Scale and Affine Invariant Local Detectors and Descriptors

Users' Guide v1.2

Gyuri Dorkó
INRIA Rhône-Alpes

1 Main Features

- Scale Invariant Detectors
  - Harris-Laplace
  - Hessian-Laplace
  - Different of Gaussian
  - Laplace of Gaussian
- Local Descriptors
  - Scale Invariant Feature Transform
  - Steerable Filters (Local Jet)
  - Intensity-based Spin-images
- Tools to export/import data to
  - MatLab,
  - ASCII text formats
- Visualization

2 Installation Instruction

2.1 Install binaries

They are two sets of binaries. The statically linked version will run on any pc but will only read and write png/pgm files. The dynamically linked version version uses ImageMagick so you have to get the executables compatible with your current installation. These executables have the advantage of reading almost any type of images (anythig that *convert* can read) and one can use *DISPLAY* as a file name instead of output image files to show the results directly in an X window.

3 Components

In general each component/program has a help which is available by the *-help* option. Please use it! The license information is also available with the *-license*

option.

Each program has some arguments (obligatory) and some options. Each option start with a - sign and usually followed by an additional parameter. The options and arguments can be specified in any order, parameters of an option must immediately follow the option. Option names are case insensitive and can be abbreviated as far as they stay unique. If a list file can be specified for a certain argument it must be indicated with an @ sign.

In order to measure the time set the TIMEFLG environment variable. A small table will be sent to the standard output. Note: In *bash* use the *export*, in *tcsh* the *setenv* commands.

In the following section we shortly describe each program in the package.

3.1 DETECT

This program run a scale invariant interest point detector on a given input image or on a series of images. The program has two obligatory arguments:

- Input: an image
- Output: a file name where all the detected regions will be stored

The default detector type is Harris-Laplace, but it can be changed with the *-dtype* option. For each detected point and orientation can be computed (which will be stored and used by ComputeDescriptor) with *-angle* option. Optional affine region estimation can be invoked with *-affine*. To process a set of images one can specify a list file instead of a single image. The list file must be an ASCII text file where each line contains a file name with full or relative path to an image. I suggest to avoid of special non-ASCII characters as well as spaces in file names. The output of such detection is still one file, however a file identifier is saved together with each detection. In case of a list file if you also plan to use *-angle* or *-affine* do not specify it here, but later in COMPUTEDESCRIPTOR instead. To indicate that a list file had been specified instead of a single image start the first argument with a @ sign (no spaces between @ and the list file name).

We have made some changes the implementation of Harris-Laplace and Hessian-Laplace. Our experience that the results are improved in average with the new implementation of Harris-Laplace, however, the old implementation is still available with the *-old* option.

Additional parameters. The scale selection for Harris-Laplace, Hessian-Laplace and LoG can be turned of with *-noscale*. this switches the detectors from scale invariant (or precisely covariant) mode to multi-scale. The detector will still be run on the whole scale pyramid, but no scale selection will be applied, therefore one can expect much more detection. The same (x, y) location can be selected many times with different scale values. The minimum and maximum scale can also be set. (The new implementation does not suppors minscale=maxscale, yet). E.g. to get the standard harris detector on the original image run:

```
Detect -old -dt har -noscale -minsc 1 -maxsc 1 a.pgm a.har
```

The scale-step, i.e. the multiplier of the pyramids can also be set with the *-scalestep*, and it is highly suggested to keep this settings within a reasonable bounds.

The detector type *dense* helps to simulate a **dense representation** by creating a set of descriptors along a grid on the image. The grid is defined on each scale

between *minscale* and *maxscale* (quantized with *scalestep* multiplier. The cell width and height computed by $currentscale * gridstep$, where *gridstep* is a parameter. The minimum scale can also be set by *-msbysize*.

For more options and precise syntax see DETECT *-help*. For example see Section 6.

