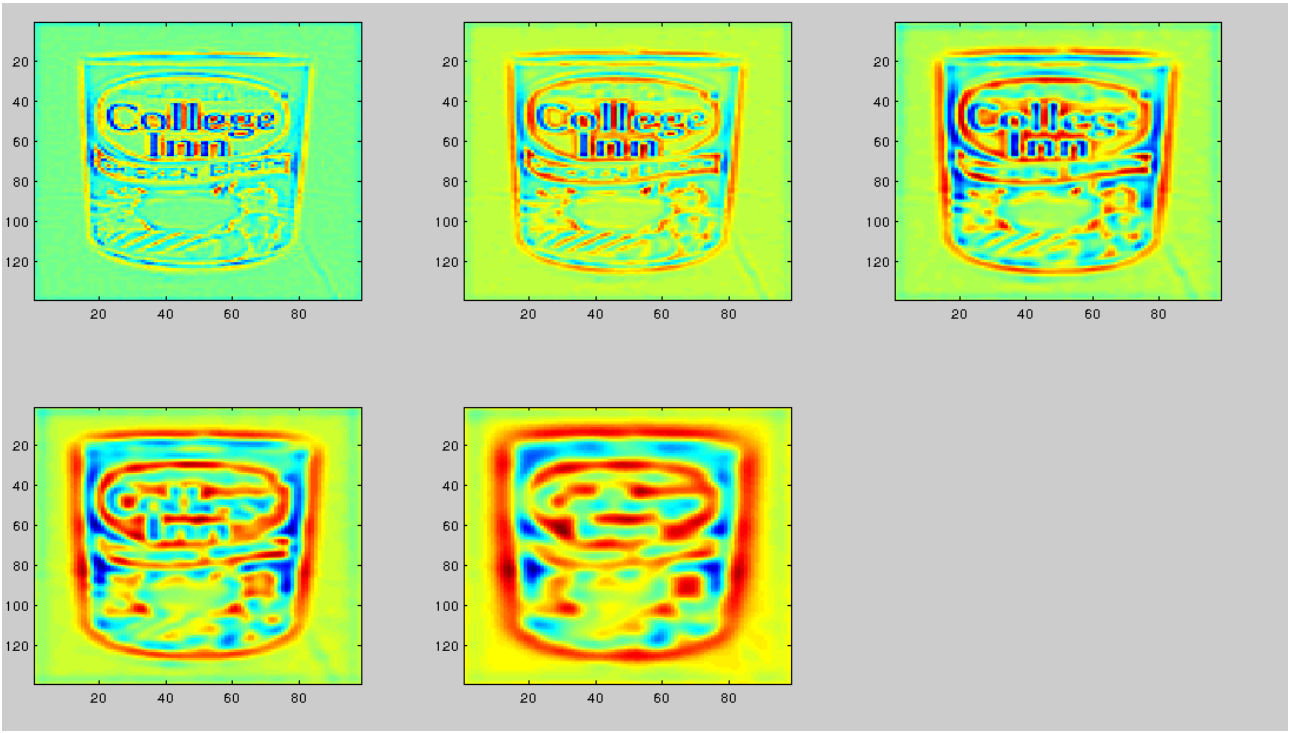
Computer Vision Assignment -3-

namank October 17, 2013.

Q1.1 In this question, I calculated DoG Pyramid by subtracting successive levels of the Gaussian Pyramid. Also, DoG_levels is returned which contains levels of the DoG Pyramid. DoG Pyramid for the image "model_chickenbroth.jpg" is shown:



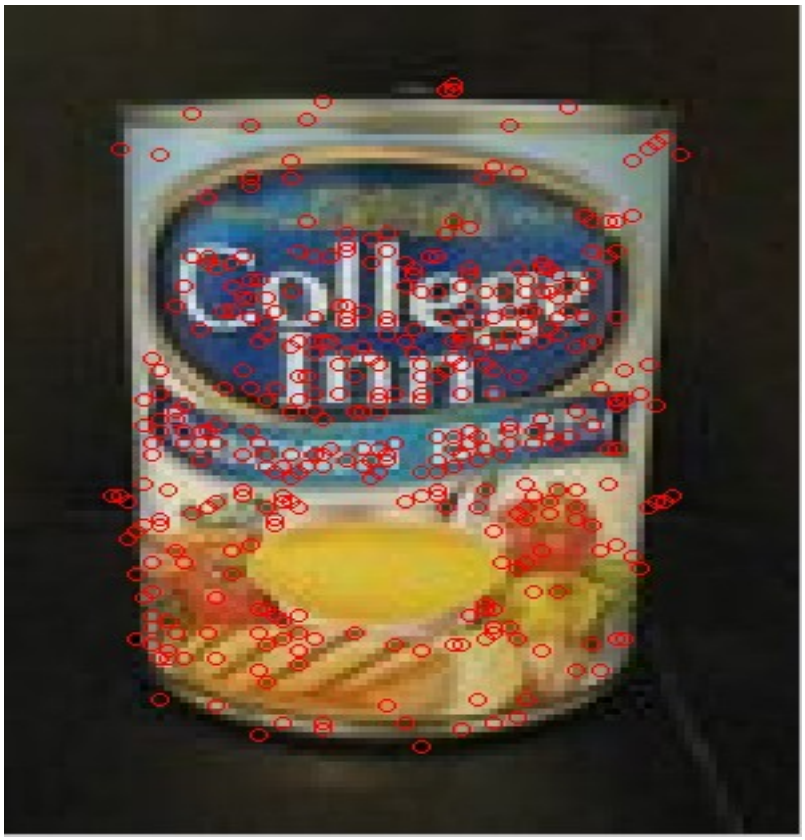
Q1.2 As edges do not represent good interest points, so we need to get rid of them and therefore in this question to perform edge suppression, we will be calculating principal curvature and will use the fact that edge points have a large principal curvature across the edge but a small one in the perpendicular direction. This function is returning Principal Curvature which is a matrix containing ratio R for the corresponding point in the DOG Pyramid, R = Trace(H)*Trace(H)./Det(H) where H is the hessian of the Difference of Gaussian.

Q1.3 In this part of the question, we will be detecting extremas in both scale and space.

a) We will consider point's eight neighbors in space and two neighbors in scale.
b) Principal Curvature of the point should be less than the threshold ratio.
c) Absolute value should be more than the threshold contrast.

Using the above conditions, we find locs which is a N X 3 matrix containing information about the interest points. First two columns contains the coordinates of the interest points and the last column contans the level of the DoG Pyramid.

Q1.4 In this question, everything is put together and image , other parameters are given as inputs and we obtain locs and Gaussian Pyramid as the output. The interest points for the image is shown below:



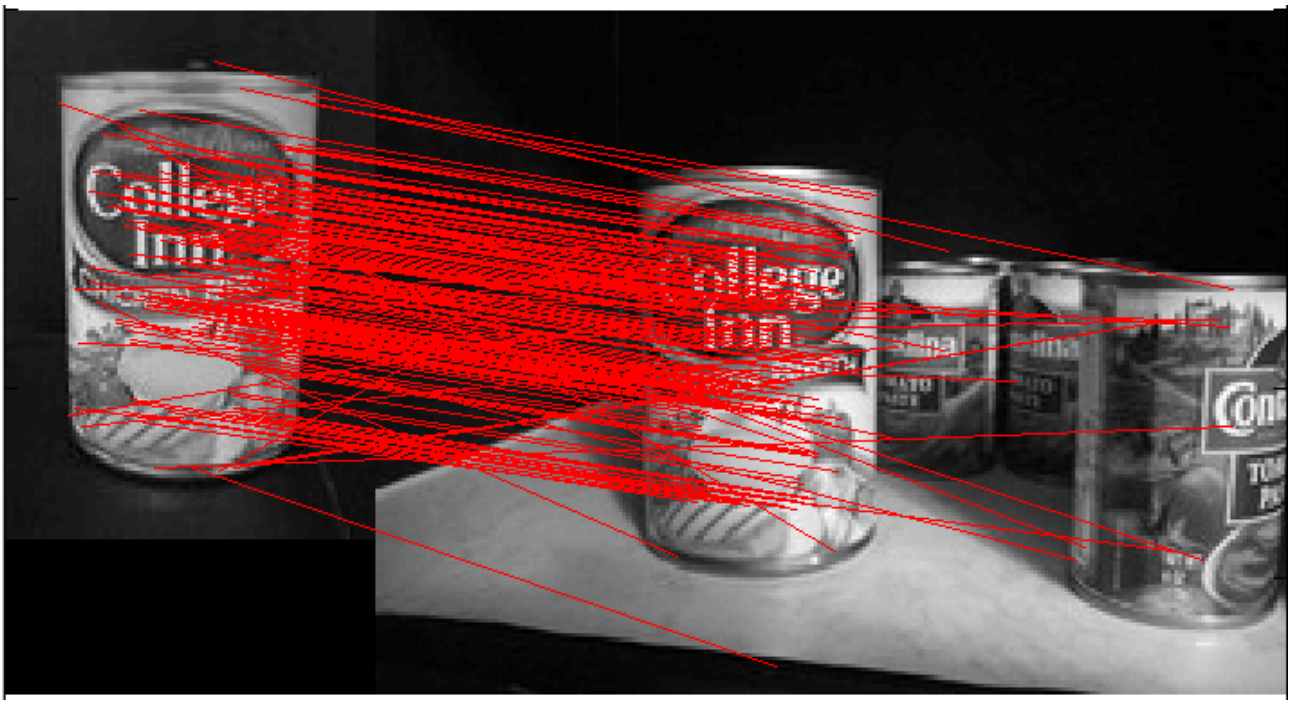
Q2.1 In this question, we will generate two 256 dimensional vectors where each value of the vector can vary from 1 to 81. The function is returning compareX and compareY which are linear indices in to the image patch. The result is saved in testpattern.mat

