DETECTING FEATURES VIA MODIFIED HARRIS CORNERS AND MATCHING THEM THROUGH SIFT

A PREPRINT

Jimut Bahan Pal*
Department of Computer Science
Ramakrishna Mission Vivekananda Educational and Research Institute
Howrah 711202
jimutbahanpal@yahoo.com

Tamal Maharaj
Department of Computer Science
Ramakrishna Mission Vivekananda Educational and Research Institute
Howrah 711202

March 18, 2020

ABSTRACT

Interpreting images spatially is a daunting task which is achieved by detecting corners and features. The most important task of detecting features is achieved by Harris Corner Algorithm. The algorithm is not robust to different scale of the same image. The algorithm may detect corner but when the image is zoomed in, the corner may appear as ridges. We use the corners detected from Harris Corner algorithm and treat these as key points to pass into Scale Invariant Feature Transform (SIFT) algorithm. The SIFT algorithm extracts descriptor vector of dimension 128 X 1 from these corners and can be used to find similarity between different images. This process is quite robust to noise, intensity, scale and occlusion and is used for matching images from a database of descriptors. We have investigated both the algorithms in this paper and made a modified version of Harris Corner algorithm by performing different kind of thresholding, both of them gave a little different result.

Keywords Harris Corners · feature detection · image matching · key points detector · auto correlation · SIFT

1 Introduction

Feature tracking algorithms are used for 3D interpretation of images [1]. Features are good when they are invariant to photometric transformations and covariant to geometric transformations. Invariance in photometric transformation occurs when the image is transformed and the corner locations don't change. Covariance in geometric transformation occurs when we have two transformed version of the same image, and same features gets detected in both the images. Harris corner uses the local auto-correlation function which happens to perform well on natural imagery. These algorithms are useful for understanding the unconstrained 3D world. Tracking algorithms needs features which are distinct and are not continuous like texture, i.e., number of features (f) $<<$ number of Image pixels (I). We need richer information that are available from corners which can help us to connect different images of different orientations.

2 Finding Good Features

In the case of unconstrained natural imagery, it shall contain both curve edges and textures of various scales. If we represent these edges of straight-line segments as features, these will be inappropriate since these may change in

*Jimut Bahan Pal is an alumni of St. Xavier's College. He is currently pursuing M.Sc. in RKMVERI, Belur.

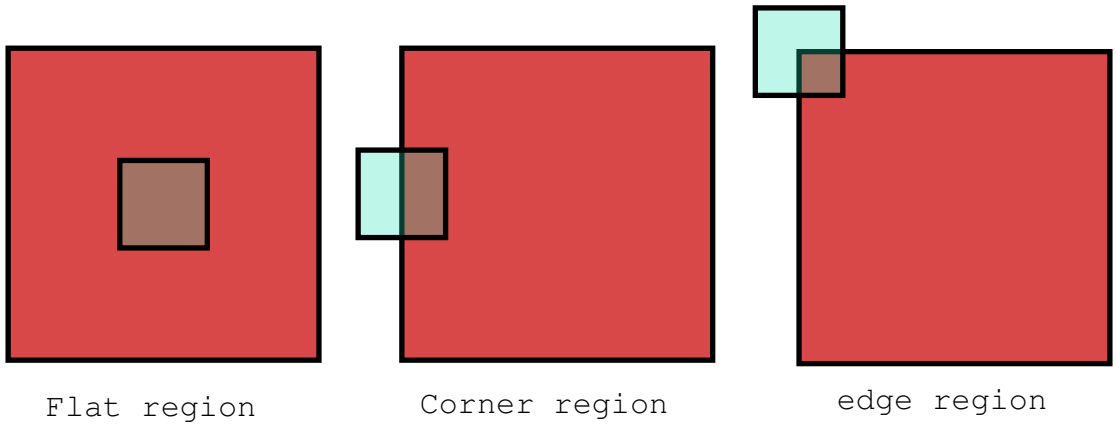
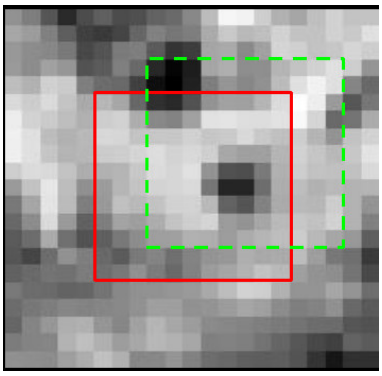
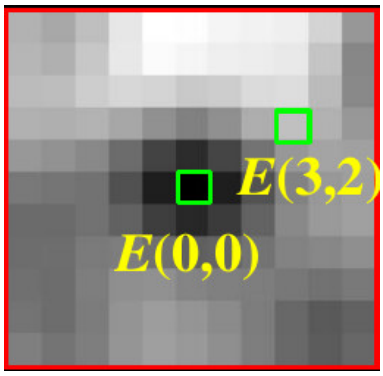


Figure 1: The three cases of Moravec's Window. (i) When the window is in a flat region, (ii) When the window straddles along an edge, and (iii) When the window is in the corner.

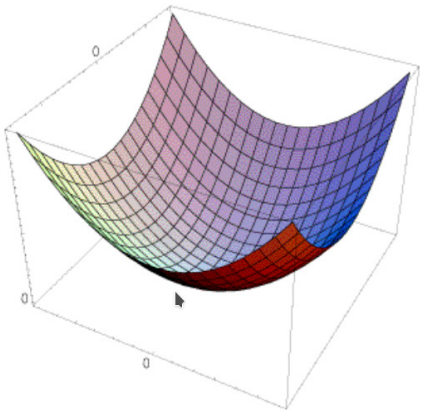


(a) The image I.

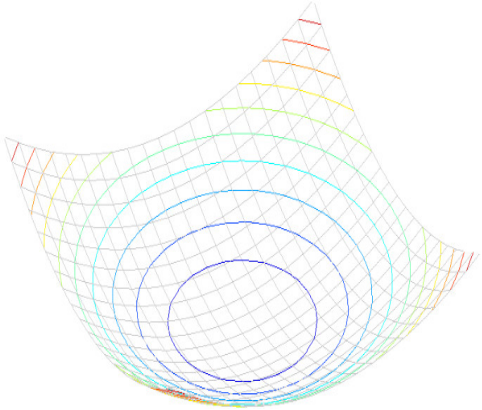


(b) The calculated edge matrix E via the window function w.

Figure 2: The figure containing the image and the corresponding edge matrix calculated via a window function. Source: Slides of Aaron Bobick.



(a) The surface of $E(u, v)$ can be locally approximated by a quadratic form.



(b) The slice of the surface of $E(u, v)$ can be considered as an ellipse.

Figure 3: A figure showing $E(u, v)$ can be represented as ellipse. Source: Slides of Aaron Bobick.

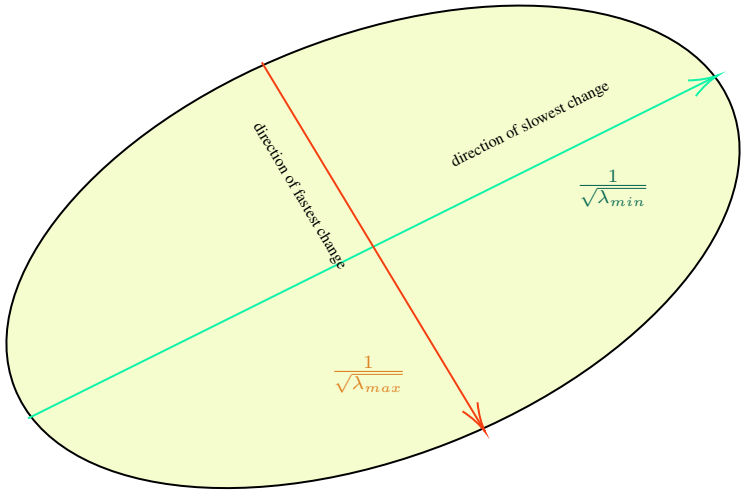
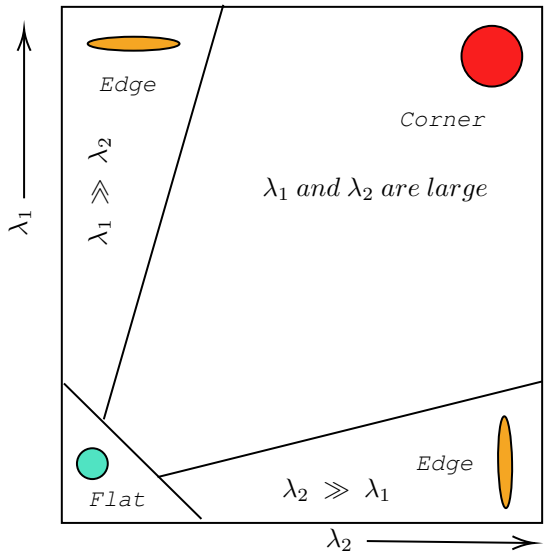
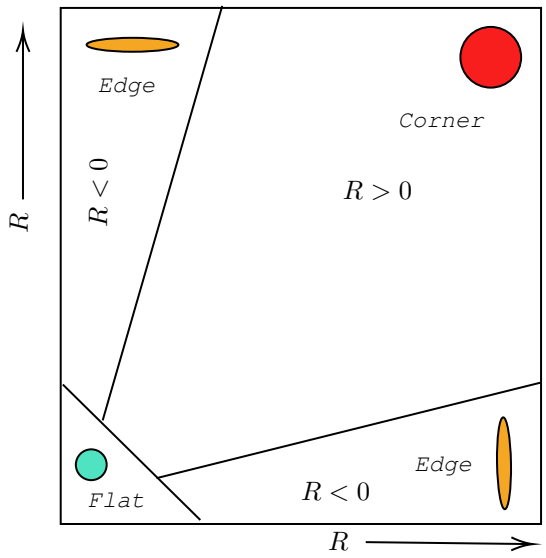


Figure 4: The direction of fastest and slowest change of eigen values λ_{max} and λ_{min}.

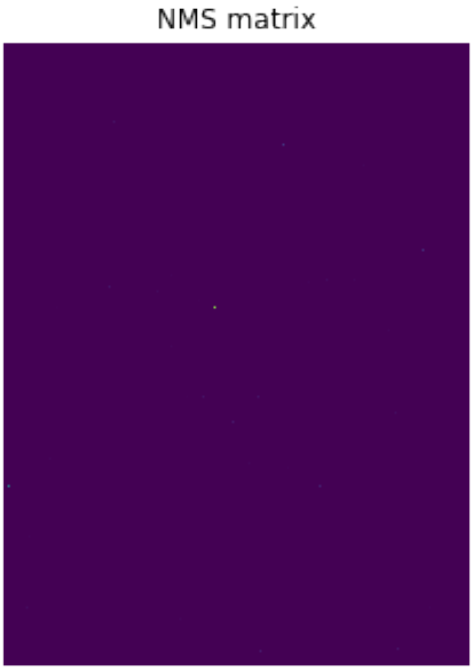


(a) The changes of eigen values (λ_1 *and* λ_2) for Harris Corner detector in the auto-correlation principal curvature space.

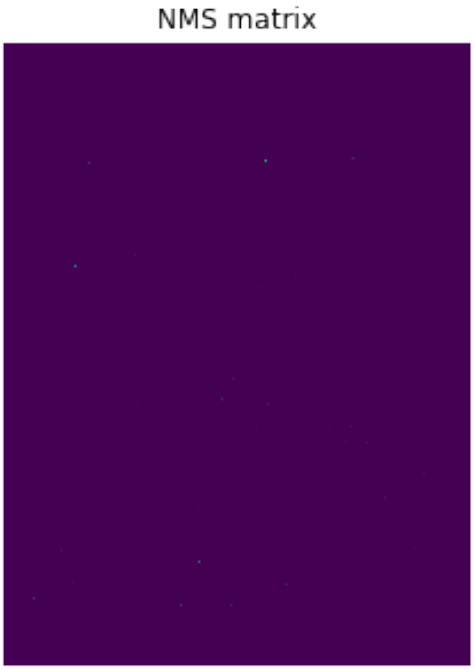


(b) The changes of response for Harris Corner detector in the auto-correlation principal curvature space.

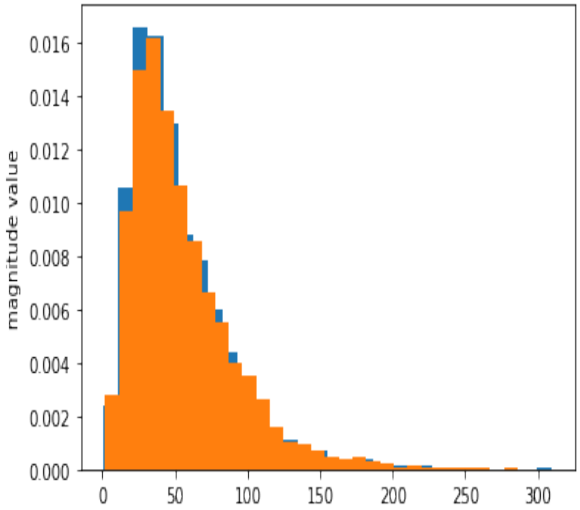
Figure 5: The change in Response (R) and Eigenvalues (λ_1 *and* λ_2). Source: Slides of Aaron Bobick.



