Face Detection via Generalized Hough Transform in Python

Introduction to Computer Vision - University of Nebraska

Zachary Swanson
April 12, 2024

Contents

1 Introduction

A common objective in computer vision applications is object detection. Like other fundamental computer vision algorithms, object detection is often a necessary component of a larger objective. This chain of dependency may be illustrated, in a simplified sense, for facial recognition where the objective is to classify the face(s) in an image with respect to a gallery of known facial identities. Facial recognition depends on detecting the shape and location of the face in the image, i.e. face detection. Furthermore, face detection depends on finding edges and other identifying information in the image, i.e. edge detection and other filtering techniques. Hence, this experiment provides the opportunity to further develop and understand foundational computer vision concepts in pursuit of more advanced algorithms and objectives.

In particular, this experiment will explore the generalized Hough transform for the purpose of face detection. The Hough transform is a parametric transform that takes advantage how a shape is interpreted, i.e. with respect to variables versus parameters. A two-dimensional line, for example, is defined by variables x and y and parameters of slope and intercept. The general concept is such that points in the image space correspond to curves in the transform space and vice-versa. Thus, a voting mechanism arises because a point of greatest intensity (greatest curve overlap) in the parameter space corresponds to the location of the shape in the image space. The generalized Hough transform extends this concept to arbitrary shapes provided a set of reference images to build a representation of the shape to be used in the transform. Thus the generalized Hough transform does not require an explicit equation to define and detect arbitrary shapes.

The primary goal of this experiment is to implement a functional generalized Hough transform based face detection engine using a modern programming language, i.e. Python. Additionally, this experiment aims to explore the effects of scale, rotation, and noise on the ability of the Hough transform to detect faces. Other parameters were also explored to achieve optimal face detection including number of quantization levels for gradient direction and the upper and lower thresholds for the Canny edge detector.

2 Background

2.1 Generalized Hough Transform

The Hough transform is a parametric transform and a concise definition of a parametric transform is provided in [1]:

> Given a point in $\mathbb{R}^d$, and a parameterized expression defining a curve in that space, the parametric transform of that point is the curve that results from treating the point as a constant and the parameters as variables.

A simple line in $\mathbb{R}^2$ is illustrative. Consider the line $y = ax + b = 3x + 1$ and two points along the line, $(1, 4)$ and $(3, 10)$. These points would correspond to the lines $b = a - 4$ and $b = 3a - 10$ in the parameter space. Furthermore, these lines in parameter space would intersect at the point $(3, 1)$, which are the parameters of the original line in the image space. Hence, it is evident that a curve in the image space corresponds to a point in the parameter space and vice-versa. Furthermore, this relationship allows points in the image space to vote for the shapes parameters in the parameter space. This is evident from the line example where parameter curves of two co-linear points intersect at parameters of the original, image space

line. This provides a certain level of robustness against noise in the image. Provided enough "true" edge pixel in an image, the votes of the true edges will outweigh the votes of the noisy pixels to determine the shape.

The line example assumes that the shape is represented by an analytic function, but more complex shapes, like faces, are difficult to describe by such functions. The generalized Hough transform allows the voting mechanism described above to be extended to arbitrary shapes. This is illustrated by a simplified example in Figure 1.

