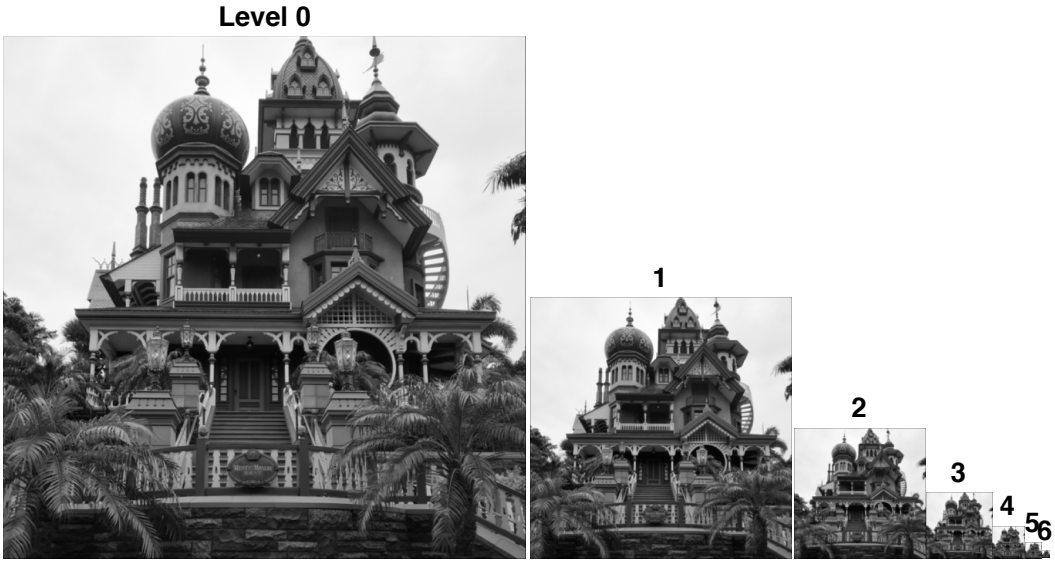
Assignment 2

Q1.
Generate a 7-level Gaussian pyramid shown as below.
Level 0 (size = 1024x1024, σ = 1) — Level 6 (size = 16x16, σ = 64)

Gaussian Pyramid



Q2.
Use the Gaussian Pyramid to generate a 6-level Laplacian Pyramid shown as below.

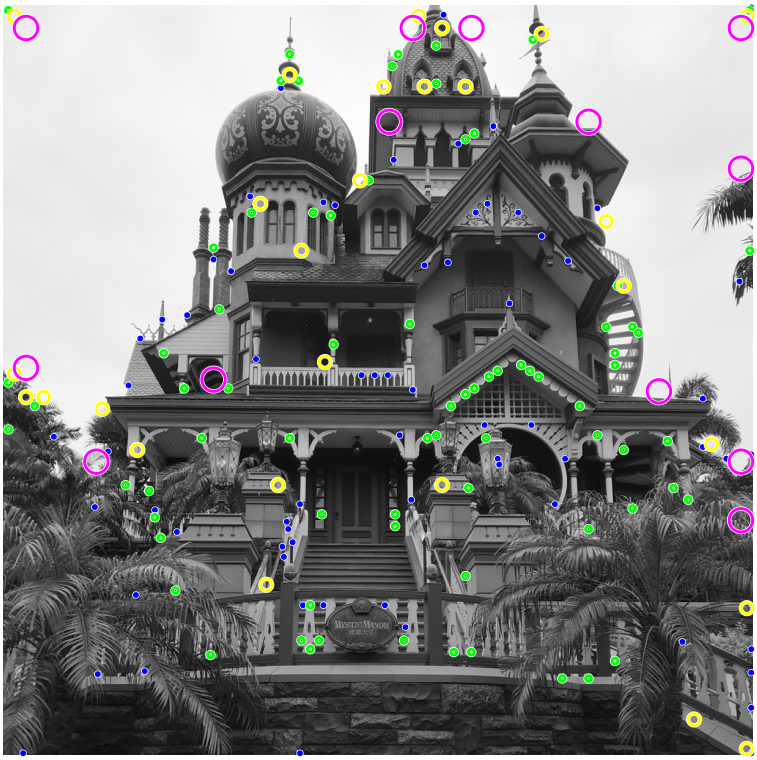
Laplacian Pyramid

Q3.

Choose a 3x3 local spatial neighborhood and find local minimums and maximums both in space and also in scale with respect to the levels above and below. In order to detect only strong extrema, values at extrema should be distinct from their neighbors in both scale and space by a threshold amount 5.

I use a 3-column matrix KeypointMatrix to store each keypoint's location and scale level (σ = 2^scale level). The scale levels of the 6-level Laplacian Pyramid (Level 0, 1, 2, 3, 4, 5) where key points can be found are 1, 2, 3 and 4. I use another 3-column matrix KeypointMatrixNormal to store each keypoint's corresponding location in the original 1024x1024 image and scale level as well.