Q2.2 In this question, we compute the descriptor for all the interest points using BRIEF. I consider 9x9 neighborhood across the interest point and compare intensities of corresponding Gaussian level to compute the descriptor. Also, I ignore the feature points which is very close to the bounday.

Q2.3 In this part, I wll put everything together in a function which accepts a double grayscale image and returns the location and descriptors of the feature points for that image.

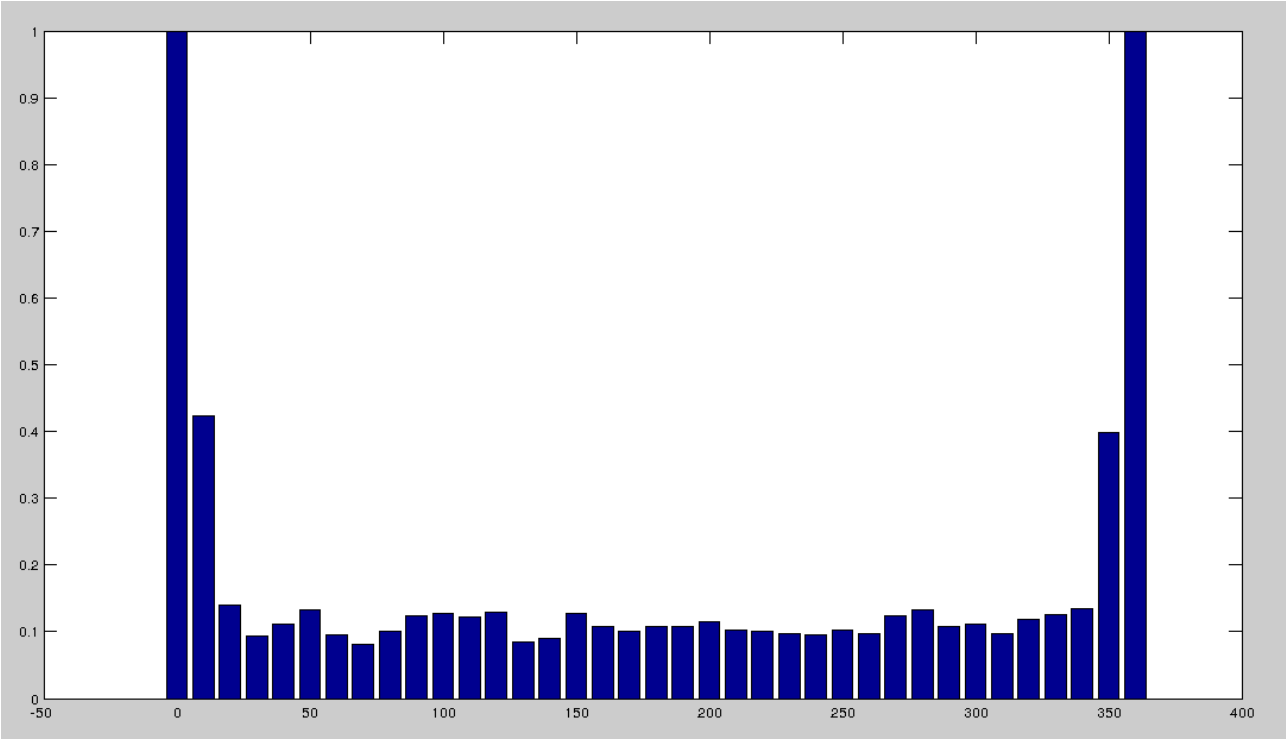
Q2.4 Once I have all the descriptors for the feature points, I have to perform Descriptor matching and I have used Hamming Distance to calculate the difference between two set of Descriptors. Also, to make sure matches are discrimative, I have calculated two closest neighbors and if the ratio of the distances to those two neighbors is greater than 0.8, that means that match is not discrimative, therefore I ignore it.
To choose ratio, I tried different values varying from 0.5 to 0.9 and I got the best matches at the ratio value of 0.8 and hence I used 0.8 for all my cases.

