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

Link to TISS Lecture

Exercise 3: Object Detection with SIFT and GHT

In the last exercise you implemented feature descriptors to calculate correspondences between two different images. Based on this idea, in this exercise you will go one step further and implement an object recognition algorithm for multiple instances using the Generalized Hough Transform (GHT).

Download files for exercise 3

Figure 1: Goal of the exercise: Detect all the instances of the object on the left in the image on the right.

Part 1: Implementation (10 points)

This time, you will not need to program your own descriptor and matching code. Instead, we will use the publicly available vlfeat library for MATLAB to calculate more robust SIFT descriptors and get correspondences between images. For object recognition, you will have to implement the Generalized Hough Transform.

1. Download and compile the SIFT library

The vlfeat SIFT implementation for MATLAB by Andrea Vedaldi can be found here. After downloading and unpacking the code, you have to run a setup command for the functions to become available. Follow the instructions in the README file or on the vlfeat website.You should change the `run` command of `main_sift.m` according to the location where you unpacked the vlfeat library. In this file, you can also set the index of the test image to be processed. **The remainder of the main file must not be changed!**

If you followed all the steps correctly, you should be able to run `main_sift.m` without errors (but of course the result will not make sense yet).

2. Generalized Hough Transform (GHT) for Object Recognition (9 points)

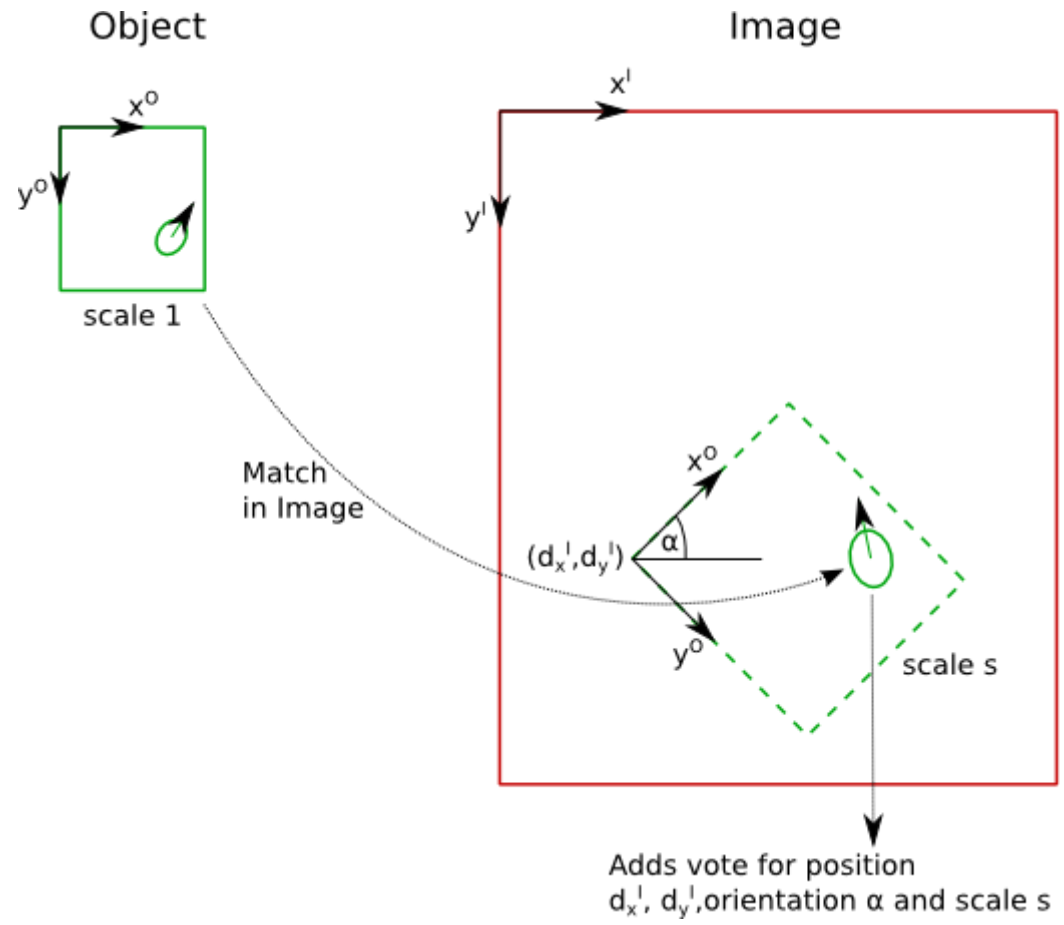
The general idea behind the Hough Transform is that objects are found through a voting procedure in a configuration or Hough space. If evidence for a particular object configuration is found in the image, a vote is added for this configuration. In our case such evidence is a matched feature vector from the training image to the test image. The **location, orientation and scale** of the feature with respect to the object's coordinate system is known from the training image, so if the feature is also found somewhere in the image, it is possible to calculate the likely location, orientation and scale of the whole object and add a "vote" for that configuration (Fig. 2a). After all matched feature vectors have voted, **maxima in the configuration space correspond to possible object instances** (Fig 2b).