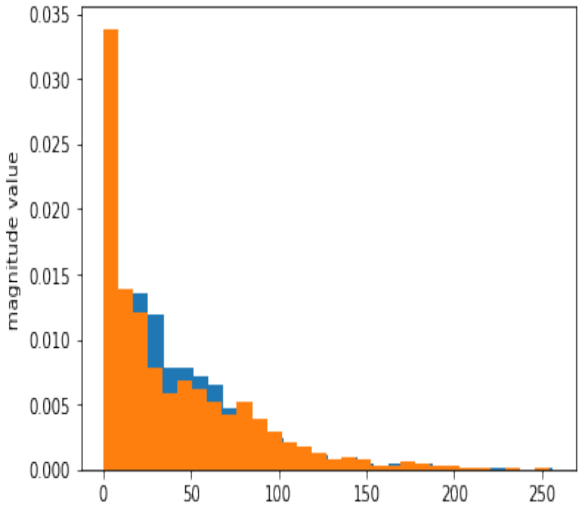
(a) The $Response\_Mat$ of image 1 of Notre Dame by applying algorithm 3



(b) The $Response\_Mat$ of image 2 of Notre Dame by applying algorithm 3

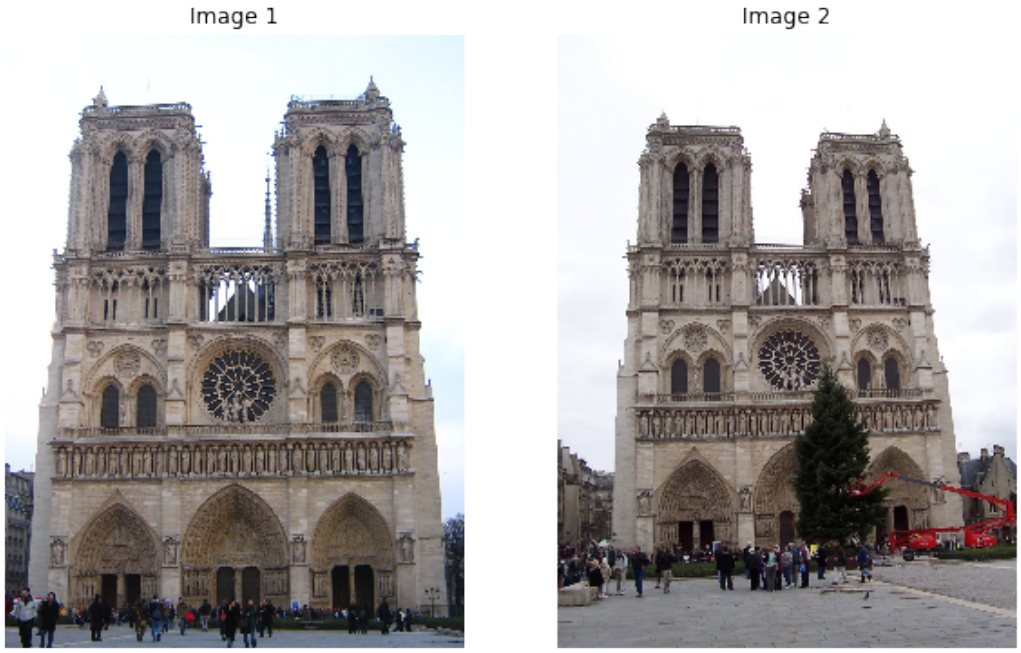


(c) The distribution of Intensity of $Response\_Mat$ of both the images of Notre Dame by applying algorithm 3

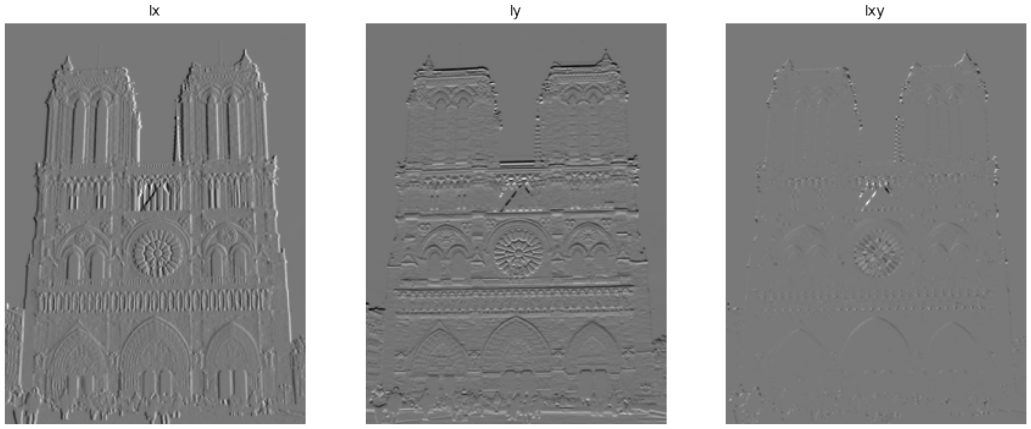


(d) The distribution of Intensity of $Response\_Mat$ of both the images of Notre Dame by applying algorithm 2

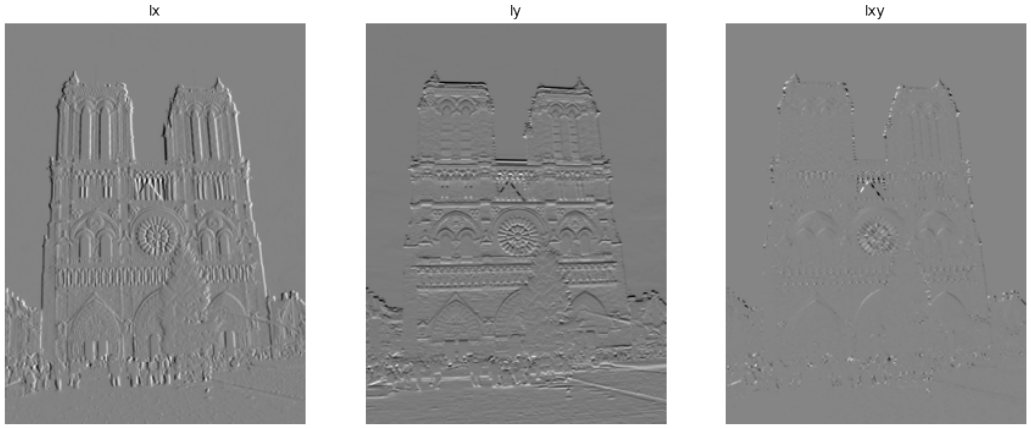
Figure 6: Image of Response matrix's intensity distribution for Notre Dame.



(a) The original image of Notre Dame.



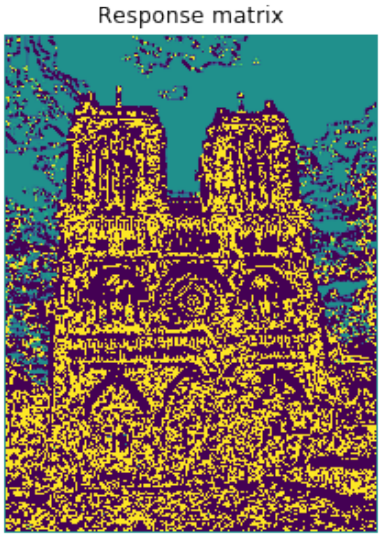
(b) Gradients of the first Image, $\frac{\partial I_1}{\partial x}$ (left), $\frac{\partial I_1}{\partial y}$ (middle) and $\frac{\partial I_1}{\partial x}\frac{\partial I_1}{\partial y}$ (right) using Sobel filter.



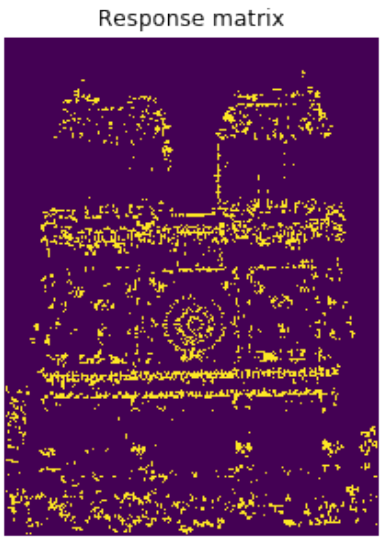
(c) Gradients of the second Image, $\frac{\partial I_2}{\partial x}$ (left), $\frac{\partial I_2}{\partial y}$ (middle) and $\frac{\partial I_2}{\partial x}\frac{\partial I_2}{\partial y}$ (right) using Sobel filter.

Figure 7: Image of Notre Dame and its Sobel derivatives.

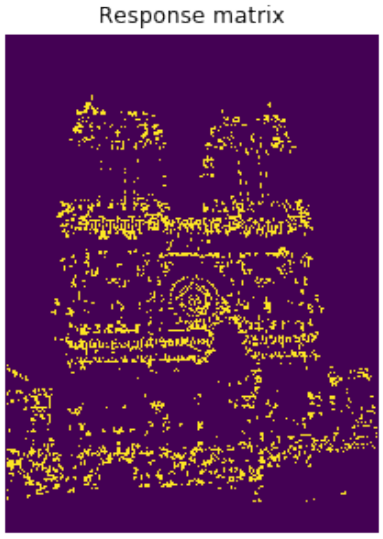
(a) The R (response matrix) calculated for first image of Notre Dame.



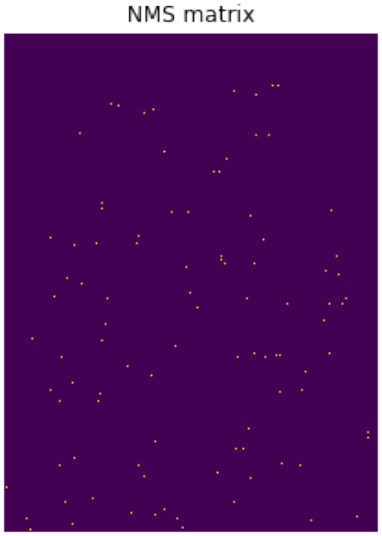
(b) The R (response matrix) calculated for second image of Notre Dame.



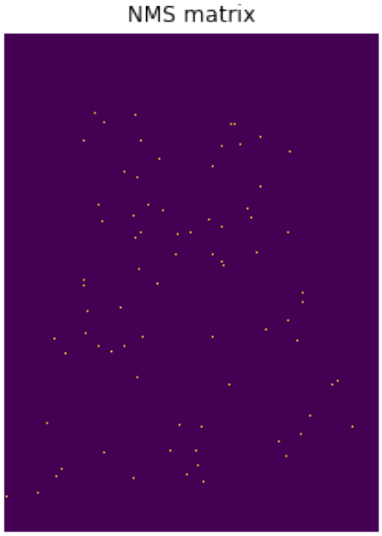
(c) After thresholding the R of first Image



(d) After thresholding the R of second Image



(e) After performing Non-Maximum-Suppression on the thresholded R of first Image



(f) After performing Non-Maximum-Suppression on the thresholded R of second Image

Figure 8: Figure showing the R value, thresholded value, and the corner detected for the images of Notre Dame.

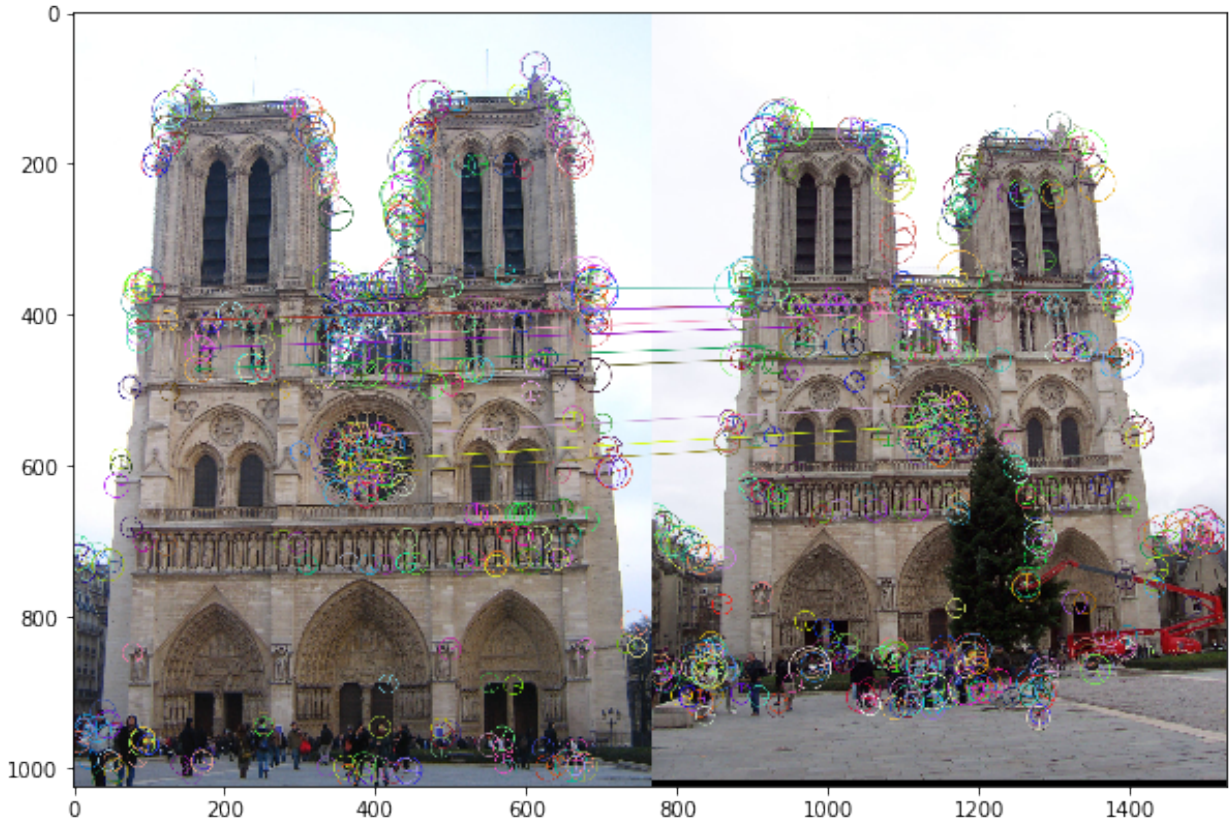


Figure 9: The feature matches of two images of Notre Dame by performing SIFT algorithm using Version-1 of Non-maximum-Suppression.

different orientation of the same image, and hence will be untraceable. The textures will vary from image to image and hence we can't take it as good features. Circular arcs cannot be used to detect corners [2]. If the edges are connected from image to image, we can get a wireframe 3D model of the region as 3D surfaces. The main aim is to find points in an image that can be found in other images, found precisely and found reliably. There shall be a fundamental matrix to recover geometry.

