Comp 558 – Assignment 2

Manoosh Samiei
McGill University

November 11, 2019

Question 1:

To generate a gaussian pyramid, I first converted the image into gray scale, and in each level blurred the original image by a sigma (2^n) and down sampled by the same factor (2^n). The result would be 7 images with different sizes and blurring scales. Then I padded these images to match their size with the size of the original image and plotted them aligned like a pyramid. (I provided more details in the comments of my code.)

Figure 1.1- A Gaussian Pyramid made up of 7 levels

It can be seen that, as we go to higher levels in the gaussian pyramid, images become more blurred and smaller.

Question 2:

To generate a Laplacian pyramid, I first reconstructed each level from the next higher level (by interpolation and up-sampling) and then took the difference between the constructed image and the original image (in each level). For interpolation, I used two methods 'nearest' and 'bilinear' (which are inputs of 'resize' function). In Nearest-neighbor interpolation; the output pixel is assigned the value of the pixel that the point falls within (No other pixels are considered). Whereas in bilinear interpolation, the output pixel value is a weighted average of pixels in the nearest 2-by-2 neighborhood. Both methods output a roughly similar pyramid; however, nearest method seems to generate more convexity and concavity in the surface of image.

I also up-sampled image by a factor of 2 in each direction for each level. Finally, I used the same strategy as question 1 to plot images of all levels like a pyramid. (more details are provided in the comments of code.)

