Feature Extraction and Object Recognition

Michael Ferko
Dept. of Electrical and Computer Engineering
University of Louisville
Louisville, USA
Email: Mike.W.Ferko@gmail.com

***Abstract*—A local feature is an image pattern which differs from its immediate neighborhood. Local features can be points (corner, blob), edges or small image patches. To localize features in images, a local neighborhood of pixels needs to be analyzed, giving all local features some implicit spatial extent. Typically, some measurements are taken from a region centered on a local feature and converted into descriptors. The descriptors can then be used for various applications (like object Matching) Good feature for one application may be useless in the context of a different problem.**

Keywords—Edge Detector, Corner Detector, SIFT, CSIFT, Pattern Recognition, Machine Intelligence, Probability & Statistics, Engineering, K-NN, RANSAC, Algorithm, Density Estimation, Digital Image Processing, Fast Fourier Transform

I. Introduction

The aim of this lab is to extract features (i.e., edges, lines, corners, etc.), and to build a local descriptor of these features that is invariant to image changes (i.e., rotation, translation, and scaling). The goal of this paper is to introduce the reader to some cutting-edge techniques that have been built for feature extraction from Images. The papers addressed will be John Canny's paper on A Computational Approach to Edge Detection, A Combined Corner and Edge Detector by Chris Harris and Mike Stephens, and David Lowe's paper on Distinctive Image Features from Scale-Invariant Keypoints.

II. PART 1: Edge Detection and Corner Detection

From John Canny's paper on Edge Detection we will be going step by step through the implementation of the algorithm. First, we should define what we are trying to achieve. From Canny's paper we can see the result is a binary representation of the edges extracted from an image. The Parts image is particularly a prime example. Also well known, is the Lenna image. We will be going step by step through the parts image and showing the result for a few images to show the robust nature of this algorithm. The following figure is the parts image we will be analyzing.

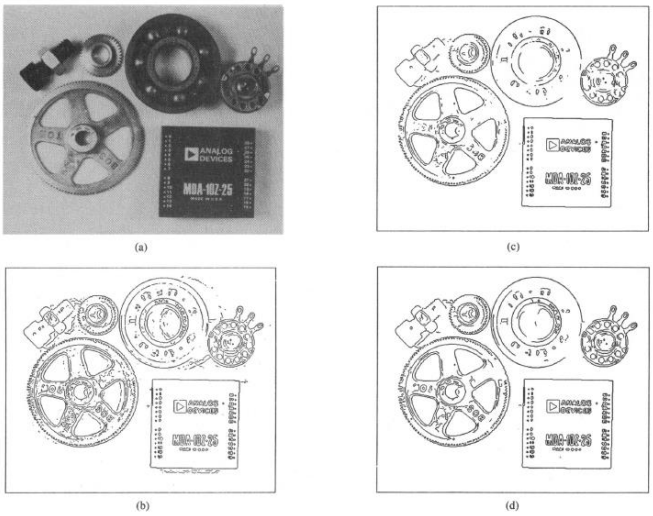


Fig. 7. (a) Parts image, 576 by 454 pixels. (b) Image thesholded at T_1. (c) Image thresholded at 2 T_1. (d) Image thresholded with hysteresis using both the thresholds in (a) and (b).

Figure 1: Parts Image Edge Detection

The original image (a) is what we will start processing. The final image (d) shows the clear edges of the image. We can now preprocess the image by normalizing the image across 0 to 255 gray scale pixel intensities. We can then reshape the image into a column vector to achieve the image intensity probability density histogram.

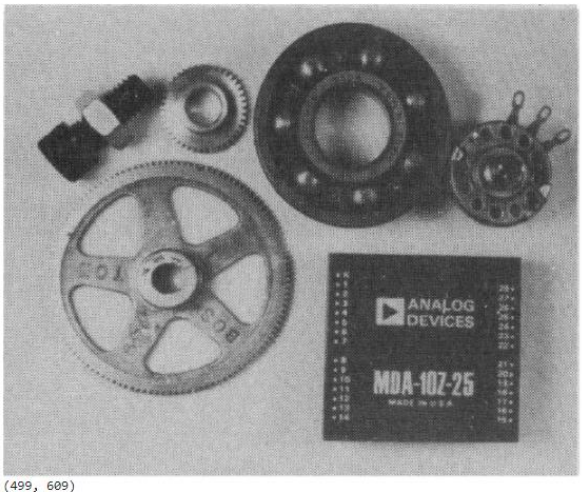


Figure 2: Imported Parts Image for Analysis

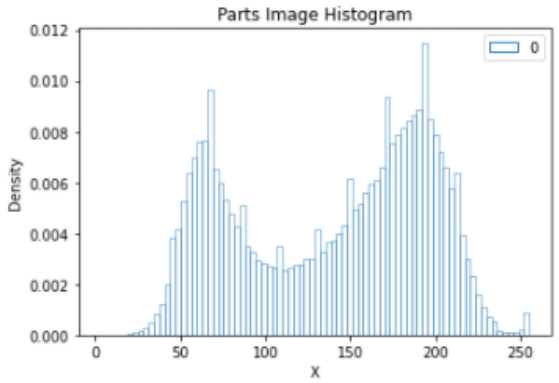


Figure 3: Histogram of Parts Image

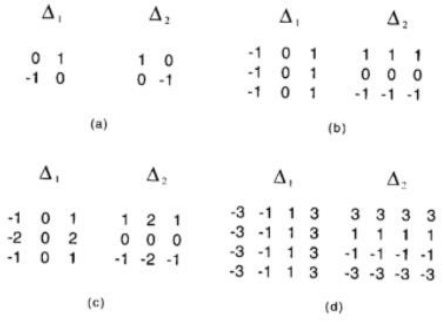
From the parts Image we can find local maxima of pixel intensity by finding the magnitude and orientation of the gradient of the image. If f is the pixel intensity in a range of 0 to 255 then we can find the gradient from equation (1) and the magnitude and orientation of the gradient of the image via equation (2) and (3).

$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right] \quad (1)$$

$$||\nabla f|| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2} \quad (2)$$

$$\theta = \tan^{-1}\left[\frac{\left(\frac{\partial f}{\partial x}\right)}{\left(\frac{\partial f}{\partial y}\right)}\right] \quad (3)$$

Various types of operator matrices have been developed to perform 2-dimensional convolution on an image to achieve localization of maxima/minima. A few of these operators include the Roberts' Cross operator, Prewitt 3x3 operator, Sobel operator and the Prewitt 4x4 operator. This achieves a gray scale gradient image of the same dimensions as the original image. (*Note: For the purpose of replicating the Canny Image, we will be working with inverted images where the background is black and the edges are white instead of the background white and edges black).



(a): Roberts' cross operator (b): 3x3 Prewitt operator
(c): Sobel operator (d) 4x4 Prewitt operator

Figure 4: Gradient Operators

To see the various operators in action we will see the gradient of the parts image.

Figure 5: Roberts' Gradient Image

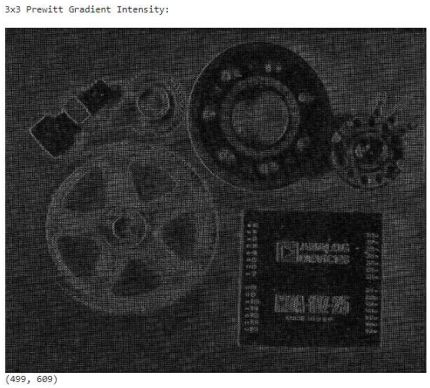


Figure 6: Prewitt 3x3 Gradient Image

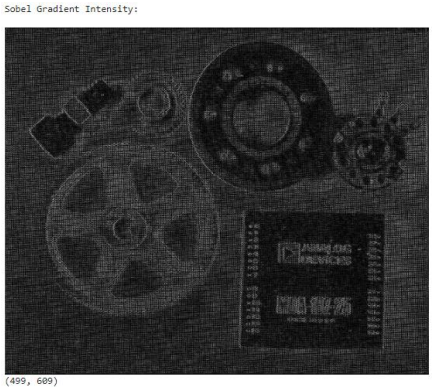


Figure 7: Sobel Gradient Image

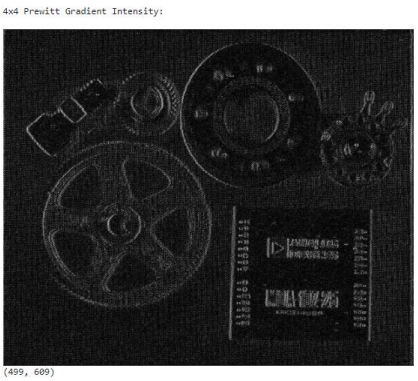


Figure 8: Prewitt 4x4 Gradient Image

Each image is very noisy, so the edges do not appear very easily. So far, we have been using the gradient of a gaussian convolution (GoG convolution) to find local maxima in the image (shown in red in Figure 9). But if we consider the zero crossing for the Laplacian of a gaussian we can see the 2nd order derivative of the gaussian convolved with the image will produce more defined edges to the image.