Good features shall have the following properties:

1. **Repeatability and Precision** - The same feature can be found in other images of the same type despite geometric and photometric transformations.
2. **Saliency and Match-ability** - Each feature has a distinctive description.
3. **Compactness and Efficiency** - There shall be very few features as compared to the number of Image pixels.
4. **Locality** - A feature occupies a small area of the image and will be robust to occlusion and clutter.

3 Use of Moravec Corner Detector

Moravec Corner detector uses a local window in the image and determines the average changes of image intensity that result from shifting the window [3] by small amount in various directions. There are three cases that are needed to be considered as shown in Figure 1.

1. If the window patch is flat and of constant intensity, then all shifts will result only in a small change.
2. If the window is lying in an edge then when we shift along the window, it will result in a small change, but when we move the window perpendicular to the edge, it will result in a large change.
3. When the window is in a corner, then a little shift in any direction will result in a large change.

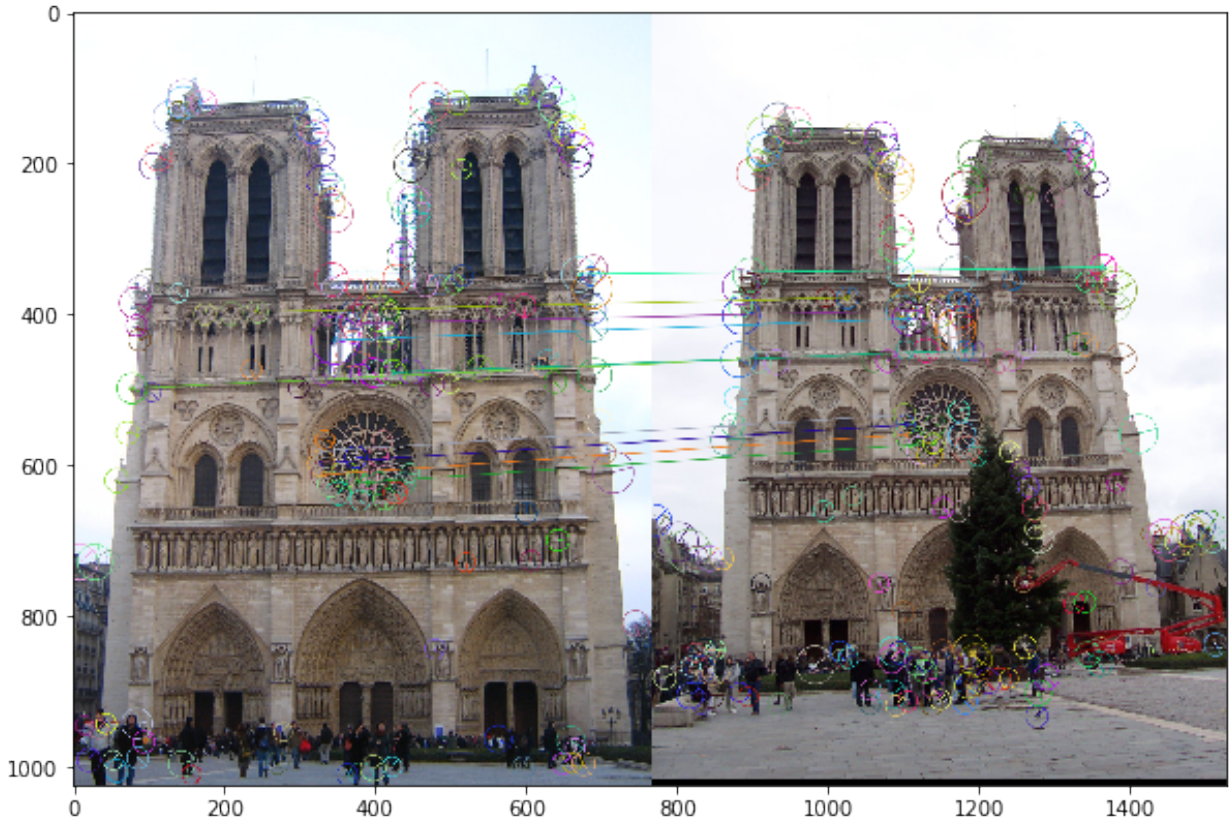
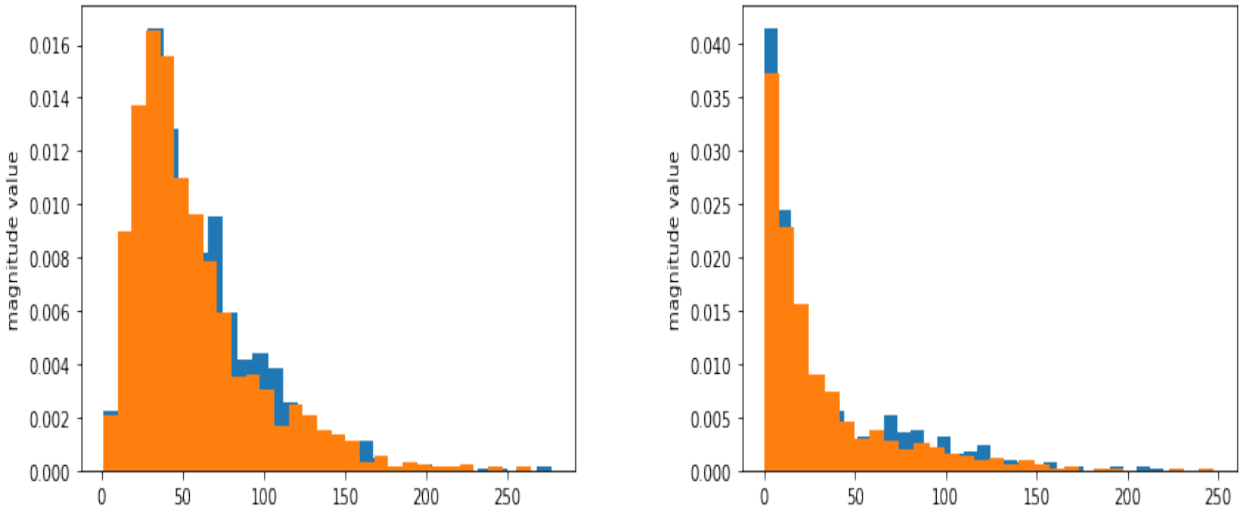


Figure 10: The feature matches of two images of Notre Dame by performing SIFT algorithm using Version-2 of Non-maximum-Suppression.

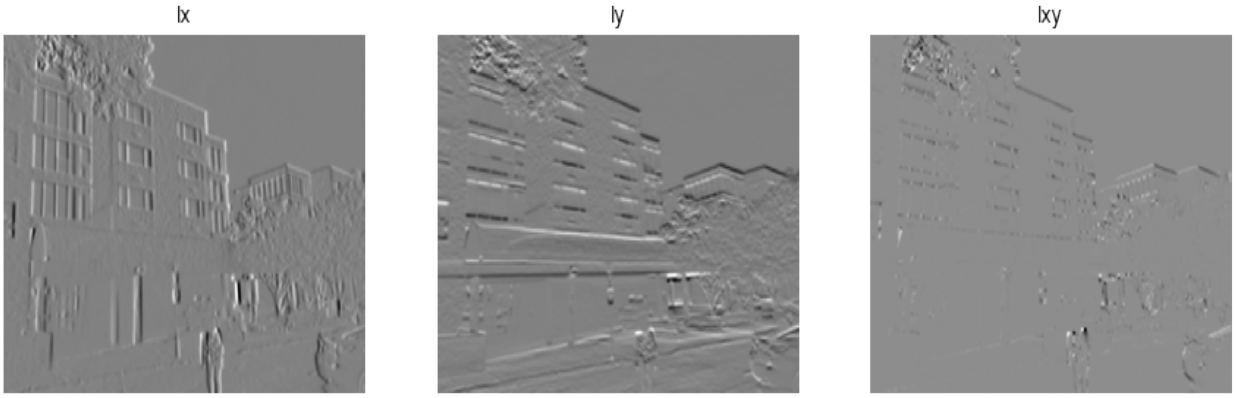


(a) The distribution of Intensity of $Response\_Mat$ of both the images of Wood House by applying algorithm 3

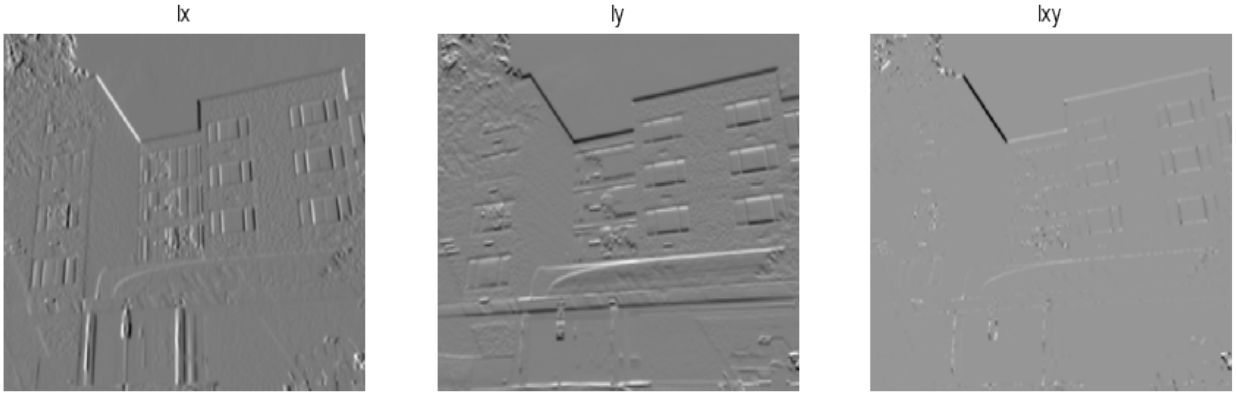
(b) The distribution of Intensity of $Response\_Mat$ of both the images of Wood House by applying algorithm 2

Figure 11: Image of Response matrix's intensity distribution for Wood House.

(a) The original image of Wood House.

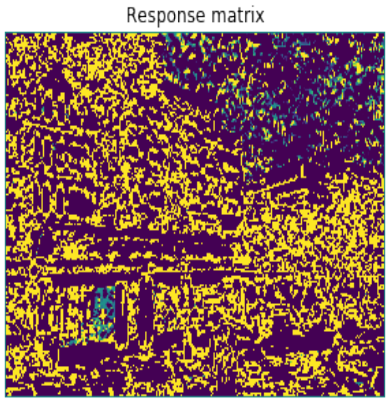


(b) Gradients of the first Image, $\frac{\partial I_1}{\partial x}$ (left), $\frac{\partial I_1}{\partial y}$ (middle) and $\frac{\partial I_1}{\partial x}\frac{\partial I_1}{\partial y}$ (right) using Sobel filter.

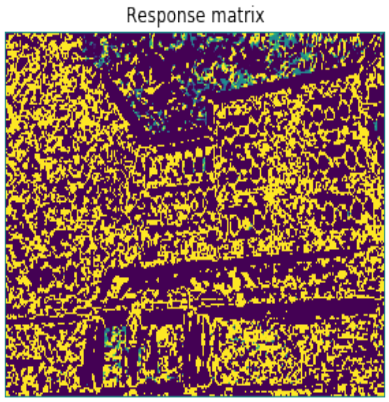


(c) Gradients of the second Image, $\frac{\partial I_2}{\partial x}$ (left), $\frac{\partial I_2}{\partial y}$ (middle) and $\frac{\partial I_2}{\partial x}\frac{\partial I_2}{\partial y}$ (right) using Sobel filter.

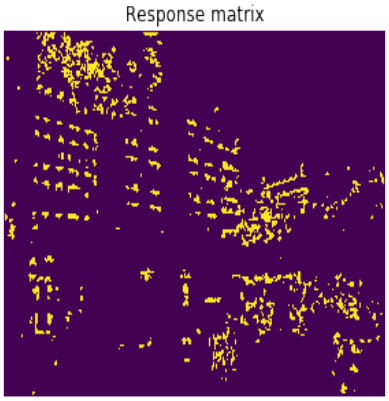
Figure 12: Image of Wood House and its Sobel derivatives.



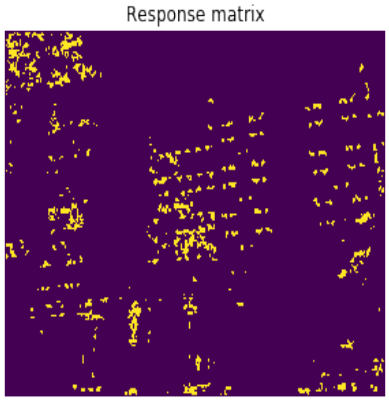
(a) The R (response matrix) calculated for first image of Wood House.