After that, I wrote a script testMatch which basically loads two images, computes descriptors and matches and finally plots the matches. The matches are as shown below:
a) Matches between model_chickenbroth.jpg and chickenbroth_01.jpg



b) Matches between model_chickenbroth.jpg and model_chickenbroth.jpg

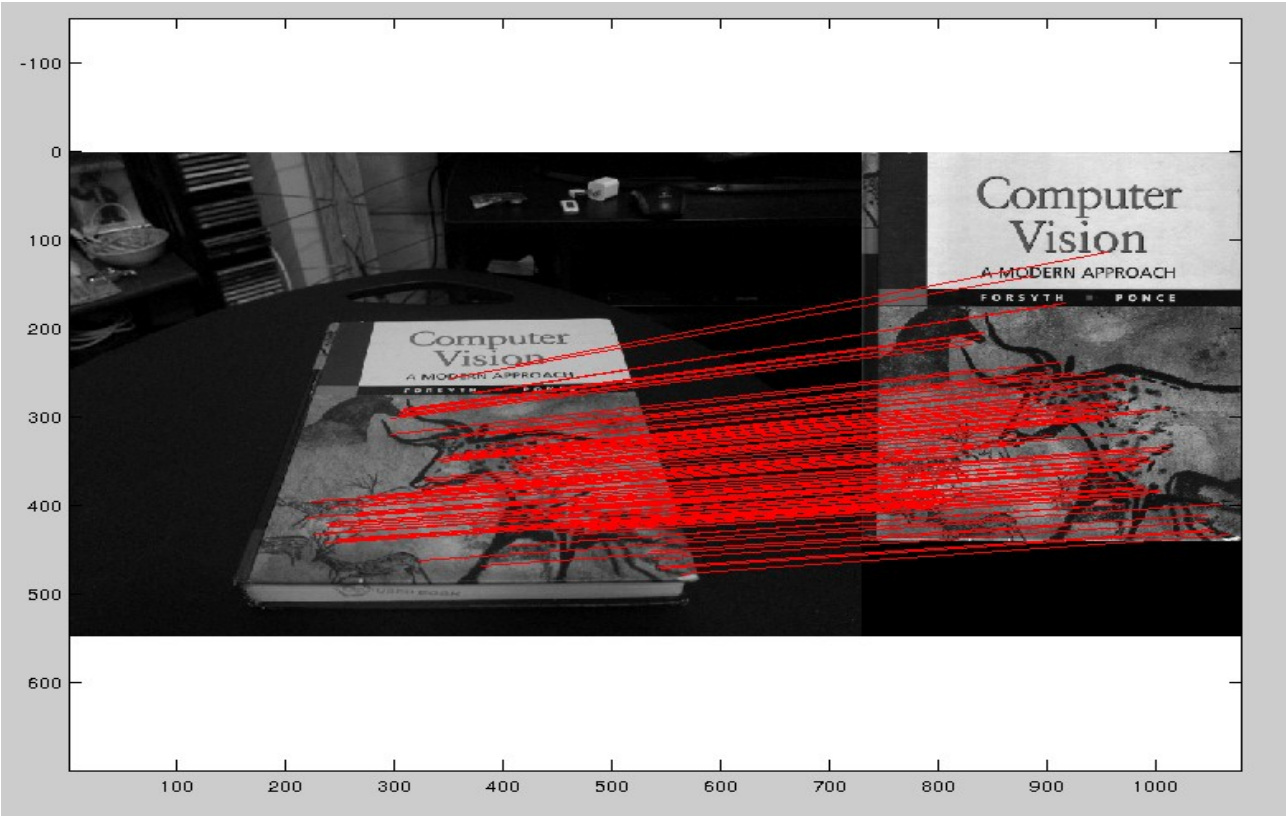
Q2.5 In this question, we have to rotate the image and compute matches between the rotated image and the original image. Then, finally plot the histogram. We get a histogram like this because as BRIEF is rotationally variant, so as image is rotated more and more, number of matches between the two images decreases. The histogram is as shown below. X axis is the angle of rotation and y axis represents matches in such a way that maximum number of matches is normalized to 1.