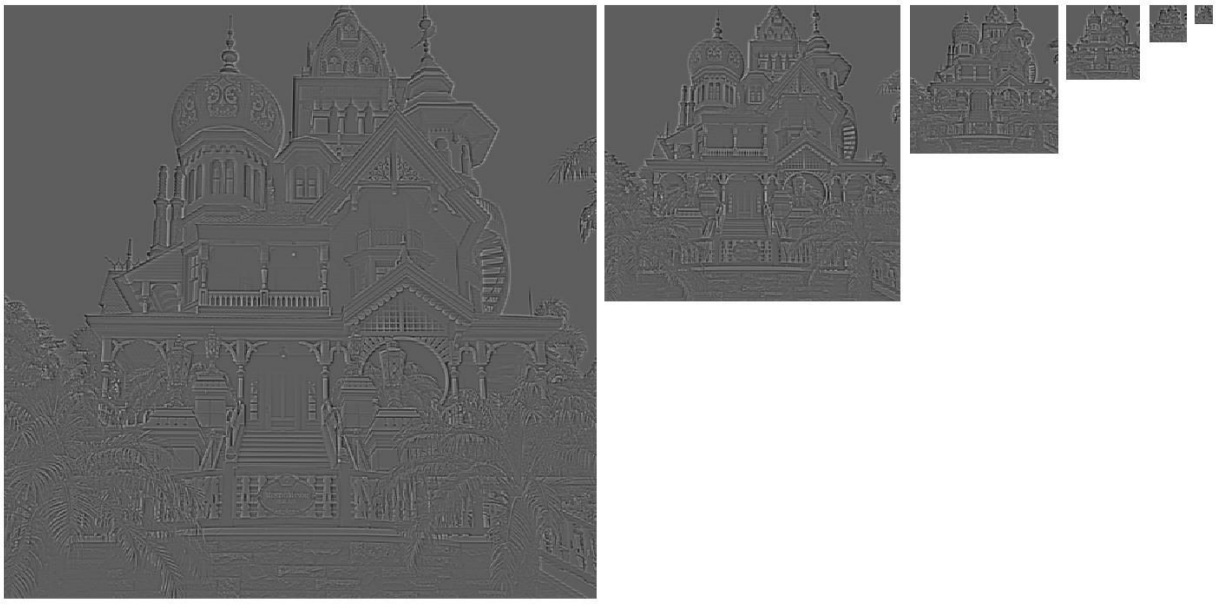

Figure 2.1- 6-level Laplacian Pyramid with K-nearest interpolation

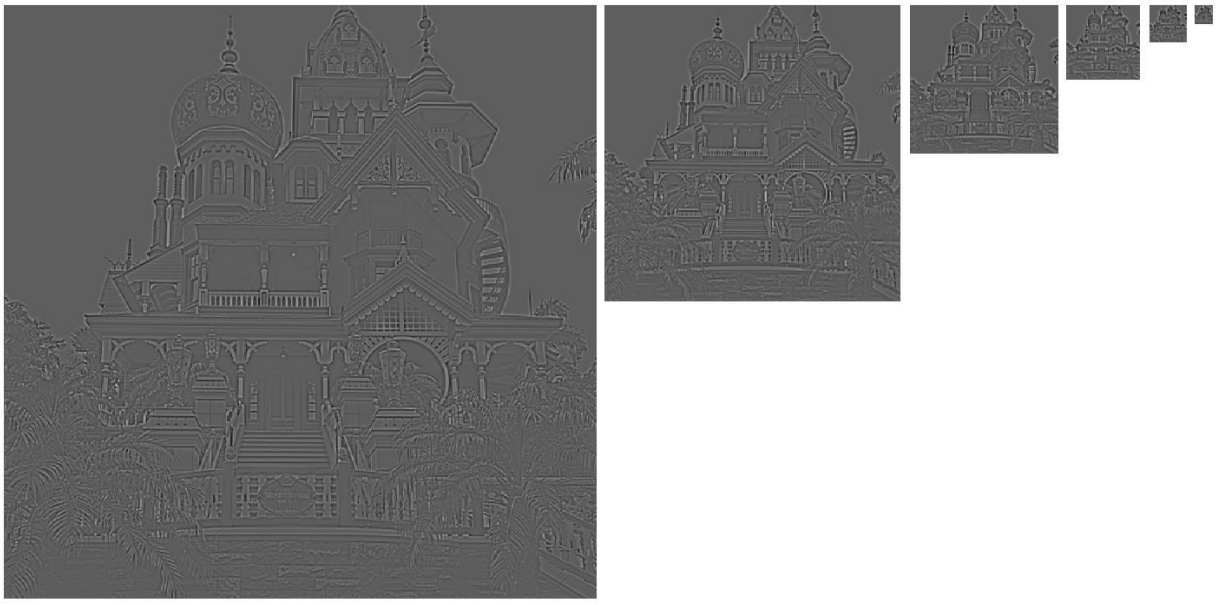

Figure 2.2- 6-level Laplacian Pyramid with bilinear interpolation

We can observe that edge-like features are highlighted at each level in the pyramids, as these features are more affected by gaussian blurring.

Question 3:

I extracted SIFT feature points from level 1 to 5 of Laplacian pyramid. To find SIFT points we need to find pixels with extremal values among pixels of each level (space), and pixels of its above and below level (scale). For finding extrema in scale, we need to up sample the above level, and down sample the below level. I considered 3 by 3 spatial neighborhoods for each pixel, which gave me 26 neighbor pixels in both scale and space.

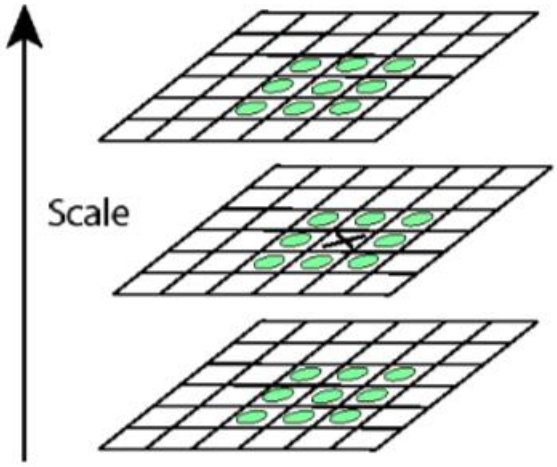

Figure 3.1- 3 by 3 neighborhoods in scale and space

To find the maxima and minima points in each level, I calculated the difference between each pixel's value and the value of all its 26 neighbor pixels. When all the differences are more than a positive threshold, it means that the pixel is a maximum point, and when all difference are less than a negative threshold, it means that pixel is a minimum point. (One mistake that I made first was that I took the absolute differences and made the condition that when all absolute differences are more than a positive value, that pixel is either minimum or a maximum. However, this is wrong. Consider 3 neighbor pixels with values such as:
-130; -110; +90. The absolute difference between the center pixel (-110) and its neighbors are respectively, 40 and 20. If we define the threshold as 10, they are both higher than threshold, and the center pixel is selected as an extremum while it is not.)

Another detail is that the level 5 is the last level in Laplacian pyramid, and we do not have any level above that to up sample. Hence, I only down sampled its below level (level 4) and calculated the difference between each pixel's value and the value of its 17 neighbor pixels. I used the same thresholding strategy as above, to detect extrema points.

After finding extrema points, I drew each point on the original image with a circle, whose color and size is chosen based on the point's scale level, using *viscircles* function in MATLAB.

Finally, I played with the threshold value to obtain a decent number of SIFT key points. When the threshold is too high, few key points are detected; while when it is set too low, lots of unnecessary and weak extrema points are detected. A threshold between 7 - 9 seems to be appropriate. Below pictures show the detected sift features points overlaid on the original image.