3.2 COMPUTEDESCRIPTOR

This program computes a descriptor (n dimensional vector) for each detected interest region. The program has three obligatory arguments:

- Input: an image
- Input: detection results (output of DETECT)
- Output: a filename where all the descriptors will be stored. the result file will contain all the information from the second input argument together with a descriptor for each detection.

The default descriptor is SIFT, it can be changed with *-dtype* option. Optional angle (rotation invariance) and affine computation (viewpoint invariance) can be specified to precede the descriptor computation. (If you have done that in DETECT it is unnecessary to do it again!) If the detection was done on a list-file you must specify here the same file with similar syntax.

SIFT descriptor. The implementation is inspired by David Lowe's implementation[1] of SIFT descriptors. Present version of COMPUTEDESCRIPTOR contains a reimplemented algorithm, however, the our old version which was based on David Lowe's code is still available with the *-old* option. SIFT, as default, is computed on a 4x4 grid, and for each cell the dimension of the orientation histogram is 8. These parameters now can be changed with *-siftis* and *-siftos*. The default settings leads us to a 128 dimensional descriptors. The *-siftws* parameter is the window size which the detected patch will be downscaled before sift computation. This is necessary for performance reasons. To further lower this setting can speed up computation, especially for detections with large scales, but can also significantly decrease the quality of the descriptors. *-siftscone* specifies the patch size (neighborhood) which the sift will be computed if the scale of the detection is 1 (only modify this if you know what you are doing!). The option *-unnormalized* switched off the normalization of the computation it is only useful if you would like to implement you own normalization. Otherwise the descriiptors are normalized to unit length.

Local Jets. the computation is based on earlier code of Krystian Mikolajczyk's implementation[2]. Steerable Filters computed until the 4th order, therefore the length of such descriptor is 15.

1:	the pixel value,
2,3:	First derivatives (Dx, Dy)
4,5,6:	Second derivatives (Dxx, Dxy, Dyy)
7,8,9,10:	Third derivatives (Dxxx, Dxxy, Dxyy, Dyyy)
11,12,13,14,15:	Forth order derivatives (Dxxxx, Dxxxy, Dxxyy, Dxyyy, Dyyyy)

All values are computed by convolution of the appropriate Gaussian kernel. The descriptors are normalized by the appropriate (co)variance matrix.

Spin Images. the implementation is inspired by Svetlana Lazebnik's code[3]. Our reimplementation provides the following parameters: *spinis* and *spinds* determines

[1] http://www.cs.ubc.ca/~lowe/keypoints

[2] http://lear.inrialpes.fr/people/Mikolajczyk/Database/index.html

[3] http://www-cvr.ai.uiuc.edu/~slazebni/

the number of bins in intensity and dinstance dimensions. The *spinws*, similarly to *siftws* is the window size which the detected patch is to be downscaled before descriptor computation, to achieve better performance. To further lower this setting can speed up computation, especially for detections with large scales, but can also significantly decrease the quality of the descriptors.

3.3 DrawCorners

This tool can be used to visualize the detection result on an image. It has three obligatory arguments.

- Input: the image
- Input: the detection result (the output of Detect or ComputeDescriptor)
- Output: an image where the results should be draw.

The program has options for different colors and pen widths. If the detection has been done on a list of images, the list file can be specified here also (in the same way) but the detection result has to be preselected for only one image (see. SelectCorners.) The second argument can be either the output of Detect, ComputeDescriptors or SelectCorners.

Create thumbnails. DrawCorners can be used to create an image with small image patches extracted from the detected location. These feature are available via *-th_...* parameters. E.g.:

```
Detect @a.lst all.har
DrawCorners -th_on @a.lst all.har a.gif
```

will run Harris-Laplace on multiple images followed by the creation of one huge image with all the detected patches. The number of patches per line the distance between patches the maximum number of patches can all be controlled by different options.