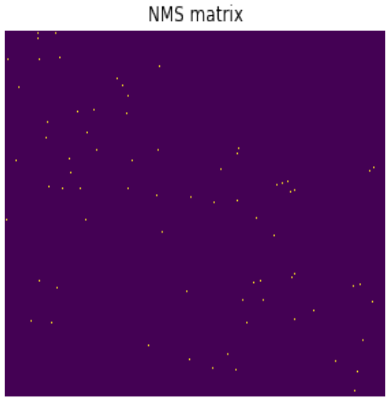
(b) The R (response matrix) calculated for second image of Wood House.



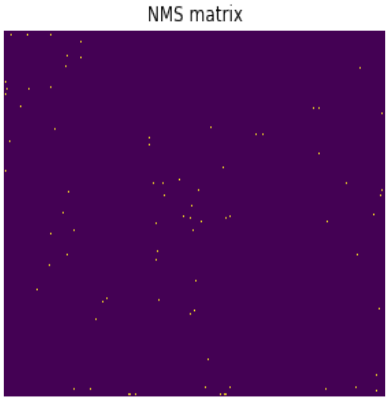
(c) After thresholding the R of first Image



(d) After thresholding the R of second Image



(e) After performing Non-Maximum-Suppression on the thresholded R of first Image



(f) After performing Non-Maximum-Suppression on the thresholded R of second Image

Figure 13: Figure showing the R value, thresholded value, and the corner detected for the images of Wood House.

4 Auto-Correlation Detector

The Edge E can be written in terms of the window function w and the Image I as:

$$E(u,v) = \sum_{x,y} w(x,y)[I(x+u, y+v) - I(x,y)]^2$$

Here, the window can be box or gaussian window. We want to find out how this function behaves for small shifts. This is very slow to compute naively. The complexity will be **O(window_width^2*shift_range^2*image_width^2)**. Suppose the window's width is 11, the shift range is 11, and the image width is 600 pixel, then this will give $O(11^2 * 11^2 * 600^2)$ = 5.2 billion, which means there will be 14.6 thousand computations per pixel in our image. This is ridiculously slow even for small sized images. The window function on an image is shown in Figure 2.

We need to approximate this function, and we can use Taylor Series Expansion for the same. The Taylor Series function can be written as:

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n$$

Which can also be written as:

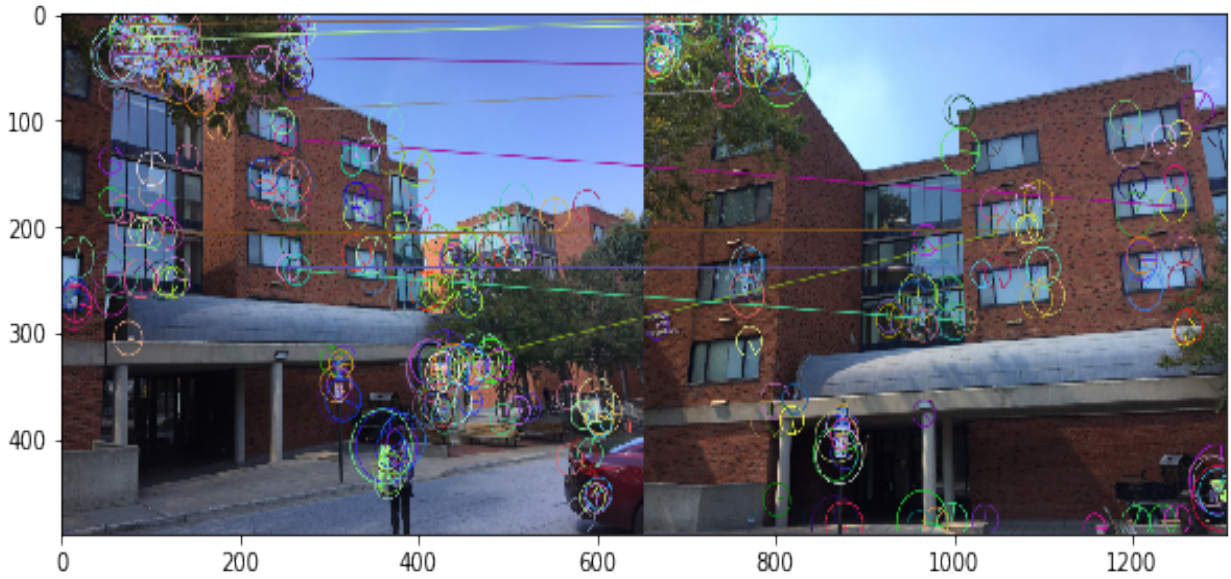


Figure 14: The feature matches of two images of Wood House by performing SIFT algorithm using Version-1 of Non-maximum-Suppression.

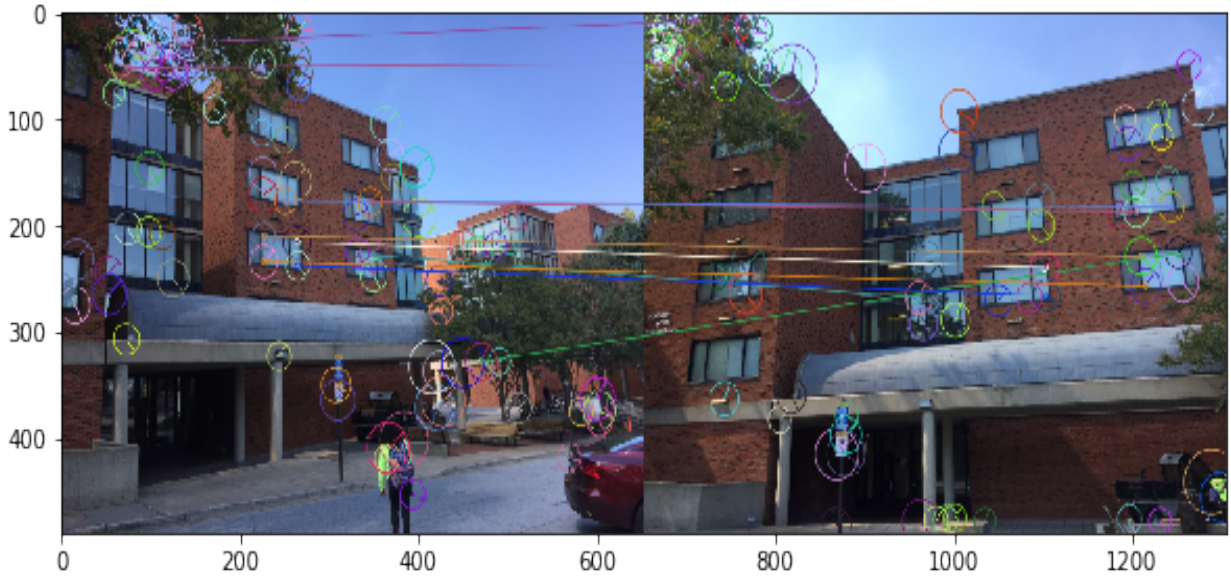
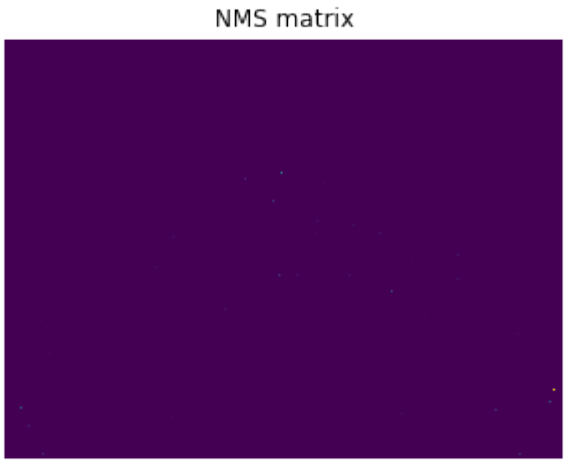
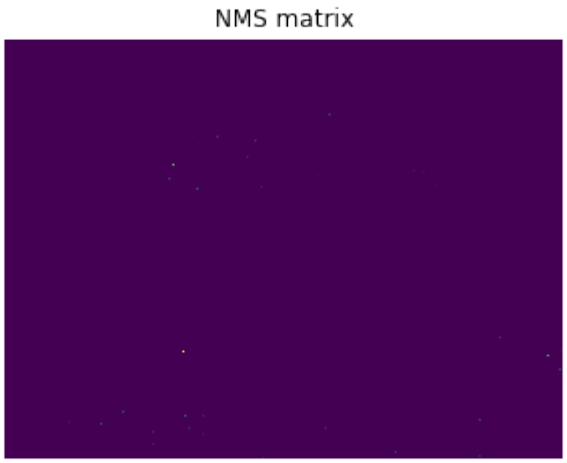


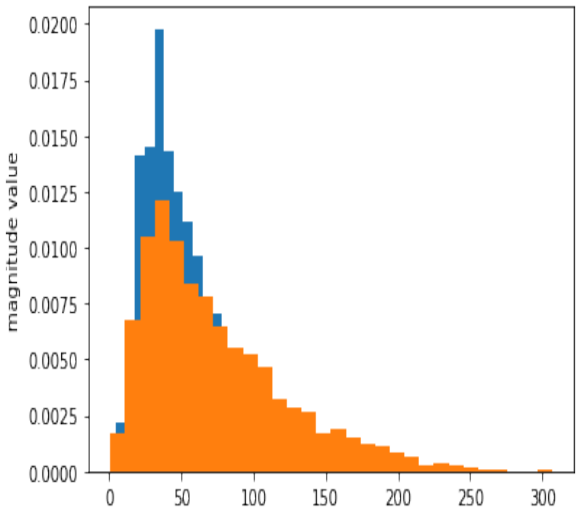
Figure 15: The feature matches of two images of Wood House by performing SIFT algorithm using Version-2 of Non-maximum-Suppression.



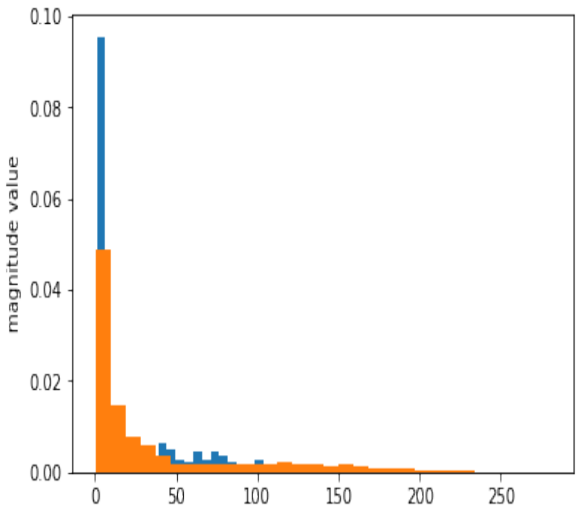
(a) The $Response\_Mat$ of image 1 of Pantheon by applying algorithm 3



(b) The $Response\_Mat$ of image 2 of Pantheon by applying algorithm 3



(c) The distribution of Intensity of $Response\_Mat$ of both the images of Pantheon by applying algorithm 3



(d) The distribution of Intensity of $Response\_Mat$ of both the images ofPantheon by applying algorithm 2

Figure 16: Image of Pantheon and its Sobel derivatives.

$$f(x) = f(a) + \frac{f^{'}(a)}{1!}(x-a) + \frac{f^{''}(a)}{2!}(x-a)^2 + \frac{f^{'''}(a)}{3!}(x-a)^3 + ...$$

So the Edge matrix E for every pixel (u, v) in image I can be approximated as:

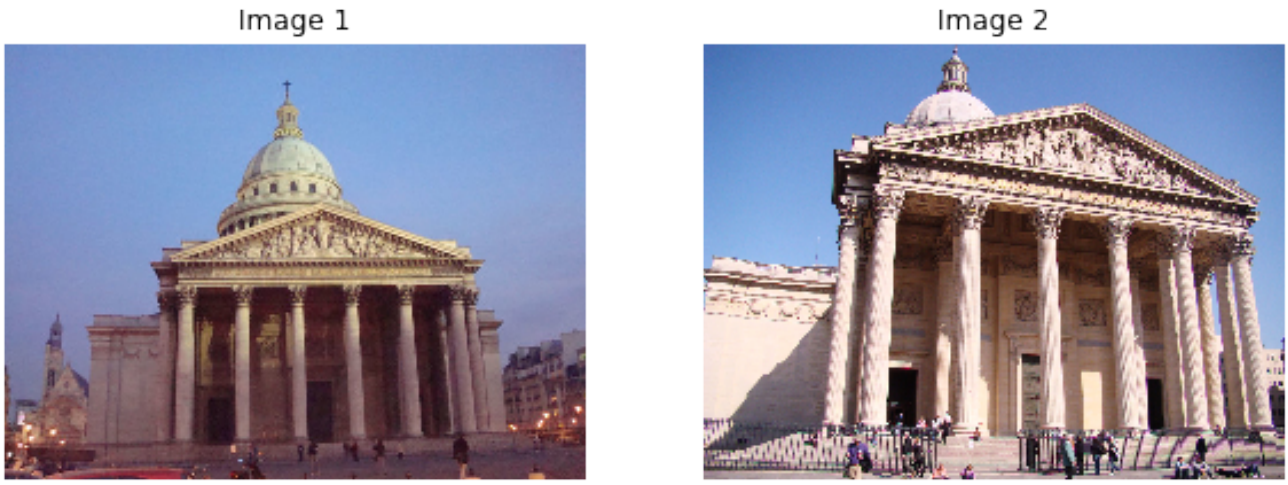
$$E(u,v) \approx E(0,0) + [u\ v] \begin{bmatrix} E_u(0,0) \\ E_v(0,0) \end{bmatrix} + [u\ v] \begin{bmatrix} E_{uu}(0,0)\ E_{uv}(0,0) \\ E_{uv}(0,0)\ E_{vv}(0,0) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