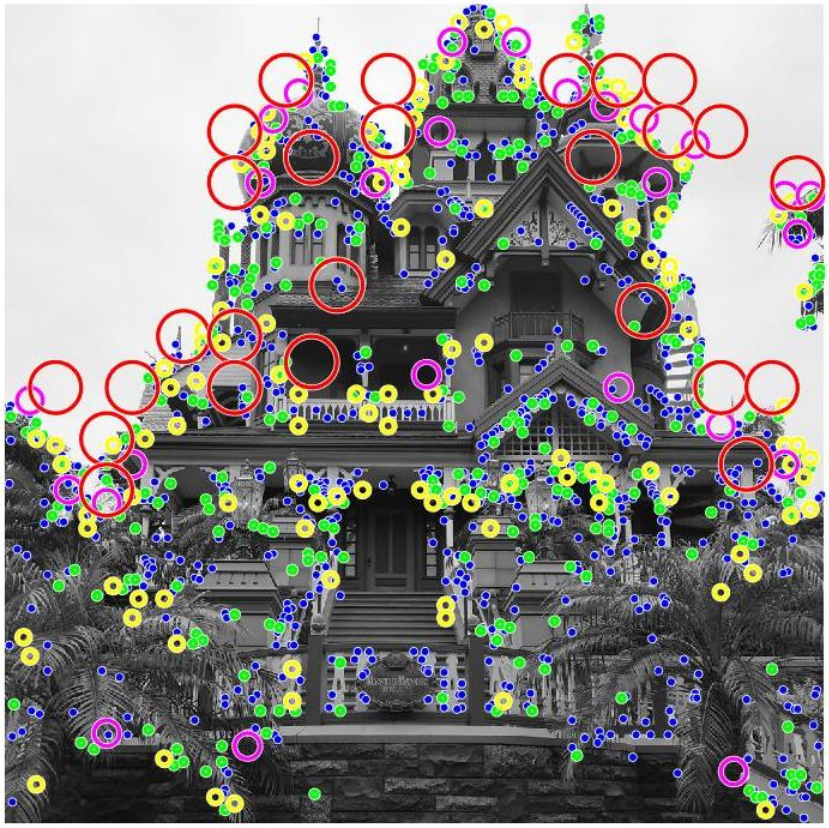

Figure 3.2- 1364 SIFT key points obtained with threshold=3, overlaid on original image

Figure 3.3- 293 SIFT key points obtained with threshold=7, overlaid on original image

Figure 3.4- 44 SIFT key points obtained with threshold=12, overlaid on original image

Question 4:

To compute SIFT feature vectors, for each SIFT key point I chose the image in the Gaussian pyramid at its scale. I used a 15x15 window centered at the key point, and for each pixel in this neighborhood calculated gradient magnitude, and orientation. Then, I multiplied each pixel's gradient magnitude by a 2D gaussian of sigma=2 and calculated a Gaussian weighted gradient magnitude. These weighted magnitudes have more value in the center of the window, where key point lies.

I only considered key points whose window is within the image boundary and ignored those whose windows are not covered among the image.

For visualization purposes, I selected the last key point whose 15x15 window is not out of boundary. (the last key point which was calculated in the loop).

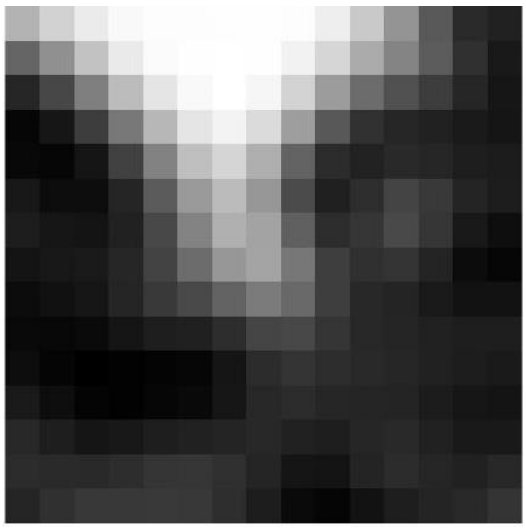

Figure 4.1- 15x15 gaussian image patch

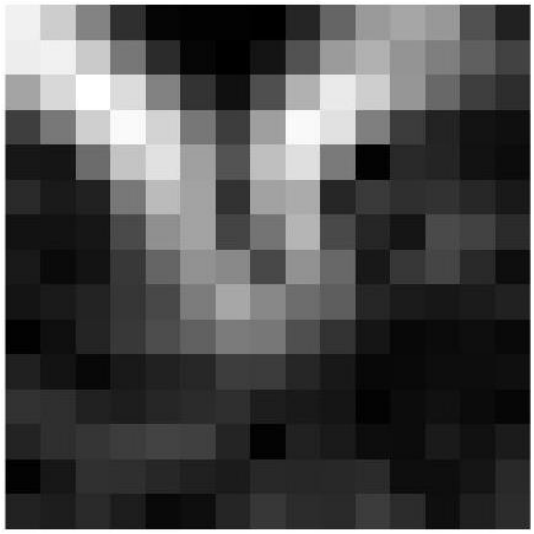

Figure 4.2- gradient magnitude of each pixel in previous figure patch

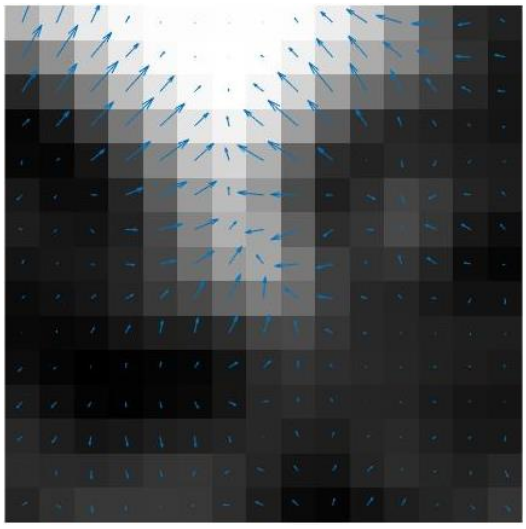

Figure 4.3- gradient orientations of each pixel in previous figure patch

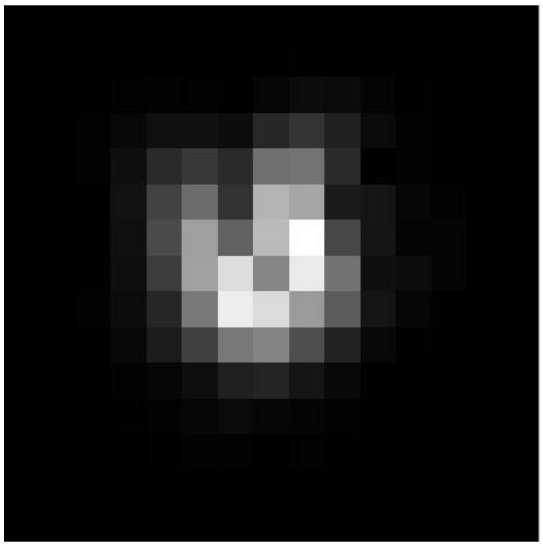

Figure 4.4- Weighted gradient magnitude (by2D Gaussian with sigma=2) of each pixel in previous figure patch

Question 5:

I created 1D histograms of gradient orientations, using the weighted gradient magnitudes, as below.