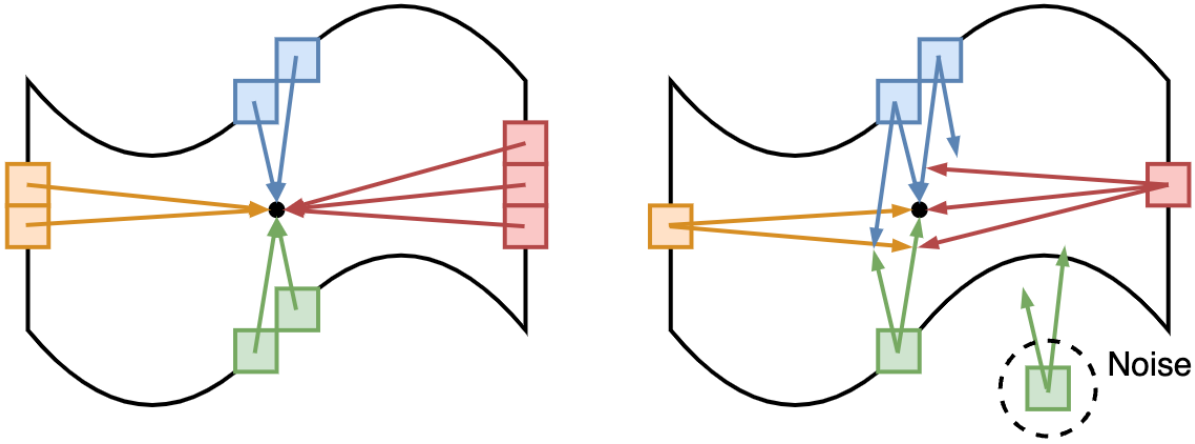

Figure 1: Simple example demonstrating a) how the R-table is constructed (left) from displacement vectors to the object reference point in a reference image for each gradient direction bin symbolized by the colors blue, red, green and orange and b) how the r-table is used to detect shape and shape location in a query image using the R-table (right).

An arbitrary curve is represented by the black line in Figure 1. The same shape is used for simplicity to represent the reference image (left) and the query image (right). Starting with the reference image (left), nine pixels are depicted by colored boxes along the curve. The color indicates the quantized gradient direction. A reference point is selected at the relative center of the shape. For every point (i.e. pixel) along the curve, a displacement vector is calculated from the pixel to the reference point and these vectors are grouped according to the quantized gradient direction. Thus, the shape is represented by the curve characteristic (gradient direction) and radius (displacement vector) at each point. The collection of displacement vectors grouped by gradient direction is referred to as an R-table. Additionally, this process may be repeated for a number of reference images to build a more robust representation.

Moving on to the query image (right) in Figure 1, six pixels are represented for simplicity: five true edge pixels and one noise pixel. As the colors indicate, the quantized gradient directions are again calculated and are necessary because they indicate which displacement vectors should be applied at that pixel. Consider the red pixels, for example. In the reference image (left), the three red vectors originate with their respective pixel and terminate at the reference point. In the query image, all three red vectors originate at the red pixel and the tip of each vector corresponds to a "vote" in the vote space. Thus, there will be votes in the vote space that do not correspond to the location of the shape, but the location with the most votes should correspond to the location of the shape in the image space. This is illustrated in Figure 1. There are a two locations that received two votes and four locations that received one vote, but there are five votes at the true shape location. Figure 1 also illustrates how the effects of noise are mitigated by this algorithm. The noisy pixel in Figure 1 added two votes but these votes

were outweighed by the votes of the true edge pixels.

In summary, the generalized Hough transform uses reference shapes (images) with a consistent reference point to build an R-table. The R-table represents the shape based the displacement vectors of the edge pixels to the reference point grouped by quantized gradient direction. For a query image, the gradient direction of the edge pixels is calculated and used as a lookup in R-table. Thus, each edge pixel gets to cast a number of votes based on the associated vector for its gradient direction. These votes are accumulated for each edge pixel in the query image. The point or group of points with greatest number of votes corresponds to the location of the shape in query image.

2.2 Face Detection

The generalized Hough transform described in Section 2.1 may be applied to face detection. Images containing faces are used as reference images to build an R-table that represents the general shape of the human face. Note that there is a lot of variability in shape and features of human faces. Thus, it is important that the reference images are scaled and aligned as best as possible in the reference images to help build a robust R-table. This face-based R-table may then be applied to query face images and the peak accumulated votes should correspond to location of the face in the image.

The reference and query images used for this experiment are consistent: similar pose, similar expression, similar lighting, etc. A thorough survey of the face detection space is provided in [2] and this references the "Wider-Face" face detection data set. This data set provides a range of images with variation factors that include scale, pose, occlusion, expression, makeup, and illumination. It may be assumed that the generalized Hough transform approach used in this experiment would not effectively detect faces across this wide-range of variations (though it does perform well within specific criteria). In fact, parametric transforms are not mentioned in this survey's description of classical face detection approaches. Haar Cascades and Histogram of Oriented Gradients (HOG) are classical approaches listed and the survey indicates that the majority of recent literature in face detection have focused on the use of deep learning-based approaches for face detection.