3.4 SelectCorners

This program chooses some interest points from a file and save it to another. It has two obligatory arguments:

- Input: interest points (output of Detect or ComputeDescriptor)
- Output: the selected interest points

Possible selection criteria:

- All descriptors that include a given pixel (see options -x and -y)
- A specified corner (see option -number)
- Random selection of n points (see -random)
- The first n points (see -firstn)
- Points larger than a given scale (see -scmin)
- Points smaller than a given scale (see -scmax)
- Interest points of a given file, in case of the detection were done on a list of files (see -fid)

SelectCorners can output to the same file as its input. It will overwrite the file the previous content will be lost (so make a backup if you are not sure). For precise options see SelectCorners -help and also see the examples in Section 6.

3.5 Export to different formats

3.5.1 dumpcontents

Dumps the content of a file (output of Detect or ComputeDescriptor) to the standard output in ASCII format. This is basically one way to convert the data to text format. with the *-size* option the program only gives the number of interest points.

3.5.2 corners2text

Creates an ascii text file where each line corresponds to a detection. the structure of the line is the following: **x y scale fileid** $m_{1,1}$ $m_{1,2}$ $m_{2,2}$ d_0 d_1 d_2 $\cdots$ d_n

where the first 4 fields (location, scale, file identifier) always appear and the others can be requested by the options: *-add affine* for the affine second moment matrix ($m_{1,1}$ $m_{1,2}$ $m_{2,2}$) and *-adddesc* for the descriptor values (d_x).

3.5.3 Conversion to matlab: corners2matlab

This program needs two arguments:

- Input: interest points (output of Detect or ComputeDescriptor)
- Output: a .mat file.

The converter creates a *struct array with fields*:

x:	the horizontal coordinate of the center of the interest point
y:	the vertical coordinate of the center of the interest point
scale:	the detected scale level
angle:	the computed orientation (dominant gradient)
fileid:	the file id (numbered from 0, e.g. 2 means the 3$^{\text{rd}}$ image in the listfile used for Detect.
affine:	the normalized 2$^{\text{nd}}$ moment matrix of the affine estimation
descriptor:	vector with the dimension of 0, 15 or 128 in case of no, localjet or sift computation.
note:	string, the image name, in case of list file only the variable part of the names

4 Import from different formats

4.1 Import from text

Programs: text2corners
This is similar to corners2text but works in the other way. It requires the same file format as the output of corners2text. It additionally supports the affine second moment renormalization of extract the scale (if it is encoded inside affine) and a different way of ellipse parameterization (see *-help* for details).

5 Known Bugs

- Sometimes some points are detected twice.

- If descriptors cannot be computed (e.g. point is too close to the border) they are not removed, their descriptor length is 0. If you'd like them removed, specify *-clean*
- DrawCorners create color images as default, even if the image extension is pgm. Solution: Use *-nocolor* in these cases. Do not use *-nocolor* with image list files.

6 Examples

6.1 First steps - Tutorial

1. Take a a gray level image of yours (e.g. a.pgm)
2. Interest Point Detection (with Harris-Laplace):

   ```
   Detect -dt har a.pgm a.har
   ```

   Note: if you use *-affine* detection also specify *-angles*
3. Compute local descriptors for each interest point (based on sift)

   ```
   ComputeDescriptor a.pgm a.har a.har.sift
   ```
4. Show result of (2) on the image:

   ```
   DrawCorners -col y -wi 2 a.pgm a.har a.har.pnm
   ```
5. Dump the result of (3) to a text

   ```
   dumpcontents a.har.sift > a.txt
   ```
6. Dump only the computed descriptors (of step (3)) to a text

   ```
   corners2text a.har.sift -adddesc a.har.desc.txt
   ```
7. Convert the results to MatLab format:

   ```
   corners2matlab a.har.sift a.mat
   ```

6.2 Detection with a list of images

1. Create a file called *images.lst*. Each line should contain a name of an image.
2. Detect interest points based on Laplace-of-Gaussian.