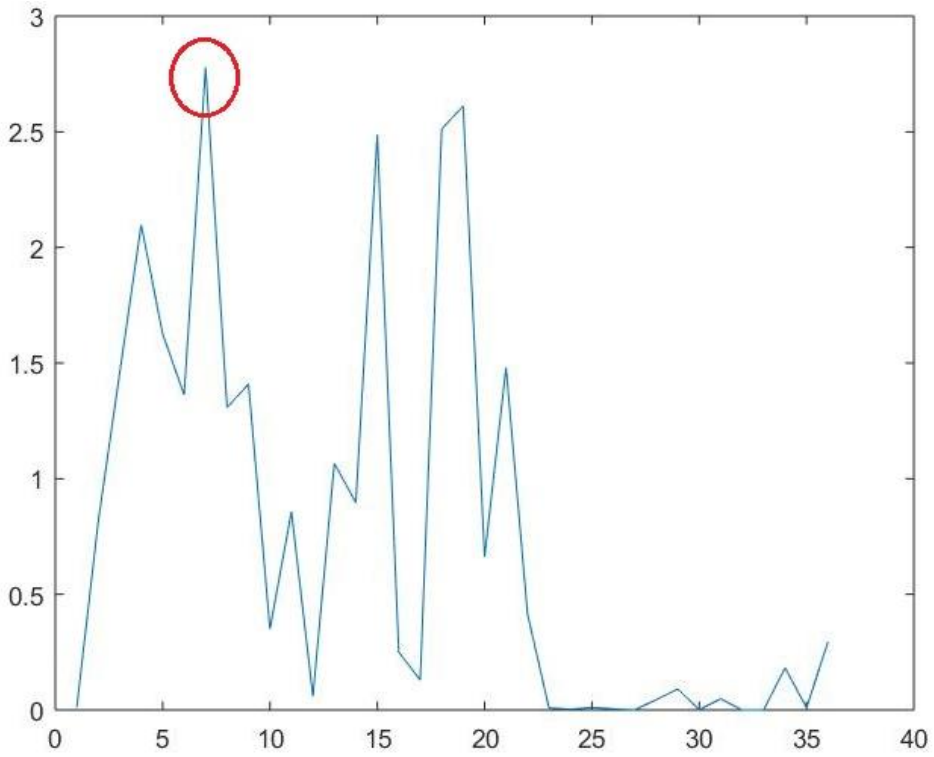

Figure 5.1- continuous plot of gradient orientations; the red circle shows the peak of the histogram.

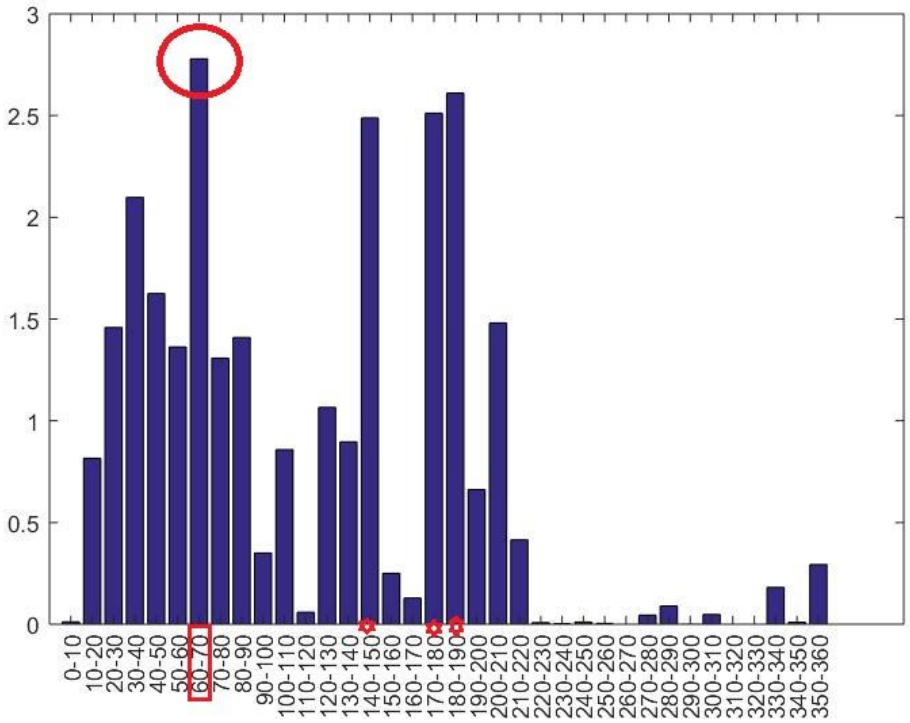

Figure 5.2- Discrete (bar) plot of gradient orientations; the red circle shows the peak of the histogram.

