CS5670 – Computer Vision
Assignment #2

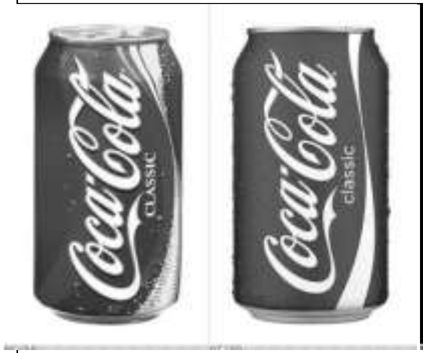
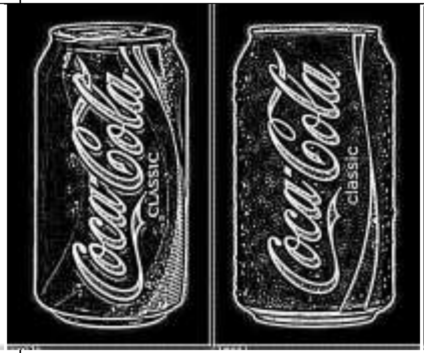
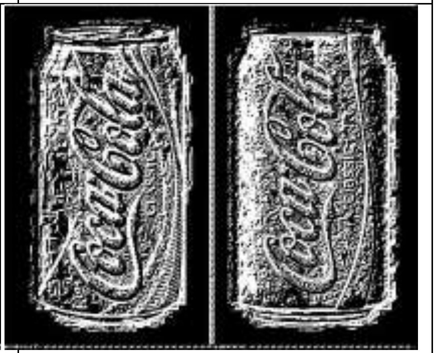
Submission details

- This assignment can be done individually or in pairs (though we strongly encourage you to work in pairs).
- Programs should be in Python. You can use either Python 2 or Python 3.
- The example Python code uses the SciPy library (http://www.scipy.org), which provides Matlab-like arrays with similar functionality. It also has some handy functions for plotting graphs of signals, which the Python standard library lacks. These operations tend to make signal and image processing algorithms much easier to code. You are not required to use this library, but it could make your life easier.
- To install (email TA if you run into trouble)
    - Ubuntu Linux:
      ```
      sudo apt-get install python-numpy python-scipy python-matplotlib ipython
      ipython-notebook python-pandas python-sympy python-nose
      ```
    - On Windows or Mac, you may have to install a Python distribution containing SciPy. We recommend the Anaconda distribution, but other distributions are also possible.
- For submission, package up your code as a zip file. Include your written answers as a pdf or word file named writeup.[pdf|docx]. Include graphs and images in your writeup, and please label them so we know which figures go with which sub-problem.
- Send the final zip file to the TA (see next bullet). Add the course name to the subject of the mail.
- If you have any questions about this assignment, please contact the TA (stinger@tx.technion.ac.il).

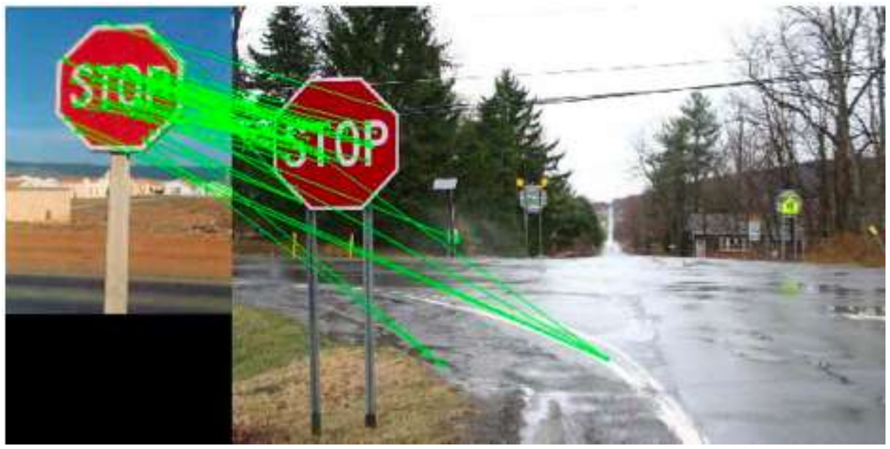
Task 1: SIFT

SIFT is a major building-block in computer vision and hence it is important to know how to use it properly, especially when you download free software. The purpose of this exercise is to clarify some delicate points regarding SIFT. These can be learned from practical use of SIFT. It is recommended to read David Lowe's IJCV'2004 paper prior to working on this exercise.

1. SIFT's goal is to find features that are mutual to different images of the same object, even when the image is noisy. Interest points are related to local changes in pixel values, and many operators have been proposed to date, for key-point selection. As an example, we show in the Figure below two input images, their DoG (Difference of Gaussians) and their Gradient magnitudes. Assuming your goal is to find interest points that are consistent under changes in the image, or noise, which of the operators will you choose?