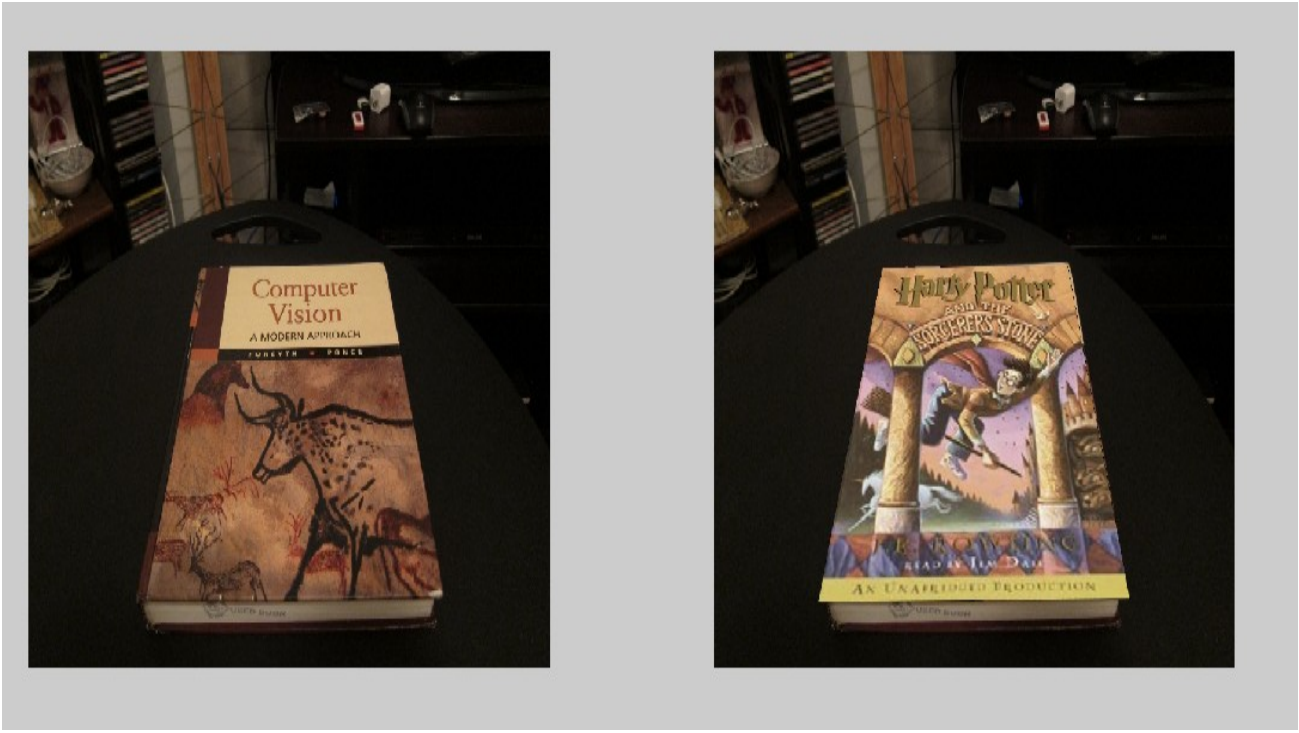


Q3.1 In this question, we have matches, locs for Image 1 and Image 2 and we will be using it to get rid of all the outliers.
a) Here, we are using RANSAC to fit any model to the data. Homograpy requires minimum 4 point pairs.
b) We select those 4 random points which have maximum number of inliers and will be using this to compute **bestH2to1**. Also, we will compute corresponding **bestError** and a vector of inliers where we will have 1 for **inliers** and 0 elsewhere.