The peak of the histogram is for gradients' orientation of 60-70 degrees. It also has a noticeable number of gradients in rotations: 140-150, 170-180, and 180-190. All these results can be justified by looking at below image (similar to fig 4.3):

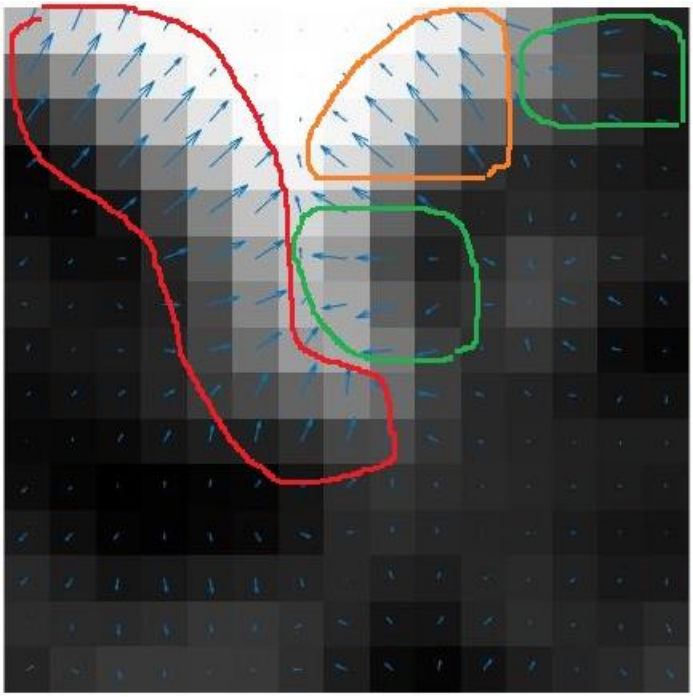

Figure 5.3- the red boundary shows the pixels with gradient orientation of 60-70 degrees which is the highest peak of histogram; the green boundary shows pixels with gradient orientation of 170-190 degrees which is the second peak of histogram; and the orange boundary shows pixels with gradient orientation of 140-150 degrees which is the third peak of histogram.

Furthermore, I created SIFT feature vectors for each key point, represented by a 39-tuple: "(x,y,σ,$w0$,$w1$,…,$w35$)". The details are presented in the code.

Question 6:

'imrotate' and *'imresize'* functions are only able to rotate and scale an image around its center point. To rotate and scale an image around another point in the image we can first pad the image with zeros (by calculating how many rows and columns to pad to create a virtual center), rotating and scaling around its center (using *'imrotate'* and *'imresize'*), and then cropping to remove the initial padding and returning the image to its original size (un-padding the image).
To correctly pad the image to form a virtual center in the location of transformation point, I divided the image into 4 sections, and padded the image depending on which section the transformation point lies.