3 Methodology

3.1 Python Implementation

Figure 2 provides a high-level diagram of the face detection system implemented Python. The functionality of the system may be divided into two sub-components indicated by the top and bottom rows of the flow diagram in Figure 2.

It may be observed that the top and bottom rows both include Canny edge detector (orange) and gradient direction (blue) blocks. The OpenCV implementation of the Canny edge detector was used for efficient edge detection. This produces a binary image where the edges are represented by white. The gradient direction of each pixel was computed by applying the 3x3 x- and y-Sobel kernels to the reference or query image and then computing $tan^{-1}(f_y/f_x)$ for each pixel, where f_y is the derivative in the y-direction and f_x is the derivative in the x-direction. The $arctan2$ operator was used to ensure proper results. The gradient directions were further quantized into a user-specified number of bins. This quantization improves the lookup capabilities of the R-table.

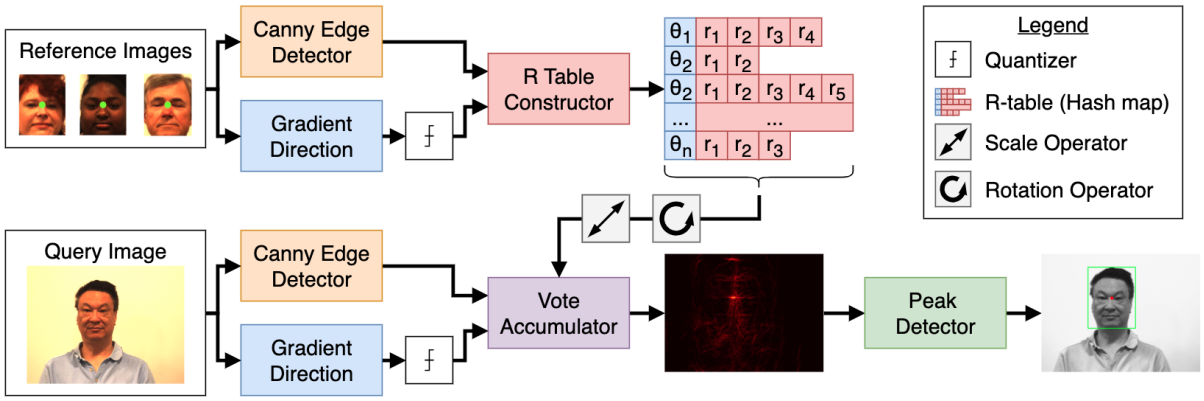

Figure 2: High-level system diagram of the face detector implementation in Python using the generalized Hough transform.

The top row of Figure 2 demonstrates how the R-table was constructed from a set of reference images. The R-table was implemented as a Python dictionary which stores key-value pairs. The key was the quantized gradient direction and the value was a list of displacement vectors. Python dictionaries are an abstracted interface for a hash table implementation. Hash tables provide an efficient mechanism for storing and accessing data and are optimal for the R-table implementation.

Continuing with the top row, the reference images were processed sequentially and the results of each reference image were added to the R-table cumulatively. The R-table constructor (red block) took as inputs a binary edge image and a quantized gradient direction image generated by passing the reference image through the respective orange and blue blocks in Figure 2. The binary edge image was raster scanned, evaluating each pixel in the image. If the pixel was an edge (i.e white), then the R-table performed two actions. First, it retrieved the quantized gradient angle from the corresponding pixel in the gradient direction image. The retrieved gradient direction was used as the key for the R-table dictionary. Second, the displacement vector from the current pixel to the reference pixel was calculated as

$$\vec{r} = ((x_{ref} - x_{edge}), (y_{ref} - y_{edge})), \tag{1}$$

where x and y correspond with column index and row index, respectively. The reference point is indicated by the green dot in the center of the faces in the reference images in Figure 2. If the gradient direction did not exist in the dictionary, an entry was added and a new displacement vector list was started with the current displacement vector. Otherwise, the displacement vector was appended to the existing list associated with the gradient direction entry.