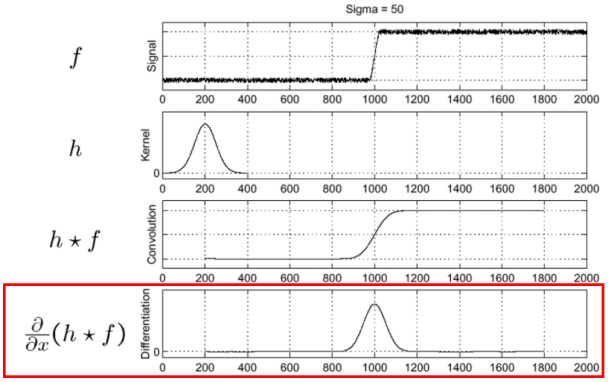


Figure 9: Gradient of a Gaussian Convolution

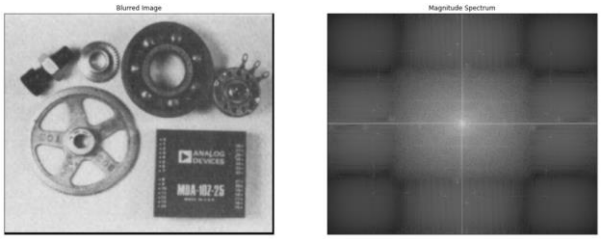


Figure 10: Gaussian 5x5 kernel Blurred Image and FFT Magnitude Spectrum

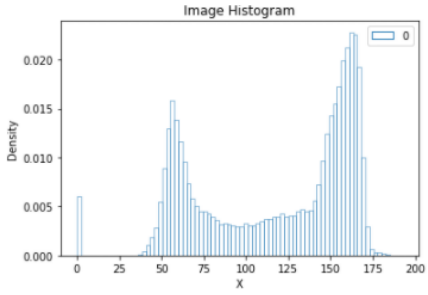


Figure 11: Gaussian Blurred Image Histogram

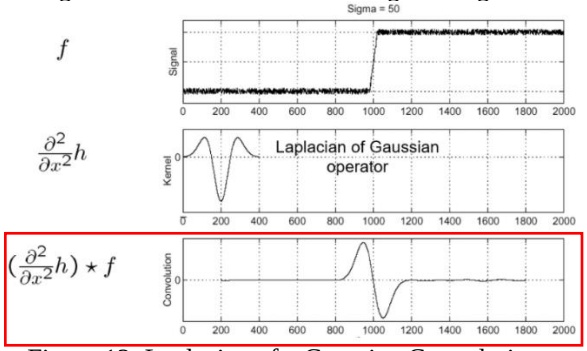


Figure 12: Laplacian of a Gaussian Convolution

The point is that when we convolve the Laplacian of a gaussian (LoG) with the image we achieve the zero crossing for the first derivative of the gaussian. And with this we can achieve more refined localization of edges.

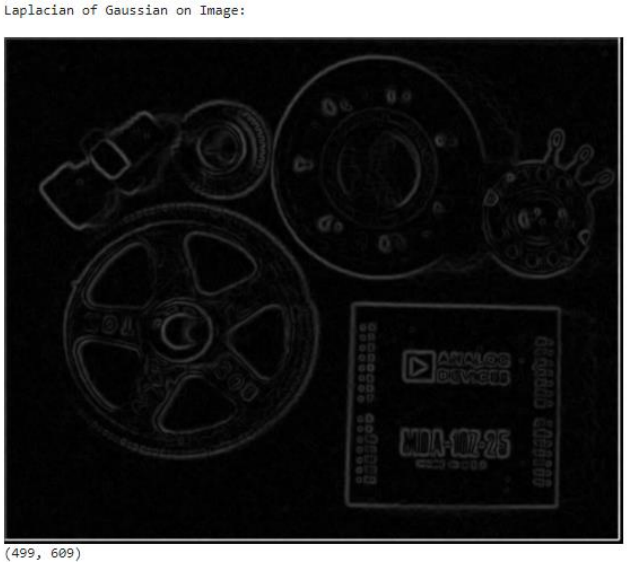


Figure 13: LoG Convolved Parts Image

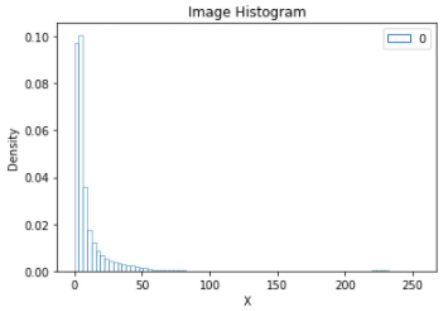
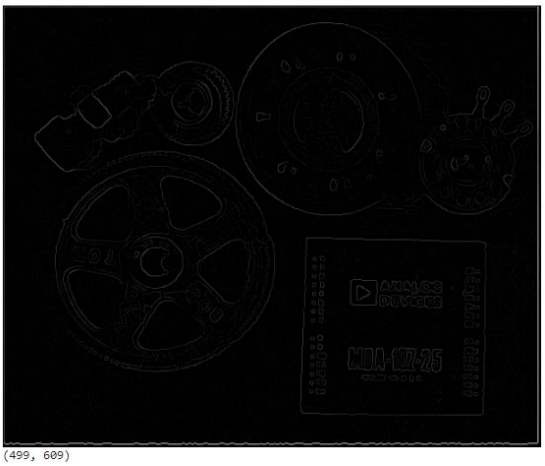


Figure 13: LoG Convolved Parts Image Histogram

From the LoG convolution of the image we achieve an exponential density histogram for the parts image. Now we can perform 3 more techniques which will achieve the Canny Image. Non-max suppression will go pixel by pixel suppressing pixels which don't have an orientation between 0 and 180 degrees to black and those that are within the 0 to 180 degree range to white.

non-max Supressed (thinning) Image:



(499, 609)

Figure 14: Non-max Suppressed Parts Image

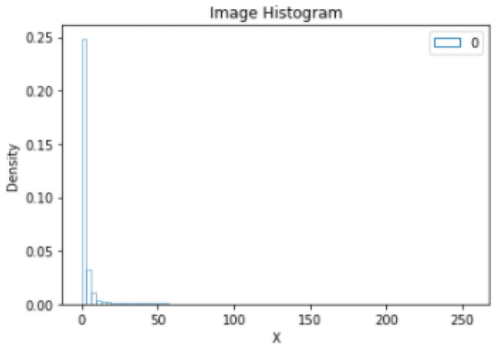
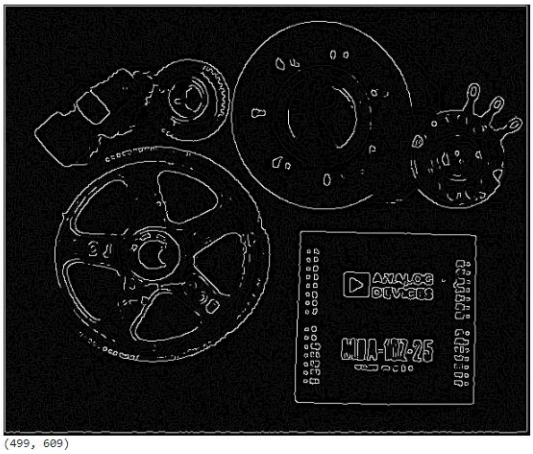


Figure 14: Non-max Suppressed Parts Image Histogram

As you can see from the histogram, we are getting closer and closer to a delta function peak at 0 (for black pixels).

Now we can double threshold the image for weak and strong pixel intensities by using a low and high threshold ratio of the intensity range of high threshold = 0.09 * Image maximum = 22.95 and low threshold = 0.05 * 0.09 * Image maximum = 1.1475. This means all the pixel intensity in the histogram between 0 and 1.1475 will be set to 0 (black), all the pixel intensities in the histogram between 1.1475 and 22.95 will be set to a gray scale intensity say 25, and all the pixels between 22.95 to 255 will be set to 255 or a white pixel intensity.