To approximate this term, we need to calculate $E_u(0,0)$, $E_v(0,0)$, $E_{uu}(0,0)$, $E_{uv}(0,0)$ and $E_{vv}(0,0)$. Since $E_{uv}(0,0)$ will be same as $E_{vu}(0,0)$. So let us calculate these terms,

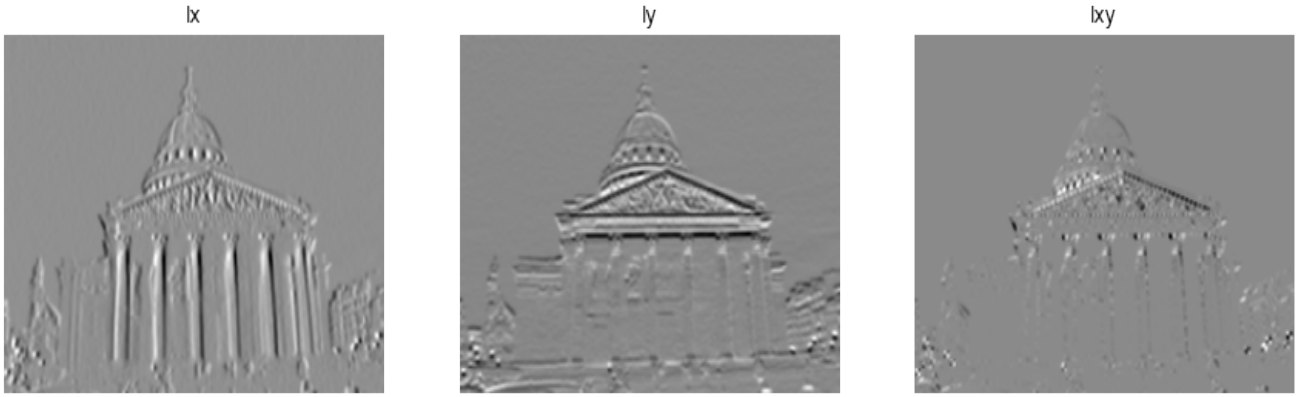
$E_u(u,v) = \sum_{x,y} 2w(x,y)[I(x+u,y+v) - I(x,y)]I_x(x+u,y+v)$

$E_v(u,v) = \sum_{x,y} 2w(x,y)[I(x+u,y+v) - I(x,y)]I_y(x+u,y+v)$

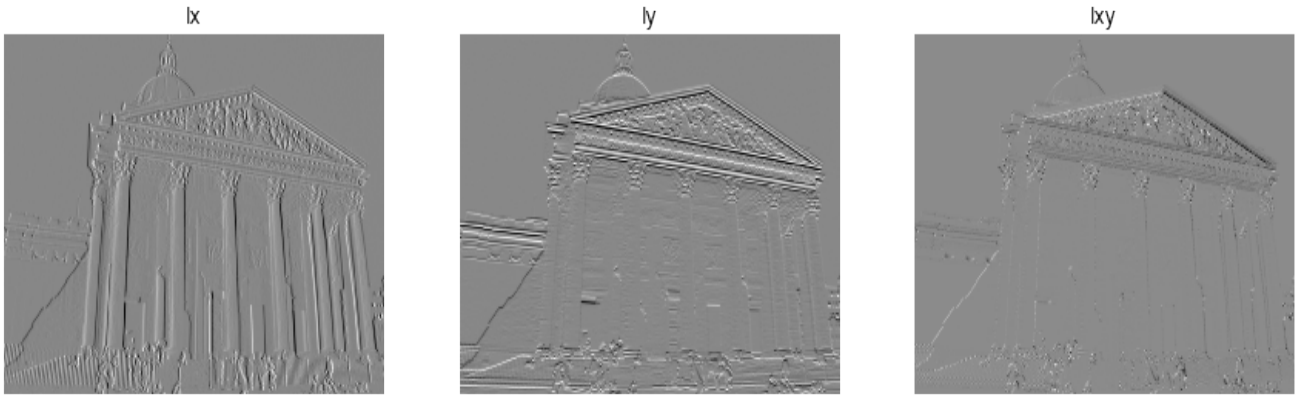
$E_{uu}(u,v) = \sum_{x,y} 2w(x,y)[I_x(x+u,y+v)]^2 + \sum_{x,y} 2w(x,y)[I(x+u,y+v) - I(x,y)]I_{xx}(x+u,y+v)$



(a) The original image of Pantheon.

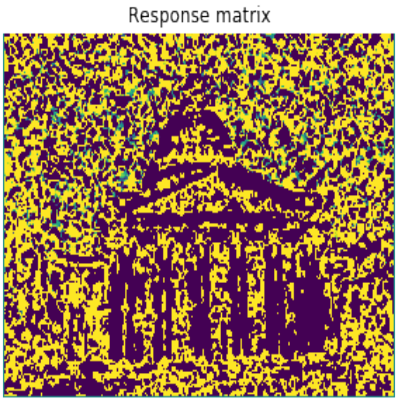


(b) Gradients of the first Image, $\frac{\partial I_1}{\partial x}$ (left), $\frac{\partial I_1}{\partial y}$ (middle) and $\frac{\partial I_1}{\partial x}\frac{\partial I_1}{\partial y}$ (right) using Sobel filter.

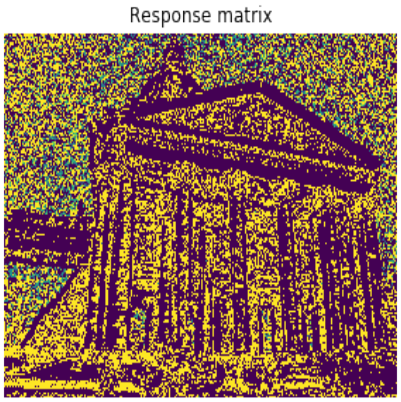


(c) Gradients of the second Image, $\frac{\partial I_2}{\partial x}$ (left), $\frac{\partial I_2}{\partial y}$ (middle) and $\frac{\partial I_2}{\partial x}\frac{\partial I_2}{\partial y}$ (right) using Sobel filter.

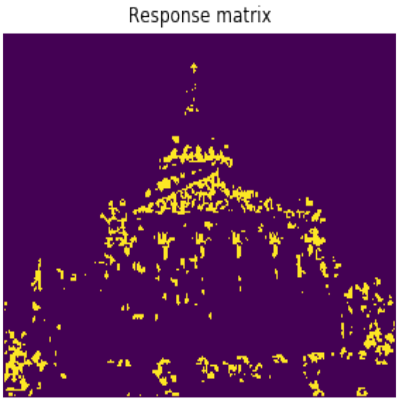
Figure 17: Image of Pantheon and its Sobel derivatives.



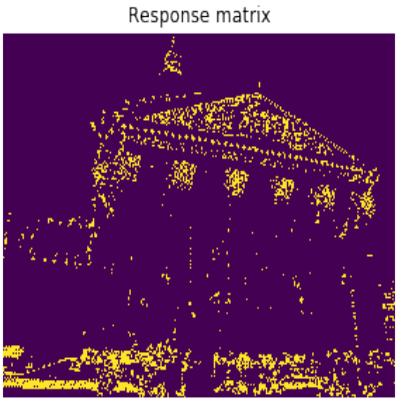
(a) The R (response matrix) calculated for first image of Pantheon.



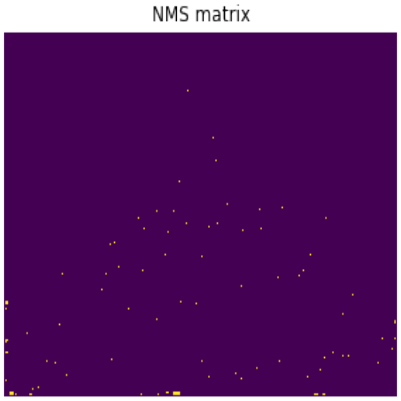
(b) The R (response matrix) calculated for second image of Pantheon.



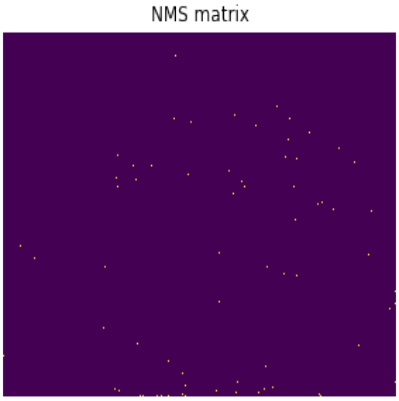
(c) After thresholding the R of first Image



(d) After thresholding the R of second Image



(e) After performing Non-Maximum-Suppression on the thresholded R of first Image



(f) After performing Non-Maximum-Suppression on the thresholded R of second Image

Figure 18: Figure showing the R value, thresholded value, and the corner detected for the images of Pantheon.

$$E_{vv}(u,v) = \sum_{x,y} 2w(x,y)[I_y(x+u,y+v)]^2 + \sum_{x,y} 2w(x,y)[I(x+u,y+v) - I(x,y)]I_{yy}(x+u,y+v)$$

$$E_{uv}(u,v) = \sum_{x,y} 2w(x,y)[I_y(x+u,y+v)I_x(x+u,y+v)] + \sum_{x,y} 2w(x,y)[I(x+u,y+v) - I(x,y)]I_{xy}(x+u,y+v)$$

These gives rise to,

$$E(0,0) = 0$$

$$E_u(0,0) = 0$$

$$E_v(0,0) = 0$$

$$E_{uu}(0,0) = \sum_{x,y} 2w(x,y)(I_x(x,y))^2$$

$$E_{vv}(0,0) = \sum_{x,y} 2w(x,y)(I_y(x,y))^2$$

$$E_{uv}(0,0) = \sum_{x,y} 2w(x,y)(I_x(x,y)I_y(x,y))$$

Hence the whole edge matrix $E(u,v)$ for a particular point can be approximated as:

$$E(u,v) \approx [u\ v] \begin{bmatrix} \sum_{x,y} 2w(x,y)(I_x(x,y))^2 & \sum_{x,y} 2w(x,y)(I_x(x,y)I_y(x,y)) \\ \sum_{x,y} 2w(x,y)(I_x(x,y)I_y(x,y)) & \sum_{x,y} 2w(x,y)(I_y(x,y))^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

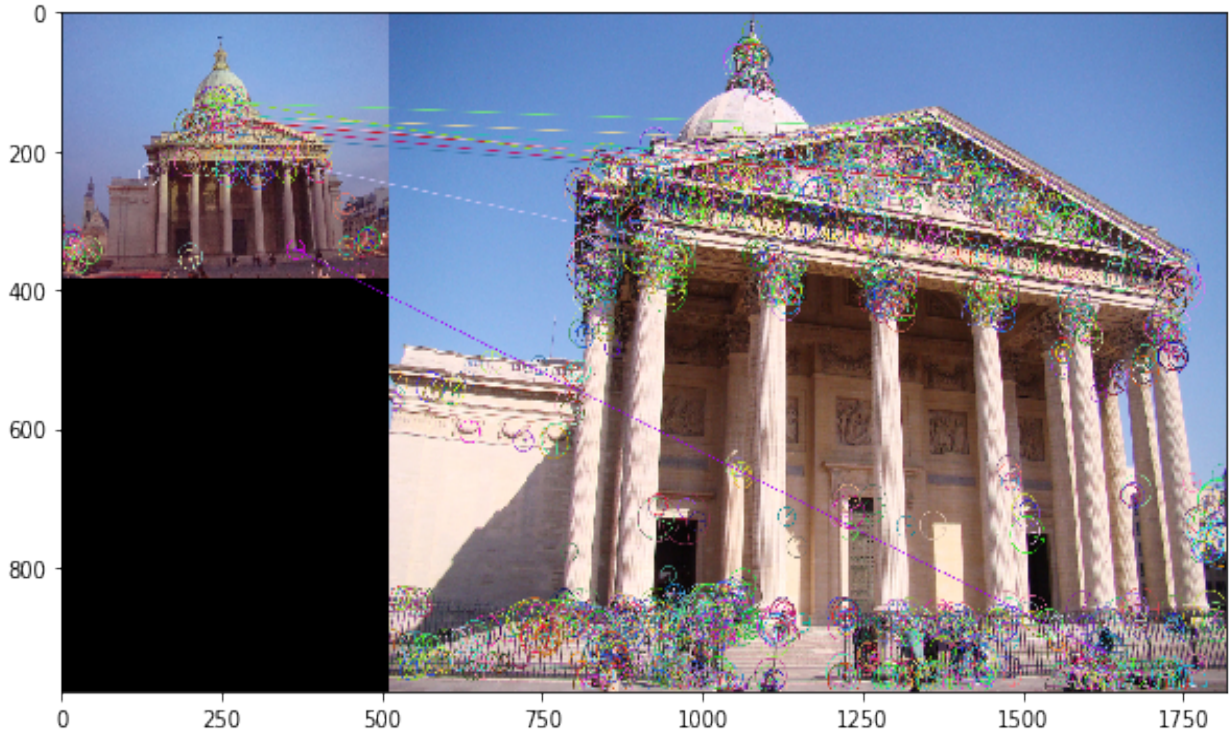


Figure 19: The feature matches of two images of Pantheon by performing SIFT algorithm using Version-1 of Non-maximum-Suppression.