   ```
   Detect -dtype log @images.lst result.log
   ```
3. Apply affine and rotation invariance for each point and compute SIFT descriptors. (the next two lines should be typed in one!)

   ```
   ComputeDescriptor -dt sift -aff -an @images.lst result.log
   result.log.sift
   ```
4. Draw the detection results (with affine estimation) on the 3rd image in the list.

   ```
   SelectCorners -fid 2 result.log.sift third.sift
   DrawCorners -color yellow @images.lst third.sift third.pnm
   ```

   Note that first we have to select the points of the third image to a different file (fid==2, because the files are numbered from 0), then we can use DrawCorners with the list file, it will load up the correct image.
5. Show the same patches but now extracted in a thumbnail format.

   ```
   DrawCorners -th_on -co ye @images.lst third.sift third_th.pnm
   ```

6.3 Examples for SelectCorners

Select interest points that have the pixel $(20, 10)$ included in their interest region:

```
SelectCorners -x 20 -y 10 source.har dest.har
```

Select 10 random interest point:

```
SelectCorners -random 10 source.har dest.har
```

Select interest points having scale between 2 and 10:

```
SelectCorners -scmin 2 source.har tmp.har
SelectCorners -scmax 10 tmp.har dest.har
```

Select the 10^{th} interest point:

```
SelectCorners -numb 9 source.har dest.har
```

Select the 10^{th} and 20^{th} interest points:

```
SelectCorners -numb '9 19' source.har dest.har
```

7 Related Scientific Work

In this section we describe where can you find some description of the different features (detectors/descriptors) provided by this package.

7.1 Articles Related to This Package

This section describes where can you find some description of the different features (detectors/descriptors) provided by this package.

If you find this package useful in your scientific work, you can refer to the following paper:

> Gy. Dorko and C. Schmid. "Object class recognition using discriminative local features." IEEE Transactions on PAMI, 2004. submitted.

DoG, SIFT:

> D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004, accepted for publication.

Harris-Laplace, Hessian-Laplace:

> K. Mikolajczyk and C. Schmid, "Indexing based on scale invariant interest points," in *Proceedings of the 8th International Conference on Computer Vision, Vancouver, Canada*, 2001, pp. 525–531.
>
> C. Harris and M. Stephens. "A combined corner and edge detector." In *M. M. Matthews, editor, Proceedings of the 4th Alvey Vision Conference*, pages 147151, 1988.

Spin Images:

A. Johnson and M. Hebert. "Using spin images for efficient object recognition in cluttered 3d scenes." IEEE *Transactions on PAMI*, 21(5):433–449, 1999.

S. Lazebnik, C. Schmid, and J. Ponce. "A Sparse Texture Representation Using Local Affine Regions." in IEEE *Transactions on PAMI*, accepted, 2005.

7.2 Other Scale Invariant Local Detectors / Descriptors

A comparative study of affine interest point detectors can be found in: [4]

K. Mikolajczyk and C. Schmid, "Scale & affine invariant interest point detectors", in *International Journal of Computer Vision*, to appear.

Performance evaluation of local descriptors:

K. Mikolajczyk and C. Schmid, "A performance evaluation of local descriptors." in *IEEE Conference on Computer Vision and Pattern Recognition*, June 2003.

more to come

8 Authors, Contributors, Acknowledgements

The software package is developed by *Gyuri Dorkó* with contributors *Michael Sdika* and *Matthijs Douze*. A large amount of our code is inspired by others' implementation, therefore we thankful for

- *Krystian Mikolajczyk* for old versions of detectors of Harris-Laplace (affine), Hessian-Laplace and Local Jet descriptors.
- *David Lowe* for detectors of Different-of-Gaussian and SIFT descriptor.
- *Svetlana Lazebnik* for the help of spin-images .

[4] http://lear.inrialpes.fr/people/Mikolajczyk/mikolajc_ijcv2004.pdf