Intro to Image Understanding (CSC420)
Assignment 2

Submission Deadline : October 12 (Sunday), 11.59pm, 2014

Max points: 10, max extra credit points: 5

1. Write a function for Harris corner detection. Please copy a snippet of your code under each of the questions below accordingly. Question (b) does not need code. For each question also plot your result for the attached image building.jpg, and add it to your pdf/doc file.

   (a) [**1 point**] Given an image (input to function), compute I_x^2, I_y^2 and $I_x I_y$. Compute matrix M in each pixel (Lecture 6).

   (b) [**1 point**] If I have a 2×2 matrix $A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$, how do I compute its determinant $\det(A)$ and $\mathrm{trace}(A)$? Write down the equation. What's the equation to compute the cornerness measure R in each pixel? Write it with determinant and trace rather than the eigenvalues.

   (c) [**1 point**] Compute R in each pixel.

   (d) [**1 point**] Threshold R and perform non-maxima suppression to find corners. Plot an image and mark the corner locations. You can do this for example by: IMSHOW(IM); HOLD ON; PLOT(X,Y,'+R'); where x and y are the horizontal and vertical coordinates of the corners you detected.

   (e) **Extra credit** [**5 points**] Detect windows in image building.jpg. You are allowed to discuss this exercise with your colleagues, and find solutions / papers online. For ideas, you can check e.g. paper:
   Jana Košecka, 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. You only need to detect the windows on the frontal facade of the building (you can ignore the building on the right and the windows on the cupola). Not detecting the door is ok, detecting it is ok too.

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 do not need to implement your own SIFT key point and feature extractor. Possible code to use:

   - Download and unpack sift-0.9.19-bin.tar.gz from
     `http://www.robots.ox.ac.uk/~vedaldi/assets/sift/binaries/`

- **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.

(a) **[1 point] Feature extraction:** Write a function that computes SIFT features for both reference.png and test.png. You can use the function IMREADBW() to read in the images as grayscale. Rescale the images to take on grayscale values in $[0, 1]$. Then use the 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 PLOTSIFTFRAME included in the link to the code above. (Remember to use Matlab's ADDPATH() if you have unpacked the SIFT package into a subfolder.)

(b) **[2 points] Matching:** Given the extracted features, 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 Matlab. You can help yourself with the attached function DIST2.M. Visualize the top 3 correspondences. 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) **[1.5 points] Affine transformation:** Use the top 3 correspondences from part (b) to solve for the affine transformation between the features in the two images. Look at Lecture 8 or the solution described in Section 7.4 of [Lowe, 2004] (pages 22-23). (Hint: $[x, y]$ and $[u, v]$ are the positions of a feature correspondence on the two images.)
(Hint: use X=A\B to solve the system $Ax = b$.)

(d) **[0.5 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].

(e) **[1 point]** What limitations are present in the above approach? What would be a more robust way of estimating the transformation? Here by robust we mean a method that is less sensitive to wrong matches (outliers). You do not need to write the code, just explain your answer well.

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