c) As far as error metric is considered, I first calculated Mean Absolute Error and then converted it to **LOGARITHM ERROR** which is the best to use in cases where we have large number of outliers.

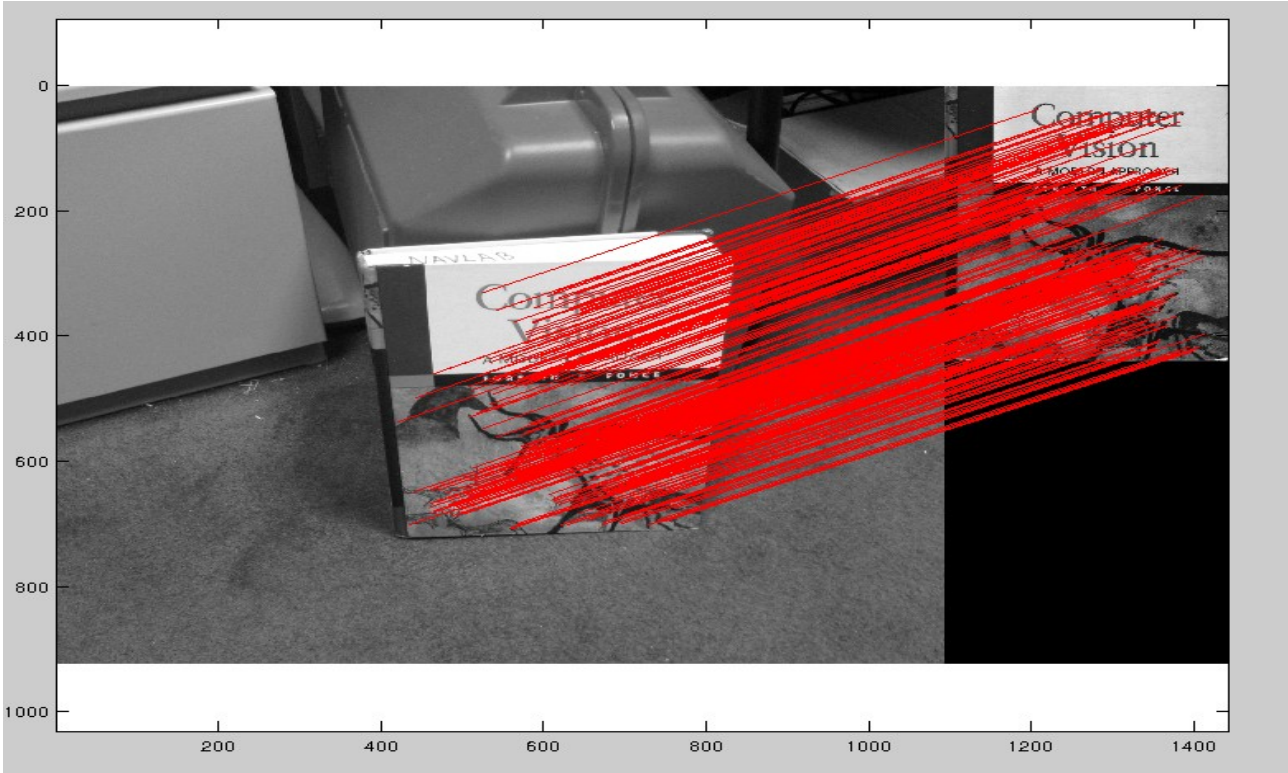
Q3.2 In this part of the question, we will put everything together. Here, we will read images, compute descriptors, compute matches, bestH2to1, bestError and inliers and then finally warp the image and display it.
Note : I am using nlfilter in matlab for the neighborhood of the interest point and therefore it takes some time to run neighborhood operation.