Double Thresholded Image:



(499, 609)

Figure 15: Double Threshold Parts Image

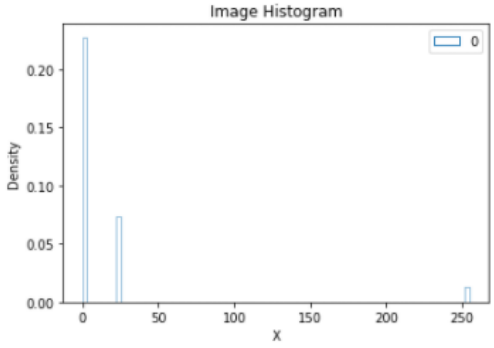
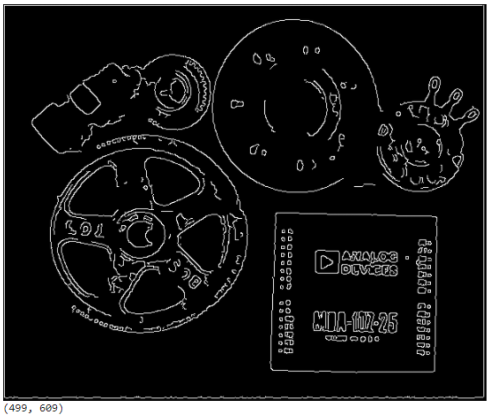


Figure 16: Double Threshold Parts Image Histogram

The last step called hysteresis will just remove the gray background that is not needed and set those gray scale pixels (25) to black (0). From this process we have achieved binary image segmentation of the edges within any image.

Canny Image:



(499, 609)

Figure 17: Hysteresis Parts Image

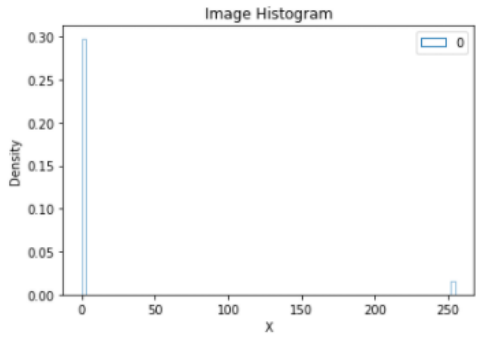


Figure 18: Hysteresis Parts Image Histogram

And if we perform the same operations, we can achieve this on any image. So, for example the very popular Canny image of Lenna:

(512, 512)

Figure 19: Lenna Image

Canny Image:

(512, 512)

Figure 20: Canny Image of Lenna

Or Messi, or license plates:

(348, 552)

Figure 21: Messi

(348, 552)

Figure 22: Canny Edges of Messi

(675, 1200)

Figure 23: License Plate

(675, 1200)

Figure 24: Canny Edges of License Plate

After John Canny developed a beautiful algorithm for edge detection Chris Harris and Mike Stephens refined Moravec's corner detector into A Combined Corner and Edge Detector algorithm. The algorithm utilizes John Canny's LoG binary image segmentation but finds the eigen values and eigen vectors in the spatial spectrum. The 2nd moments matrix M (A.K.A. the autocorrelation matrix) paints a description (descriptors will become a new data structure when talking about the SIFT) of a pixel's local neighborhood. The cornerness (or Harris response, R) can be calculated as shown:

$$M = \sigma_D^2 g(\sigma_I) * \begin{bmatrix} I_x^2(\mathbf{x},\sigma_D) & I_x(\mathbf{x},\sigma_D)I_y(\mathbf{x},\sigma_D) \\ I_x(\mathbf{x},\sigma_D)I_y(\mathbf{x},\sigma_D) & I_y^2(\mathbf{x},\sigma_D) \end{bmatrix}$$

$$I_x(\mathbf{x},\sigma_D) = \frac{\partial}{\partial x} g(\sigma_D) * I(\mathbf{x})$$

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

$$\text{cornerness} = \det(M) - \lambda \operatorname{trace}(M)$$

Figure 25: Cornerness (R) Calculated from Moments Matrix (M)

From the moments matrix we achieve a few different autocorrelation convolutions of an image that describe this neighborhood.

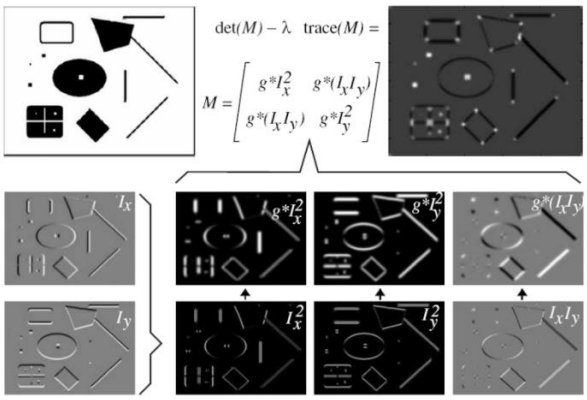


Figure 26: Autocorrelation Matrix Images

The eigen values can be plotted versus each other and the magnitude of the Harris response will be above 0 for corners less than 0 for edges and small or equal to zero for flat regions.

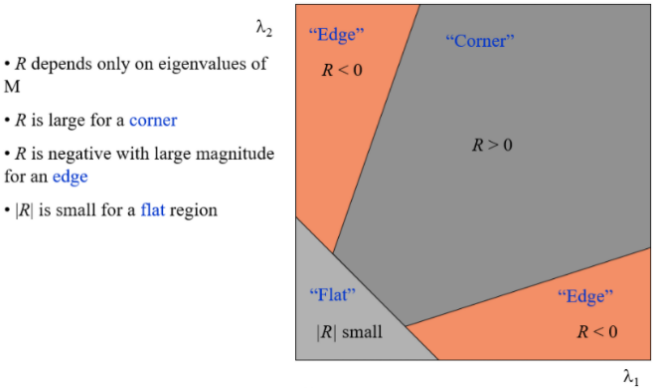


Figure 27: Eigen Value Map for Harris Response

```python
# Use Gaussian Blurring
kernelsize = (size, size)

Ixx = cv2.GaussianBlur(Ixx, kernelsize, Sigma)
Iyy = cv2.GaussianBlur(Iyy, kernelsize, Sigma)
Ixy = cv2.GaussianBlur(Ixy, kernelsize, Sigma)
Iyx = cv2.GaussianBlur(Iyx, kernelsize, Sigma)
# print(Ixy.shape, Iyx.shape)

R = np.zeros((w, h), np.float32)
# For every pixel find the corner strength
# i.e. compute the harris response R
for row in range(w):
  for col in range(h):
    '''
    M = | Ixx Ixy |
        | Iyx Iyy |
    '''
    M = np.array([[Ixx[row][col], Ixy[row][col]], [Iyx[row][col], Iyy[row][col]]])
    '''
    R = detM - k(traceM)^2
    '''
    R[row][col] = np.linalg.det(M) - (k * np.square(np.trace(M)))
```

Figure 28: Python Implementation of Harris Response Mathematics

Through experimenting with the eigen value mathematics it became evident that creating lists of which point pixels belonged to corners, edges and flat regions and performing local non-maximum suppression gave better results.

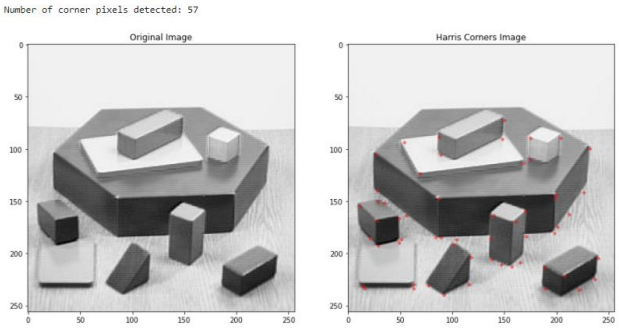


Figure 29: Corner Detection on Blocks Image

As much as we have achieved by implementing and detecting these corners. The implementation of the Harris corner detector algorithm needs to be robust to translation, rotation and scaling of this blocks image. So the following are various corner detections for different rotation and scaling of the blocks image.