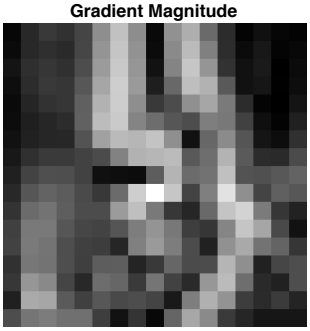
I Illustrate the keypoints by drawing circles at their locations overlayed on the original 1024x1024 image. For each scale level, I use a circle of a different color: blue (level 1), green (level 2), yellow (level 3), magenta (level 4) and the radius of a circle is σ. The result is shown as below.



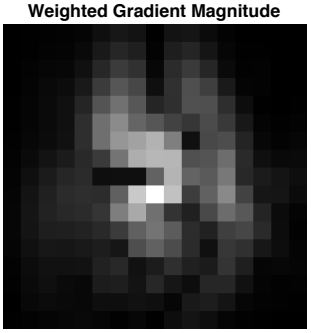
Q4.

Firstly, I delete those keypoints whose distances from bound are less than window_size/2. For each SIFT keypoint, I choose the image in the Gaussian pyramid at its scale and use a 17x17 window centred at the keypoint, and I calculate the gradient in X and Y direction for each pixel in this neighborhood. Then I get the gradient magnitude, orientation and a 2D Gaussian (σ = 4) weighted version of the gradient magnitude for each keypoint. For illustrative purposes, I select a keypoint which is located in (335, 83) of level 1 and create a visualization of the 16x16 patch, image gradient magnitude, image gradient orientation and weighted gradient magnitude shown as below.

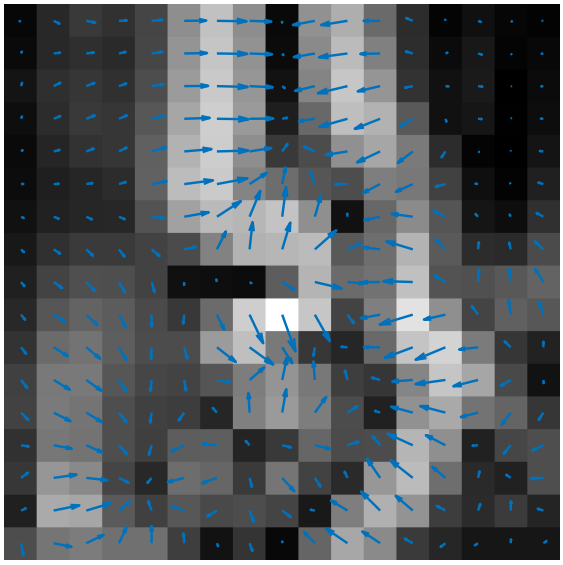
Gradient Magnitude



Weighted Gradient Magnitude

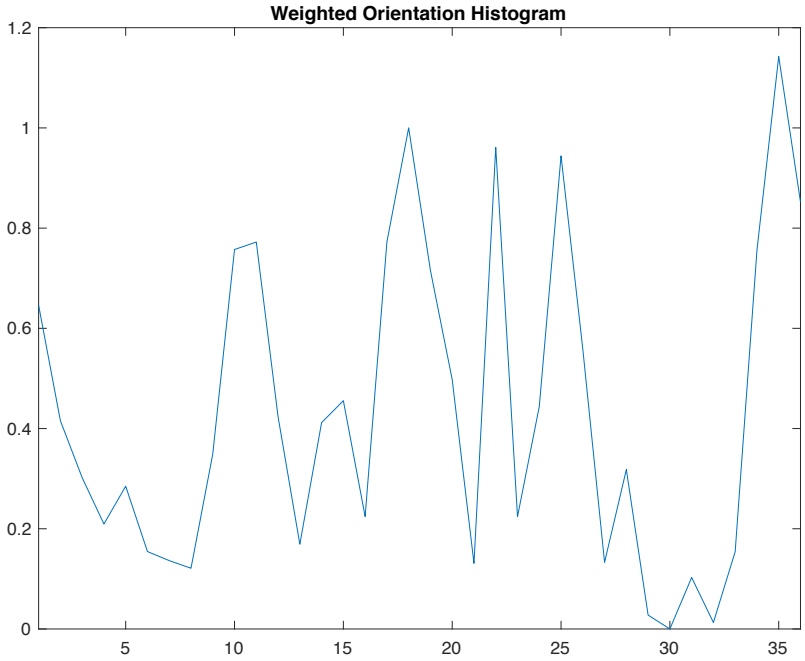


Gradient Orientation



Q5.

