Exercise 1

Guidelines:

- Implement the code in each question using either Matlab or Python, including their libraries for basic operations like reading/writing images, overlaying in a figure, and off-the-shelf functions that are permitted in each question. Recommended are the Matlab Computer-Vision and Image-Processing toolboxes, the vl_feat library as well as Python Pillow, Opencv and scikit-image libraries.
- For each question, in addition to the required outputs and explanations to submit, you should submit <u>all the code files that you have written</u> (not including other libraries you have used).
- The input images for each question are organized in the respective folders Q1, Q2, Q3.
- In addition to the code files per question, you should submit <u>one document</u>, preferably in pdf format, with all the text and output figures required in each question.
- Submit the exercise to the email: cv.checker.21@gmail.com with the title "תרגיל בית 1". You may send updated versions (before the deadline) as well as grade appeals using the same thread.
- The submission deadline is Thursday April 22 at 23:59. Every late day will incur a reduction of 10 points off the grade.

Questions:

1. Hough Transform on equilateral triangles (35 pts)

The goal is to automatically detect equilateral triangles in natural images using a variant of the Hough Transform. You will need to apply your detector on 2 images (of your choice) from each of the folders triangles_1 and triangles_2, where the result should be overlayed on the images. Each detected triangle should be drawn (automatically by your code) using thin lines of a visible color. In each image you will attempt to detect all equilateral triangles <u>of a fixed edge length</u>, at all possible locations and orientations.

<u>Suggested stages</u>:

a. Compute a binary edge map of the image (you may use any off-the-shelf detector, such as Canny) - set its parameters (possibly per image) for best results.
b. Compute edge directions using a gradient computation.
c. Suggest a parameterization for the Hough transform and describe how an (oriented) edge point contributes to the Hough accumulator.
d. Implement the Hough loop, filling the parameterized table - Choose the triangle edge length and parameter step sizes per image, for best results.

e. Choose a threshold (per image) on the minimal vote count to detect triangles. If needed, apply non-maximal suppression to improve the result.

Submit:

1. Short description of the suggested parameterization and table filling method (step c. above)
2. For each of the 4 (2 from each folder) input images:
    a. The edge map computed
    b. The chosen edge length and resolution (step size) of each parameter
    c. The input image with overlayed detected triangles
    d. A short discussion (one short paragraph) on the quality of each result and the reasons for any imperfections (false detections, inaccurate detections, mis-detections) and suggestions as to how they could be dealt with.

2. SIFT-based correspondences (30 pts)

The goal is to compute and visualize matches between related image pairs, based on the SIFT method.

Suggested stages:

a. Use any off-the-shelf SIFT detector to run on both images in a pair of related images. The result of the detector, for each detected interest point (IP), is a 4 dimensional vector [x, y, r, t], where x and y are the image coordinates and r, t are the scale and orientation, as well as a 128-dim descriptor vector.
b. Given that there are N and M IPs in the first and second image, compute the N x M distance matrix between all pairs of descriptor vectors.
c. Extract the matches (correspondences) that have passed:
    i. the ratio-test (with ratio 0.8)
    ii. the bidirectional-test (require first-nearest neighbor in both directions)

Submit:

1. Write your own code to visualize the SIFT features for the single image UoH.jpg. You need to show the location+scale+orientation of each IP. This can be done using a single arrow, like we saw in class. You may draw a (random) subset of the IPs, if there are too many and the image is very crowded.
2. Write your own code to visualize (using connecting thin lines) the correspondences for two (of your choice) out of the three image pairs. Visualize separately for the case of the ratio test and the bidirectional case. If it is too crowded, you may visualize a smaller subset of correspondences.
3. Give a short discussion on the quality of each of the image pair results.

3. Image warping and RANSAC-based fitting (35 pts)

The goal is to estimate the 2D image transformation between a pair of images and warp one to the other accordingly.

Part 1: Warp image Dylan.jpg into the parallelogram and the quadrilateral in image frames.jpg.

Suggested stages:

a. Find off-the-shelf affine (3 match) and homography (4 match) minimal solvers.
b. Find an off-the-shelf warping function, which can take as input an affine or homography transformation.
c. Manually collect the relevant minimal seed correspondences and feed them into the minimal solvers to obtain the 2D transformations (affine, homography).
d. Apply the warps and overlay the warped source image in both target locations.

Submit:

1. The image frames.jpg with the image Dylan.jpg warped over it in both locations.

Part 2: Warp one stereo image over the other using a RANSAC fitted transformation

Suggested stages:

a. Run your code from question 2, in order to get a list of putative (candidate) correspondences for each image pair.
b. Implement a simple RANSAC loop with both options of affine and homography minimal solvers.
c. For each image pair, warp one of the images over the other (choose the direction), using the estimated transformations.

Submit:

1. For each image pair, show the result (one image warped over the other) for both affine and homography options.
2. Shortly discuss the result quality, including the difference between the affine and homography versions.