The bottom row of Figure 2 demonstrates how a query image was processed and how the R-table was applied to determine the face location. Again, the Canny edge detector was used to determine edges and quantized gradient direction was calculated for every pixel. These were passed the accumulator (purple) block along with the R-table dictionary created previously. At every edge pixel, the corresponding gradient direction was used to perform a lookup in the R-table dictionary. For every displacement vector returned for that gradient direction, a vote was added at the pixel corresponding to the current edge pixel plus the displacement vector as demonstrated in Section 2.1. After accumulating the votes for every edge pixel, the accumulator produced a vote intensity image, shown between the accumulator and peak detector blocks in Figure 2. Finally, the peak detector (green) block found the maximum peak and produced a

bounding box centered on the maximum peak as shown in the bottom right image of Figure 2.

It is worth noting that some experimentation was done using K-means clustering for the peak detector. However, acceptable results were achieved by tuning other parameters and the peak detector was implemented such that it selected the pixel with the maximum number of votes.

It may also be observed that there are scaling and rotation operators on the line connecting the R-table to the vote accumulator block in Figure 2. These were necessary to experiment with the effects of scale and rotation of the detected shape. These operators may be modeled as

$$\begin{aligned} x' &= s * (x\cos\theta - y\sin\theta) \\ y' &= s * (x\sin\theta - y\cos\theta), \end{aligned} \tag{2}$$

where s is the scaling factor θ is the rotation factor. Alternatively, this may be viewed in terms of row indexes, as was the case for the implementation, where x corresponds to columns and y corresponds to rows.

3.2 Evaluation

Qualitative evaluation was used initially to achieve a working prototype. The reference images were used as the query images in this initial phase because they did not include extraneous edges like the actual test images, which may skew the maximum peak away from the true face center.

After achieving a working prototype, a "ground truth" point was arbitrarily selected on each of the test images near the bridge of the noise. The bridge of the nose was selected because this was reference point in the reference images. The ground truth point was used to compute the Euclidean distance between the ground truth point the maximum accumulated peak given by

$$||\boldsymbol{x}\hat{\boldsymbol{x}}|| = \sqrt{(x^2 - \hat{x}^2) + (y^2 - \hat{y}^2)}, \tag{3}$$

where x, y are the ground truth coordinates and $\hat{x}, \hat{y}$ are the predicted coordinates.

The Euclidean distance in Equation 3 allowed for various parameter optimizations to be performed by finding the parameters that minimized the Euclidean distance. This included finding the optimal combination of the number of gradient direction quantization bins, the lower Canny edge detector threshold, and the upper Canny edge detector threshold. The average Euclidean distance across the three test image was used in this case. Additionally, the optimal scaling and rotation for each test image was determined by tracking the value combination that minimized the Euclidean distance for each image.

The effect of noise in the reference images and in the query image was also studied. For this experiment, additive white Gaussian noise (AWGN) was applied to the images before passing them to their respective blocks of the face detector. The Euclidean distance in Equation 3 provided a metric for evaluating the effect of increasing noise.

4 Results

4.1 Qualitative Results

Figure 3 provides visualizations of the R-tables generated from the reference images using different parameters for the gradient direction quantizer and the Canny edge detector. The

general shape of the face and facial features (mouth, eyes, nose, brows, etc.) are observable in both images. The image on the right had higher threshold for the Canny edge detector and fewer edges were detected as a result. This explains the lack of some facial features when compared to the image on the left. However, the lower threshold of the left image permitted additional edges like wrinkles and hair that don't necessarily benefit detection of a general human face. These extraneous edges provide a level of noise that distract from the most important shape and facial features. As will be discussed further in Section 4.2, the parameters that produces the R-table on the right provided the best results across the three test images.

Figure 3: Visualizations of the R-tables generated from the three reference images with different number of gradient direction bins and low and high thresholds for the Canny edge detector. These parameters were (4, 20, 150) for the left and (32, 100, 200) for the right.

Figure 4 provides test images with bounding boxes indicating the results of the face detection implementation. The results were achieved after optimizing the gradient direction quantizer and the Canny edge detector parameters. Furthermore, there was a difference in scale and rotation when compared to the reference images, as will be discussed in Section 4.2. These results were achieved by applying the optimal rotation and scale factor for each test image. Qualitatively, the results in Figure 4 indicate a functional Python implementation of face detection using the generalized Hough transform.