In our case, the configuration (or voting) space for the object is 4-dimensional, the dimensions being **x**, **y**, the **scale** and the **orientation**.

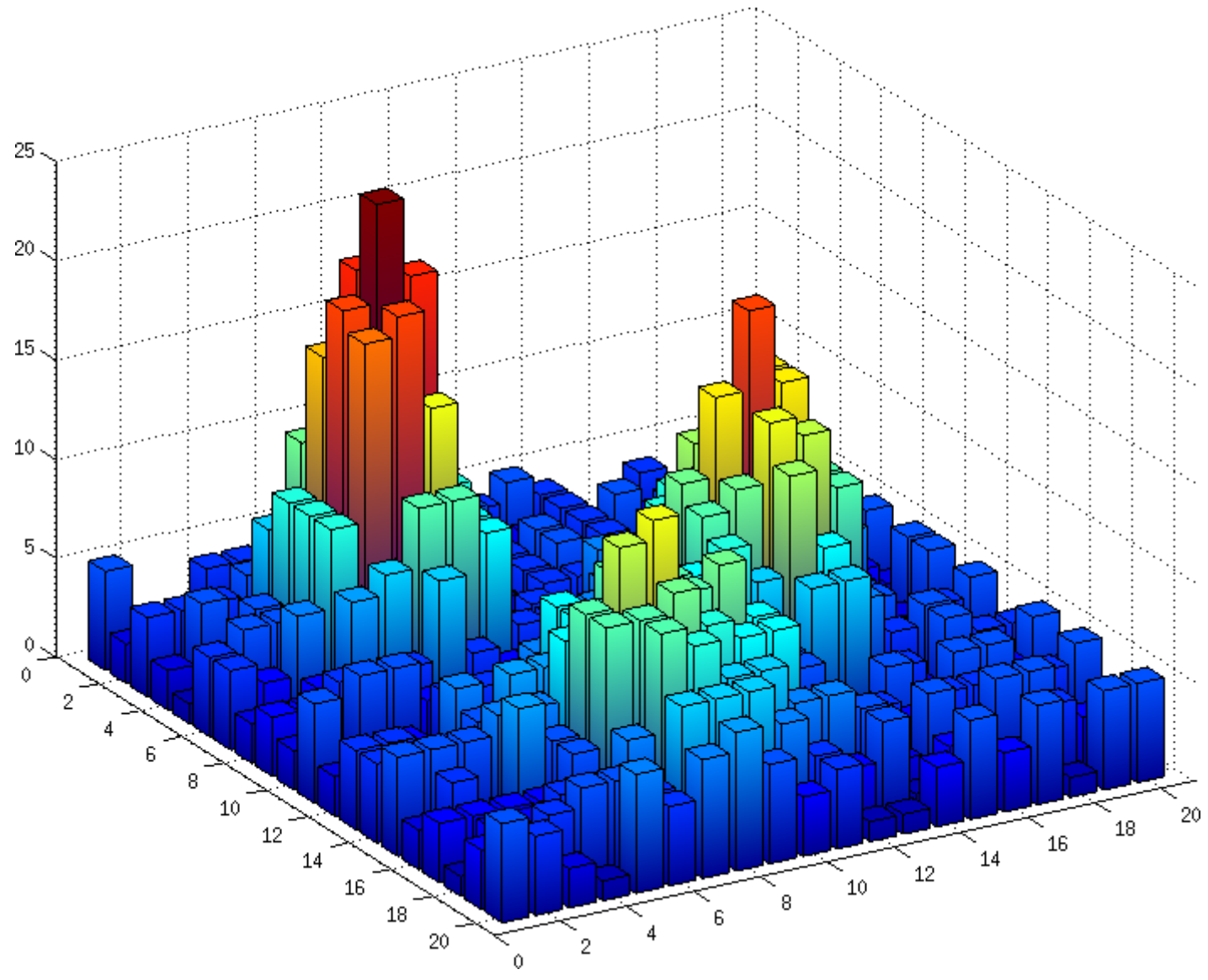
To be able to accumulate votes, the configuration space has to be discretised into bins. If a configuration should get a vote, the vote is added to the single corresponding bin. **The width of the bins (or the "grid resolution") can be chosen independently for each dimension and is crucial for the quality of the final result.** If the grid is too fine, votes might spread across too many neighboring bins without a clear maximum. If on the other hand the grid is too coarse, the accuracy of the detection will be bad. For the spatial dimensions, it makes sense to set the bin size with respect to the object's dimensions, for the orientation, a coarse binning of 15-20° is a good starting point. To find a useful grid size for the scale dimension, you can check the object's dimensions in the object and the test images.