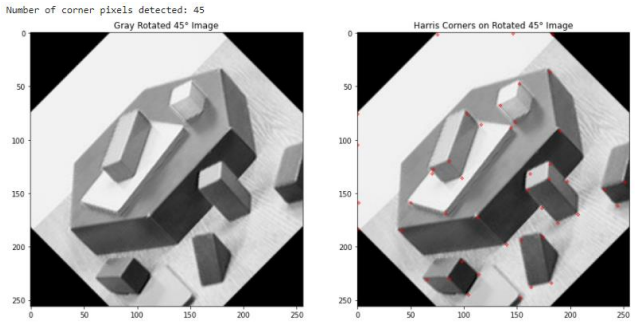


Figure 30: 45° Rotation and Corner Detection

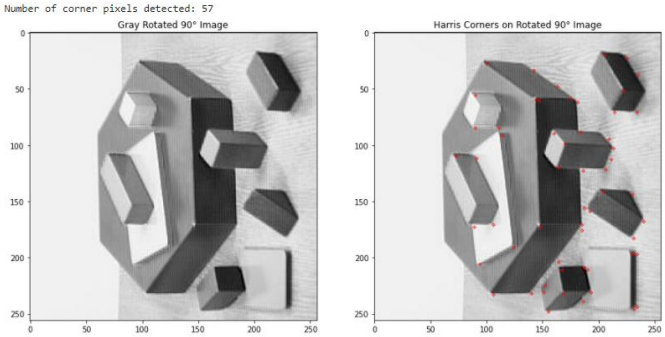


Figure 31: 90° Rotation and Corner Detection

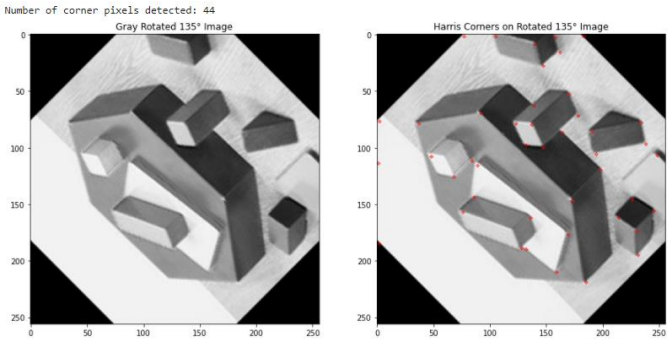


Figure 32: 135° Rotation and Corner Detection

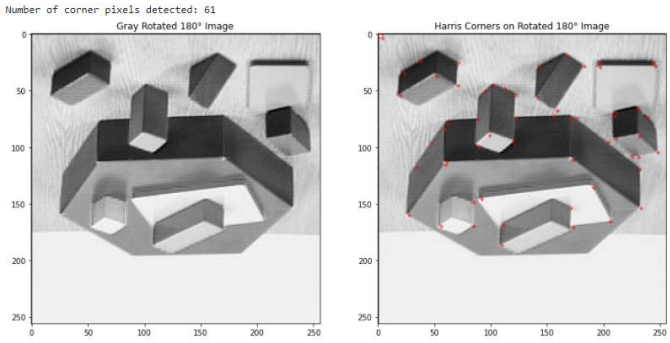


Figure 33: 180° Rotation and Corner Detection

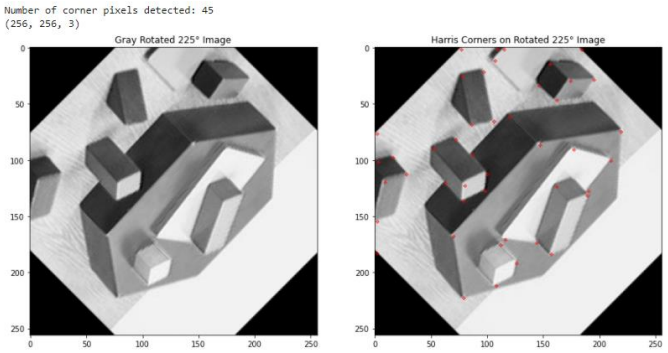


Figure 34: 225° Rotation and Corner Detection

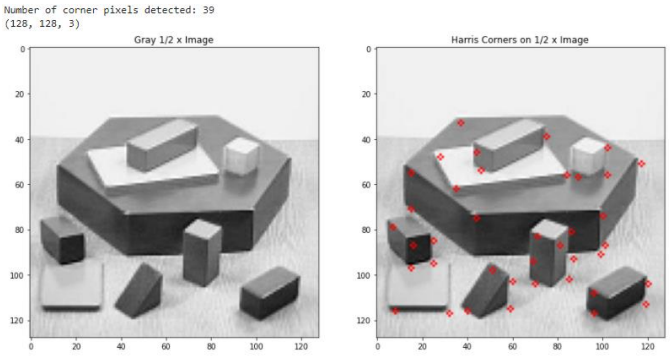


Figure 35: ½ Scaling and Corner Detection

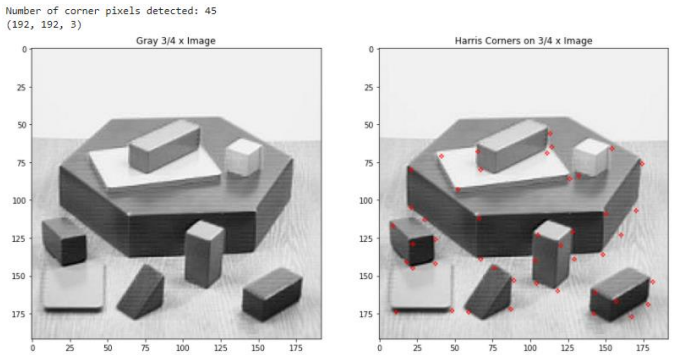


Figure 36: ¾ Scaling and Corner Detection

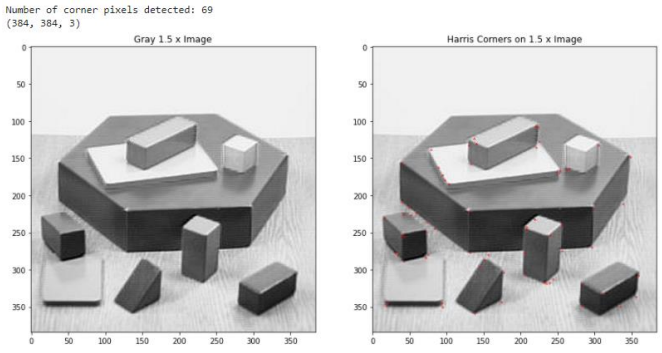


Figure 37: 1½ Scaling and Corner Detection

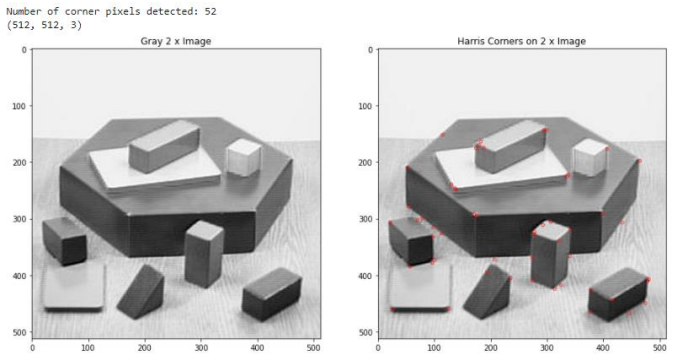


Figure 38: 2x Scaling and Corner Detection

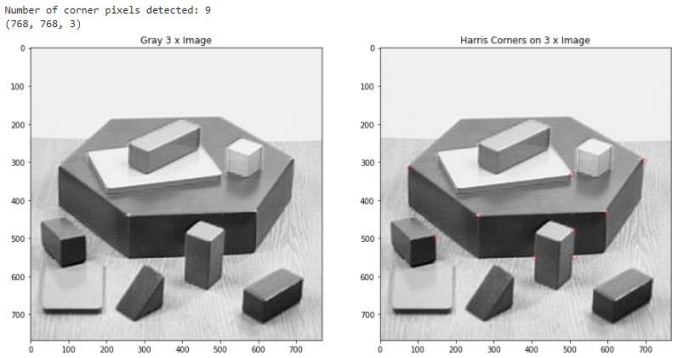


Figure 39: 3x Scaling and Corner Detection

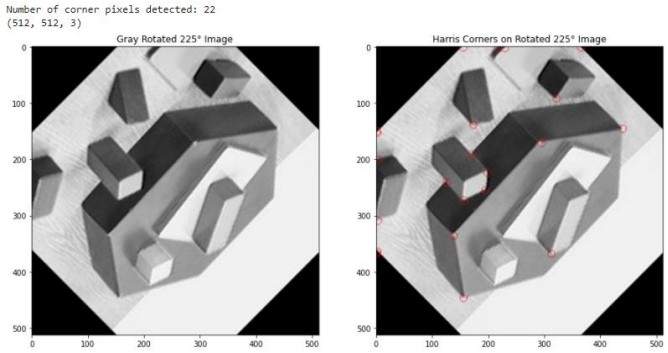


Figure 40: 225° Rotation with 2x Scaling and Corner Detection

From these 11 images we have proven that this implementation of the Harris Corner Detection Algorithm is robust to change (i.e., rotation, translation, and scaling).

