Project Report

Introduction

In this project, I have Implemented the basic logic of comparing the image pairs using computer vision techniques. To achieve this, I implemented the Harris corner detection and got the corner points from both of the images; applied adaptive non-maximum suppression to get the key points; implemented SIFT descriptors to get the feature vectors for all of the key points from the two images and then match them to get the final accuracy score. This report will also touch on the algorithm implemented, configuration changes to get better results; finally, accuracy results by comparing the three different image pairs.

Implementation Details

- **Harris corner detection:** This is the first step to find the corner points by finding the gradient intensity across the image. Sobel operator was used to getting the gradient change in the x, y-direction(Ix, Iy in the algorithm) of the image pixel. We calculated gradient covariances(Ix_Ix, Iy_Iy, Ix_Iy) from the above gradient intensity values using the light Gaussian smoothing technique.
  - We know from the formulae that $\mathrm{H} = \begin{bmatrix} Ix\_Ix & Ix\_Iy \\ Ix\_Iy & Iy\_Iy \end{bmatrix}$ and $\alpha = \frac{\lambda 1\, \lambda 2}{(\lambda 1 + \lambda 2)}$

    $\lambda 1\ \lambda 2$ are eigenvalues

    $$R = det(H) - \alpha\, \mathrm{trace(H)}^2$$

    R > 0 are the corners and the local maxima point.
    R ≅ 0 are flat region and
    R < 0 are the pixels with edges in the image.
  - Configuration parameters available in my programme for this step are **kernel size(ksize), sigma values(sigmaX,sigmaY), no of maxima to return(n)**.

- **Adoptive Non-Maximal Suppression:** In this step,
  - I will make the window of size n=4 which divides the whole image by 4*4 windows.
  - For each window, I am considering the local maxima, which are significantly important w.r.t the image's global maxima. It can be achieved by using **Significance Percentage** parameter field. **E.g.** If the global maxima value is 8.0 and significance percentage is 10%, it will only consider the local maxima with value more significant than (8*10%).
  - Next step is to find the distance between the maxima. I have used the vectorisation technique to find the distance matrix, which **gave an excellent performance in the distance calculation**.
  - It was then finding the minimum of distances to a keypoint's stronger neighbours. This is achieved by sorting the distance row-wise, which gives the minimum distance to the strongest key point(minimum suppression radius).
  - The last step is to filter the minimum suppression radius by the threshold radius value and return the maxima coordinates.
  - Config parameters available here are **significance percentage, suppression radius.**

- **SIFT like Feature description:** SIFT descriptor generates a histogram of the image gradients in characterising a keypoint's appearance. The feature vector, which is the SIFT descriptor's output, details the patch's orientation data like each pixel's magnitude and angle.
  - The angle can be calculated as $\mathbf{tan^{-1}}\left(\frac{Iy}{Ix}\right)$ at each pixel
    - Ix, Iy are the intensity of gradient at x,y direction.

        - Magnitude is calculated as $\sqrt[2]{(Ix^2 + Iy^2)}$
    - Next step is to calculate the orientation's histogram for each key point where the x-axis is divided into angle bins, and y-axis has magnitude strength.
    - I have also Implemented the normalisation and clamping technique to get better results for feature matching.
    - Configuration parameter are **feature_width, kernel size,sigma.**
- **Feature matching**: In this step, I have used the technique to match the feature vector's local feature matching generated using the "nearest neighbour distance ratio test" method.
    - Here we calculate the distance between the key points of each feature vector from two images.
    - We will then calculate the ratio test between the distances and filter based on the threshold.
    - Configuration parameters are **threshold.**

Configuration Settings

Feature width setting

We can see the impact of feature width setting on the different image pairs from the below results. We can recognised from the below table that if the image pixels are enlarged and need a larger context, we should keep the **higher feature width** for that image to get better results.

Episcopal Gaudi, **Accuracy = 49%(Improved)**		Episcopal Gaudi, Accuracy = 5%	
Image 1, feature width = 64	Image 2, feature width = 32	Image 1, feature width = 16	Image 2, feature width = 16
Notre Dame, **Accuracy = 95% (Improved)**		Notre Dame, Accuracy = 84%	
Image 1, feature width = 24	Image 2, feature width = 16	Image 1, feature width = 16	Image 2, feature width = 16
Mount Rushmore, **Accuracy = 97% (Improved)**		Mount Rushmore, Accuracy = 92%	
Image 1, feature width = 32	Image 2, feature width = 32	Image 1, feature width = 16	Image 2, feature width = 16

Normalised patch and SIFT feature description.

