Feature detection and matching

October 2015

Abstract

This laboratory is devoted to experiment with an implementation of the SIFT descriptor detection and matching.

Contents

1 Introduction **2**

2 Feature detection **2**

3 Feature matching **4**

4 Practicum submission **5**

1 Introduction

In order to perform this laboratory, the code available at the following link has to be downloaded

`http://www.vlfeat.org/download/vlfeat-0.9.16-bin.tar.gz`

This code is an implementation of the SIFT descriptor detection and matching algorithm described work [1] (it is not recommended to download newer versions for Linux). Although it does not correspond to the original author's implementation, the descriptors extracted by this implementation correspond nearly exactly to the ones extracted by the original SIFT.

The code has an API that can be called from MATLAB and C language. For the former case, the library already includes precompiled binaries that can be used under Windows, Linux or Mac.

In order to install the library, follow the next instructions:

1. Download the file and decompress it to a folder where you may easily change directory in MATLAB. When decompressing the folder `vlfeat-0.9.16` will be created.

2. Download from the campus the utilities files and decompress it inside the folder `vlfeat-0.9.16`.

3. Run MATLAB and change directory to the folder `vlfeat-0.9.16` resides.

4. In order to configure the `vlfeat` library, run the following command. Avoid copying the text from the PDF file since MATLAB will complain about incorrect characters. You may ignore XML warning messages when running the command.

   ```
   run('vlfeat-0.9.16/toolbox/vl_setup');
   ```

5. To check if installation went right, run

   ```
   vl_version
   ```

2 Feature detection

We are now going to perform some experiments with the SIFT descriptor detection algorithm. For that issue run the following commands (remember! avoid copying the commands from the PDF file since MATLAB will complain; you'll need to introduce them by hand)

```
I = vl_impattern('roofs1');
I = single(rgb2gray(I));
imshow(I);
[f,d] = vl_sift(I);
```

The SIFT keypoints are detected and described by means the `vl_sift` MATLAB command. This command requires a single precision gray scale image. The image range may be in [0,255] or [0,1]. In this example the image is within the range [0,1].

The `vl_sift` command returns the following information:

- The matrix `f` has a column for each keypoint. A keypoint is characterized by a center `f(1:2)`, scale `f(3)` and orientation `f(4)`.
- Each column of `d` is the descriptor of the corresponding keypoint in `f`. A descriptor is a 128-dimensional vector of class `uint8`.

We may view the descriptors by using

```
show_keypoints(I,f);
```

Note that a large number of keypoints have been detected in the image. For each keypoint, the application draws a green colored circle indicating its position, orientation and scale at which it was detected. The bigger the scale at which the key point was detected the bigger the circle is painted. From an intuitive point of view, the size of the circle is associated to the size of the (circular) patch that is used to compare the patch with the neighboring ones.

One may view only a small subset of the previous keypoints by randomly selecting some of them

```
show_keypoints(I,random_selection(f,50));
```

***Question**: Take at your own choice several keypoints that have been detected at different scales. Using the theory given in the lectures, comment on the reasons of why do think that a keypoint has been detected at that position and at that particular scale. You may repeat the experiment with another image (such as 'river1') to understand what a significant keypoints is.*

The `vl_sift` command allows to use some parameters to control how keypoints are detected. The SIFT detector of this lab is controlled mainly by two parameters: the peak threshold and the (non) edge threshold.

- The peak threshold filters peaks of the Difference of Gaussian D scale space (see slides) that are too small in absolute value. That is, for a given keypoint at location p one checks if $|D(p)|$ is above a given threshold.
- The edge threshold eliminates peaks of the Difference of Gaussian scale space whose curvature is too small (such peaks yield badly localized keypoints). Let H be the autocorrelation matrix defined as

$$H = \begin{pmatrix} \sum_{\mathbf{p}\in R} \left(\frac{\partial D}{\partial x}\right)^2 & \sum_{\mathbf{p}\in R} \frac{\partial D}{\partial x}\frac{\partial D}{\partial y} \\ \sum_{\mathbf{p}\in R} \frac{\partial D}{\partial x}\frac{\partial D}{\partial y} & \sum_{\mathbf{p}\in R} \left(\frac{\partial D}{\partial y}\right)^2 \end{pmatrix}$$