Features from Scale-Invariant Keypoints.

III. PART 2: SIFT

The story so far has been Canny developed the edge detection, Harris developed the corner detection, and we have implementations for these two features that are robust to change in the image. Now we want to develop this even further. David Lowe developed and patented the SIFT (Scale-Invariant Feature Transform) in his paper Distinctive Image Features from Scale-Invariant Key points. The idea behind this SIFT algorithm is to scale an octave of gaussians and get a Difference of gaussians (DoG) between each octave. The DoG octaves achieve the Laplacian of a Gaussian (LoG) that was seen to detect the edges of an image very well. So just like with the Canny Edge detector and Harris Corner Detector we are developing a new method. For the David Lowe Scale-Invariant Feature Transform (SIFT) we have 4 key steps. The SIFT algorithm is shown as follows: * Please Note: David Lowe has patented his algorithm for the SIFT and so we should only be using tools available to us instead of trying to implement the algorithm ourselves. There are attempts at implementing David Lowe's algorithm online and they are quite good and accurate, but for the purpose of avaoiding any issues with the patent only openCV's module for the SIFT was used. So instead we will just analyze the algorithm and understand the implementation process. Also Note that the OpenCV source code is available and already covered the original color issue with the SIFT. Also, the time it takes to implement this algorithm was not available at our disposal so there were several factors that attributed to the decision to rely on another person's implementation rather than to implement the algorithm directly. Also Note the opencv module can default to a version where the SIFT function does not operate properly.

1. Scale-space extrema detection
2. Keypoint localization
3. Orientation assignment
4. Keypoint descriptor

Figure 41: David Lowe's SIFT Algorithm

1. Scale-Space Extrema Detection:

We define the scale space as follows:

Given a signal: $f : \mathbb{R}^N \rightarrow \mathbb{R}$

the scale-space representation $L : \mathbb{R}^N \times \mathbb{R}_+ \rightarrow \mathbb{R}$

is defined as: $L(\vec{x}, t) = g(\vec{x}, t) * f(\vec{x})$

where: $L(\vec{x}, 0) = f(\vec{x}) \forall \vec{x} \in \mathbb{R}^N$

and $g(\vec{x}, t)$ is the scale-space kernel

Figure 42: Scale-Space Mathematical Representation

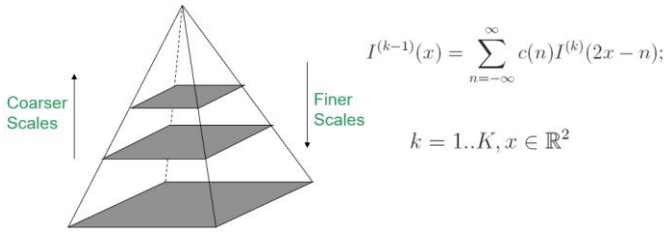


Figure 43: Mutli-Scale Pyramid Octave Structure

We build the scale space by blurring the image and creating and octave pyramid structure as shown above. Then

we can take the DoG of the scale space pyramid octave representation.

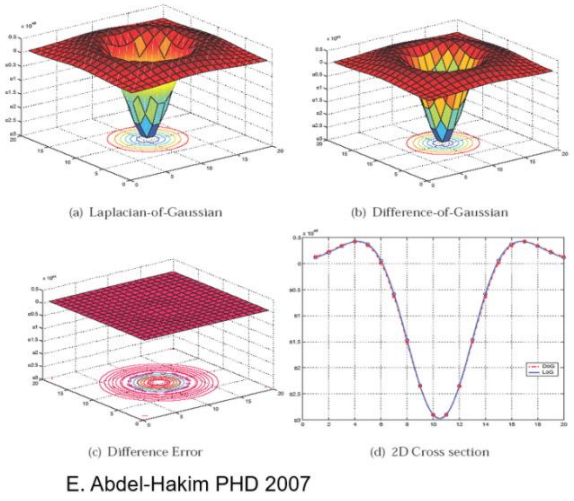


Figure 44: (a) Laplacian of a Gaussian (LoG) versus (b) Difference of a Gaussian (DoG)

As we can see from Figure 43 the Difference of Gaussian Octaves achieves the same curve as the Laplacian of a Gaussian.

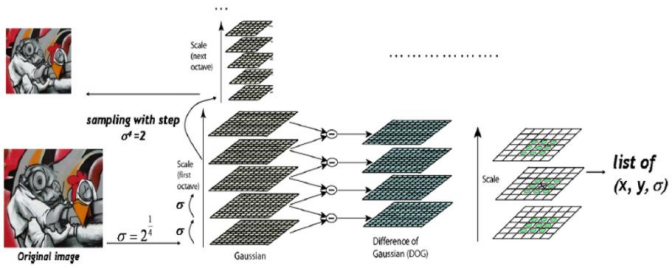


Figure 45: DoG from Gaussian Octave Scales

The reason we want to use the DoG to approximate the LoG is because the LoG requires more computational resources (i.e., storage space or execution time) than that of the DoG. The DoG is achieved by smoothed image subtraction:

$$D(x,y,\sigma) = \big(G(x,y,k\sigma) - G(x,y,\sigma)\big) * I(x,y)$$
$$= L(x,y,k\sigma) - L(x,y,\sigma)$$

Figure 46: DoG calculation Equivalence to LoG

Step 2: Keypoint Localization:

Fit keypoint at $\underline{x}$ to nearby data using quadratic approximation.

$$D(\underline{x}) = D + \frac{\partial D^T}{\partial \underline{x}}\underline{x} + \frac{1}{2}\underline{x}^T \frac{\partial^2 D^T}{\partial \underline{x}^2}\underline{x}$$

Differentiate and set to 0 then

$$\hat{\underline{x}} = -\frac{\partial^2 D}{\partial \underline{x}^2}^{-1} \frac{\partial D}{\partial \underline{x}}$$

Discard local minima

$$D(\hat{\underline{x}}) < 0.03$$

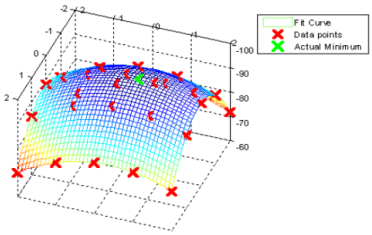


Figure 47: Low Contrast Point Elimination

We now use the calculated DoG to construct a 2nd order quadratic expression creating a hill where we can eliminate local minima or points of contrast from the DoG. Next we find the Hessian matrix of the DoG from the eigen vectors and eigen values and check the Harris Response (cornerness or R) and get rid of the Edges that are very sensitive to noise. Again developing the skills from edge detection and corner detection found in the Canny and Harris papers.

Step 3: Orientation Assignment:

We can now use the Sobel operator to find the gradient magnitude and orientation in a similar fashion where we convolve the Sobel operator kernel with the image using 2D convolution and assign weights to the local gradients.

Compute Gradient for each blurred image

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2}$$
$$\theta(x,y) = \tan^{-1}((L(x,y+1) - L(x,y-1))/(L(x+1,y) - L(x-1,y)))$$

Figure 48: Calculating Magnitude and Orientation of an Image

Step 4: Keypoint Descriptors:

The descriptor is a totally new data structure. Local descriptors are list like structures that contain the local magnitude and orientation of each keypoint in an image. The histogram is interpolated and smoothed with a gaussian window that has a standard deviation of half the window size.

