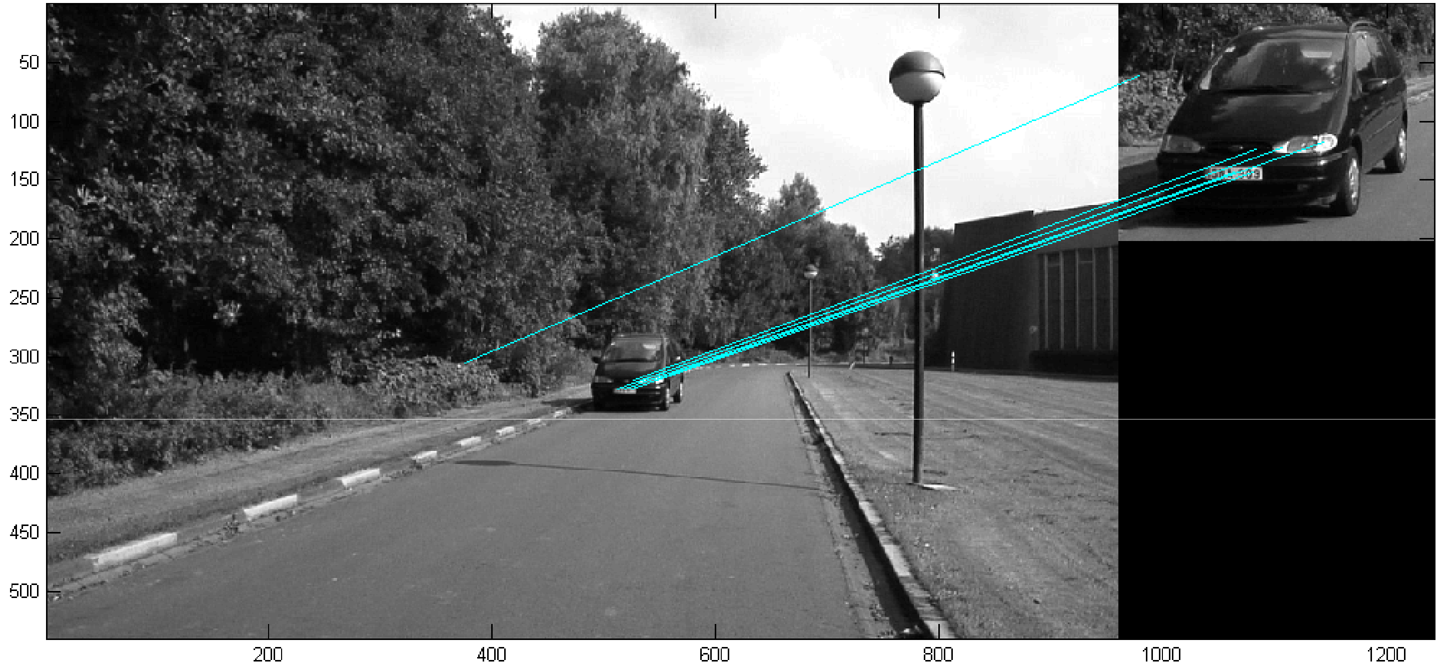
t = 800 ms



Six points from the template are correctly identified in the scene.

