Exercices (5*th* Day)

(SIFT)

In this exercise the so-called Scale Invariant Feature Transform (SIFT) is introduced. This method automatically determines points of interest in the image (keypoints) and afterwards extracts local features for each of the keypoints' sourrounding areas. The method was originally developed to be used in image registration and object detection. The features acquired in the process are to a large extend invariant to scaling and rotations in the image plane (partially also to affine transformations, changes in perspective and lighting).

For the solution of this exercise an implementation of the SIFT keypoint detection and feature extraction is available: To use it make sure the „sift" folder is in your matlab path by typing $path(path,'C:\...\sift')$ The most fundamental function you will need is $[frames, descriptors] = sift(grayScaleImage)$

A short introduction to other functions of this implementation can be found in $...sift\backslash doc\backslash sift.pdf$.

Exercise 1: Load the image $sift1.png$ into the memory and and calculate the keypoints as well as the SIFT descriptors. After that draw the keypoints into the image. Where are the keypoints located and why are they found there?

Now use the $plotsiftframe()$ function to draw some of the so-called frames. What do these obviously display? How is the invariance of the descriptors against changes in scale and rotation achieved?

Exercise 2: Implement a naive matching algorithm which is able to relate similar keypoints to each other. Therefore use the image $sift1.png$ as the training sample. Calculate the SIFT frames and descriptors and store them in a suitable way. Afterwards implement a simple nearest-neighbour-matching which compares a passed list of descriptors with those in your „ training database" and returns the one with minimal euclidian distance as the match.

Test your algorithm by calculating the SIFT descriptors of the image $sift2.png$ and by determining the respective best matches. Visualise the result (hint: draw both images next to each other and connect the related keypoints by lines).

Exercise 3: The quite naive approach from Ex. 2 will normally not yield satisfying results. Test this by applying your solution to the image $sift4small.png$.

The „trick" exploited by the SIFT approach is (instead of matching each single descriptor) to match groups of descriptors which describe the same object pose (scale and rotation). Extend the algorithm from Ex. 2 in such a way that for each match the relative pose (compared to the training sample) is determined:

- for each keypoint compute the point in the image which would correspond to the top left corner of the sample image.
- generate two histograms from your data to display the distribution of the relative object sizes on one hand and the relative rotations on the other.

Test this newly developed procedure and visualise the results for the images $sift2.png$ and $sift3.png$.

Exercise 4: Now deploy the knowledge you gained in Ex. 3 to implement a pose-based matching. To achieve this make use of the generalised hough transform by letting the retrieved descriptors vote for the object pose they relate to. Use this method to localise the training object in the image $sift4.png$. Make sure you have appropriate classification thresholds for the descriptor distance within the nearest-neighbour matching step and for the detection of local maxima in the hough pose-space. Visualise your results.

Further instructions: To visualise results you can use the *line([x1 x2 ...] [y1 y2 ...])* function. Using *hold on* and *hold off* while plotting onto an image within a matlab figure prevents overwriting of the image. As for the more complex samples the nearest-neighbour matching can be quite slow, you can utilise a fast kd-tree implementation which can be found in the *kdtree* folder. Have a look at the *kdtree_demo.m* to find out how this implementation is applied.