RE4017 Project 2:
Harris Corner Detection and Image Alignment

Date: 29/04/20

Name: Bryan Tiernan

ID: 16169093

Table of Contents

Harris Corner Detection Overview:

For this project, the goal is to create a Harris corner detector and use it to be able to align any applicable pair of images.

The Harris Corner Detector is a corner detection operator that is commonly used in computer vision algorithms to extract corners and infer features of an image. It was first introduced by Chris Harris and Mike Stephens in 1988 upon the improvement of Moravec's corner detector.[1] The Harris Corner Detector improves on this previous iteration by taking the differential of the corner score into account with reference to direction directly, instead of using shifting patches for every 45-degree angles. This method has been proven to be much more accurate in detecting the differences between edges and corners. This method has been adopted and is still widely used in many algorithms to pre-process images for a given application.

The detector operates on a basic level by finding 'corners' in an image, which is essentially an intersection of 2 edges in an image, where an edge is a change in the brightness of the image. Using these corners, interest points in the image can be found and plotted, matched across images and aligned. This is covered in more detail in the Implementation of the project.

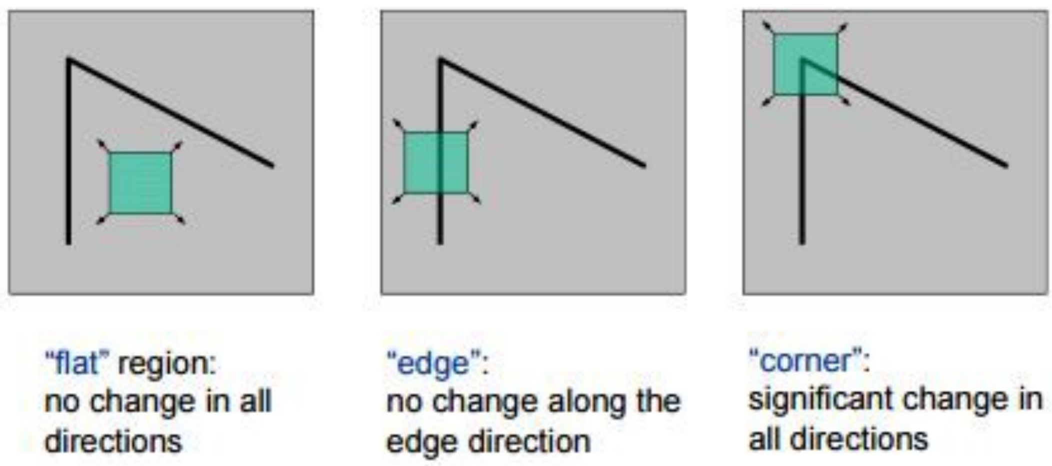

Figure 1: Corner Detection

Implementation of Project:

The code for this project can be split into several distinct steps:

Harris Matrix:

From both images, a Harris matrix is generated. A Gaussian filter is applied to reduce/smooth out noise. This operator is also used to compute the components of the Harris matrix which is used to calculate the determinant M and trace M. These matrices represent all the potential interest points or corners in the given image.

