376.054 Machine Vision and Cognitive Robotics (VU 4,0) 2018W

Link to TISS Lecture

Exercise 2: Interest Points and Descriptors

In this exercise you will implement an interest point detector, the Harris corner detector. You will then compute descriptors at each of the interest points. These descriptors will then be used to match similar interest points from consecutive image frames with each other.

Download files for exercise 2

Part 1: Implementation (10 points)

You should write code so that it can be understood by someone else (the tutors, for example). Comment sections and lines which might not be easy to understand. If you've thought about a line or section for more than a couple of minutes, it probably warrants a comment. Use descriptive variable names.

Your code does not have to run extremely fast, but it should not take too long to run. Points may be deducted for slow code, even if your solution is correct.

1. Harris corner detector (4 points)

In the file **harris_corner.m** you should write code for the Harris corner detector presented in the lecture. The input parameters are a grayscale image with values ranging from 0-255 and a `parameter` struct containing the following 4 settings:

- `parameters.k`: Coefficient of the Harris formula
- `parameters.threshold`: Threshold for corners
- `parameters.sigma1`: Sigma of Gaussian filter used in the first step of the algorithm
- `parameters.sigma2`: Sigma of Gaussian filter used after the calculation of the derivatives in x and y direction (should be larger than `sigma1`)

For debugging purposes, the result of every intermediate step should be returned as a parameter. For a detailed description of the output parameters refer to the comments in the file.

The basic steps you will have to implement are:

1. Normalize image to double values in the range of 0-1.
2. Compute the horizontal and vertical derivatives of the image (I_x and I_y) by convolving the image I with derivatives of Gaussians in x- and y-direction. You can create the Gaussians with the `fspecial` command as in exercise 1, using `sigma1`. The width of the filter can be calculated like you did in exercise 1. To get a Gaussian derivative you can filter a Gaussian kernel with the small filter [-1 0 1], or use the `gradient` command.
3. Compute the Harris feature value for each pixel and normalize the values to range [0, 1]. Use another Gaussian (with `sigma2`) as the weighting kernel.
4. Threshold the feature values and apply non-maximum suppression.

To test your implementation you can call the script main_harris_image.m. You can also try and use live images from a connected webcam with main_harris_video.m. You may need to install some additional matlab packages and play around with video codecs for this to work. You are not required to do this.

2. Patch descriptor (1 points)

Once the corner detector is complete, we want to match corners in similar images to find correspondences. To be able to do that, you first have to implement a feature descriptor characterising each detected corner. For the basic descriptor, take the image intensity values around a corner and concatenate them to get a large feature vector (this is a one-line operation).

The compute_descriptors function is a helper function which passes a valid image patch to a descriptor function, ensuring that descriptors are not computed on patches which would go outside the bounds of the image.

Complete the code in **patch_basic.m** and **compute_descriptors.m**.

3. Matching interest points (1 points)

Now that we have a feature vector describing each interest point, we can try to match similar interest points across different images. The matching function has to be implemented in the file **match_descriptors.m**, which takes the feature descriptors of two images as input parameters. For each descriptor x_A of image A, you have to calculate the distance to each descriptor x_B of image B. As a distance measure, you can simply use the euclidean distance:

$$d = |\mathbf{x}_A - \mathbf{x}_B|$$

The x_B with the smallest distance value corresponds to the best match for x_A. Some descriptors can be very ambiguous and cause false matches. To reduce the number of false matches, you should add the following heuristic: Consider the two "closest" matches x_{B1} and x_{B2} to x_A with distances d_1 and d_2. x_{B1} should only be considered as the unique best match for x_A if $d_1/d_2 < 0.8$. That way, many ambiguous and probably wrong matches will be discarded.

An example of a good matching result can be seen in Figure 1 below.