Input images	DoG	Gradient
		

2. Use SIFT to find interest points and descriptors to all the provided input images. You should not implement SIFT yourself. Here are some links to software you could use:
    a. http://www.vlfeat.org/~vedaldi/code/sift.html
    b. http://opencv-python-tutroals.readthedocs.org/en/latest/py_tutorials/py_feature2d/py_sift_intro/py_sift_intro.html
    c. http://www.janeriksolem.net/2009/02/sift-python-implementation.html
    d. Find your preferred software. Make it clear which one you used.
3. Write a function "findMatches" that gets as input two sets of descriptors and their locations in the image (one set per image). The function should return a list or an array of corresponding descriptors (their coordinates). The matching should be based on similarity between the descriptors.
4. Write a function "showMatches" that gets as input two images, and the list (or array) of matches from the previous item. The function will generate an image where the matches are displayed. See an example bellow of a good visualization – it connects matching pixels by highlighted lines. Please generate these images for the following image pairs: StopSign1-StopSign2, StopSign1-StopSign3, StopSign1-StopSign4



5. Your next task is to use geometry to try and improve the matches. Write a function "affineMatches" that takes a list of matches between two images, uses RANSAC to find an affine transformation between the pair of images and rejects the outlier matches. You can implement RANSAC yourself, or download code for it. The function should return the computed affine transformation and the inlier matches. Generate again the visualizations of the image matches, this time using only inliers. Did RANSAC do what you expected it to do? How many matches did you have before and after RANSAC? Are all the remaining matches consistent geometrically?
6. We would like to examine the accuracy of the transformation you computed. You will write a function "alignImages" that gets as input a pair of images and the affine transformation between them. The function will apply the transformation to one of the images (warp) and will "merge" the warped image with the other image. Merging will be done as follows: Take the red and green channels from one image and the blue channel from the other image. Use them to generate a new image. How can you tell if the transformation you got is accurate? Did you get satisfactory results? When does it succeed and when does it fail?
7. Come up with a quantitative measure to assess the success of the alignment. The measure you propose and its results should make sense and explain well your results. Discuss the quantitative results you got.
8. Repeat Items 5 and 6, this time for a homography. Compare the quantitative results of with affine transformations and homographies. Can you explain the results?

<u>Task 2: Epipolar Geometry and Camera Calibration</u>

<u>Part 1</u>

Figure 2.1 shows a pair of cameras, each with a focal length of unity, whose principal axes meet at a point. The y-axes of both cameras are parallel and point out of the page. Assume that the left camera (center C_L) lies at the origin of the world coordinate system.

1. Write down the camera matrices for this configuration and verify that the fundamental matrix F is:

$$\begin{pmatrix} 0 & -d/2 & 0 \\ -d/2 & 0 & -\sqrt{3}d/2 \\ 0 & \sqrt{3}d/2 & 0 \end{pmatrix}$$

   Hint: Take care when constructing M', the projection matrix of the right camera. Do it by deriving X' in terms of X, via a sequence of stages:
   (i) a translation of the coordinate frame from C to the intersection point;
   (ii) a rotation of the coordinate frame about the intersection point and
   (iii) another translation to C'.

2. Compute the epipolar line in the right image corresponding to the homogeneous point

$x = (1,1,1)^T$ in the left from $l = Fx$.

3. Using Figure 2.1 determine where potential correspondences to the left image point $(x, y) = (0,0)$ can lie in the right image.
4. Describe the rotation and translation that should be applied to the left camera that would make the epipolar lines in the two images horizontal.