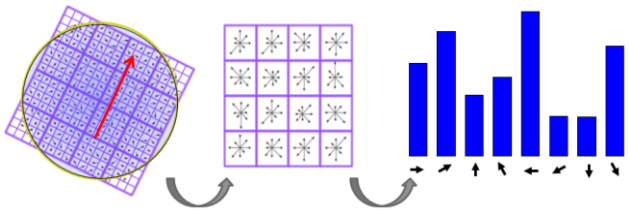


Figure 49: Local Keypoint Descriptors

Step 5: Keypoint Descriptor Matching:
Although it is painfully obvious we forgot about this step it is vital to the application of David Lowe's Algorithm. Kernel Nearest Neighbor (K-NN) can be used to match versus discard the keypoint descriptors in our data structure. Now the distance measure is an L2 distance. The difference between L1 and L2 is that distance just like moments have an Lp structure. So, an Lp distance is as follows:

where the l_p-$norm$ $||.||_p$ is defined by

$$||x||_p = (\sum_{i=1}^{n} |x_i|^p)^{\frac{1}{p}}$$

Figure 50: Lp Norm Distance Measure

The more common name for an L2 distance is the Euclidean distance. So we are performing 2-dimensional K-NN on each pixel neighborhood using Euclidean distance on each kernelled pixel.

However, L2 does not threshold very well, so we can use the ratio thresholding as done with the canny edge detector. Let's see how the SIFT algorithm performed on a set of image data. First, we structure the image data into a set of the images in order to perform the SIFT on the whole data set as needed.

Figure 51: Louisville Mosque Image Data Set

However, the features are too small to see without making the Figure large so we will have to display pages of the Features detected as follows:

Figure 52: Louisville Mosque 1 SIFT Feature Detection

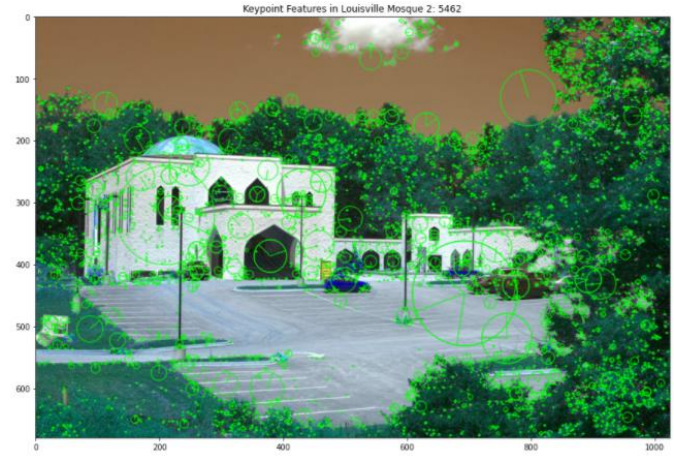


Figure 53: Louisville Mosque 2 SIFT Feature Detection

Figure 54: Louisville Mosque 3 SIFT Feature Detection

Figure 55: Louisville Mosque 4 SIFT Feature Detection

Figure 56: Louisville Mosque 5 SIFT Feature Detection

Figure 57: Louisville Mosque 6 SIFT Feature Detection

Figure 58: Louisville Mosque 7 SIFT Feature Detection

Figure 59: Louisville Mosque 8 SIFT Feature Detection

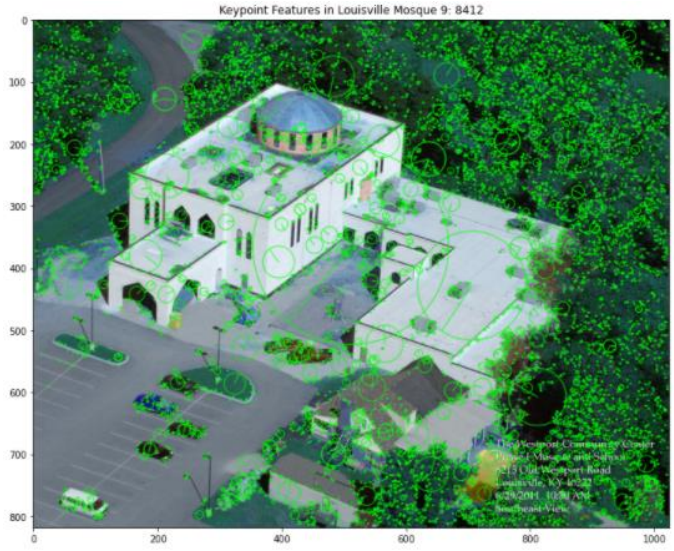


Figure 60: Louisville Mosque 9 SIFT Feature Detection

Figure 61: Louisville Mosque 10 SIFT Feature Detection

Figure 62: Louisville Mosque 11 SIFT Feature Detection

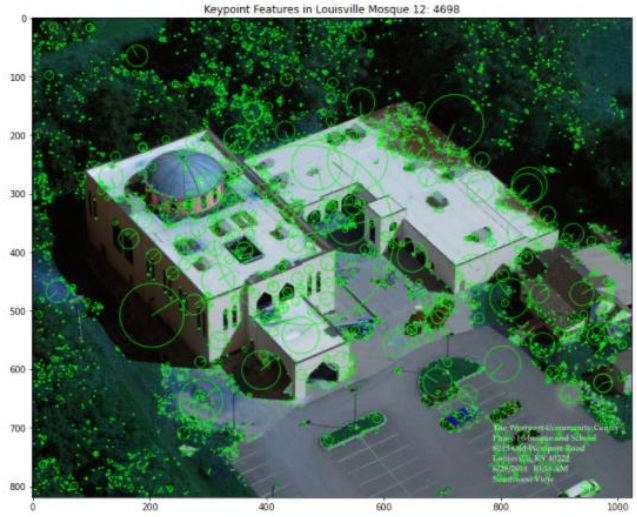


Figure 63: Louisville Mosque 12 SIFT Feature Detection

Now that we have found the features between 5000 to 11000 features from each image, we can use David Lowe's ratio test to determine good features. And then time the detection and matching of each image. To save time we can just shown this process for Image 1 to Image 2, Image 3 to Image 4, Image 5 to Image 6, Ect…

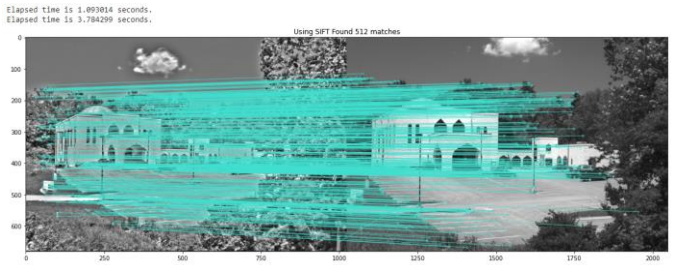


Figure 64: Louisville Mosque 1 and 2 SIFT matching Found 512 Matches, Detection time: 1.093014 seconds, and Matching Time: 3.784299 seconds

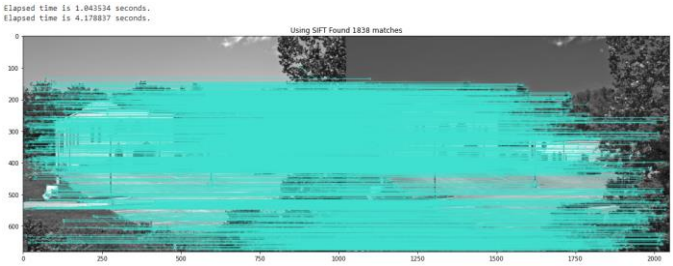


Figure 65: Louisville Mosque 3 and 4 SIFT matching Found 1838 Matches, Detection time: 1.043534 seconds, and Matching Time: 4.178837 seconds

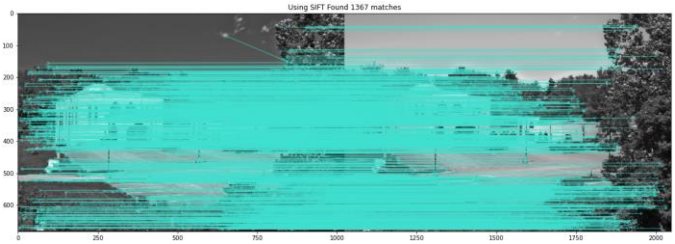


Figure 66: Louisville Mosque 5 and 6 SIFT matching Found 1367 Matches, Detection time: 0.942934 seconds, and Matching Time: 2.919465 seconds

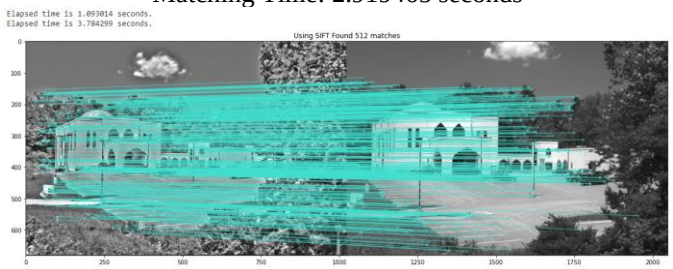


Figure 67: Louisville Mosque 7 and 8 SIFT matching Found 3200 Matches, Detection time: 1.024387 seconds, and Matching Time: 4.473762 seconds