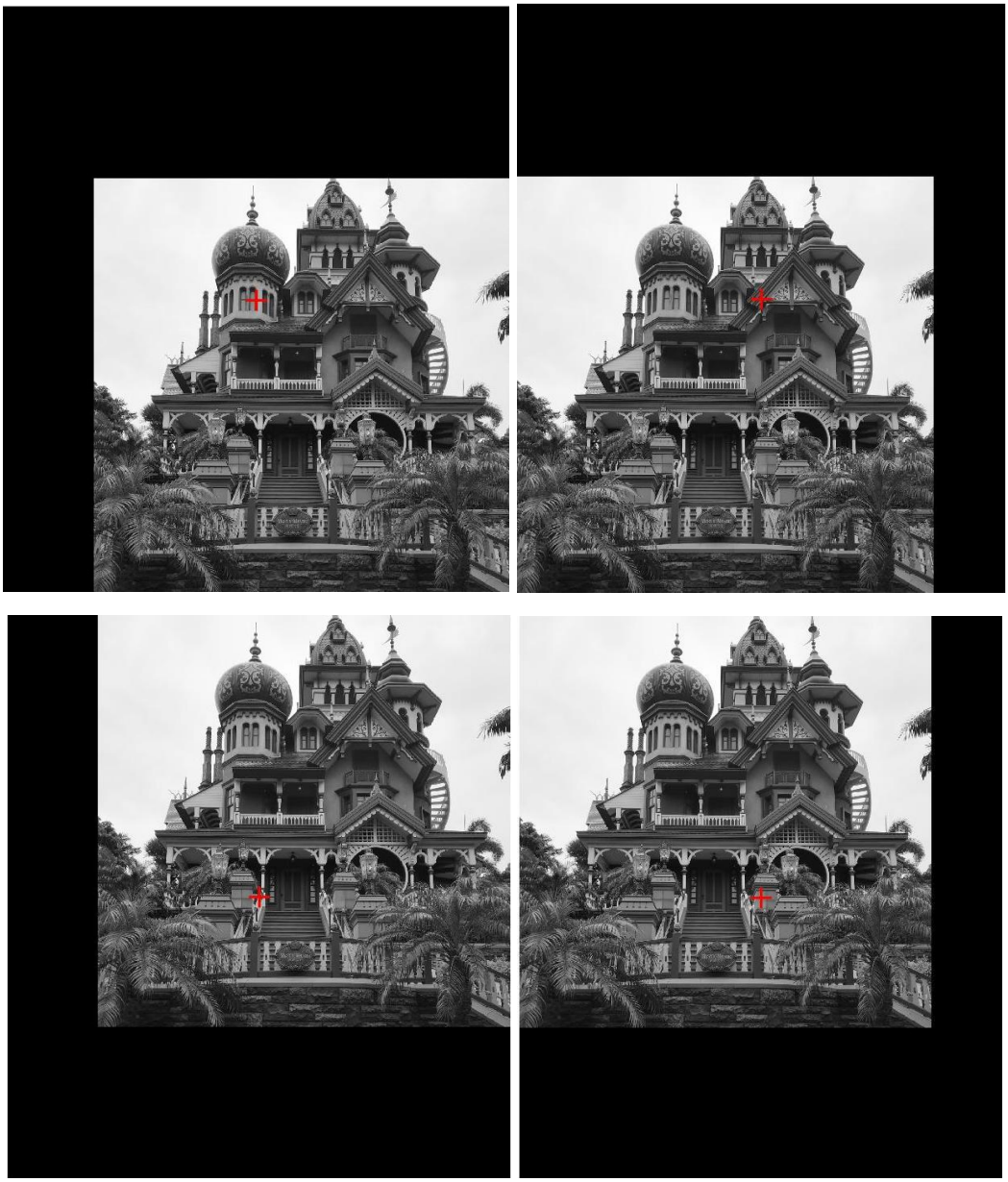

Figure 6.1- The transformation point (x,y) coordinates are (300,400) in upper left image; (600,400) in upper right image, (300,700) in lower left image; (600,700) in lower right image. The red cross shows the center of the padded image, which corresponds to the location of the transformation point.

The algorithm is as follows: first, I calculate the center of the image. Second, I calculate the amount of padding I should add the columns and rows of the image to make the transformation point a virtual center. Third, I calculated the difference between the coordinates of the image center and transformation point, and based on the sign of these differences, which determines in which quarter of image the transformation point lies, decided where to concatenate these zero rows and columns to the image. forth, I rotated and scaled the padded image, using *imrotate* and *imresize* functions in MATLAB. Finally, I cropped the image around the transformation point to return its final size back to 1024x1024.

Fig 6.2- The upper left image is the original image. The upper right image is rotated by 30 degrees around point (400,300) (the point is shown with a red cross in all images). The lower left image is scaled by factor 2 around the same point as above. The lower right image is both rotated by 30 degrees and scaled by factor 2 around the same point.

Fig 6.3- The upper left image is the original image. The upper right image is rotated by 45 degrees around point (400,700) (the point is shown with a red cross in all images). The lower left image is scaled by factor 3 around the same point as above. The lower right image is both rotated by 45 degrees and scaled by factor 3 around the same point.

Question 7:

To test that my matching feature algorithm works, I used the same original image and checked whether all sift key points are correctly matched.

Figure 7.1- checking the algorithm using the same original image for sift feature matching.

Sift points in image scaled by factor 3:

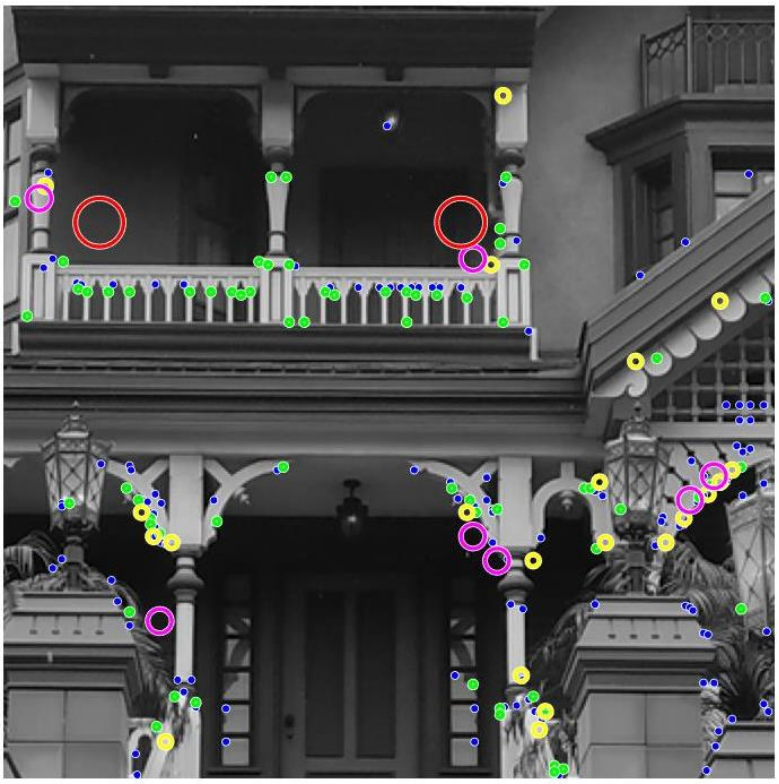

Figure 7.2- Sift points in image scaled by factor 3.

Matched points between original image and a scaled version of original image by factor 3:

Figure 7.3- The left image is original image and the right image is a scaled version by factor 3. The correctly matched points are marked with green circles. Here a region of interest is defined in the original image. While no ROI is defined for the scaled image. (3 points are showed here)

Figure 7.4- The left image is original image and the right image is a scaled version by factor 3. The correctly matched points are marked with green circles. (4 points are showed here) Here a region of interest is defined in the original image. While no ROI is defined for the scaled image.

