

Intro to Image Understanding (CSC420)

Assignment 2

Posted: Feb 2, 2018 Submission Deadline : Feb 10, 11.59pm, 2018

Instructions for submission: Please write a document (either pdf, doc, etc) with your solutions (include pictures where needed). Include your code inside the document. Please submit through MarkUs. You are expected to work on the assignment **individually**.

Max points: 12 + max extra credit points: 4

1. Interest point detection:
 - (a) **[3 points]** Implement a function to perform Harris corner detection. Plot your result for the attached image `building_1.jpg`, and add it to your pdf/doc file.
 - (b) **[2 points]** Write pseudo-code (describe the algorithm in detail) for Lowe's scale-invariant interest point detection. Let the number of scales per octave be a parameter of your pseudo-code.
2. For this exercise you will use the Scale-Invariant Feature Transform (SIFT) for matching. You will extract SIFT features from two images and use them to find feature correspondences and solve for the affine transformation between them. Please include code under each question. You are allowed to use existing code for SIFT key-point and descriptor extraction (but not for matching). Possible code to use:
 - Download code from
<http://www.robots.ox.ac.uk/~vedaldi/assets/sift/binaries/> or
<http://vision.ucla.edu/~vedaldi/code/sift.html> or
https://docs.opencv.org/3.1.0/da/df5/tutorial_py_sift_intro.html
 - **OR** download and install the VLfeat package from <http://www.vlfeat.org/>. This package is in general very useful for basic computer vision algorithms. However, it's bigger in size than the package above and it might be slightly more difficult to install. There are also other packages available online, and you are free to use any of them.
- (a) **[1 points] Feature extraction:** Compute SIFT features for `reference.png`, `test.png`, and `test2.png`. You can use e.g. function `SIFT()` to output, for each image, a list of feature descriptors and a list of their corresponding frames. Please visualize the detected keypoints on the image. By visualize we mean: plot the image, mark the center of each keypoint, and draw either a circle or rectangle to indicate the scale of each keypoint. For clarity, please plot only the first 100 keypoints. You can help yourself with the function `PLOTSIFT-FRAME` included in the link to the code above. (If you are using Matlab, remember to use `ADDPATH()` if you have unpacked the SIFT package into a subfolder.)
- (b) **[3 points] Matching:** Given the extracted features on `reference.png` and `test.png`, describe a simple matching algorithm to find the best feature matches (correspondences) for the features in `reference.png` and features in image `test.png`. Implement the algorithm in your favorite programming language. If you are using Matlab you can help yourself with the

attached function `DIST2.M`. Visualize the top 3 correspondences. Please do the same using the test image `test2.png`. Show each image and visualize each correspondence by indicating the feature's position and scale in the appropriate image. Use a separate color for each correspondence.

- (c) **[2 points] Affine transformation:** Use the top 3 correspondences from part (b) to solve for the affine transformation between the features in the two images.
- (d) **[1 point]** Visualize the affine transformation. Do this visualization by taking the four corners of the reference image, transforming them via the computed affine transformation to the points in the second image, and plotting those transformed points. Please also plot the edges between the points to indicate the parallelogram. If you are unsure what the instruction is, please look at Figure 12 of [Lowe, 2004].

Link to [Lowe, 2004]: <http://www.cs.ubc.ca/~lowe/papers/ijcv04.pdf>

3. **Extra credit [3 points]** (You can be up to two days late (for free) if you do well in this exercise) Detect windows in images `building*.jpg`. For ideas, you can check e.g. paper:

Jana Košečka, Wei Zhang, Extraction, matching and pose recovery based on dominant rectangular structures. *Computer Vision and Image Understanding*, 100(3):174293

<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.129.9184&rep=rep1&type=pdf>

You can assume the axes of the windows are aligned with the horizontal and vertical axis in the image. If it can be of some help, a function `findLines.m` to find lines in an image is attached. Write down your method, include the code in the pdf. Plot the results: a detection of a particular window is a bounding box (a rectangle) tightly fit around a window. The points you get will be based on how many windows you get correct/incorrect. You only need to detect windows on the frontal facade of the building (you can ignore windows that are not axis-aligned rectangles). Not detecting the door is ok, detecting it is ok too.

Additional credit [1 point] Can you find floors in each image? Here the lowest row of windows represents the first floor, the row of windows above this is the second floor, etc. Indicate each floor with a (one) horizontal line.