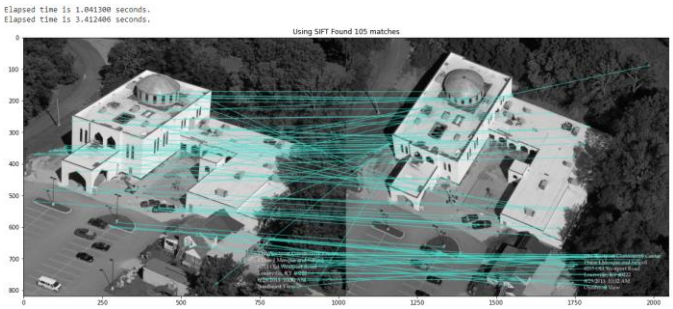


Figure 68: Louisville Mosque 9 and 10 SIFT matching Found 105 Matches, Detection time: 1.041300 seconds, and Matching Time: 3.412406 seconds

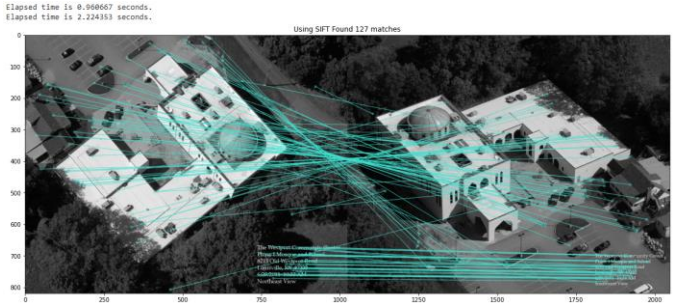


Figure 69: Louisville Mosque 11 and 12 SIFT matching Found 127 Matches, Detection time: 0.960667 seconds, and Matching Time: 2.224353 seconds

Now, Since there can be a lot of false matches with the SIFT algorithm; the RANSAC algorithm modification to the SIFT was so that outlying keypoint descriptors could be eliminated and only the inlying keypoint descriptors were matched between images. To see the difference with the RANSAC we can replot all the previous matched figures and see the difference in time and number of matches. The timing function was imported using a module called pytictoc which essential functions the exact same way that tic toc functions in MATLAB. Another method is to get system operating times using the sys module.

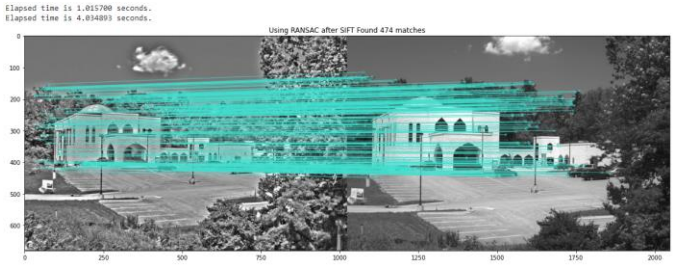


Figure 70: Louisville Mosque 1 and 2 RANSAC after SIFT Found 474 Matches, Detection time: 1.049232 seconds, and Matching Time: 3.840096 seconds

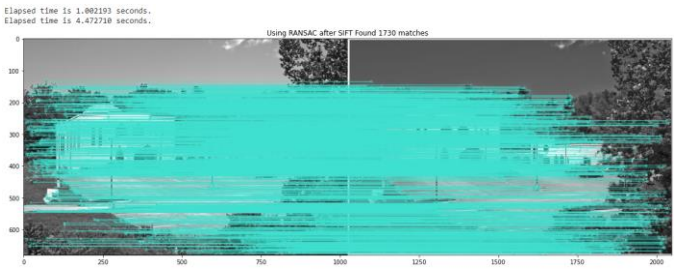


Figure 71: Louisville Mosque 3 and 4 RANSAC after SIFT Found 1730 Matches, Detection time: 1.002193 seconds, and Matching Time: 4.472710 seconds

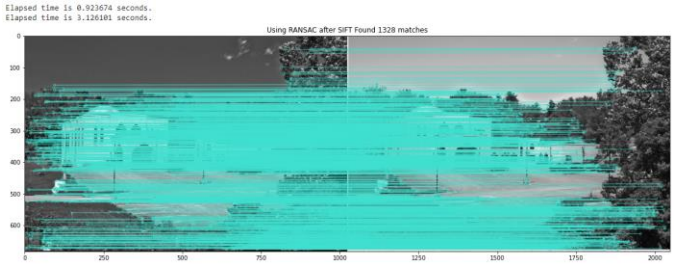


Figure 72: Louisville Mosque 5 and 6 RANSAC after SIFT Found 1328 Matches, Detection time: 0.923674 seconds, and Matching Time: 3.126101 seconds

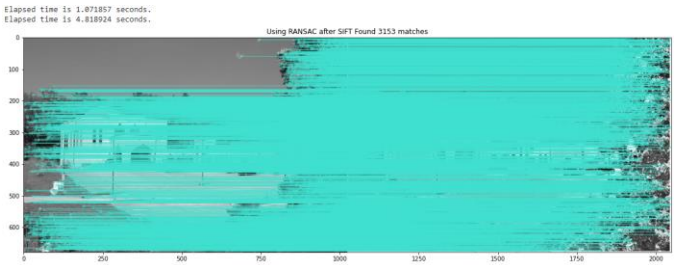


Figure 73: Louisville Mosque 7 and 8 RANSAC after SIFT Found 3153 Matches, Detection time: 1.071857 seconds, and Matching Time: 4.818924 seconds

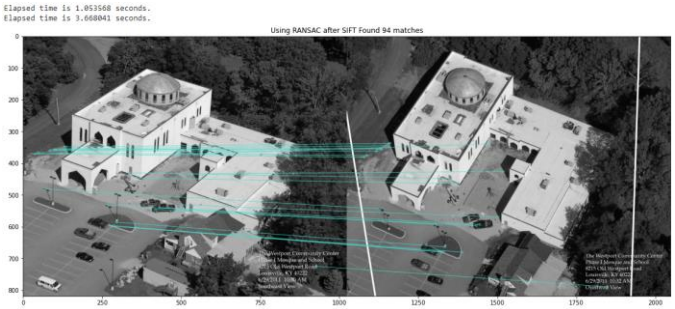


Figure 74: Louisville Mosque 9 and 10 RANSAC after SIFT Found 94 Matches, Detection time: 1.053568 seconds, and Matching Time: 3.668041 seconds

Figure 75: Louisville Mosque 11 and 12 RANSAC after SIFT Found 118 Matches, Detection time: 0.947795 seconds, and Matching Time: 2.322261 seconds

IV. PART 3: CSIFT (COLOR SIFT)

So, for the CSIFT we want the added importance of color to be useful to distinct between objects. Now opencv does not have a CSIFT, but we can evaluate the performance using colored images in MATLAB at least from the provided function and images. The added Red, Green and Blue (RGB) channels can give a more refined feature descriptor for our object-oriented programming (i.e., OOP). A solution to color invariance is the Kubelka-Munk theory which models the photometric reflection in an image by:

$$E(\lambda,\vec{x}) = e(\lambda,\vec{x})(1-\rho_f(\vec{x}))^2 R_\infty(\lambda,\vec{x}) + e(\lambda,\vec{x})\rho_f(\vec{x})$$

λ: The wavelength
e : The illumination spectrum
ρ_f : The Fresnel reflectance
R_∞: The material reflectivity
E : The reflected spectrum in the viewing direction

Figure 76: Kubelka-Munk Model for Color Invariance

$$E(\lambda,\vec{x}) = i(\vec{x})[\rho_f(\vec{x}) + (1-\rho_f(\vec{x}))^2 R_\infty(\lambda,\vec{x})]$$

By differentiating with respect to λ:

$$\left.\begin{aligned} E_\lambda &= i(\vec{x})(1-\rho_f(\vec{x}))^2 \frac{\partial R_\infty(\lambda,\vec{x})}{\partial\lambda} \\ E_{\lambda\lambda} &= i(\vec{x})(1-\rho_f(\vec{x}))^2 \frac{\partial^2 R_\infty(\lambda,\vec{x})}{\partial\lambda^2} \end{aligned}\right\} H = \left(\frac{E_\lambda}{E_{\lambda\lambda}}\right) = \frac{\partial R_\infty(\lambda,\vec{x})}{\partial\lambda} \Big/ \frac{\partial^2 R_\infty(\lambda,\vec{x})}{\partial\lambda^2} = f(R_\infty(\lambda,\vec{x}))$$

Independent of viewpoint, surface orientation, illumination direction, intensity, and Fresnel reflectance coefficient.