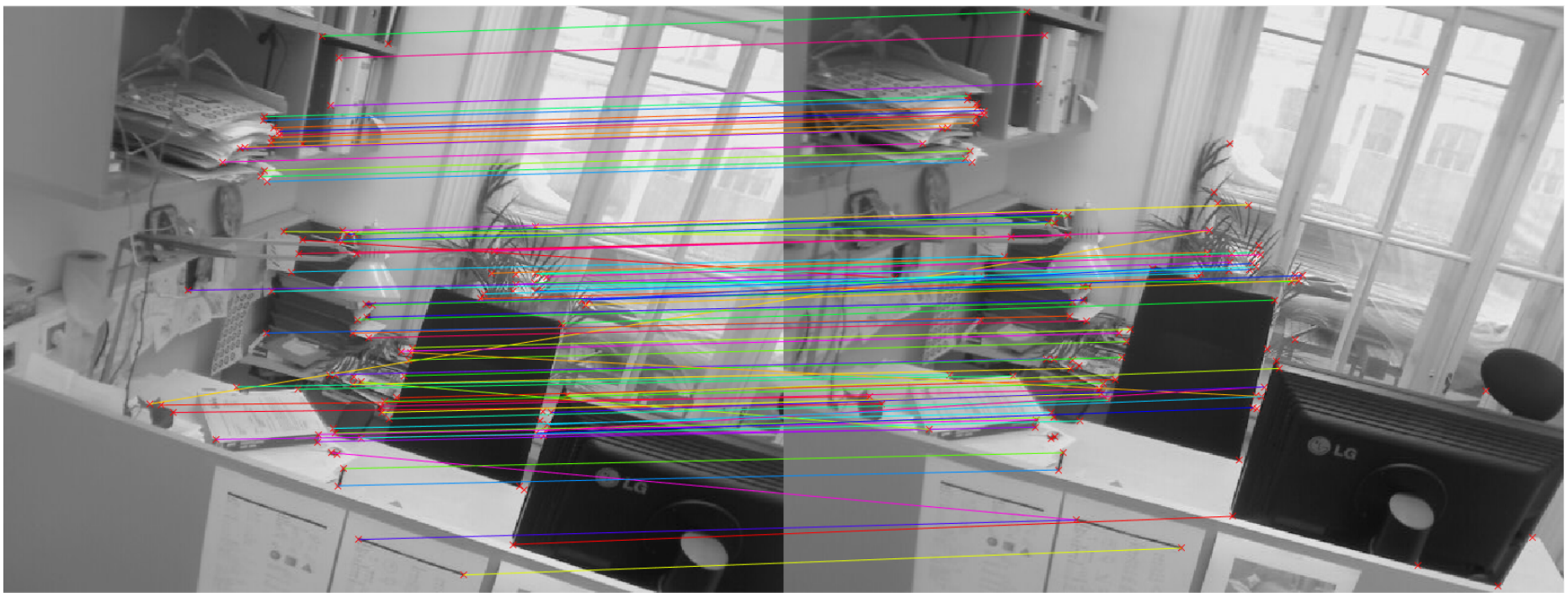

Figure 1: Good matching result (only 5 wrong matches).

4. Better descriptors (4 points)

Extend the basic patch descriptor. These functions should not require more than 3 lines of code.

1. Normalise the descriptor (**patch_norm.m**)
2. Sort the normalised descriptor (**patch_sort.m**)
3. Extract pixels in a circular region of the patch, and sort them after normalising. You can use the `circle_mask` function to help (**patch_sort_circle.m**)

Implement an orientation-based descriptor (**block_orientations.m**). This descriptor will operate on blocks of 4x4 pixels on a 16x16 image patch. Compute the gradient magnitude and orientation of each pixel as you did in the previous exercise, using the simple filter [1 0 -1]. You do not need to apply additional blur to the patch. The final descriptor will be a concatenation of orientation histograms computed on the 16 4x4 blocks. Each block should produce a histogram with 8 bins in the range [-pi, pi]. Each bin should be weighted by the sum of the gradient magnitudes of the pixels in that bin. Normalise each block's histogram individually before concatenating it. The `discretize` function and/or functions for histograms should be helpful.