Matched points between original image and a scaled version of original image by factor 2:

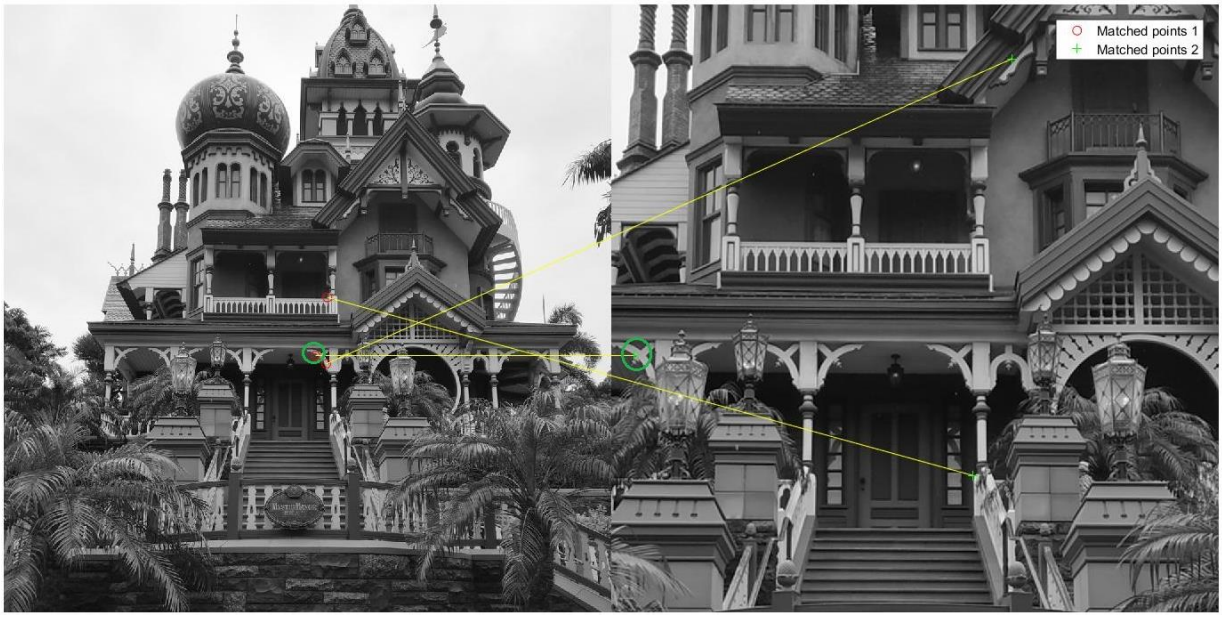

Figure 7.5- The left image is original image and the right image is an scaled version by factor 2. The correctly matched points are marked with green circles. Here a region of interest is defined in the original image. While no ROI is defined for the scaled image. (3 points are showed here)

Figure 7.6- The left image is original image and the right image is an scaled version by factor 2. The correctly matched points are marked with green circles. Here a region of interest is defined in both the original image and the scaled image. (3 points are showed here)

It can be seen from the two above images that some times region of interest can improve performance, while it is not true for all the cases.

Also, increasing the scale by a big factor will negatively affect matching features:

Figure 7.7- very big scale

Matched points between original image and a rotated version of original image:

Figure 7.8- The left image is original image and the right image is a rotated version by 30 degrees. The correctly matched points are marked with green circles. Here a region of interest is defined in both the original image and the scaled image.