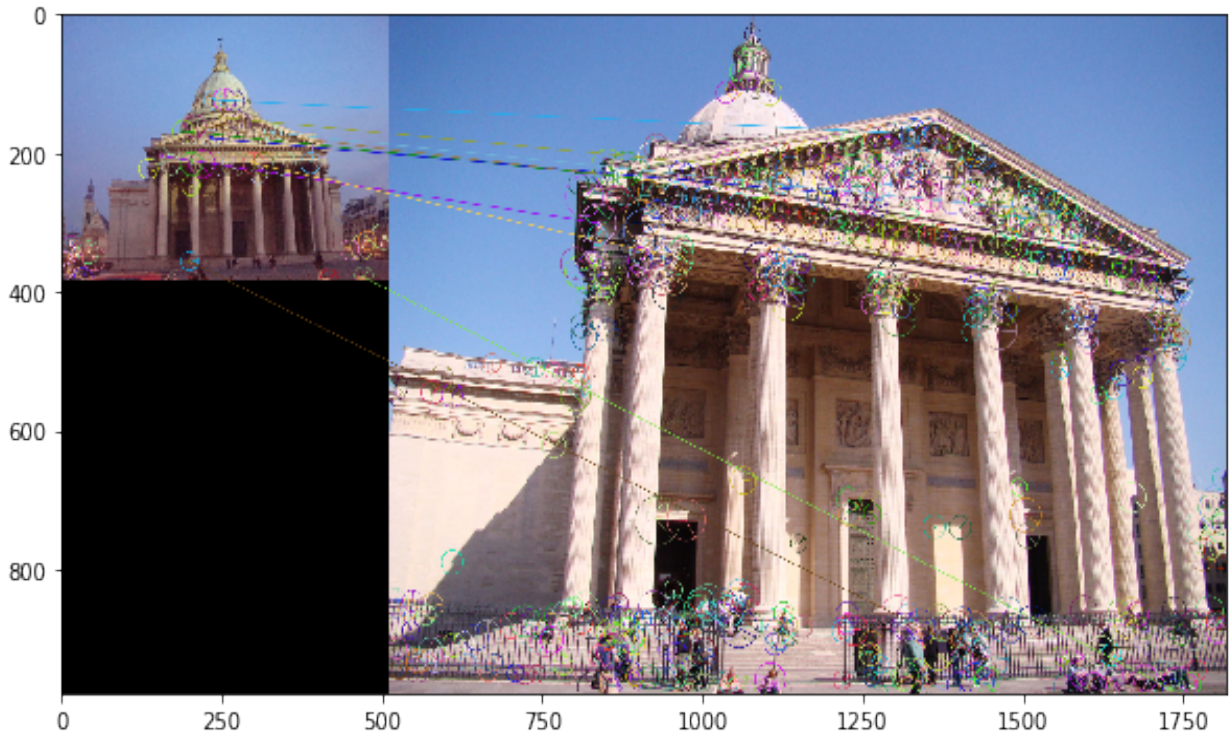
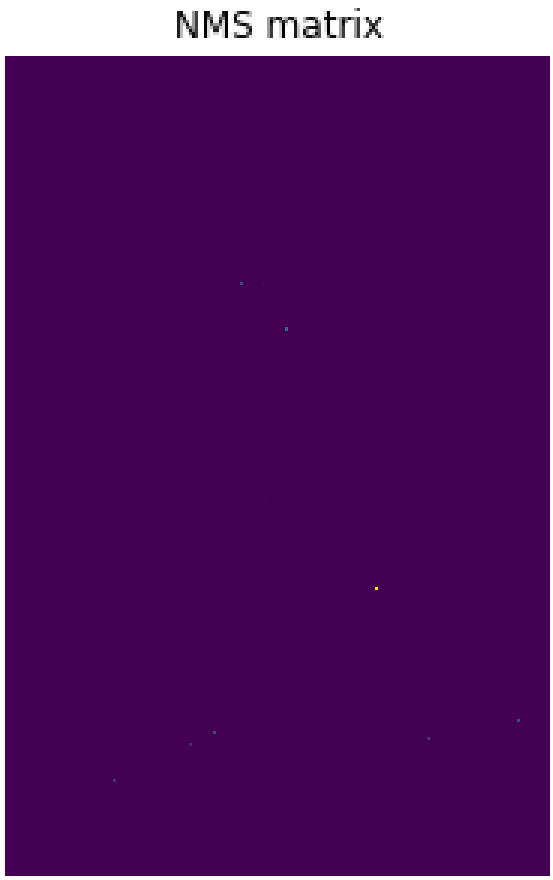
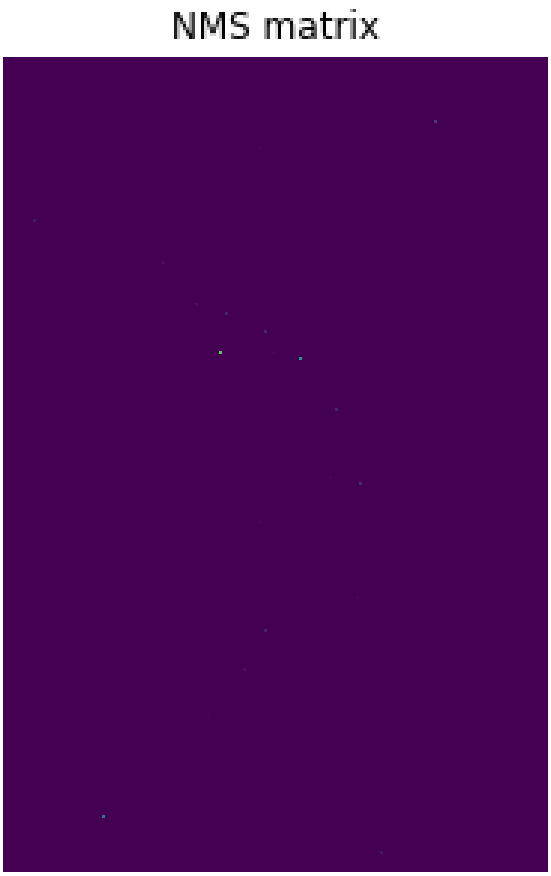


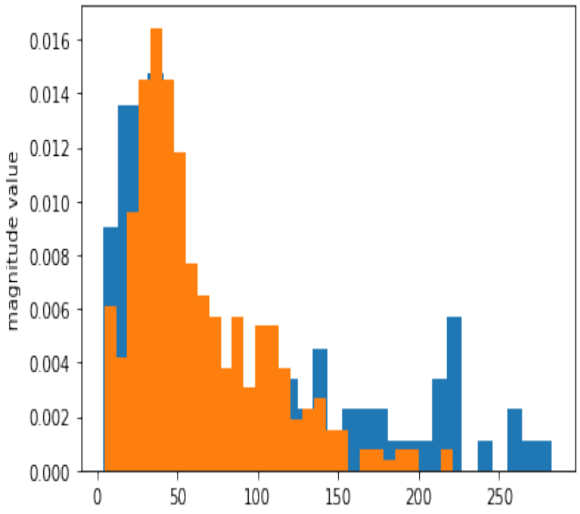
Figure 20: The feature matches of two images of Pantheon by performing SIFT algorithm using Version-2 of Non-maximum-Suppression.



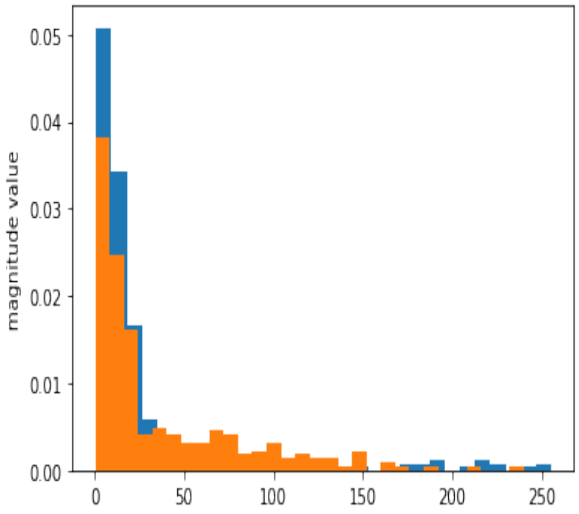
(a) The $Response\_Mat$ of image 1 of Statue of Liberty by applying algorithm 3



(b) The $Response\_Mat$ of image 2 of Statue of Liberty by applying algorithm 3



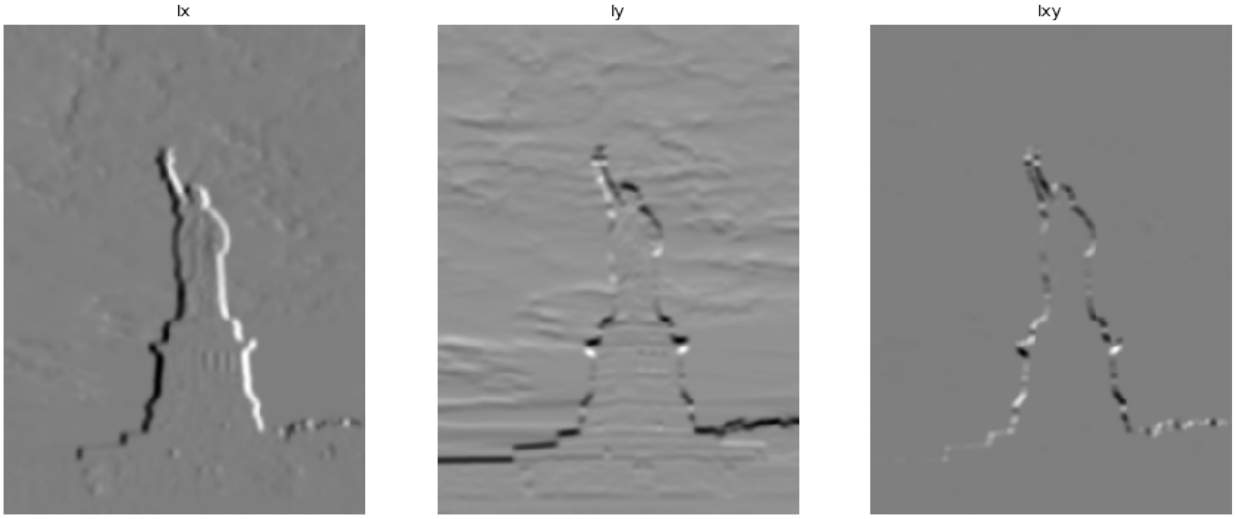
(c) The distribution of Intensity of $Response\_Mat$ of both the images of Statue of Liberty by applying algorithm 3



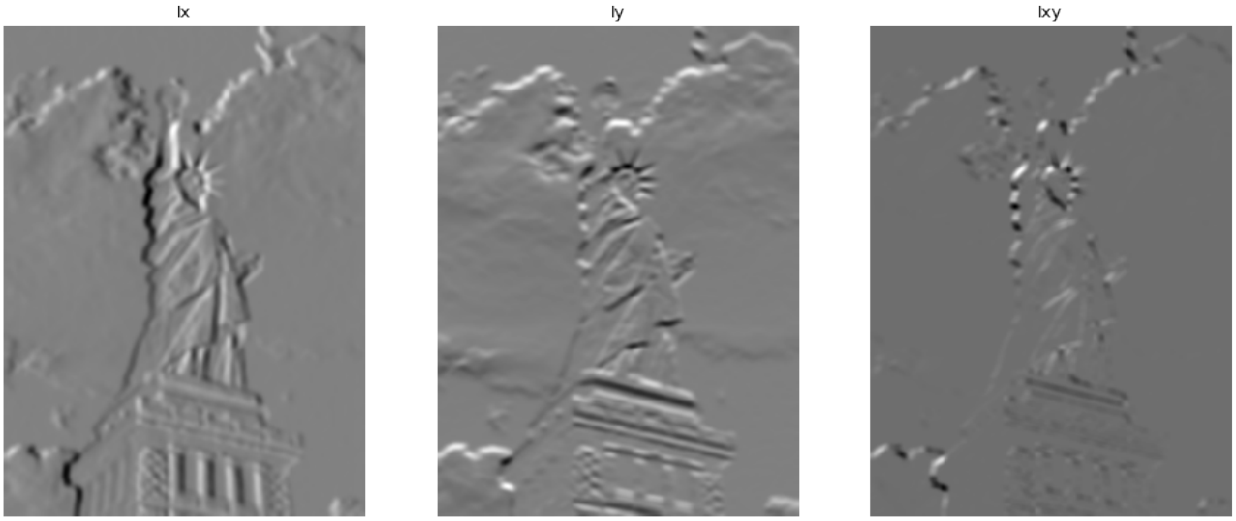
(d) The distribution of Intensity of $Response\_Mat$ of both the images of Statue of Liberty by applying algorithm 2

Figure 21: Image of Statue of Liberty and its Sobel derivatives.

(a) The original image of Statue of Liberty.

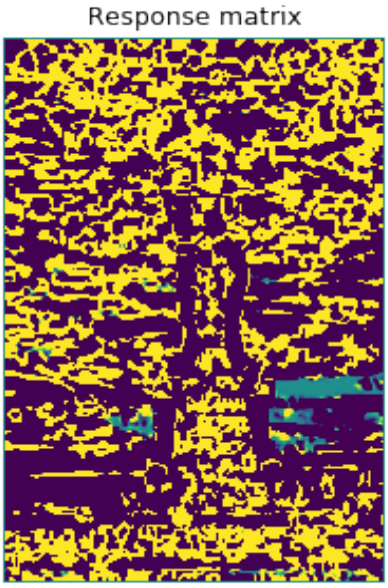


(b) Gradients of the first Image, $\frac{\partial I_1}{\partial x}$ (left), $\frac{\partial I_1}{\partial y}$ (middle) and $\frac{\partial I_1}{\partial x}\frac{\partial I_1}{\partial y}$ (right) using Sobel filter.