I create an orientation histogram with 36 orientation bins for each SIFT keypoint through adding up the weighted gradient magnitudes of all gradients in the 16x16 neighborhood whose orientations fall within that particular bin. I use a 36-column matrix (orientationhistogram) to store the results for all SIFT keypoints. For the same keypoint located in (335, 83) of level 1, its orientation histogram is shown as below. Then, I select the strongest peak and use it as the first entry in the histogram vector, using a CCW direction for the other entries. At the end of this part, I use a 39-column matrix to store the SIFT feature vector for each SIFT keypoint. Each SIFT keypoint is represented as (row(y), col(x), σ = 2^scale level, w0, w1,…, w35). The additional 36 entries w0, w1,…, w35 are the entries in the peak-aligned histogram vector. For example, the feature vector for the chosen keypoint is (83, 335, 2, 1.14258365385942, 0.852508717608413, 0.645968788142396, 0.415556029020691, 0.301941574313022, 0.209511320146868, 0.284902152746204, 0.154492293998944, 0.136444043976036, 0.121203288075670, 0.349080046648849, 0.757335211955444, 0.772162308301517, 0.423619948742393, 0.169152187991676, 0.411822193299117, 0.455595844530048, 0.224413666482644, 0.772314043330504, 1.00013026162325, 0.715523914899597, 0.497704465956189, 0.131009910771258, 0.961176836001227, 0.223998373740195, 0.443838780610157, 0.944159829055817, 0.558538768823207, 0.132957032716676, 0.318233409505772, 0.0277038044138713, 0, 0.102739896785083, 0.0129635299183218, 0.154298579692792, 0.755555109812943)



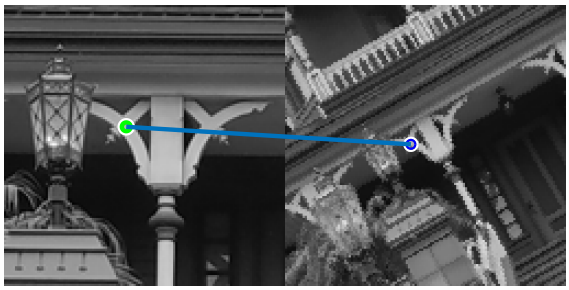
Q6.

First, I use built-in Matlab function imresize to get the scaled version of the original image. The chosen location (row_i, col_j) becomes (row_i*scale factor, col_j*scale factor). Then, I pad the scaled matrix with zeros to make sure that (row_i*scale factor, col_j*scale factor) can become the new center of the new matrix so that I can use the Matlab built-in function rotate to get the scaled and rotated matrix whose center is the same point as the chosen location in the original image. I crop the scaled and rotated matrix with the center unchanged to get the final 1024*1024 result matrix. If the size of scaled and rotated matrix is less than 1024*1024, pad it with zeros symmetrically. I choose two examples—(550, 500, 60, 2) and (400, 600, 30, 0.5). The results are shown below.

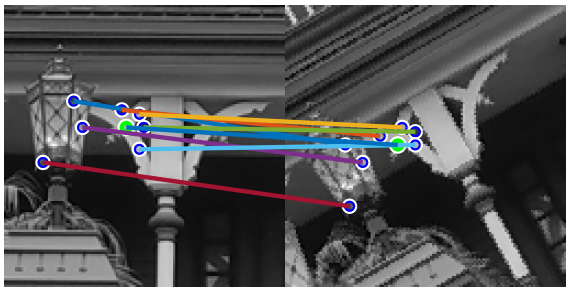
Q7.

For this question, in order to get more SIFT keypoints for comparison, values at extrema should be distinct from their neighbors in both scale and space by a threshold amount 1 instead of 5 as before. I choose the region of interest of size 128*128 centered at (400,600) and (550,500) and I set the threshold for the Bhattacharya coefficient as 0.25. For each SIFT feature vector, if the minimal Bhattacharya coefficient between itself and a SIFT feature vector in the transformed image is less than this threshold, these two SIFT keypoints are considered to be matched. For the transformed image, I choose several sets of roation degree and scale factor. The results are shown as below.

Center=(400,600) Roatation=30 ScaleFactor=0.5

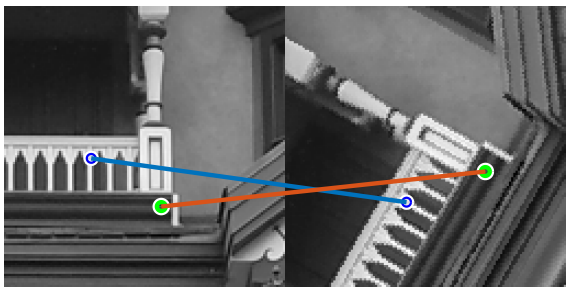


Center=(400,600) Roatation=30 ScaleFactor=0.95



Center=(500,550) Roatation=60 ScaleFactor=2

Center=(500,550) Roatation=60 ScaleFactor=1.05



For the results above, it can be seen that for the case (center=(500.550), rotation=60, ScaleFactor=2), there is no matched SIFT features between the original image and transformed image. It may be due to the large scale factor (the original image is expanded and the transformed image show less contents). For all other 3 cases, some SIFT features are matched in a correct way and there is no wrong match. It can also be concluded that, when the scale factor is nearer to 1, which means that the image is less scaled, more SIFT features in the original and transformed image can be matched.

Q8.

When we match each keypoints independently between two images, many of initial matches can be incorrect due to ambiguous features or feature that arise from background clutter. Therefore, a possible strategy is to match clusters of several keypoints (3 or more) since clusters have a higher probability of being correct than individual feature matches. The assumption for this strategy is that features from the same cluster should agree on the same object or its pose.

Reference:
Lowe D G. Distinctive image features from scale-invariant keypoints[J]. International journal of computer vision, 2004, 60(2): 91-110.