Demo – recognition of a car

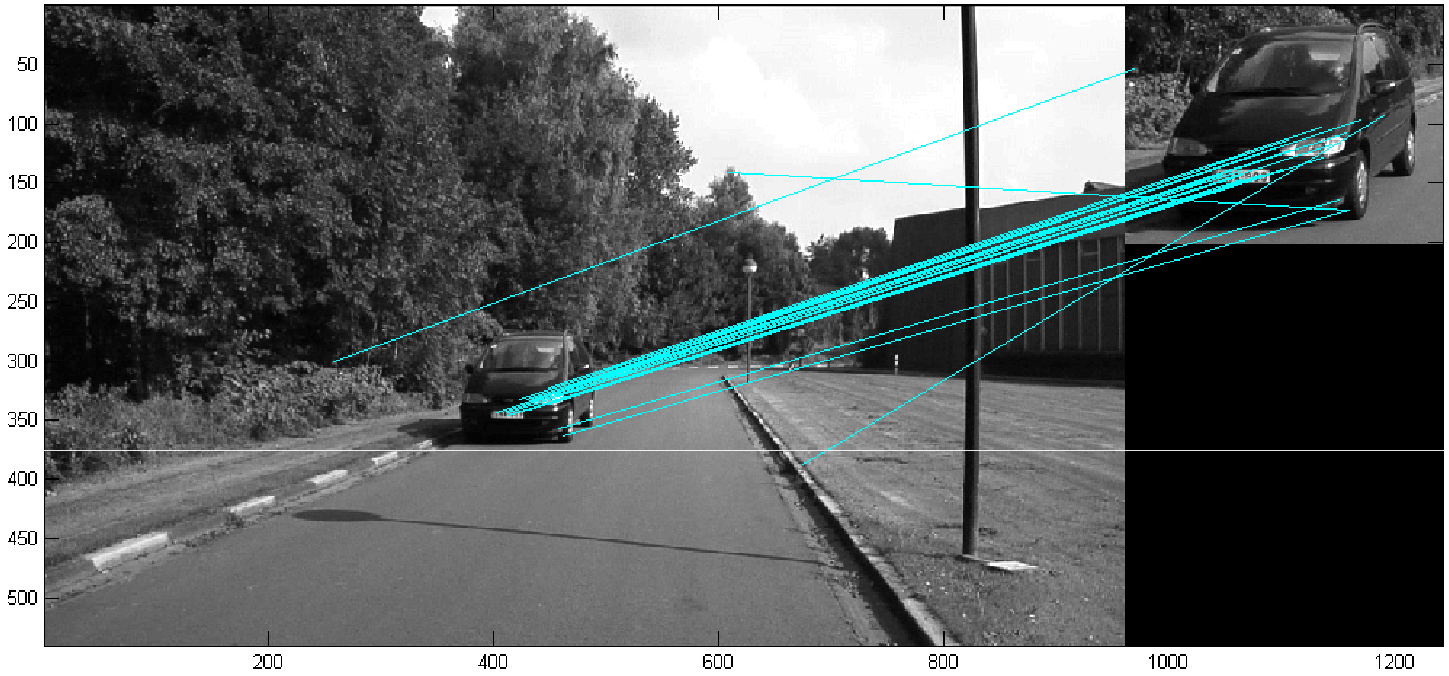
t = 1200 ms



six points from the template are correctly identified in the scene (one point dos not belong to the car however).