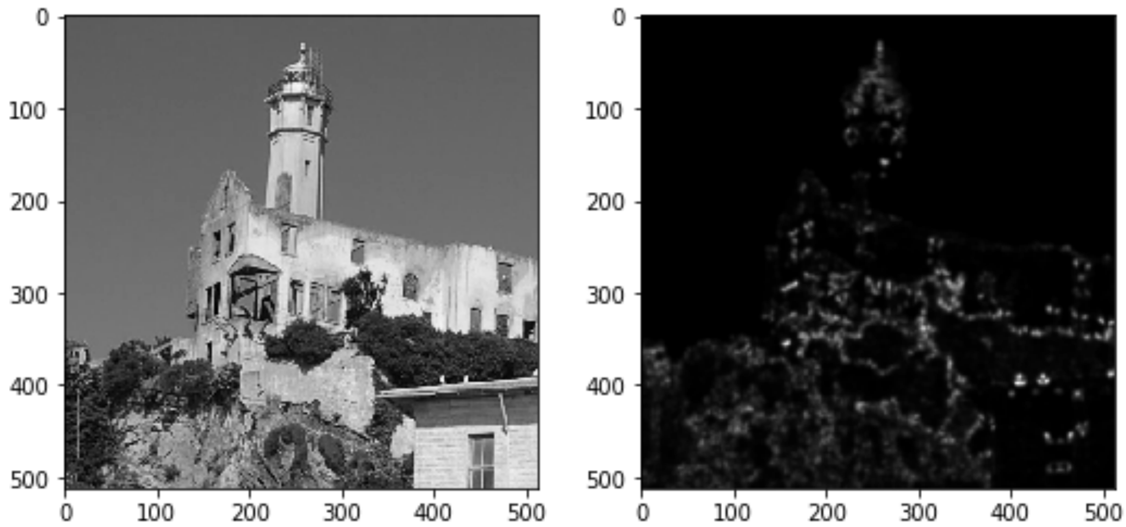

Figure 2: Original image(left) and Harris Matrix(right)

Harris Points:

The Harris points or interest points of the image are then found by utilising the Harris Matrix. Using the Harris matrices and a defined threshold (threshold =0.1 in this case), we can find the best corners in the image. The co-ordinates for these points are then stored alongside their respective response values, checked to ensure they are acceptable point locations. The best points are then selected using nonmax suppression and the coordinates are returned.

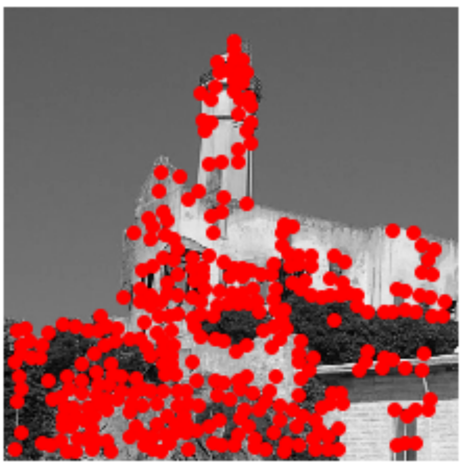

Figure 3: Image with interest/Harris points plotted

Descriptors:

Using the Harris points or interest points, we can then create a patch around these points. A patch is just an image patch or box around the points of interest. These are what we will use to compare the 2 images and ultimately match and align them.

For both images, a list of patches or descriptors is created using the interest points of each respective image and the coordinates of each point. It is at this point that we can begin to match the images.

In order to be able to compare the descriptors, one of the array of descriptors is transposed. Using the dot product of these arrays a non-thresholded response matrix can be generated. By then comparing the response matrix rows and columns to the threshold (0.95) and row maximum we have generated our matched pairs of points, as well as the thresholded matrix.