Since we want to detect multiple instances of the same object, we cannot simply consider only the maximum value in the voting space! Instead, the `detectObject` function needs to be able to detect multiple peaks and return the corresponding object configurations (x, y, orientation and scale) for each peak.

In Figure 2 you see a visual explanation of the general idea of the Hough Transform you should implement.



explanation of hough transform



hough voting space

(a) (b) Figure 2: (a) Principal idea of the Generalized Hough Transform for object recognition. (b) Simple 2-dimensional voting space (e.g. only x and y coordinates). Clearly, 3 object instances have been detected in different configurations.

Summary of necessary implementation of the GHT:

- Set up voting space, discretization with reasonable grid sizes for x, y, scale and orientation. (1 point)
- Compute the (x,y,s,o) configuration for each match pair, using your implementation of `match_to_params.m` and add it to the voting space. (2.5 points)
- Extract the (x,y,s,o) configuration for each peak in the voting space. (1 point)
- At this point, you will have many possible configurations for objects (see Fig. 3). Reduce the number of returned configurations as much as you can, such that you eventually end up with only one bounding box for each object (as in Fig. 1). Your bounding boxes do not have to be perfect. You may find `kmeans` or `bboxOverlapRatio` to be useful here. (4.5 points)

Figure 3: All possible configurations extracted from votes in the GHT. These need to be processed so that (ideally) there is one configuration per object

3. Set up Transformation Matrix (1 points)

After the configurations of the objects have been found, they should be indicated by bounding boxes placed on top of the query image (see Fig. 1 and 3). The coordinates of the rectangles are defined in the object coordinate system in the main script. To display the rectangles at the correct position in the image, the coordinates first have to be transformed to the image coordinate system, according to the detected translation, scale and rotation of the object. In the file `transform_points.m`, you have to implement this transformation. Set up the 3-dimensional transformation matrix and apply it to all input points. **Use homogeneous coordinates to be able to vectorize the transformation!** You do not have to add code to draw the rectangles, this is done by `draw_rectangles`. The main script only needs to be adapted to set the vlfeat path and change the image index.

Useful commands

`ind2sub`

Part 2: Documentation (6 points)

- Answer each numbered question separately, with the answer labelled according to the question number. You do not need to include the question text.
- Include images to support your answers.

For the experiments, you are provided with a template image of the object (object.png) and four test images containing the object several times in more or less cluttered scenes (image1.png to image4.png).

1. Include the detection results for all the images, and write a short interpretation of them. Does your code work well on all images? If there are issues in any of the images, add a few sentences explaining what you think might be the cause. (2 points)
2. Describe your method to detect peaks in the voting, and your approach to reducing the number of possible object configurations. (2 points)
3. Using the image your code performs best on, show results using a bad parameter setting for the position resolution, orientation resolution, and scale resolution, and explain what you see. Test each parameter independently. For the parameters which you are not testing, use a setting which you found to give good results. (2 points)

For all result images, use the same parameters (e.g. grid resolutions) and mention them in the documentation so we can reproduce your results.

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

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 `detect_object.m`, `transform_points.m` and `match_to_points.m` as well as your documentation (pdf-format) **as one ZIP-file** via TUWEL.

Submission status

Submission status	No attempt
Grading status	Not marked
Due date	Monday, 19 November 2018, 12:00 PM
Time remaining	13 days 23 hours
Last modified	-
Submission comments	Comments (0)

Add submission

You have not made a submission yet

◄ Generalizing the Hough transform to dete… Jump to... Files for Exercise 3 ►

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