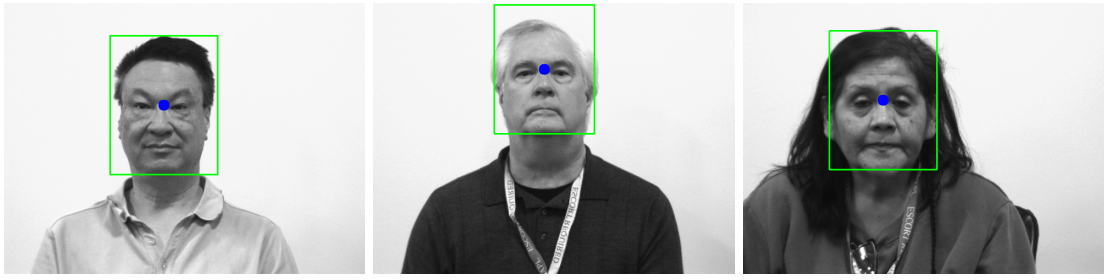

Figure 4: Best results achieved by face detection implementation after optimizing parameters and finding appropriate scale and rotation factors for each image.

4.2 Quantitative Results

The initial goal was to optimize the number of bins for the gradient direction and the lower and upper thresholds for the Canny edge detector. This was achieved by applying a number of combinations of bin numbers and threshold to find the combination that minimized the Euclidian distance from the predicted point to the manually determined ground truth point. In total, 195 combinations were tested with number of bins ranging from four to 64, lower threshold from 20 to 150, and upper threshold from 100 to 230. The truncated results of the optimization are provided in Table 1.

Table 1: Results of parameter optimization with parameters listed as (number of bins, lower threshold, upper threshold). Error corresponds to the Euclidean distance.

Parameters	**Per Image Error**	**Average Error**
(32, 100, 200)	[75.802, 46.648, 54.918]	59.123
(8, 100, 200)	[75.802, 70.214, 54.918]	66.978
(32, 50, 230)	[76.322, 72.139, 54.918]	67.793
...	...	...
(16, 150, 230)	[75.802, 387.737, 54.918]	172.819
(4, 50, 150)	[154.797, 224.404, 149.482]	176.228
(4, 20, 150)	[170.074, 223.486, 149.482]	181.014

The results in Table 1 indicate that a generally higher edge detection threshold produced the best results. The evidence for the number of quantization bins is less clear; however, the results seem to indicate the eight or more bins seem to produce better results. Figure 5 demonstrates the results when using the optimal parameters (top) and the worst-case parameters (bottom). The best parameters still haven't located the desired point (e.g. the bridge of the noise) but has detected the face based on the bounding boxes.

Figure 5: Detection results for optimized parameters (top) and worst parameters (bottom).

It is also noted that the worst-case parameters were also more computationally expensive that the best parameters. The best parameters could detect the face in approximately two

seconds. The worst-case parameters resulted in face detection that took more that one minute. This is due to the worst-case parameters detecting many edge points with the lower thresholds as demonstrated in Figure 3. Hence, the R-table is considerably larger and the larger R-table is applied to more detected edges in the query image. This multiplicative factor results in significantly worse performance with the lower thresholds.

The best results in Figure 5 have detected faces but have not located the face at the optimal location in the image with respect to the reference point used, i.e. the bridge of the nose. The cause was hypothesized to be variations of scale and rotation between the reference images and the query images. A rough estimate of the scale difference was determined to be roughly 0.5 from reference to query. This was determined by manually measuring pixels in the images. Thus, for each query image, a scale factor was applied ranging from 0.5 to 1.0 in increments of 0.05. The faces did not appear to be rotated significantly, but a rotation factor was also applied from $-5°$ to $+5°$ in increments of $0.5°$. This resulted in 231 scale-rotation combinations for each test image. The optimal results are shown in Table 2 and the associated face detection results are shown in Figure 4.

Table 2: Best results of scale-rotation optimization. Error corresponds to Euclidean distance.