Demo – recognition of a car

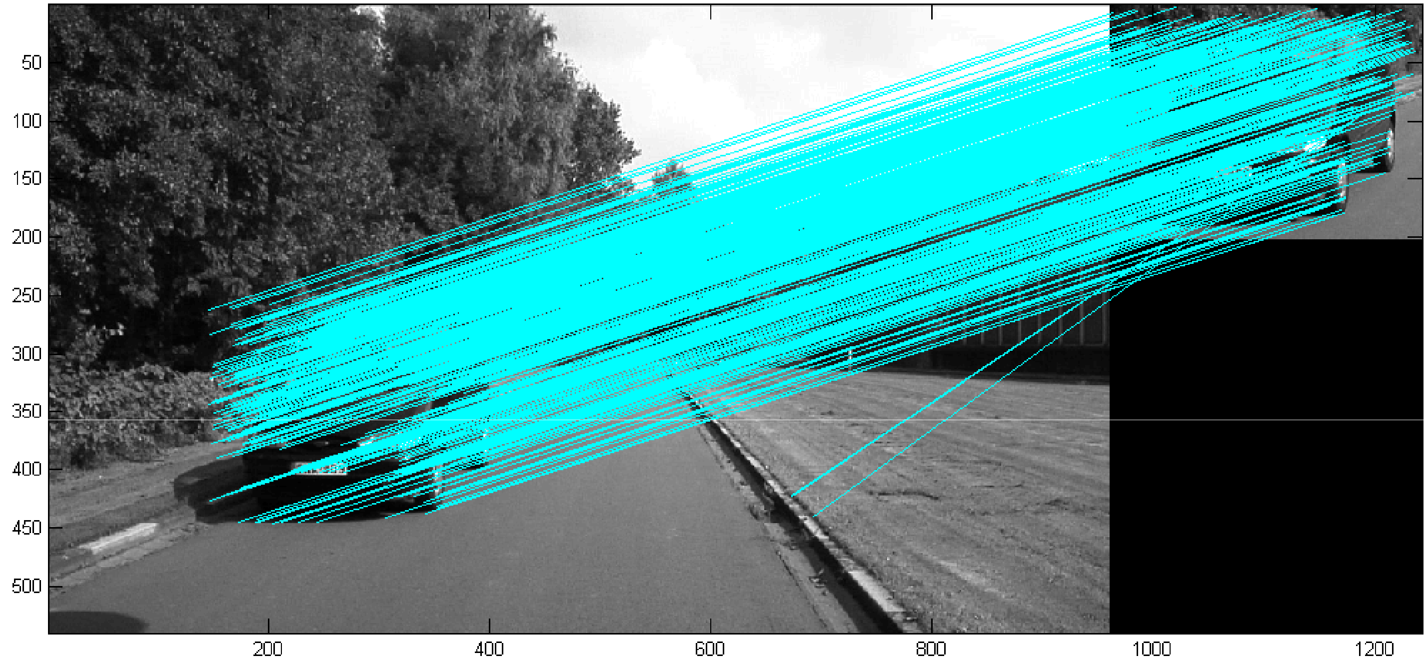
t = 1600 ms



More points are being recognized, two points are wrongly matched

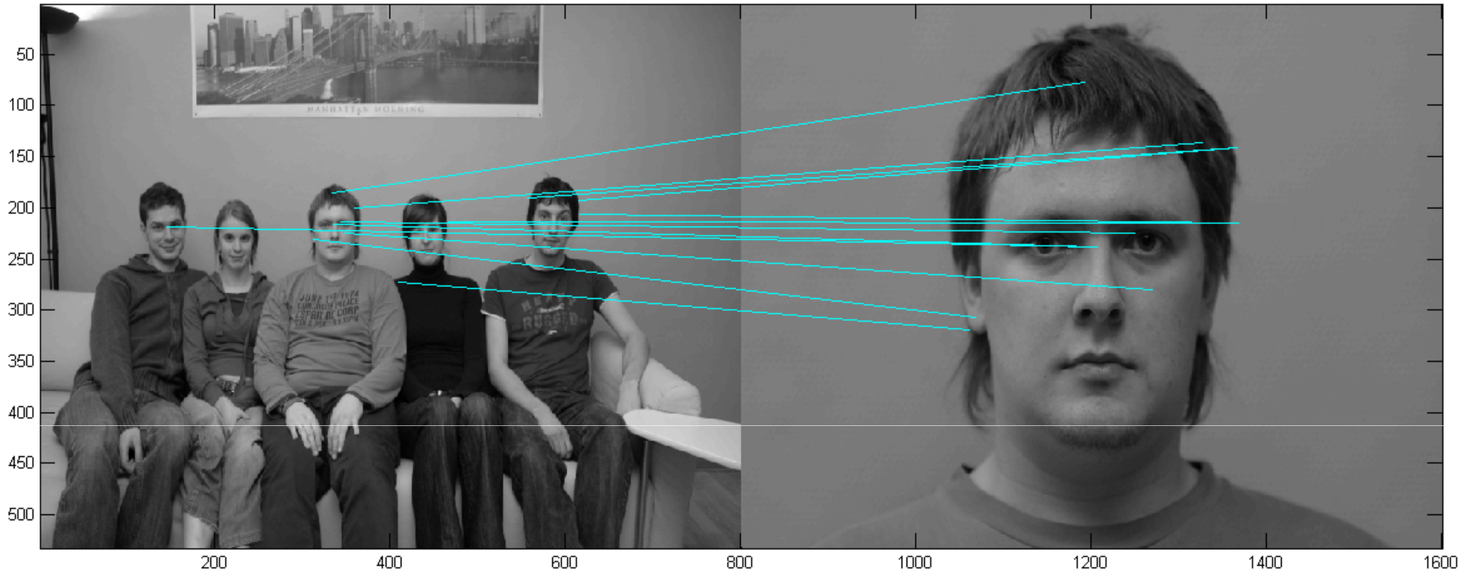
Demo – recognition of a car

t = 2000 ms



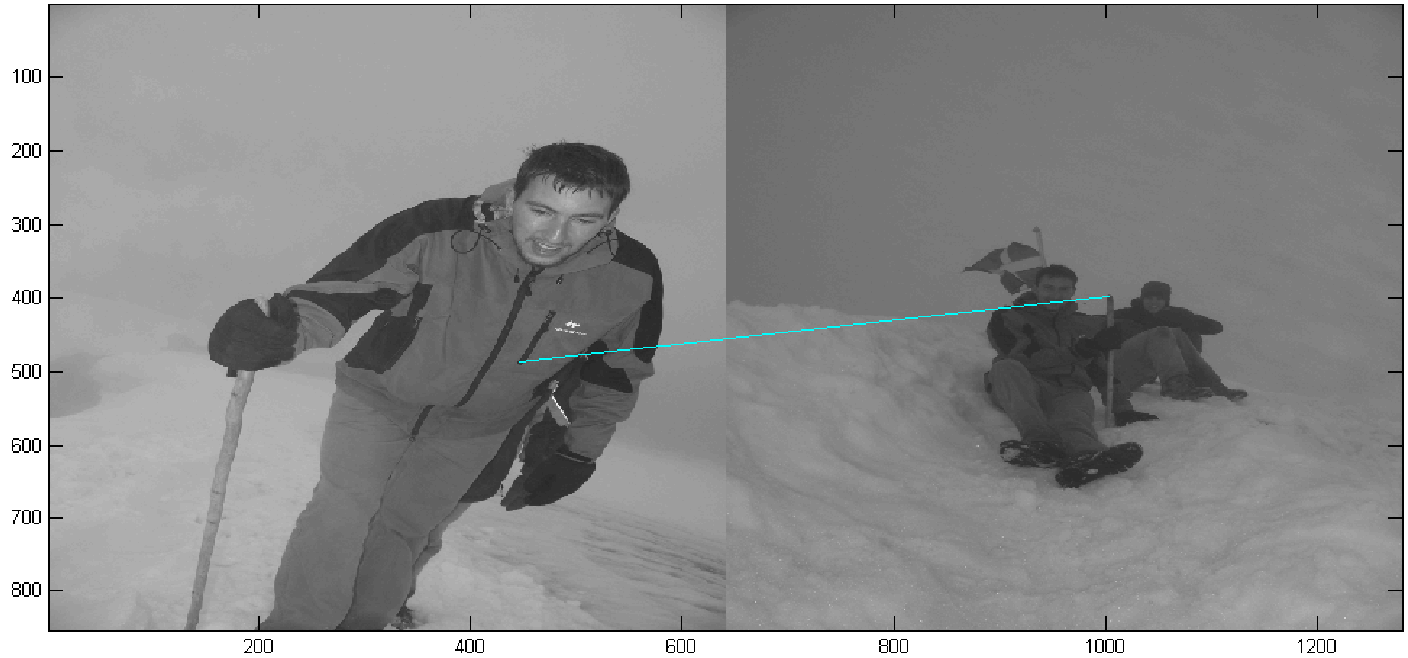
A lot of points are correctly matched (this is to be expected since the template was derived from this image). Two points are incorrectly matched

Demo – recognition of people



Most points are reliably matched, however there are outliers, these could be removed by using a model for consistency in the mapping process

Demo – recognition of people



Clearly the algorithm falls short in matching the person in this scene, taking into account the fact that there is a big difference in viewpoint, illumination and scale. Notice that even for humans the matching process is not straightforward.

Results

To some point, the technique appears to be robust against image rotation, scaling, substantial range of affine distortion, addition of noise, change in illumination

Extracting large numbers of features leads to robustness in extracting small objects among clutter

However in depth rotation of the image of more than 20 percent results in a much lower recognition