When changed to SIFT like feature description from the normalised patch feature description, I got great results. This is because of the SIFT's ability to provide more details of angle and magnitude around the key point than the normalised patch feature. Below are the results:

After SIFT Descriptor Change	With Normalised patch feature description
Episcopal Gaudi, **Accuracy = 15%(Improved)**	Episcopal Gaudi, **Accuracy = 2%**
Notre Dame, **Accuracy = 95% (Improved)**	Notre Dame, **Accuracy = 69%**
Mount Rushmore, **Accuracy = 94% (Improved)**	Mount Rushmore, **Accuracy = 34%**

Sigma and kernel size settings

We can see from the below results for different combinations of the sigma values and impact on the accuracy. We can say that changing the sigma value to a certain extent produced excellent results because it can improve the keypoints maxima intensity and get better accuracy.

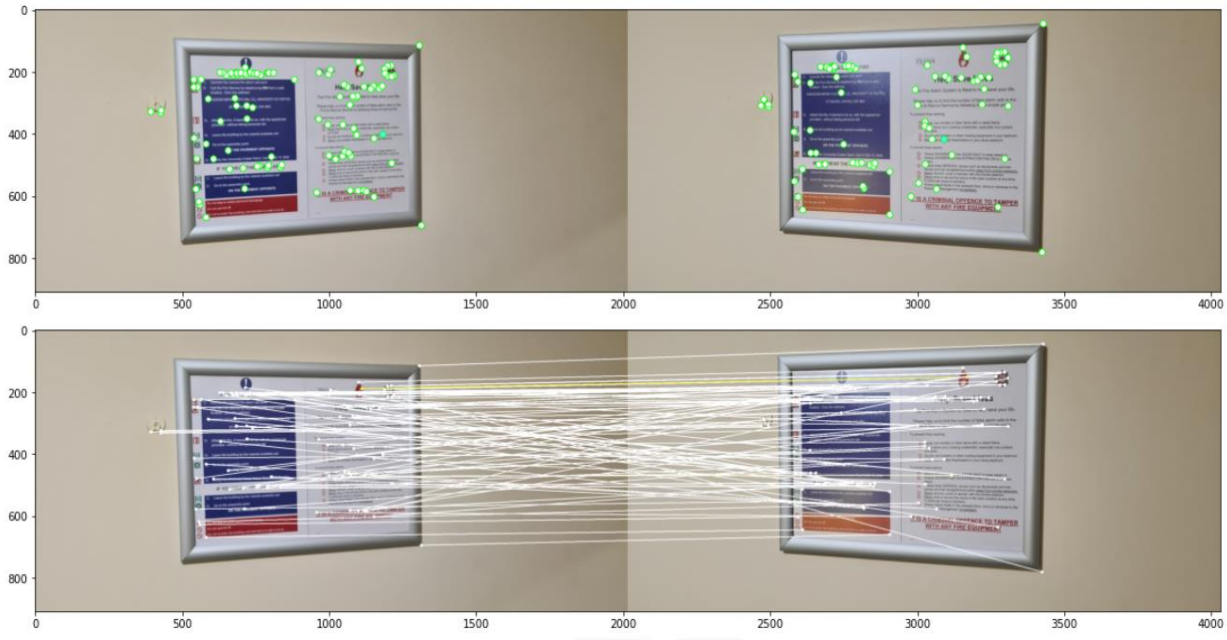
After Sigma Value Change		Before Sigma Value Change	
Episcopal Gaudi, **Accuracy = 49%(Improved)**		Episcopal Gaudi, **Accuracy = 1%**	
Image 1, sigma=12	Image 2, sigma=12	Image 1, sigma=3	Image 2, sigma=3
Notre Dame, **Accuracy = 95% (Improved)**		Notre Dame, **Accuracy = 47%**	
Image 1, sigma=9	Image 2, sigma=9	Image 1, sigma=3	Image 2, sigma=3
Mount Rushmore, **Accuracy = 97% (Improved)**		Mount Rushmore, **Accuracy = 49%**	
Image 1, sigma=9	Image 2, sigma=9	Image 1, sigma=3	Image 2, sigma=3

Scale settings

By changing the scale, we can get the good results; I got exceptionally well results for the Mount Rushmore, Accuracy = 94% (Improved) from 34%. This is because of the reduction in the difference between the pixel intensity between the images after the rescaling.