a) pf_desk matches and warping

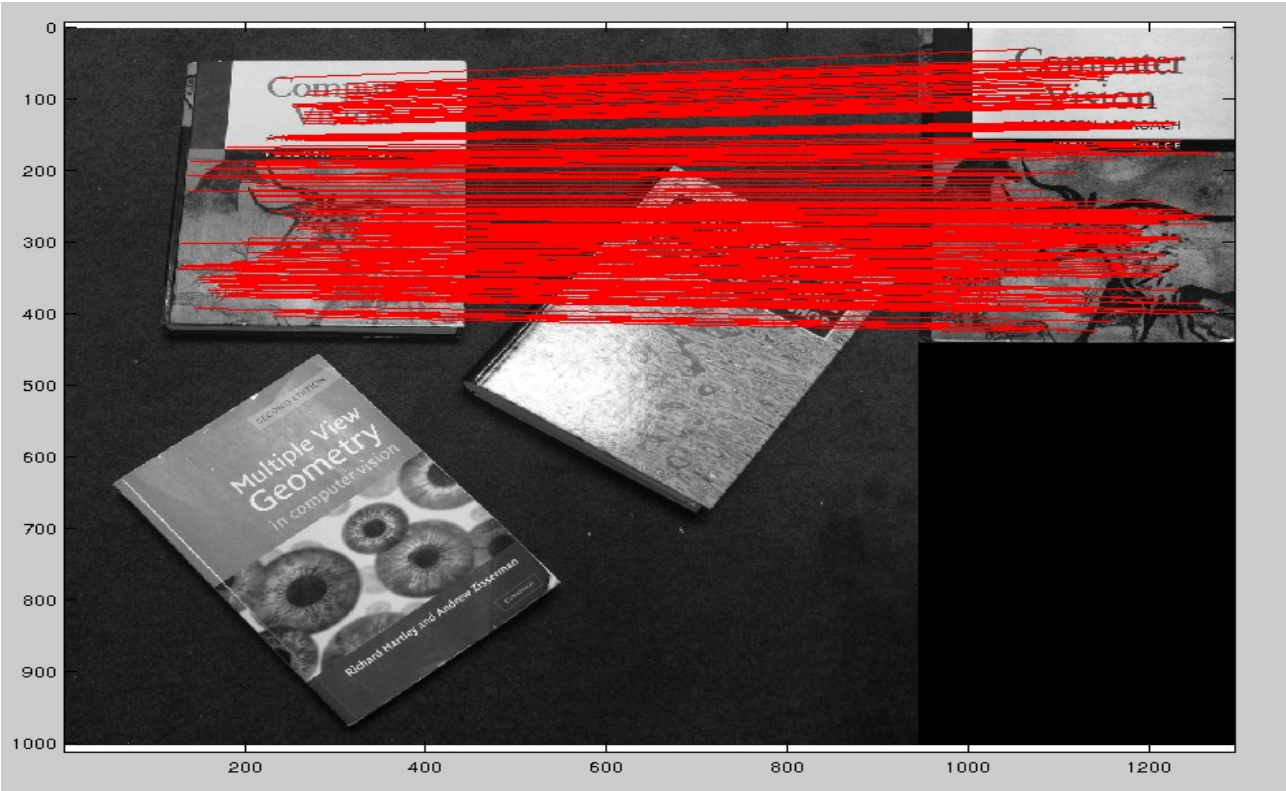




b) pf_stand matches and warping



c) pf_floor matchs and warping



Q3.3 Theory

Q 3.3.1

We have a plane with a known equation in 3D space.

P_i is a world coordinate $= \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$

p_i is the image point $= \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix}$ where $u_i = \frac{X}{Z}$, $v_i = \frac{Y}{Z}$

$\Rightarrow$ P_i is a world coordinate which is constrained to a plane with the equation:

$$aX + bY + cZ + d = 0$$

Now, the relation P_i and p_i is

$$p_i \equiv K[R\ t]\, P_i \quad \text{Ans}$$

where K is the camera matrix
R is the camera rotation
t is the camera translation.