Test Image	**Scale**	**Rotation**	**Error**
001	0.7	-0.5°	1.0
002	0.65	-4.0°	1.0
003	0.7	-1.5°	4.0

Figure 6 provides a comparison of the images produced by visualizing the accumulated votes for test image 001 with the optimal scale and rotation provided in Table 2 (right) and without (left). There's a clear peak in the right image with optimal scale and rotation. There appear to be peaks in the left image but they seem to be distributed around the desired peak. This seems to indicate that the vectors in the R-table are overshooting the mark, such that there is a relatively equal accumulation of votes on both sides, equidistant from the ground truth. This matches expectations because the face in the test image was relatively smaller (in terms of pixels) than the faces in the reference images. A similar pattern is expected if the query face was scaled larger. In such a case, the R-table vectors would undershoot the desired location but still create a sort of distribution around the location.

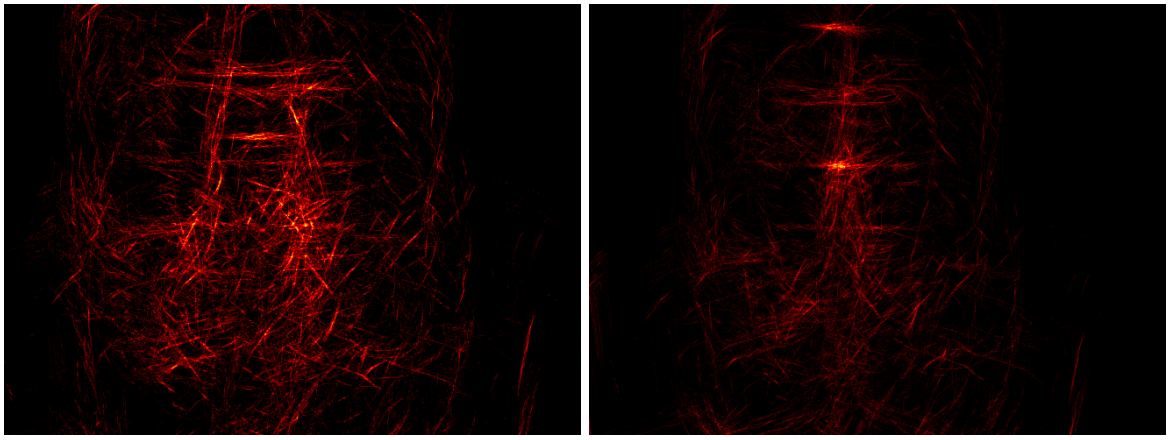

Figure 6: Accumulator images without optimal rotation and scale (left) and with (right).

Lastly, the effects of noise at the reference image and at the query image were explored. For this test, zero-mean additive white Gaussian noise (AWGN) was applied to the reference images, the test images, or both the test and reference images and the standard deviation was applied at 1, 2, 5, 10, and 20. This resulted in 15 noise combinations across the three test images. Note, the range was originally extended to include 50 and 100, but face detection took too long at these noise levels due to the large number of edges that were detected. The effect of the noise was analyzed with respect to the average Euclidian distance of the prediction to the ground truth point. The results are plotted in Figure 7.

