Image Warping 50 marks

[You may not use any functions of the image processing toolbox in this implementation]

a. Write a MATLAB function which will take as input the name of a gray scale image and six affine parameters a_1 to a_6. The output of the program should be another image file which is of the same size as the input file, but has the original image warped according to the six parameters. Any portion of the output image which falls outside the bounds of the input image should be cropped. Using your program, generate the following transformations:
    1. Counter-clockwise rotation of 15°
    2. Translation of +50 rows and -30 columns
    3. Scaling of 1.5 on x-axis (row direction) and 0.8 along y-axis.
    4. Scaling of 0.8 on x-axis, followed by a 45° (cc) rotation.
    5. 45° (cc) rotation, followed by a scaling of 0.8 on x-axis.
b. Try additional transformations on your own and report their results.
c. Figure out a transformation through trial and error such that transformed mecca06.pgm looks identical (or as close as you can get) to mecca06t.pgm?
d. Extend the warping function (implemented in part a) for color images. Try on mecca06.ppm and report your results.

e. Modify the above program so that the complete transformed image is always visible, i.e. there is no cropping of the transformed image at the corners.
f. Recovering affine transformation from control points, using least-squares (or pseudo inverse) approach:
Here, the input to the program will be two images, mecca06.pgm and mecca06t.pgm and a set of (at least three) control points (also called correspondences) between these images. The correspondences may be specified through a graphical interface by clicking (you can use `getpts` function), or may simply be read from a text file. The affine transformation between mecca06.pgm and mecca06t.pgm needs to be recovered first. After that, mecca06.pgm can be transformed by the recovered affine parameters. Then, the transformed image should look identical to mecca06t.pgm. Display transformed image and mecca06t.pgm together to highlight differences if any. [You do not have to implement generic code for this. You may do this part of the assignment for this specific case, manually]. Subtracting the two, taking the square and summing over the whole image can compute the error between them. Comment on the error image.

g. Implement projective warping in a similar fashion to part (a) of this question.