You can check if your block extraction is correct by operating on the following matrix:

```
kron(reshape(1:16, 4, 4)', ones(4))
```

Your blocks should come in row major order, i.e. be blocks of all 1s, 2s, 3s and so on (as opposed to 1,5,9... for column major order).

Useful commands

`imfilter` - Filtering with linear filter kernel
`fspecial` Generation of filter kernels
`imread` - Load an image
`imshow` - Display an image (for RGB image, value format is uint8 with range [0, 255], for grayscale, value format is double with range [0, 1])
`rgb2gray` - Image conversion to grayscale
`double` - Converts to double-precision floating point format
`uint8` - Converts to 8-bit unsigned integer format
`sort` - Sort a vector or matrix

Part 2: Documentation (6 points)

Perform the following experiments with the given images and at least one picture set you took yourself, and answer the questions in a few sentences. Document the results using representative example images. Only change the parameters in the main_... files at the marked places.

- Answer each numbered question separately, with the answer labelled according to the experiment and question number. For example, experiment 1 question 2 should be numbered 1.2. You do not need to include the question text.
- Include images to support your answers.

Maximum 1500 words, arbitrary number of pages for result images (referenced in the text). Violating the text limit will lead to point deductions!

Experiment 1 (2.5 points)

Calculate the Harris Corners with different sigma values for the Gaussian filtering and test their influence on the results. Call the script main_harris_image.m and/or main_harris_video.m.

1. How do different sigma values for the initial Gaussian filtering influence the output of the detector?
2. How does the second Gaussian filtering improve the output of the detector?
3. The Harris detector is not scale invariant. Give an example of an image patch which would fire the detector at one scale, but not at another.
4. How does not being scale invariant limit the effectiveness of a detector? Use a simple example to explain.
5. What is the effect of varying the k parameter?

Experiment 2 (3.5 points)

Match the patch descriptors of several similar images using different parameter settings. You should look at the effect of rotated images, and images with different illumination (you can use lightened images with suffix `cl` in the `desk` directory), or modify your own images using the curves or levels tools in photoshop or GIMP. Also try with non-sequential images, e.g. use desk 00 and desk 03.

1. How does changing the patch size of the affect the accuracy of the matching with the patch descriptors?
2. How are the descriptors affected by rotation of the image? Approximately at which point do the patch descriptors no longer produce good results? Use `rotfill` to make sure the detector fires in the same locations.
3. What problem does normalising the patch descriptor mitigate? Is it still as effective as the basic descriptor?
4. What problem does sorting the patch mitigate? Are the results better with the circular version? Why is this?
5. How does the matching performance of the block descriptor compare to the others?
6. What are some benefits of using a histogram to describe parts of the patch?
7. What are some benefits of using blocks within the patch? Are there any disadvantages?

Assistance

Please keep to the following hierarchy whenever there are questions or problems:

1. Forum: Use the TUWEL forum to discuss your problems.
2. Email for questions or in-person help requests: machinevision@acin.tuwien.ac.at

Upload

Please upload the Matlab files **harris_corner.m**, **patch_basic.m**, **patch_norm.m**, **patch_sort.m**, **patch_sort_circle.m**, **block_orientations.m**, **compute_descriptors.m**, **match_descriptors.m**, and your documentation (pdf-format) **as one ZIP-file** via TUWEL.

Submission status

Submission status	No attempt
Grading status	Not marked
Due date	Monday, 5 November 2018, 8:00 AM
Time remaining	6 days 21 hours
Last modified	-
Submission comments	Comments (0)

Add submission

You have not made a submission yet

◄ A Combined Corner and Edge Detector

Jump to...

Files for Exercise 2 ►

My courses
183.585 Computer Vision (VU 3,0) 2018W
186.112 Heuristic Optimization Techniques (VU 2,0) 2018W
376.054 Machine Vision and Cognitive Robotics (VU 4,0) 2018W
All my courses
Course overview
Search courses
New course
Help
TUWEL Tutorials (Students)
TUWEL Tutorials (Teachers)
TUWEL scenarios
E-Mail Ticket Support
General Information
Terms of Use
Teaching Support Center