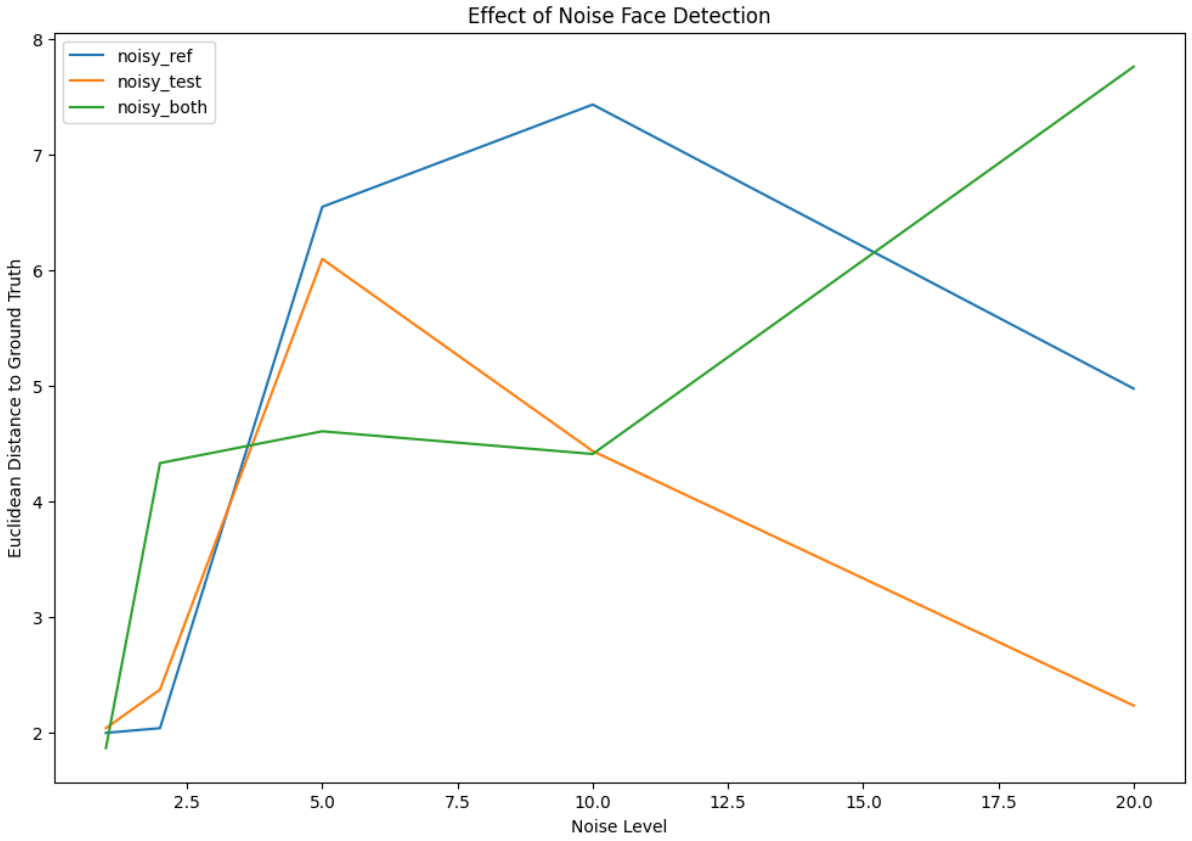

Figure 7: Results of the error at different standard deviations of added Gaussian noise applied to the reference images, to the test images, and to both.

As the graph in Figure 7 indicates, the generalized Hough transform is noise invariant. Only when noise is added to both the reference and the test images does there seem to be a consistently upward trend in the error. When $\sigma = 20$ noise applied to both the reference and the test images, the average error is only 8 pixels from the ground truth. This is a low error, especially when considered against the 60-pixel error in Table 1 prior to rotation and scale optimization. The minimal effect of noise on the generalized Hough transform is further illustrated by the images produced in Figure 8 when $\sigma = 20$ Gaussian noise is applied. The R-table is distorted by noise but the general shape and location of eyes, noise, and mouth are visible. This is sufficient to produce a clear peak in the accumulator and an accurate prediction. However, it may be observed that if too much noise is added to the reference images, it will overwhelm the votes of the true edges and begin to deteriorate the results.

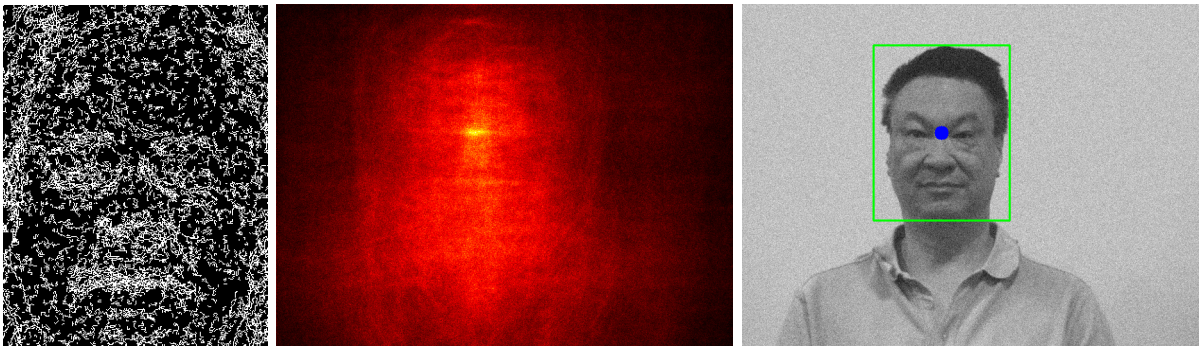

Figure 8: Visualized R-table (left), visualized accumulator (middle), and detected face in noisy image (left) when $\sigma = 20$ noise is added to both the reference and test image.