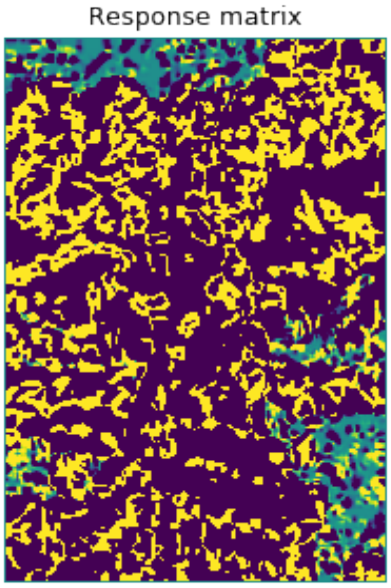


(c) Gradients of the second Image, $\frac{\partial I_2}{\partial x}$ (left), $\frac{\partial I_2}{\partial y}$ (middle) and $\frac{\partial I_2}{\partial x}\frac{\partial I_2}{\partial y}$ (right) using Sobel filter.

Figure 22: Image of Statue of Liberty and its Sobel derivatives.

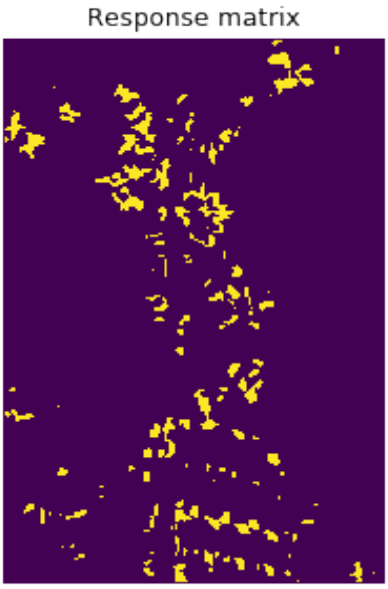


(a) The R (response matrix) calculated for first image of Statue of Liberty.

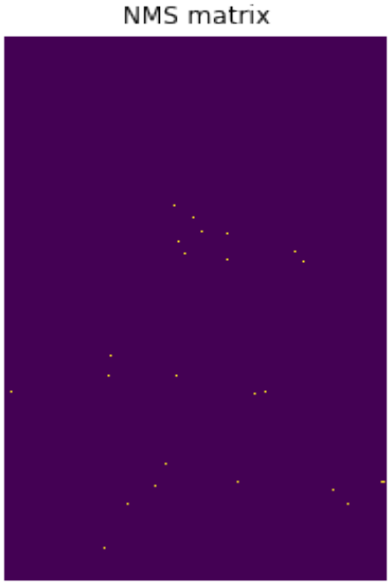


(b) The R (response matrix) calculated for second image of Statue of Liberty.

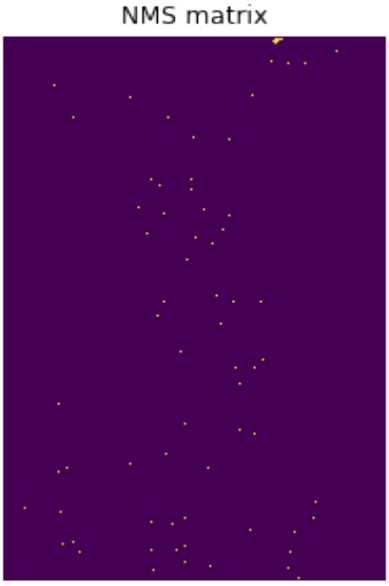
(c) After thresholding the R of first Image



(d) After thresholding the R of second Image



(e) After performing Non-Maximum-Suppression on the thresholded R of first Image



(f) After performing Non-Maximum-Suppression on the thresholded R of second Image

Figure 23: Figure showing the R value, thresholded value, and the corner detected for the images of Statue of Liberty.

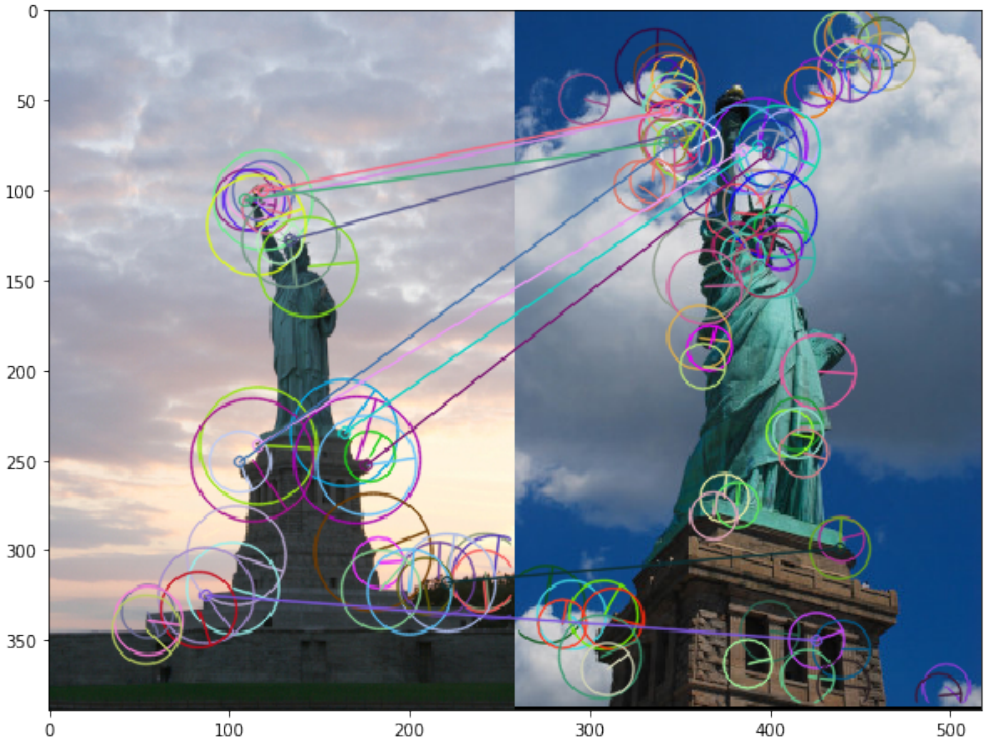


Figure 24: The feature matches of two images of Statue of Liberty by performing SIFT algorithm using Version-1 of Non-maximum-Suppression.

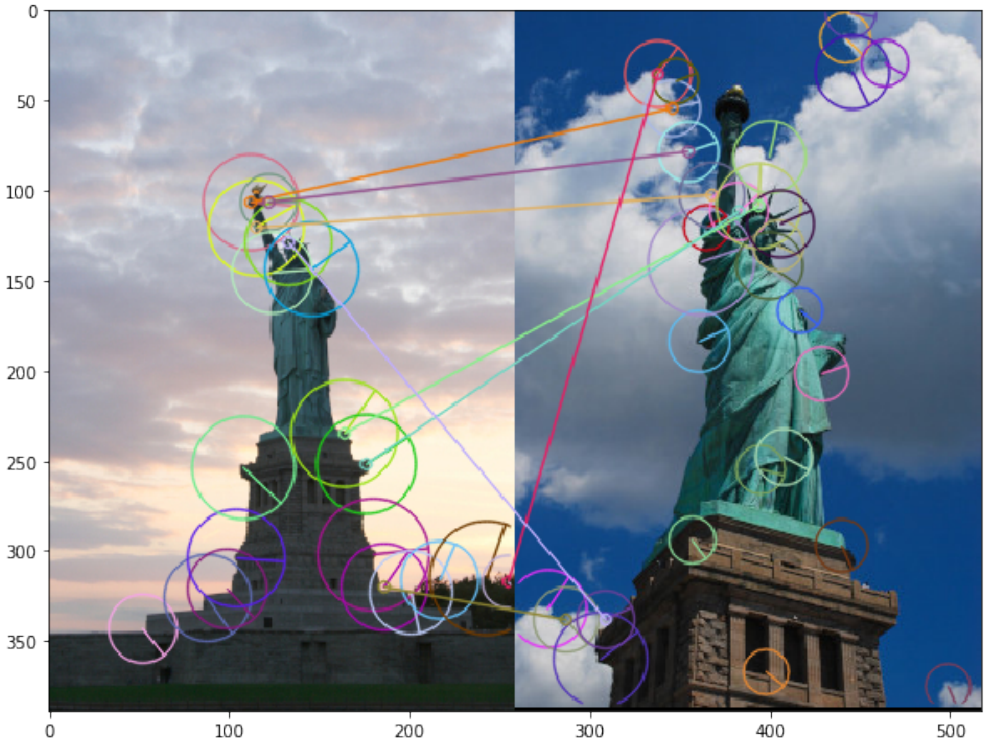


Figure 25: The feature matches of two images of Statue of Liberty by performing SIFT algorithm using Version-2 of Non-maximum-Suppression.

In simple terms, if we neglect the 2w and write it as a derivative of Image I,

$$E(u,v) \approx [u\ v] \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

Which can be written as:

$$E(u,v) \approx [u\ v] M \begin{bmatrix} u \\ v \end{bmatrix}$$

Here M is the second moment matrix,

$$M = \begin{bmatrix} \sum_{x,y} I_x I_x & \sum_{x,y} I_x I_y \\ \sum_{x,y} I_x I_y & \sum_{x,y} I_y I_y \end{bmatrix}$$

Hence, in a more compact notation,

$$M = \sum_{x,y} \begin{bmatrix} I_x \\ I_y \end{bmatrix} [I_x\ I_y] = \sum_{x,y} \nabla I (\nabla I)^T$$

Corners are the distinctive interest points. We can also write the gradient of the image in x-direction [4] as $I_x = \frac{\partial I}{\partial x}$, the gradient of the image in y-direction as $I_y = \frac{\partial I}{\partial y}$ and the gradient of the image in xy-direction as $I_x I_y = \frac{\partial I}{\partial x}\frac{\partial I}{\partial y}$. The surface of $E(u,v)$ is locally approximated by a quadratic form as shown in Figure 3a. This may be considered analogous to the contours obtained by applying gradient descent to least square fit. If we take a horizontal slice of the contour as shown in Figure 3b, we will get an equation of ellipse.

$$I_x^2 u^2 + 2 I_x I_y uv + I_y^2 v^2 = k$$

We can further interpret the second moment matrix as,

$$M = \sum_{x,y} \begin{bmatrix} I_x \\ I_y \end{bmatrix} [I_x\ I_y] = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$

The gradients can be either vertical or horizontal. If either of the λ is close to 0 then this is not a corner. We need to look for locations where the value of λs are large, i.e., both λ_1 and λ_2 are high. The diagonalization of M, the moment matrix gives rise to the equation,

$$M = R^{-1} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} R$$

The axis lengths are determined by the eigenvalues and the orientations is determined by R. The visualization of this is shown in Figure 4. We can also classify the image points using eigenvalues of M as shown in Figure 5a. We see that when the value of λ_1 and λ_2 are low, it represents a flat surface in the corner space. When one of them is high, then it represents an edge. When both of them are high, it represents a corner, since a small change in any one direction will result in a large change when we are sliding the window in corner as shown in Figure 1.

The response function can be also written as

$$R = det(M) - \alpha\, trace(M)^2 = \lambda_1 \lambda_2 - \alpha(\lambda_1 + \lambda_2)^2$$

Here α is a constant and varies from 0.04 to 0.06. R depends on eigen values of M, but we don't need to compute them via square root, so this formulation is computationally very fast. if R is large, then it is a corner. If R is negative with large magnitude then it is an edge. If R is small then it is a flat region. The variation of R is shown in Figure 5b.