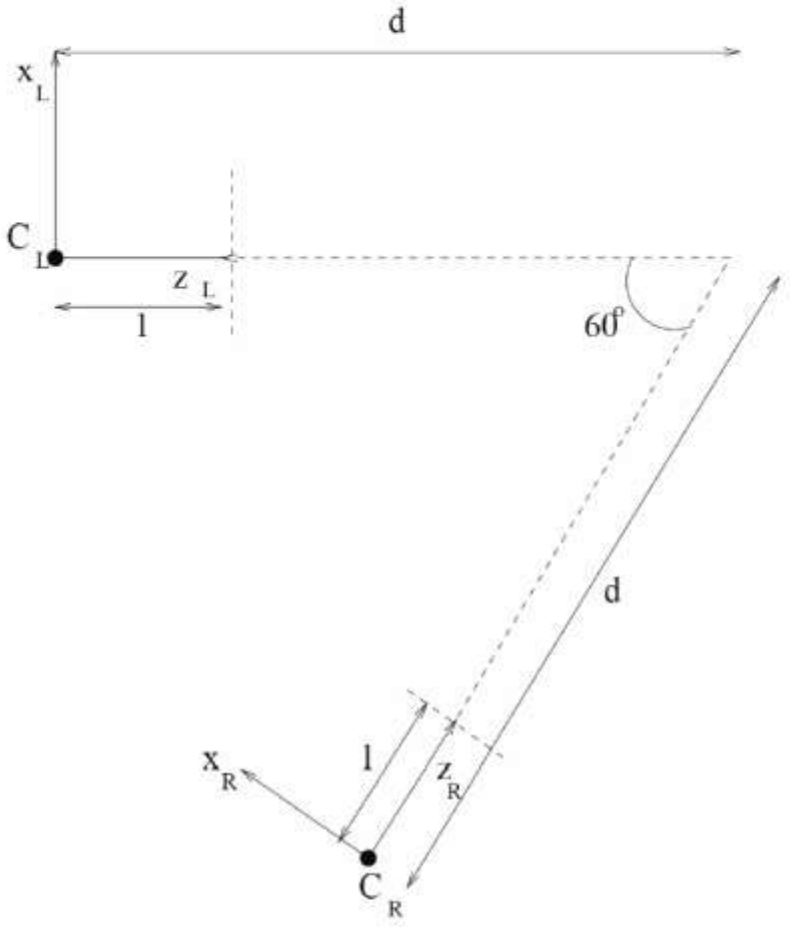


Figure 2.1

Part 2

Here the goal is the compute the 3x4 camera matrix P describing a pinhole camera, given the coordinates of 10 world points and their corresponding image projections. Then you will decompose P into the intrinsic and extrinsic parameters. You should write a simple Python script that works through the stages below, printing out the important terms.
Use the provided ASCII files, world.txt and image.txt. The first file contains the (X,Y,Z) values of 10 world points, and the second file contains the (x,y) projections of those 10 points.
Add to your writeup significant values resulting from the stages below.

1. **Find the 3x4 matrix P that projects the world points $\underline{X}$ to the 10 image points $\underline{x}$. This should be done in the following steps:**
   - Since P is a homogeneous matrix, the world and image points (which are 3 and 2-D respectively), need to be converted into homogeneous points by concatenating a 1 to each of them (thus becoming 4 and 3-D respectively).
   - We now note that $\underline{x} \times P\underline{X} = 0$, irrespective of the scale ambiguity. This allows us to setup a series of linear equations of the form:

     $$\begin{bmatrix} \mathbf{0}^T & -w_i\underline{\mathrm{X}}_i^T & y_i\underline{\mathrm{X}}_i^T \\ w_i\underline{\mathrm{X}}_i^T & \mathbf{0}^T & -x_i\underline{\mathrm{X}}_i^T \\ -y_i\underline{\mathrm{X}}_i^T & x_i\underline{\mathrm{X}}_i^T & \mathbf{0}^T \end{bmatrix} \begin{pmatrix} P_1^T \\ P_2^T \\ P_3^T \end{pmatrix} = \underline{\mathbf{0}}$$

     for each correspondence $\underline{\mathrm{x}}_i \leftrightarrow \underline{\mathrm{X}}_i$, where $\underline{\mathrm{x}}_i = (x_i, y_i, w_i)^T$, w_i being the homogeneous coordinate, and P_j is the jth row of P. But since the 3rd row is a linear combination of the first two, we need only consider the first two rows for each correspondence. Thus, you should form a 20 by 12 matrix A, each of the 10 correspondences contributing two rows. This yields $A\underline{p} = \underline{\mathbf{0}}$, $\underline{p}$ being the vector containing the entries of matrix P.
   - To solve for $\underline{p}$, we need to impose an extra constraint to avoid the trivial solution $\underline{p} = \underline{\mathbf{0}}$. One simple one is to use $\|\underline{p}\|_2 = 1$. This constraint is implicitly imposed when we compute the SVD of A. The value of $\underline{p}$ that minimizes $A\underline{p}$ subject to $\|\underline{p}\|_2 = 1$ is given by the eigenvector corresponding to the smallest singular value of A. To find this, compute the SVD of A, picking this eigenvector and reshaping it into a 3x4 matrix P.
   - Verify your answer by re-projecting the points $\underline{\mathrm{X}}$ and checking that they are close to $\underline{\mathrm{x}}$.
2. **Now that we have P, we can compute the world coordinates of the projection center of the camera $\underline{C}$. Note that $P\underline{C} = \underline{\mathbf{0}}$, thus $\underline{C}$ lies in the null space of P, which can again be found with an SVD. Compute the SVD of P and pick the vector corresponding to this null-space. Finally, convert it back to homogeneous coordinates and to yield the (X,Y,Z) coordinates.**