Computationally efficient

Applications

View matching for 3D reconstruction
=> Structure from motion

Motion tracking and segmentation

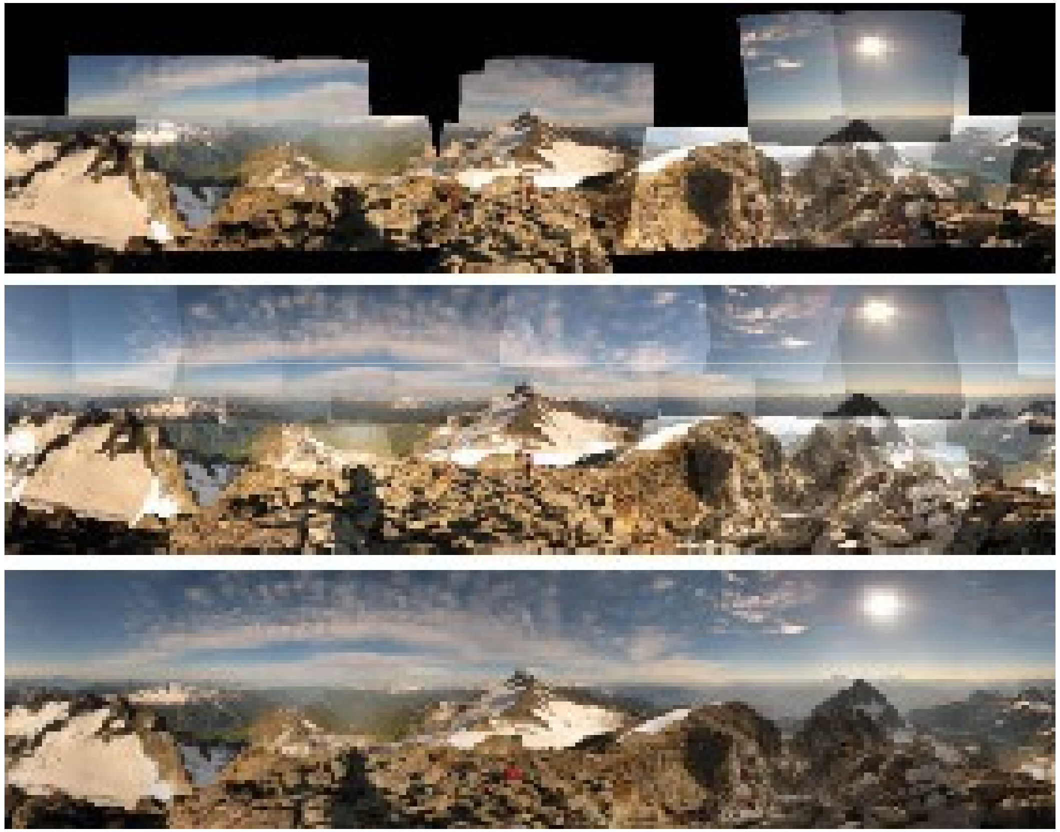
Robot localization

Image panorama assembly

Epipolar calibration

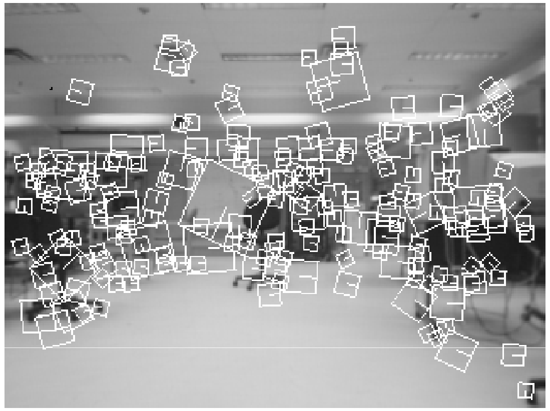
Applications

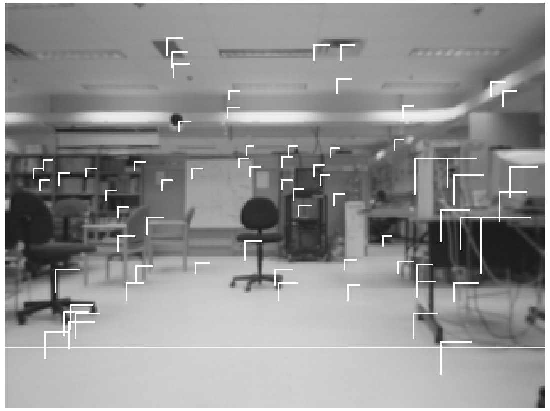
Image panorama assembly



Applications

Robot localization, motion tracking





Applications

Sony Aibo (Evolution Robotics)

SIFT usage:
Recognize charging station
Communicate with visual cards

Future work

Evaluation of the algorithm in matching faces in a cluttered environment

While systematically varying scale, rotation, viewpoint and illumination

Future work

Using the algorithm for long range tracking of objects

⇒Filtering using a priory knowledge
⇒For example in video we have an estimate for the speed vector calculated from previous frames
⇒Integration gives a bounding box where the match is to be found

Integration of new techniques:

SURF: Speeded Up Robust Features

GLOH (Gradient Location and Orientation Histogram)
=> using principal component analysis

Future work

Evaluation other descriptors
⇒e.g. incorporation of illumination invariant color parameters

Incorporation of texture parameters (descriptor build of several scales rather than one current scale)

Dynamic descriptor rather than a static one,
⇒training determines which parameters should be used

⇒Closer study on recent achievements in biological studies of the mammalian vision

⇒It is clear that mammals are still much better at recognition than computer algorithms => promising opportunities