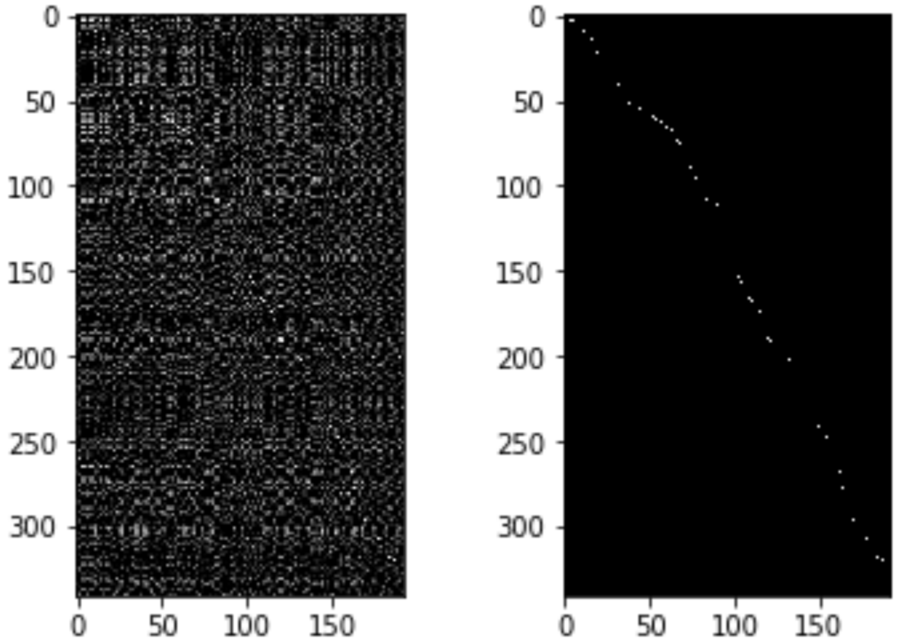

Figure 4: non-thresholded response matrix(left) and thresholded matrix(right)

From this point plotting the pairs is simply a matter of creating a concatenated image of image 1 and image 2, and plotting the line between the co-ordinates using the indices from these pairs.

Figure 5: Matches plotted between images

Figure 5 shows the matches between in the interest or Harris points between the 2 images. However, we can still see some outliers present in the images that need to be dealt with.

RANSAC:

RANSAC stands for Random Sample Consensus. Due to the fact that mapping using descriptors is inherently a noisy process, there will be some outliers as can be seen in fig.5. RANSAC is a process for dividing a dataset into inliers and outliers by randomly sampling multiple times with random choices for offsets. It will return the offset of model with the most support or largest consensus set, which in this case is ideal as we do not know the model initially ourselves.

In the RANSAC method, a list of offsets is built from the matches found between the 2 images. The offsets are the compared, with the best match, best row and best column offset for each sample being stored and compared for each of the given offsets. The best row, column offset are returned from the method in order to create the composite image, as well as the best match count.

Append images:

Having found the best row and column offset for the image, creating the complete composite image is achieved by pasting together the original images using the newly generated offsets to create the final image.

Figure 6: Composite image

Figure 6 shows the final or composite image, with both images appended to one another.

Performance & Conclusion:

As shown in the last step of the implementation, the Harris corner detector created for this project works quite well.

Figure 7: Other composite images

Figure 7 shows the other images provided to test the code on appended correctly as well. One of the questions posed was does the Harris detector work on rotated and differently scaled images ? The Harris corner detector algorithim is rotation invariant. This is because the eigenvalues or value of R for each given patch remains unchanged when rotated.

However, the Harris corner detector is non-invariant to scaling. This is because when you scale an image down, points in the image which were classified as edges can become corners, changing the results.

To the extent that I have tested it, I have found that the code written does seem to work on differently scaled images. However, the underlying non-invariant nature of the algorithim, the more downscaled the image the less reliable the matching and alignment will be, even more so with differently scaled images to the point at which it may not be able to make correct matches between descriptors.

Unfortunately, I was unable to make my code work on rotated images. This is due to the way the code is implemented, the region descriptors and points are compared and contrasted by rows and columns. However, when I attempted to change the code to reflect the invariance of the algorithim I was unable to do so.

In conclusion, I feel that aside from implementing invariance to rotation the project has been a success. For 2 suitable images, this implementation of a Harris corner detector and image aligment application will be able to generate a composite image by implementing corner detection.

While only one of the set of images was used to show the implementation of the project, running the code given alongside the project should illustrate that the code works on more than just one set of images.

References:

[1] C. Harris and M. Stephens, 'A Combined Corner and Edge Detector', in *Procedings of the Alvey Vision Conference 1988*, Manchester, 1988, pp. 23.1-23.6, doi: 10.5244/C.2.23.