Figure 77: Kubelka-Munk Model for Color Invariance Cont.

$$C = \left(\frac{E_\lambda}{E}\right) = \frac{1}{R_\infty(\lambda_{,\mathbf{x}})}\frac{\partial R_\infty(\lambda_{,\mathbf{x}})}{\partial\lambda}$$

Independent of viewpoint, surface orientation, illumination direction, intensity.

$$W_x = \left(\frac{E_\mathbf{x}}{E}\right) = \frac{1}{R_\infty(\lambda_{,\mathbf{x}})}\frac{\partial R_\infty(\lambda_{,\mathbf{x}})}{\partial\mathbf{x}}$$

Independent of illumination intensity.

$$N_{\lambda x} = \left(\frac{E_{\lambda x}E - E_\lambda E_x}{E^2}\right)$$

Independent of viewpoint, surface orientation, illumination direction, intensity, and illumination color.

Figure 78: Kubelka-Munk Model for Color Invariance Cont.

$$ang^O_\mathbf{x} = \tan^{-1}\left(\frac{\sqrt{3}(R_\mathbf{x}-G_\mathbf{x})}{R_\mathbf{x}+G_\mathbf{x}-2B_\mathbf{x}}\right),$$

$$ang^S_\mathbf{x} = \tan^{-1}\left(\frac{(G_\mathbf{x}R - R_\mathbf{x}G)(\sqrt{R^2+G^2+B^2})}{R_\mathbf{x}RB + G_\mathbf{x}GB - B_\mathbf{x}(R^2+G^2)}\right).$$

Specularities, edge sharpness, and lighting geometry invariants.

Vand De Weijer and Schmid (2006)

$$hue = \tan^{-1}\frac{O_1}{O_2} \qquad O_1 = \frac{R-G}{\sqrt{2}}, \quad O_2 = \frac{R+G-2B}{\sqrt{6}}$$

Figure 79: Kubelka-Munk Model for Color Invariance Cont.

Repeatability $\quad r(\varepsilon) = \dfrac{|R(\varepsilon)|}{\min(|\mathcal{IP}_1|,|\mathcal{IP}_2|)}$

Recall $\quad Recall = \dfrac{|M_c|}{\min(|\mathcal{IP}_1|,|\mathcal{IP}_2|)}$

Precision $\quad Precision = \dfrac{|M_c|}{|M_c|+|M_f|}$

Figure 80: Descriptor Matching Evaluation Criteria

Now that we understand the mathematics behind the CSIFT; let's use it on a few images. The .ppm file will not be read by any software available to us on a standard computer so we can still use our Ipython Notebook to at least visualize the images we are working on. From the COLSIFT function we are using in MATLAB we get a .keys file that gives us the descriptors data structure. We could utilize these descriptors data to do the matching between the images in our Ipython Notebook but given the time we have available to us we should just use the MATLAB implementation to speed up our analysis. The descriptors .key file can be opened and viewed with notepad++ or any text editor:

Figure 81: .key file for Mosque

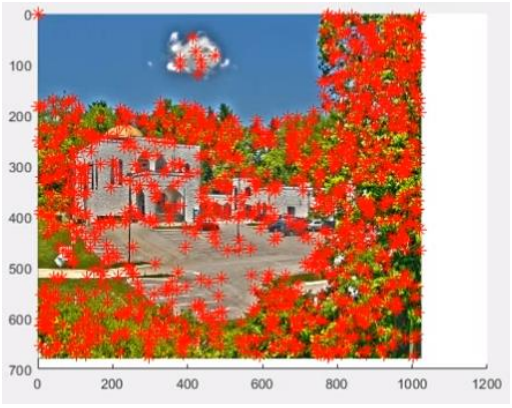
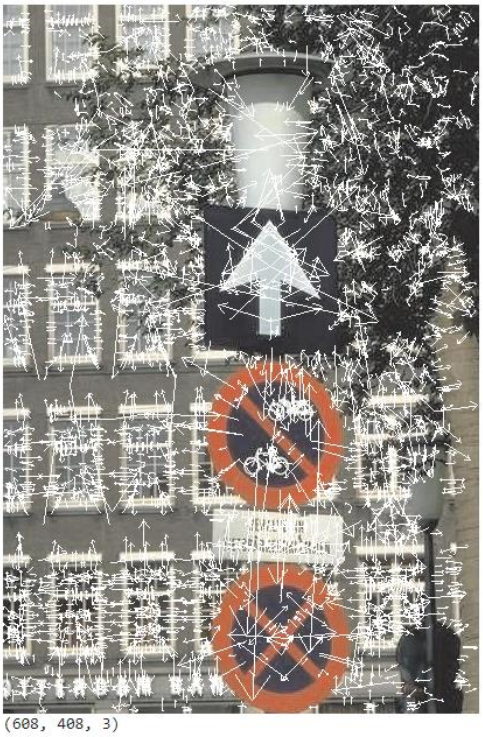


Figure 82: keypoint painting on Louisville Mosque using CSFIT

(608, 408, 3)

Figure 83: abn.ppm: Street Signs



(608, 408, 3)

Figure 84: resabncol.ppm: Street Signs with Color Keys

The color keys are vectors of all the features that we can then match and so the keypoints can be plotted over the image. As shown in Figure 82.

After matching we achieve something as shown in Figure 85

Figure 85: Louisville Mosque CSIFT Matching

And after the RANSAC we reject the outliers and only match the inliers just as before.

Figure 86: Louisville Mosque CSIFT with RANSAC Matching

V. Conclusions

From the beginning we wanted to build a detector. We started with John Canny's Edge Detector step by step we implemented his algorithm into the well-known Canny Image of Lenna utilizing the Laplacian of a Gaussian (LoG). Then we wanted to detect the corners and so we developed the fundamental mathematics with Chris Harris and Mike Stephens algorithm for detecting corners, edges, and flat regions in the eigen space of the image. The Harris Response (cornerness, or R) was used to define which region either corners or edges or flat the pixels in the image belonged. From this we proved that the Harris corner detector was robust to image changes (i.e., rotation, translation, and scaling). From this we moved onto David Lowe's SIFT algorithm where we developed an understanding that we could approximate the LoG with an octave pyramid structured Difference of Gaussians (DoG). We analyzed the techniques, compared the detection and matching times for the 12 Louisville Mosque Image Data Set and did the same for the 12 images when we used the RANSAC for rejecting outliers. Finally, we discovered an extra bit of information could be utilized from the RGB three channels to analyze these images and obtain more features. And from this we used the RANSAC to eliminate outliers and achieve an extremely robust algorithm for object detection via the extraction of features from the images gathered. It should also be stated that these algorithms extend into real-time video object detection.

VI. References

[1] CANNY, JOHN. "A Computational Approach to Edge Detection." *Readings in Computer Vision*, 1987, pp. 184–203., doi:10.1016/b978-0-08-051581-6.50024-6.

[2] Farag, Introduction to Probability Theory with Engineering Applications, Cognella Academic Publishing, 2021.

[3] Farag, Biomedical Image Analysis Statistical and Variational Approaches.

[4] Golub, Gene Howard, and Van Loan Charles F. *Matrix Computations.* The Johns Hopkins University Press, 2013.

[5] Harris, C., and M. Stephens. "A Combined Corner and Edge Detector." *Procedings of the Alvey Vision Conference 1988*, 1988, doi:10.5244/c.2.23.

[6] Lowe, David G. "Distinctive Image Features from Scale-Invariant Keypoints." *International Journal of Computer Vision*, vol. 60, no. 2, 2004, pp. 91–110., doi:10.1023/b:visi.0000029664.99615.94.

[7] Reuven Rubinstein and Dirk P. Kroese Simulation and the Monte Carlo Method, 3rd Edition, Wiley, New Jersey, 2017

[8] R. O. Duda, P. E. Hart and D. G. Stork: Pattern Classification, 2nd Edition, Wiley, 2000.

VII. Appendix: IPython Code:

The IPython Notebook is available at: https://github.com/MichaelFerko