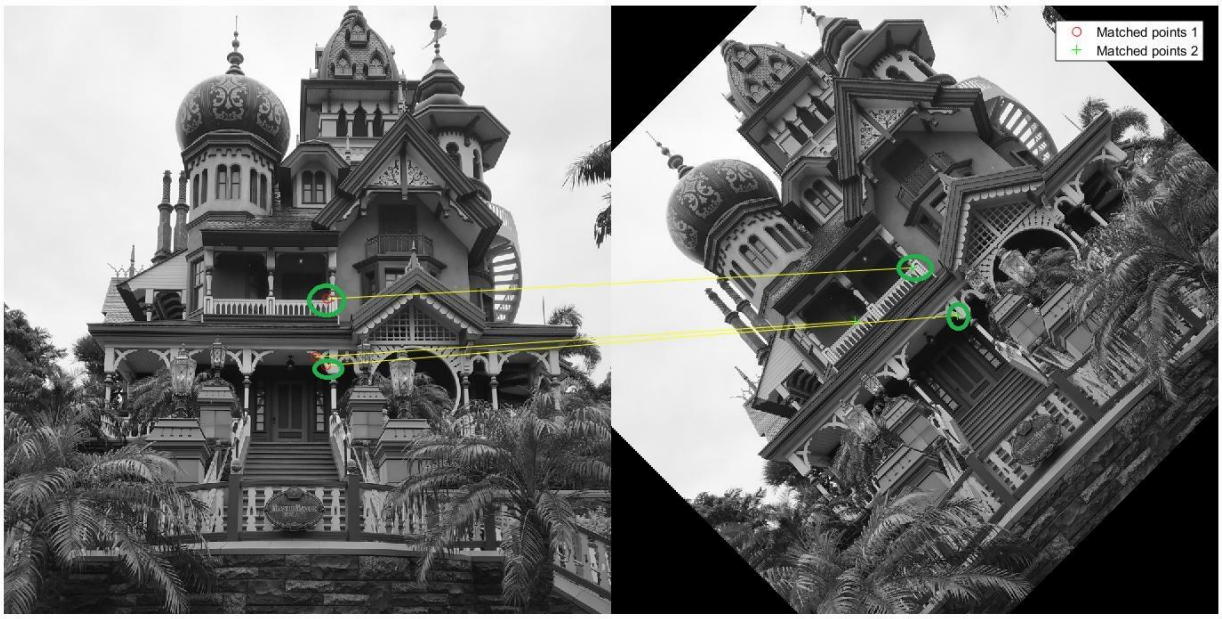

Figure 7.9- The left image is original image and the right image is a rotated version by 45 degrees. The correctly matched points are marked with green circles. Here a region of interest is defined in both the original image and the scaled image.

Matched points between original image and a rotated and scaled version of original image:

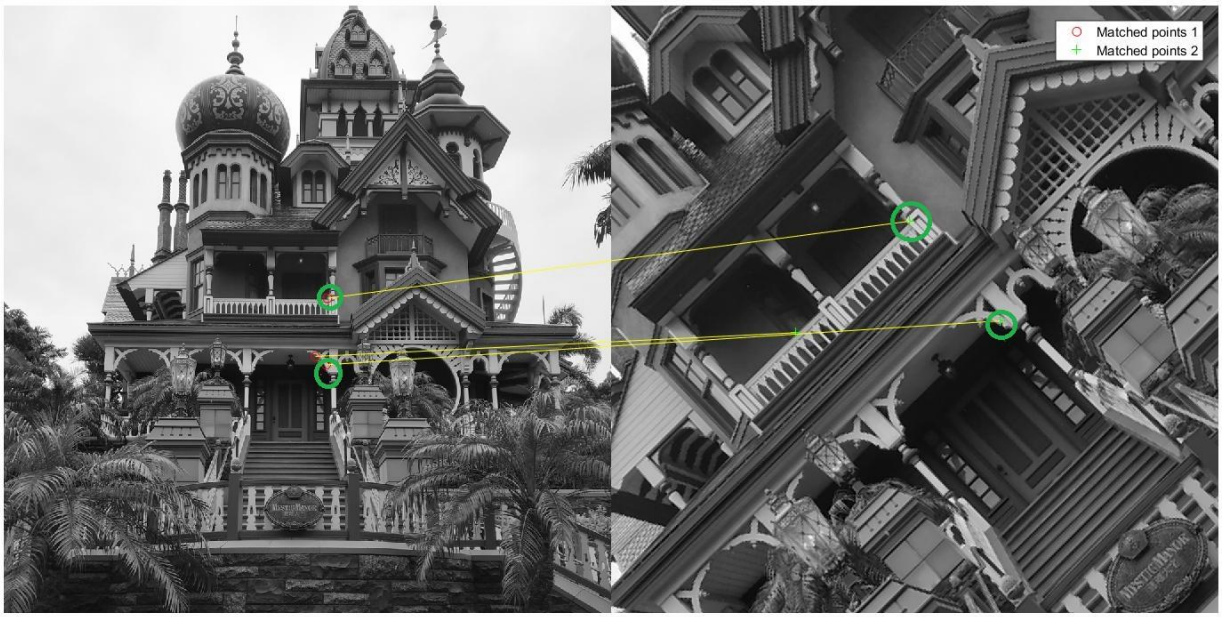

Figure 7.10- The left image is original image and the right image is a scaled and rotated version by respectively factor 2 and 45 degrees. The correctly matched points are marked with green circles. Here a region of interest is defined in both the original image and the scaled image.

Figure 7.11- The left image is original image and the right image is a scaled and rotated version by respectively factor 2 and 45 degrees. The correctly matched points are marked with green circles. Here a region of interest is defined in both the original image and the scaled image.

Sometimes the image works better on only scaled and only rotated images compared to both scaled and rotated images, but sometimes that is not the case. This might be because of the amount of noise in the picture, and also a large number of outliers.

Also, it can be seen that many structures are repeated in the image, and sometimes without defining an ROI in the transformed image, we match the point to a similar structure in transformed image, which is not locally correct with that in the original image.

Question 8:

By considering the spatial relationship between points, we can increase the efficiency of the algorithm. Meaning that we can take three sift key points next to each other, and try to find three points with the same relative distance and organization in the transformed image.

Because each point in the triplet of features has a position (we get a triangle) and a scale in the new image, the matching features must obey a similar relationship on their scale and position. If the features scales F1; F2; F3 are related by some ratio s1 : s2 : s3 in the original image, then this constrains the scales of the matching features. Such confronts allow us to prune away many possibilities.

Hence, in general, we can use clusters of key points in the original image and match these clusters with the transformed image.

For those three feature points to match three other points, all of them should cast a high vote for the new triplet, with which they are matching. This condition might pose some problems: Maybe one of the points in the triplet is an outlier, or due to image noise doesn’t cast a high vote for the selected triplet; hence we may not be able to correctly identify the best matching triplet. In this method we have more difficulty in finding the best matching triplet, as we have to calculate the matching distance for all three feature points.