Exercise 1: The Scale-Invariant Feature Transform

August 9, 2010

1 Introduction

The Scale-Invariant Feature Transform (SIFT) is a method for the detection and description of interest points developed by David Lowe [Lowe, 2004]. The method has been successfully applied in many computer- and machine-vision applications, such as object recognition, image retrieval, and robotic localization and mapping.

In this exercise, we will use SIFT for the representation of the environment of a robot. We will use the detector to find interest points in the environment, and we will describe those interest points with the SIFT descriptor, so that we will be able to store and recognize the points. The main goal of this exercise is to play around with the different parameters of the detector and descriptor to get a better grasp on the method.

2 Exercises

The detector is based on the filtering of the image with Difference of Gaussians: $D(\sigma) = G(\sigma) - G(k\sigma)$, a so-called Mexican-hat function. Lets visualize the Difference of Gaussian:

2.1 DIfference-of-Gaussian

- Start matlab
- Change directory to `exercise_1`.
- Create a Gaussian filter kernel of size 101×101 and $\sigma_1 = 20.0$ with
  `G1 = fspecial('gaussian', [101 101], 20.0);`
  Mind that the size of the kernel is much bigger than is actually used in the SIFT method, but this is for displaying purposes.

- Create another Gaussian kernel of the same size, but with a $\sigma_2 = k\sigma_1$, where $k = 1.2$.
- `D = G1 - G2;`
- `mesh(D);`

The Difference of Gaussians (DoG) can be compared to the center-surround receptive fields found in the early visual processing in the human brain. You can visualize the receptive field with `imagesc(sign(D));`. As the center-surround processing, the DoG filter kernel will respond to white blobs on a black background (or the reverse if you use `D = G2 - G1;`). Let us have a look at the result of filtering an image with a DoG:

- Open an image: `Irgb = imread('circles_png');`
- Show image: `imshow(Irgb);`
- Convert it to gray scale: `I = rgb2gray(Irgb);`
- Convert to double precision values: `I = im2double(I);`
- Make a DoG kernel with $\sigma_1 = 7.0$.
- Do the filtering: `Idog = imfilter(I,D);`
- Display the result: `imagesc(Idog);`

To detect blobs of different sizes, the image needs to be filtered with DoGs of different sizes.

- Create a for loop with increasing σ_1 and see how the filter response develops.

2.2 Separable filter

The above calculations are very slow due to the large size of the filter kernel. To increase the speed, a number of tricks are applied in the SIFT method:

- Much smaller kernel sizes are used, with $1.6 \leq \sigma \leq 3.2$ and $size = 6 \cdot \sigma$.
- Instead of growing σ to larger and larger values, the image is down-sampled with a factor of two when $sigma > sigma_0$.
- Instead of using a DoG kernel, the image is filtered with two Gaussian kernels with different σ values, and the two Gaussian-filtered images are subtracted. The main advantage is that the 2D Gaussian kernel is separable into two independent 1D Gaussian kernels in the x and y direction. Confirm that this
  ```
  G2D = fspecial('gaussian', [15 15], 1);
  If1 = conv2(I, G2D)
  ```

gives the same result (except for some small differences in rounding) as
`G1D = fspecial('gaussian', [15 1], 1);`
`If2 = conv2( conv2(I, G1D), G1D' );`
and time the calculations by placing `tic;` before and `toc;` behind the code.
NB. `imfilter` already tries to separate 2D kernels into two 1D kernels.

2.3 Finding optima in scale-space

SIFT detects interest points in the image at points that are a local optimum (maximum or minimum) in the DoG-filtered images both in xy-position as well as in scale (neighborhood of $3 \times 3 \times 3$), in order to get points that are unique in position and scale. We will create a matlab script to calculate the local optima on different scales and octaves of the image.