The example is further illustrated in Figure 9, where the $\sigma = 20$ noise is added to the reference images and the test image noise is increased to $\sigma = 50$. The effect of the noise is starting to be seen in the accumulator, but a maximum peak still exists in the accumulator that leads to an accurate prediction. This demonstrates a considerable robustness to high levels of noise.

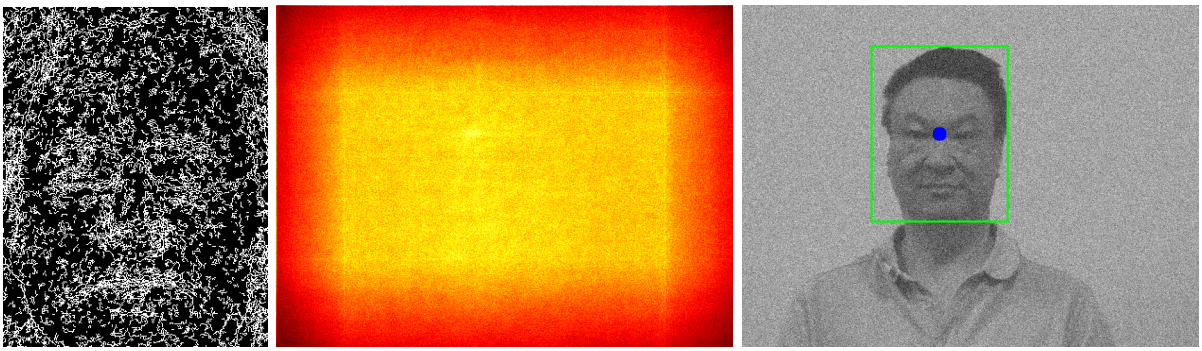

Figure 9: Visualized R-table (left), visualized accumulator (middle), and detected face in noisy image (left) when $\sigma = 20$ noise is added to the reference images and $\sigma = 50$ noise is added to the test image.

5 Conclusion

Parametric transforms like the Hough transform provide a mechanism for detecting shapes. The generalized Hough transform allows this mechanism to be extended to arbitrary shapes that may not be described by an analytic function, such as a face. Such was the focus of this experiment: the application of the generalized Hough transform to face detection.

The primary goal of the experiment was to implement a functional face detection algorithm in a modern programming language (Python) using the generalized Hough transform. The Python implementation details were discussed in Section 3.1 and the qualitative and quantitative results in Section 4 indicate the functionality of the implementation.

The secondary goal was to explore the effects of scale, rotation, and noise on the face detection results. After optimizing parameters for gradient direction and edge detection, it was shown that scale and rotation had an impact on the accuracy of the predicted result. Finding

the appropriate scale and rotation factors resulted in a highly accurate detection, within pixels of the desired location. However, detections were still located on the face and the bounding box encompassed most of the face when scale and rotation adjustments were not applied.

Testing the effects of noise revealed that the generalized Hough transform featured significant robustness against noise. This was true whether the noise was added to the reference images or the query images. At a certain level, the noise begins to overwhelm the true edges of the image, but this threshold is relatively high when compared to the amount of noise that one would expect in a general image. Furthermore, smoothing and other restoration techniques would be applied in practice.

With these results, all the objective of the experiment were successfully demonstrated. The experiment was a valuable exploration of the generalized Hough transform and face detection. The experience provides an other foundational computer vision tool that may be applied in pursuit of more advance algorithms and objectives.

References

[1] W. E. Snyder and H. Qi, *Fundamentals of Computer Vision*. Cambridge University Press, 2017.

[2] S. Minaee, P. Luo, Z. Lin, and K. Bowyer, *Going deeper into face detection: A survey*, 2021. arXiv: 2103.14983 [cs.CV].