We have implemented the Harris Corner Algorithm using the same algorithm shown in Algorithm 1. We have performed two version of the Non-Maximum-Suppression stage, and both gave different results. The first version of the non-maximum-suppression is similar to the one given by Harris, and is shown in Algorithm 3. This uses the original version of Thresholding and performing non maximum suppression on a given window size. OpenCV's dilation version would

Algorithm 1: Modified Harris Corner Detector (Jimut Version)

Input: $Im \leftarrow Image, ws \leftarrow window\_size, \alpha \leftarrow alpha, thres \leftarrow threshold\ and\ nms\_size \leftarrow non-maximum\ suppression\ window\ size$

Result: $Corners, I_x, and\ I_y$

$Image\_Gray \leftarrow Im$; // Convert color Image to Gray Scale

$R \leftarrow dim(Im)$; // Initialise response matrix of same dimension of image

$I \leftarrow Image\_Gray * G_\sigma(x, y)$; // Smooth Grayscale image by applying Gaussian filter

$I_{x\_} \leftarrow \frac{\partial I}{\partial x}$; // Apply Sobel filter to get gradient in X direction

$I_{y\_} \leftarrow \frac{\partial I}{\partial x}$; // Apply Sobel filter to get gradient in Y direction

$I_{xy\_} \leftarrow \frac{\partial I}{\partial x}\frac{\partial I}{\partial y}$; // Apply Sobel filter to get gradient in X-Y direction

$I_x \leftarrow I_{x\_} * G_\sigma(x, y)$; // Smooth the I_x Image via Gaussian Filtering

$I_y \leftarrow I_{y\_} * G_\sigma(x, y)$; // Smooth the I_y Image via Gaussian Filtering

$I_{xy} \leftarrow I_{xy\_} * G_\sigma(x, y)$; // Smooth the I_{xy} Image via Gaussian Filtering

$I_x^2 = I_x * I_x$

$I_y^2 = I_y * I_y$

$padding \leftarrow \frac{window\_size}{2}$

for $y, x \leftarrow 0$ **to** $Image.height(), Image.width()$ **do**

- $windowI_x^2 = I_x^2[y - padding : y + padding + 1, x - padding : x + padding + 1]$;
- $windowI_{xy} = I_{xy}[y - padding : y + padding + 1, x - padding : x + padding + 1]$;
- $windowI_y^2 = I_y^2[y - padding : y + padding + 1, x - padding : x + padding + 1]$;
- // Calculate the response in the window for each of the terms in the Moment equation
- $S_x^2 = sum(windowI_x^2)$; ; // Calculate the sum of putting in moment equation
- $S_{xy} = sum(windowI_{xy})$;
- $S_y^2 = sum(windowI_y^2)$;
- $det = S_x^2 * S_y^2 - S_{xy}^2$; // Get the determinant of the window
- $trace = S_x^2 + S_y^2$; // Get the trace of the window
- $R[x, y] = det - \alpha\ (trace)^2$; // Calculate Response

end

$Perform\ Thresholding$;

$Perform\ Non\ Maximum\ Suppression$ using **version-1** or **version-2** to detect corners;

Algorithm 2: Non-Maximum-Suppression Version - 2

Input: $R \leftarrow Response\ Matrix, thres \leftarrow threshold\ and\ nms\_size \leftarrow non-maximum\ suppression\ window\ size$

Result: $Corners$

$corners \leftarrow \phi$;

$Response\_Mat \leftarrow dim(R)$; // Initialise Response Matrix to the dimension of R

$nms\_pad = \frac{nms\_size}{2}$;

for $y, x \leftarrow 0$ **to** $nms\_pad\ ...R.height() - nms\_pad, R.width() - nms\_pad$ **do**

- $r \leftarrow max(R[y - nms\_pad, y + nms\_pad, x - nms\_pad, x + nms\_pad])$;
- // Find the maximum of the sub-window
- $x, y \leftarrow coordinates\ of\ r\ i.e.,\ the\ maximum\ value$;
- $Response\_Mat[y - nms\_pad, y + nms\_pad, x - nms\_pad, x + nms\_pad] = 0$;
- // Make the contents of sub-window to 0
- $Response\_mat[y][x] = r$;
- $Corners.append(list(x, y))$;
- $x \leftarrow x + nms\_size$;
- $y \leftarrow y + nms\_size$;

end

also give feasible result, but this is different from that algorithm. The second version of the algorithm is shown in Algorithm 2.

We have taken the famous Notre Dame's image as shown in Figure 7a. We the blur the image by applying Gaussian filter [5] and perform the sobel derivative [6] in the x-direction as shown in Figure 7b for Image 1. We perform the same for the other image too as shown in Figure 7c. We calculate the response matrix as accordance with the algorithm and the value of the Response matrix is shown in Figure 8a for Image 1 and Figure 8b for Image 2. We perform the thresholding of the response matrix of the first image 8a via algorithm 3 and get the modified thresholded Response matrix as shown in Figure 8c. Similarly, we perform the thresholding of the response matrix of the second image 8b via algorithm 3 and get the modified thresholded Response matrix as shown in Figure 8d. We then perform the Non-maximum-suppression using algorithm 3 and get the final response matrix detecting the modified Harris corners in Figure 8e and Figure 8f.

Algorithm 3: Non-Maximum-Suppression Version - 1

Input: $R \leftarrow Response\ Matrix, thres \leftarrow threshold\ and\ nms\_size \leftarrow non-maximum\ suppression\ window\ size$
Result: $Corners$
$corners \leftarrow \phi$;
$Response\_Mat \leftarrow dim(R)$; // Initialise Response Matrix to the dimension of R
for $y, x \leftarrow 0$ **to** $R.height(), R.width()$ **do**
  $r \leftarrow R[y, x]$;
  if $r > thres$ **then**
    $Response\_Mat \leftarrow 255$
  else
    $Response\_Mat \leftarrow 0$
end
$nms\_pad = \frac{nms\_size}{2}$;
for $y, x \leftarrow 0$ **to** $nms\_pad\ ...R.height() - nms\_pad, R.width() - nms\_pad$ **do**
  $r \leftarrow R[y, x]$;
  if $Response\_Mat[y][x] \equiv 255$ **then**
    $Response\_Mat \leftarrow 255$;
  end
  $Response\_mat[y - nms\_pad : y + nms\_pad, x - nms\_pad : x + nms\_pad] = 0$;
  $Response\_mat[y][x] = 255$;
  $Corners.append(list(x, y))$;
  $x \leftarrow x + nms\_size$;
  $y \leftarrow y + nms\_size$;
end

This algorithm keeps the magnitude of the gradient and selects the maximum from a given window and setting every other element in that window to 0. This gives better results since this collects the best possible corner candidates from the region of interest. The $Response\_Matrix$ obtained via algorithm 2 for the image 7a (left) is shown in Figure 6a. Similarly, the $Response\_Matrix$ obtained via algorithm 2 for the image 7a (right) is shown in Figure 6b. We can clearly see that the $Response\_Matrix$ obtained via algorithm 2 gives a **Chi-Squared** distribution as shown in Figure 6d. The equation for the same can be written as,

$$\frac{1}{\Gamma(k/2)}\,\gamma\left(\frac{k}{2}, \frac{x}{2}\right)$$

where k is a constant and Γ comes from the Gamma function. Corners detected from this algorithm are barely visible, since after a certain amount of thresholding, the number of corners exponentially decreases over time in general, so the corners are barely visible from naked eye. The $Response\_Matrix$ obtained via algorithm 3 gives a **Gaussian** distribution as shown in Figure 6c. The equation for the same can be written as,

$$G_\sigma(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x-y)^2}{2\sigma^2}}$$

Here σ is the standard deviation. The corners are visible with equal intensities and are used to create keypoints for the Scale Invariant Feature Transform (SIFT) algorithm [7]. Our aim here is to match two images via SIFT algorithm by

creating keypoints. SIFT algorithm extracts features that are invariant to image scale and rotation, affine distortion, change in 3D viewpoint, addition of noise and change in illumination. The features detected are highly distinctive and a single feature can be correctly matched with high probability against a large database of features from many images. This algorithm can even recognize objects by performing a fast nearest-neighbor algorithm, giving a near real-time performance. SIFT uses a Difference of Gaussian using a pyramidial approach to detect scale invariant features. Since a corner in one scale may not appear as corner but a ridge when zoomed in. It has a descriptor of size 128 for each keypoint, created by histogramming the magnitude of the image gradient. Figure 9 shows the feature matches of both the images as in Figure 7a by applying SIFT algorithm and performing non-maximum suppression using algorithm 3. Similarly, Figure 10 shows the feature matches of both the images as in Figure 7a by applying SIFT algorithm and performing non-maximum suppression using algorithm 2. There is a minute difference of feature match and both gives good result. There is a certain amount of threshold needed to control the number of feature matches, and for this case it is about **100**. If we decrease the threshold intensity, then there will be a greater number of feature points.

We do the same for a series of images. We take the image of Wood House as shown in Figure 12a. We perform the sobel derivatives using Sobel filter to get the magnitude of the gradient in the X and Y directions. The sobel derivative of the first image is shown in Figure 12b. We do the same for second image as shown in Figure 12c. The response matrix of the first image is shown in Figure 13a and for the second image is shown in Figure 13b. Performing thresholding using 2 gives the response matrix as shown in figure 13c for image 1 and Figure 13d for image 2 using algorithm 2. The thresholded version of both the images after this stage is shown in Figure 13e and Figure 13f. After generating keypoints and matching images via OpenCV's SIFT algorithm we get the result as shown in Figure 14 for algorithm 2 of Non-maximum-Suppression and Figure 15 for algorithm 3.

Again, we take the image of Pantheon as shown in Figure 17a. We perform the sobel derivatives using Sobel filter to get the magnitude of the gradient in the X and Y directions. The sobel derivative of the first image is shown in Figure 17b. We do the same for second image as shown in Figure 17c. The response matrix of the first image is shown in Figure 18a and for the second image is shown in Figure 18b. Performing thresholding using 2 gives the response matrix as shown in figure 18c for image 1 and Figure 18d for image 2 using algorithm 2. The thresholded version of both the images after this stage is shown in Figure 18e and Figure 18f. After generating keypoints and matching images via OpenCV's SIFT algorithm we get the result as shown in Figure 19 for algorithm 2 of Non-maximum-suppression and Figure 20 for algorithm 3.

We see that for all the images, algorithm 3 gives **Gaussian** distribution as shown in Figures 6c, 11a, 16c and 21c. The algorithm 2 gives a **Chi-Squared** distribution as shown in Figures 11b 6d 16d, and 21d.

Finally, we take the image of Statue of Liberty as shown in Figure 22a. We perform the sobel derivatives using Sobel filter to get the magnitude of the gradient in the X and Y directions. The sobel derivative of the first image is shown in Figure 22b. We do the same for second image as shown in Figure 22c. The response matrix of the first image is shown in Figure 23a and for the second image is shown in Figure 23b. Performing thresholding using 2 gives the response matrix as shown in figure 23c for image 1 and Figure 23d for image 2 using algorithm 2. The thresholded version of both the images after this stage is shown in Figure 23e and Figure 23f. After generating keypoints and matching images via OpenCV's SIFT algorithm we get the result as shown in Figure 24 for algorithm 2 of Non-maximum-suppression and Figure 25 for algorithm 3.

5 Conclusion

The main purpose of this investigation [8] is to visualize how Harris Corner Works. We have also looked at the various ways to find corners by performing two type of non-maximum-suppression algorithm. We have built two types of algorithm for the same. The traditional Harris Corner detector have some limitation, for example a corner may not be a corner in different scale of the same image, since a corner may tend to be ridges when zoomed in. We also looked at the various ways by which we can match the feature vectors in terms of descriptors for two images. Scale Invariant Feature Transform (SIFT) [9, 10, 11] is used to create a descriptor of size 128 which represents the keypoint determined by Harris Corners. These keypoints are robust to light intensity and some occlusion. We have seen that SIFT performs good in a near real-time system which is capable of detecting matching feature vectors between two images in different orientations. These are the traditional algorithms for matching images for a large database of feature vectors. We have seen that we can change the value of α and tune certain hyperparameters to get different corner points. We can also tune the threshold for getting varying amount of corner points. If we make the threshold very low, then we may get unnecessary corners which are not so unique when we compute the SIFT descriptors. We can also tune the size of non-maximum-suppression window size and the algorithm itself to get different type of corners. The version - 1 of the algorithm 2 gave quite a distinct and visible set of features since they were clipped. When we used the version - 2 of the algorithm 3 we saw that since it is taking the maximum of the window, some corners were not visible to the

naked eye in the response matrix, since different values of corner were caught without clipping. It performed better than the other algorithm since it got the best candidate for the corners. We also found that this algorithm took longer than the previous one but is more effective. The larger the size of the image to be passed to the Harris detector and SIFT algorithm, the larger the window size is needed to calculate the response. The SIFT's descriptor's gradient direction is given in the radius of the circle to determine the direction of the descriptor in whole.

6 Acknowledgements

It is ritual that scholars express their gratitude to their supervisors. This acknowledgement is very special to me to express my deepest sense of gratitude and pay respect to my supervisor, Tamal Maharaj, Department of Computer Science, for his constant encouragement, guidance, supervision, and support throughout the completion of my project. His close scrutiny, constructive criticism, and intellectual insight has immensely helped me in every stage of my work.

I'm grateful to my father, Dr. Jadab Kumar Pal, Deputy Chief Executive, Indian Statistical Institute, Kolkata for constantly motivating and supporting me to develop this documentation along with the application. Finally, I acknowledge the help received from all my friends and well-wishers whose constant motivation has promoted the completion of this project.

References

[1] Chris Harris and Mike Stephens. A combined corner and edge detector. In *In Proc. of Fourth Alvey Vision Conference*, pages 147–151, 1988.

[2] Jimut Bahan Pal. Traditional lines and circles detector - osf, 2020. `https://dx.doi.org/10.31219/osf.io/sf4ye` [Online; accessed 17-March-2020].

[3] Aaron Bobick. Features 1 – harris and other corners. `https://www.cc.gatech.edu/~afb/classes/CS4495-Fall2014/slides/CS4495-Features1.pdf`, 2014. last accessed 17-March-2020.

[4] Wikipedia contributors. Image gradient — Wikipedia, the free encyclopedia, 2019. `https://en.wikipedia.org/wiki/Image_gradient` [Online; accessed 17-March-2020].

[5] Jimut Bahan Pal. A deeper look into hybrid images. `https://arxiv.org/abs/2001.11302` last accessed on 17.03.2020, 2020.

[6] Wikipedia contributors. Sobel operator — Wikipedia, the free encyclopedia, 2020. `https://en.wikipedia.org/wiki/Sobel_operator` [Online; accessed 17-March-2020].

[7] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, November 2004.

[8] Tamal Maharaj. Project 3: Detecting harris corners and matching images, 2020. `http://cs.rkmvu.ac.in/~tamal/CV_S20/proj3/proj3.html` [Online; accessed 17-March-2020].

[9] E. R. Davies. Computer and machine vision theory, algorithms, practicalities, 2012.

[10] Richard Szeliski. Computer vision algorithms and applications, 2011.

[11] Rafael C. Gonzalez and Richard E. Woods. *Digital image processing*. Prentice Hall, Upper Saddle River, N.J., 2008.