Q 3.3.2

$$H \equiv KA$$

p is the image point.
P is the world point
M is the projection matrix

$$\Rightarrow p \equiv MP$$

$$p \equiv K[Rt] \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$p \equiv K[Rt] \begin{bmatrix} x \\ y \\ -\frac{d}{c} - \frac{a}{c}x - \frac{b}{c}y \\ 1 \end{bmatrix}$$

(As point is on the plane)

$$p \equiv K[Rt] \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -a/c & -b/c & 0 & -d/c \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$p \equiv K[Rt] \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -a/c & -b/c & -d/c \\ 0 & 0 & 1 \end{bmatrix}}_{\hookrightarrow B} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$p \equiv \underbrace{K[Rt]B}_{H} P \Rightarrow H \equiv K[Rt]B$$

and $H \equiv KA$

$$\Rightarrow A = [R\ t]B$$

Now let $A = [A_1\ A_2\ A_3]$ and $R = [R_1\ R_2\ R_3]$

then

$$A_1 = R_1 + \left(\frac{-a}{c}\right) R_3$$

$$A_2 = R_2 + \left(\frac{-b}{c}\right) R_3$$

$$A_3 = R_3\left(\frac{-d}{c}\right) + t$$

A_1, A_2, A_3 is 3×1

Ans

Q3.3.3

We know $\Rightarrow A = [R\ t]\,B$

$$\Rightarrow A_1 = R_1 + \left(\frac{-a}{c}\right)R_3$$

$$A_2 = R_2 + \left(\frac{-b}{c}\right)R_3$$

$$A_3 = R_3\left(\frac{-d}{c}\right) + t$$

As $A = K^{-1}H = H'$ (say)

Then $A_1 = H_1'$

$A_2 = H_2'$

$A_3 = H_3'$

$$\Rightarrow H_1' = R_1 + \left(\frac{-a}{c}\right)R_3 \Rightarrow R_1^T H_1' = R_1^T R_1 + \left(\frac{-a}{c}\right) R_1^T R_3$$

$$H_2' = R_2 + \left(\frac{-b}{c}\right)R_3 \qquad \boxed{R_1^T H_1' = 1}$$

$$H_3' = R_3\left(\frac{-d}{c}\right) + t$$

Similarly $\boxed{R_2^T H_2' = 1}$

$$R_3^T H_3' = -\frac{d}{c} + R_3^T t \Rightarrow \boxed{R_3^T (H_3' - t) = -\frac{d}{c}} \quad -(3)$$

Also, we know $\sqrt{R_{11}^2 + R_{12}^2 + R_{13}^2} = 1$

$\sqrt{R_{21}^2 + R_{22}^2 + R_{23}^2} = 1$ $\qquad (H' = K^{-1}H)$

$\sqrt{R_{31}^2 + R_{32}^2 + R_{33}^2} = 1$

From these equations, we get R_3.

Now we have R_1, R_2 and R_3 and we can use equation (3) to compute t.

$\Rightarrow$ Also, we will have more than one solutions

as $R_1^T H_1' = 0 \Rightarrow$ if $H_1' = [3 \quad 4 \quad 5]$

then R_1^T can be $\begin{bmatrix} 1/3 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1/4 \\ 0 \end{bmatrix}, \ldots$

$(H' = K^{-1} H)$

Also, we use $R_1 R_2$ to compute R_3 and hence they are interdependent.

Note: The above question 03.3.3 was discussed with Nitish, Kartik, Dhaval, Namrata and Akshay.

Q4. To make BRIEF rotationally invariant, there are lot of options like:
a) Assign most dominant orientation to each pixel and use this so that it becomes rotationally invariant as done in SIFT.
b) Also, we can rotate the neighborhood along with the pixel so that the neighborhood around the pixel does not change and therefore total number of matches will remain same.

I used part b) and by rotating the patch also, we can obtain rotational invariance. I used this concept but I was facing some issues in implementation and was not able to finish it successfully but this was the approach which I was following. I have also attached the code for this question named **Q4.m**