- Create a for loop to run through 3 different octaves. After each loop the image is down-sampled by a factor of two by
  `I = G(1:2:end, 1:2:end, 1);`,
  where `G(:,:,1)` is the Gaussian-filtered image at the first scale in the octave.
- For each octave:
  - Create a 3D matrix `G` with the Gaussian-filtered images at different scales:
    `G(:,:,sc) = imfilter(I, fspecial('gaussian', sz, s));`
    with `s = 1.6` for the first scale and `s = k*s` for subsequent scales where `k = 2^(1/nrScales)`. Set the size of the Gaussian kernel to 6 times s and make sure that the size is always an odd number, so that there is a central cell. Mind that if you want S number of scales, we need $S + 2$ number of Difference of Gaussians (due to finding the optima in a $3 \times 3 \times 3$ neighborhood), and therefore $S + 3$ scales of Gaussian images.
  - Create a 3D matrix `DoG` with the $S + 2$ DoG images by subtracting the Gaussian images:
    `DoG(:,:,sc) = G(:,:,sc) - G(:,:,sc+1);`
  - Find the local optima in the DoG matrix using `Opt = localOptimum(DoG, th);`.
    The function returns a matrix of the same size as DoG with 1 for every local optimum that has an absolute value higher then th. You can use `find()` to get the xy-position of the maxima.
    NB. You might have to recompile: `mex localOptimum.cpp`
  - Plot the Gaussian images and the corresponding local optima for every octave and scale. Mind that the first and last scale of Opt contain zeros only.
- Play around with some different parameters and see how that influences the results. You can also try different images.

2.4 SIFT detection and description

Depending on the image and on the parameters, you might find many interest points on edges in the image. These points, however, are not very unique. The SIFT detector therefore tests all local optima to see if they are truly on a peak or on a ridge in the DoG image. Furthermore, the positioning of the interest points is improved by fitting a quadratic function to the data. This becomes especially important for positioning the points in the image calculated on higher octaves. We will not go into the details of these calculations, but rather proceed with the SIFT package of Andrea Vedaldi[1].

- Include the sift package in Matlab's path: `addpath('../sift');`
- Have a look at the documentation
- In the remaining of the exercise, we will use images from the KTH-IDOL2 database, containing a sequence of images recorded by a robot driving through an office building. Get sequence information
  `sequenceInfo = getIDOLSequenceInfo('min_cloudy1');`
- Load the first image of the sequence
  `I = imread(sequenceInfo(1).pngFileName);`
- Convert to gray-scale and double using `rgb2gray` and `im2double`.
- Calculate the SIFT interest points and descriptors:
  `[frames, descriptors] = sift(I);`
- View them:

```
    imshow(I);
    hold on;
    plotsiftframe(frames);
```

- Try some different settings for a number of the parameters, for instance `FirstOctave`, `NumOctaves`, `NumLevels` and see how that effects the interest points. You can use `size(frames,2)` to get the number of points.

The SIFT descriptor describes the neighborhood of an interest point using Histograms of Oriented Gradients (HOGs). The size of the neighborhood depends on the scale at which the interest point is detected. This makes that the descriptor is invariant to changes in scale. With `plotsiftdescriptor`, the descriptor can be displayed. Standard, the neighborhood is split up into 4×4 squares. For each square, a histogram is made representing the orientations of the image gradients in the square. This histogram contains 8 bins. This results in a feature vector of 128 elements. With the descriptor, interest points can be stored and compared to other points.

[1]http://www.vlfeat.org/~vedaldi/code/sift.html

- Calculate the SIFT frames (interest point location, orientation, and scale) and descriptors:
  `[frames, descriptors] = sift(I);`.
- Plot some of descriptors:

  ```
  imshow(I);
  hold on;
  plotsiftdescriptor(descriptors(:,i), frames(:,i));
  ```

- Get two consecutive images from the KTH-IDOL2 database and calculate the SIFT frames and descriptors.
- Find the matching interest points
  `matches = siftmatch(descriptors1, descriptors2);`
- View the matches: `plotmatches(I1, I2, frames1, frames2, matches);`
- In the matching process, the descriptor distance to the nearest neighbor and to the second-nearest neighbor is found. Only when the distance to the nearest is clearly shorter the match is accepted. This assures unique matched. The threshold of the next-best-to-best ratio can be set as a third argument to `siftmatch`. The default is 1.5. Try different settings and view the results.

2.5 SIFT for scene interpretation

We will now use the SIFT detector and descriptor to find interest points to describe the environment of the robot. The interest points should be repeatable, meaning that they should be redetectable in subsequent images, and they should be unique, meaning that they should not match with interest points in different parts of the environment. We will setup an experiment to test the repeatability and uniqueness of the interest points

- Implement a measure for the repeatability and the uniqueness
- Measure the repeatability and uniqueness at different parts of the environment and calculate the mean and standard error.
- Test the influence of a number of SIFT parameters (for the detector and for the matching). Make plots to view the repeatability and uniqueness as a function of the parameters.

References

[Lowe, 2004] Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110.