The principals curvatures are estimated as the eigenvalues of the previous matrix. Let r be the ratio between the largest magnitude eigenvalue and the smaller one, so that $\lambda_{max} = r\,\lambda_{min}$. Then

$$\frac{\mathrm{Tr(H)}^2}{\mathrm{Det(H)}} = \frac{(\lambda_{max} + \lambda_{min})^2}{\lambda_{max}\lambda_{min}} = \frac{(r+1)^2}{r}$$

which depends only on the ratio r rather than the individual eigenvalues. The quotient is minimum when the two eigenvalues are equal and it increases with r. Therefore, to check that the ratio of principal curvatures is below a given threshold, r, we only need to check

$$\frac{\mathrm{Tr(H)}^2}{\mathrm{Det(H)}} < \frac{(r+1)^2}{r}$$

to accept a keypoint. By default a threshold of $r = 10$ is used.

Let us see some examples. First we test the SIFT detector by using different peak threshold

```
[f,d] = vl_sift(I,'PeakThresh', 0.01);
show_keypoints(I,f);
```

Note that the small threshold is due to the fact that the range of gray-values of the input image is within [0,1].

Question: *Try to slowly increase or decrease the threshold. Comment why the number of detected keypoints decreases when the threshold is increased. Is this the expected behavior according to the way the threshold is defined?*

We now try to experiment with the edge threshold. For that you need to fix the peak threshold

```
[f,d] = vl_sift(I,'PeakThresh', 0.04, 'EdgeThresh', 10);
show_keypoints(I,f);
```

Question: *Try to slowly increase or decrease the threshold. Comment why the number of detected keypoints decreases when the threshold is increased. Is this the expected behavior according to the way the threshold is defined?*

3 Feature matching

We will now use the SIFT descriptors to perform matching between two images. For that let us load two images and compute their associated descriptors

```
Ia = vl_impattern('roofs1');
Ib = vl_impattern('roofs2');
Ia = single(rgb2gray(Ia));
Ib = single(rgb2gray(Ib));
[fa, da] = vl_sift(Ia);
[fb, db] = vl_sift(Ib);
```

We compute the keypoint matching with the following command

```
[matches, scores] = vl_ubcmatch(da, db);
```

For each descriptor in `da`, `vl_ubcmatch` finds the closest descriptor in `db` (as measured by the L2 norm of the difference between them). The index of the original match and the closest descriptor is stored in each column of matches and the distance between the pair is stored in scores. The function uses the algorithm suggested by D. Lowe [1] to reject matches that are too ambiguous.

Matches also can be filtered for uniqueness by passing a third parameter to `vl_ubcmatch` which specifies a threshold. Here, the uniqueness of a pair is measured as the ratio of the distance between the best matching keypoint and the distance to the second best one (see `vl_ubcmatch` for further details).

Two view the matchings we use the following command

```
show_matches(Ia,Ib,fa,fb,matches);
```

As before, one may view a reduced set of matchings by executing

```
show_matches(Ia,Ib,fa,fb,random_selection(matches,50));
```

Note that not all the matches are correctly performed. We may improve the matching strategy by using a threshold

```
[matches, scores] = vl_ubcmatch(da, db, 2.0);
show_matches(Ia,Ib,fa,fb,matches);
```

Question: *try to modify the previous threshold. What is the effect of this threshold? Why do results improve as the threshold is increased? Comment your response.*

4 Practicum submission

You are requested to deliver a PDF document including the experimentation performed during this first part of the lab and the second part. The report should included the answers to the questions proposed in this lab and the experimental results for the next lab. The PDF document should include all necessary images to fully understand your discussion. There is no need to deliver a long report. I only expect you to answer the questions in such a way that I'm able to see that you have understood what you have been doing during this lab.

References

[1] Lowe, "Distinctive image features from scale-invariant keypoints", International Journal of Computer Vision, 60, 2 (2004), pp. 91-110.