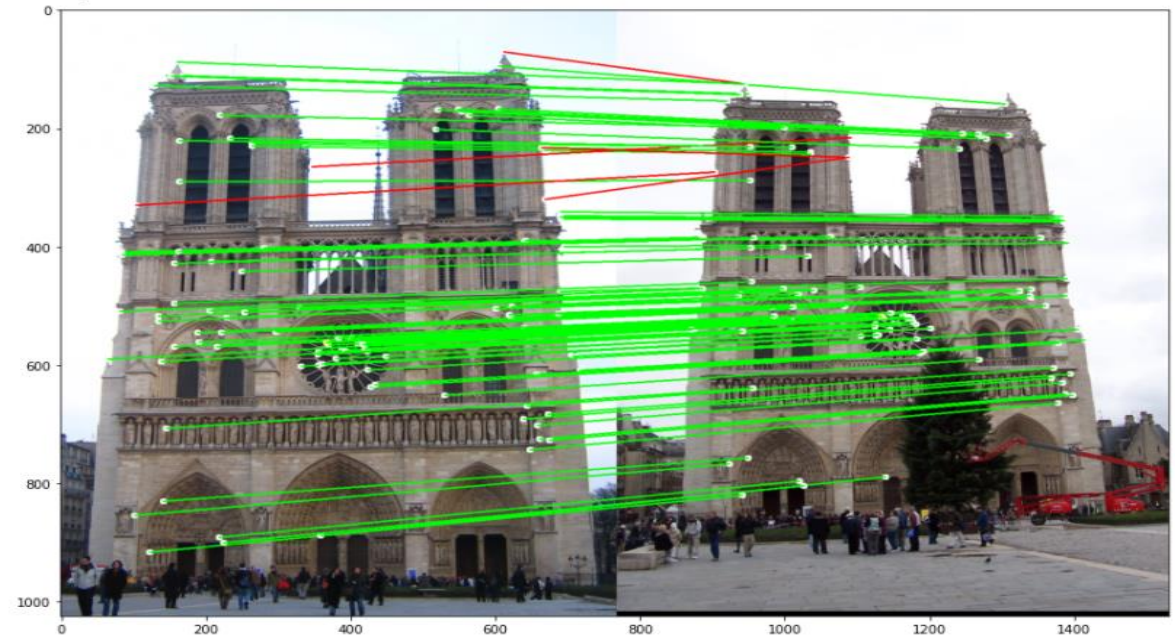
When Scale_factor = 0.5	Scale_factor = 1
Episcopal Gaudi, Accuracy = 46%	Episcopal Gaudi, **Accuracy = 49%(Improved)**
Notre Dame, **Accuracy = 95% (Improved)**	Notre Dame, **Accuracy = 94%**
Mount Rushmore, **Accuracy = 94% (Improved)**	Mount Rushmore, **Accuracy = 34%**

Special additional Test(Custom Image): I also conducted to see whether my test can find the key point, get feature description using my SIFT descriptor and match the custom image provided by me. It did well to identify 755 matches from 963 corners and found the 100/100 required matches. This is because the image was taken at different angles and with different light intensity. Since SIFT descriptors are invariant to angle, it could match the key points from the different images. Attaching the sample picture for reference.

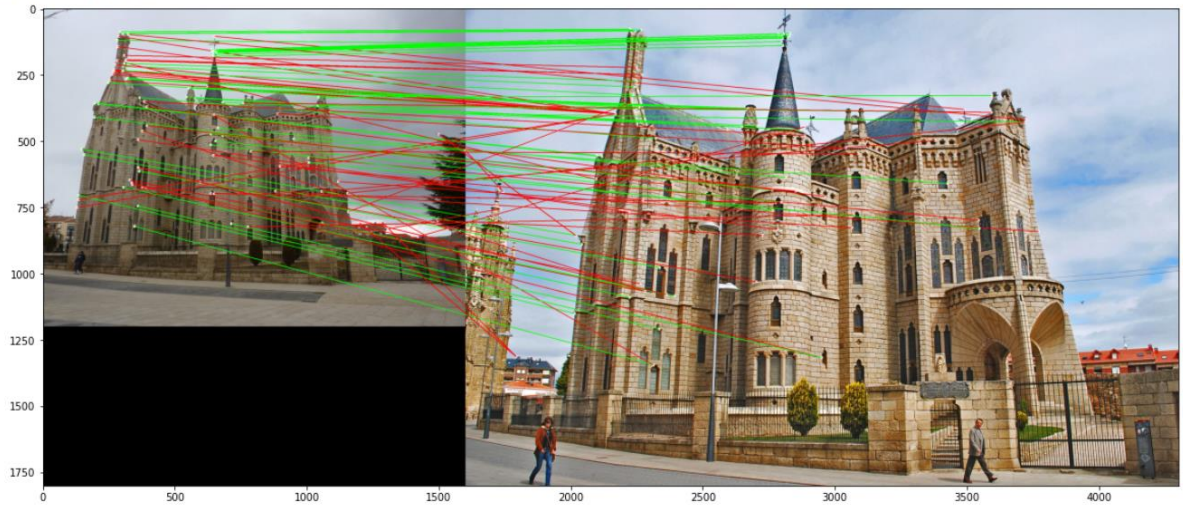


Top performing Results:

```
You found 100/100 required matches
Accuracy = 0.950000
```




```
You found 100/100 required matches
Accuracy = 0.490000
```




```
You found 100/100 required matches
Accuracy = 0.970000
```