As expected, both perform much better than the standard Hough transform. In addition, Xu states that RHT cannot be used for ”curves expressed by equations which are nonlinear with respect to parameters,” which includes ellipses. Robert McLaughlin [2] experimented with RHT and compared it against the standard HT and Probabilistic Hough Transform (PHT is similar to HT but only a small portion $\alpha$ of the pixels in the image, where $2\% < \alpha < 15\%$), are transformed.

McLaughlin achieved good results with RHT vs PHT and HT. He found RHT had ”higher accuracy that both HT and PHT, in noise free images with multiple ellipses” . Also, RHT was ”less subject to false alarms in both noise-free and noisy images. Moreover, RHT proved to be faster than either SHT or PHT ... [and required] substantially less memory” [2]. These results help verify Xu’s statements on RHT. However, contrary to Xu, McLaughlin performed all experiments with ellipses, which Xu states cannot be found by RHT. McLaughlin does this by deriving a linear equation for ellipses.

The author’s goal was to implement the Randomized Hough Transform described and implemented by McLaughlin [2]. In addition, Andrew Schuler’s implementation[3] of McLaughlin’s work served as a reference for this project. Both papers contain high level information, and a few equations, on parameterizing an ellipse and find it with a RHT, however, they leave many of the implementation details to the reader. This caused difficulty to the author in implementing the RHT because much research was required to determine the correct ellipse equations and their application to this problem. These equations and implementation are described below.

## 2 Algorithm

### 2.1 High Level Detail

The overall Randomized Hough Transform algorithm implemented is described below:

```
while( we find ellipses OR not reached the maximum epoch ) {
    for( a fixed number of iterations  ) {
         Find a potential ellipse.

         If( the ellipse is similar to an ellipse in the
         accumulator) average the two ellipses and replace the
```

```
 7          one in the accumulator. Add 1 to the score.
 8
 9          Else insert the ellipse into an empty position in the
10          accumulator with a score of 1.
11      }
12      Select the ellipse with the best score and save it in a
13          best ellipse table.
14      Remove the best ellipse's pixels from the image.
15      Clear the accumulator.
16  }
```

The algorithm executes for a number of epochs, where an epoch is the processing that occurs to find ellipses through accumulation. The algorithm completes when the maximum number of epochs is reached or it does not find ellipses for a specified number of epochs. This allows the user to specify a large epoch maximum and still not waste computing time if the algorithm stops finding ellipses.

The main body of processing occurs in the `for` loop starting on line 2. During the loop, ellipses found are accumulated and given scores. The larger the number of iterations the more likely multiple similar ellipses will be accumulated into a single ellipse and given a higher score. At the end of the `for` loop the accumulator is searched for ellipses with high scores, which are placed in a best ellipse table. To reduce redundant work, the best ellipses found are removed from the image. Because these best ellipses should no longer exist the accumulator is cleared. In this way, previously found ellipses will not generate high scores in the accumulator overshadowing ellipses not found.

## 2.2 Ellipse Dissection

As can be seen from the High Level Algorithm Details previously described the algorithm it self is fairly straight forward. The difficulty arises in actually parameterizing the ellipse such that it can be accumulated. This section describes in detail what is necessary to accomplish this.

Ellipse Equation: $A(x-p)^2 + 2B(x-p)(y-q) + C(y-q)^2 = 1$
With restriction: $B^2 < 4AC$

An ellipse can be described in two ways, either by its center coordinate, semimajor axis length, semiminor axis length, and orientation $(p, q, a, b, \theta)$, or by its center coordinate, radius out from the foci 1, radius out from foci 2, and orientation $(p, q, r_1, r_2, \theta)$. The quin-tuple definition $(p, q, a, b)$ was used in this imple-

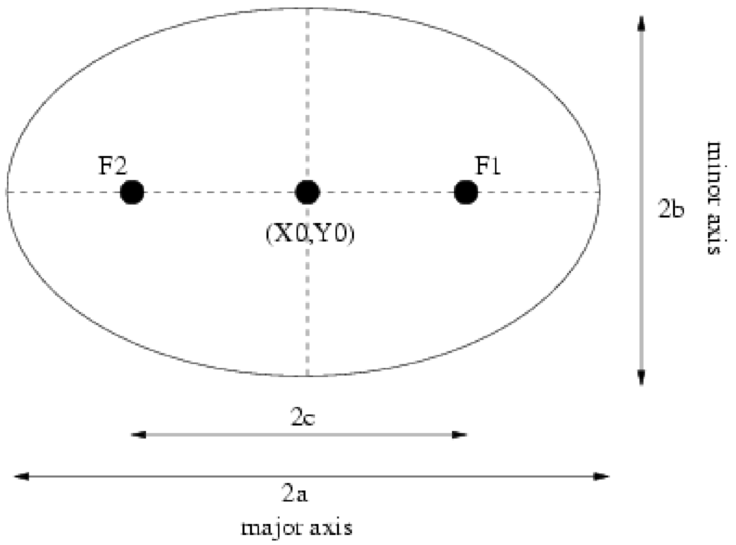

Figure 1: Ellipse Anatomy

mentation because it followed from the equations derived to find the ellipse in an image. The orientation, θ, was not used because no suitable equations could be found to derive it with the information in the image. This is also the reason the second form, i.e. using the radii, was not used.

It is worthing noting, McLaughlin [2] and Schuler [3] use the second form because it seemed to uniformally distribute the ellipse parameters across the hough space. The equations to produce the second form were not forth coming from those papers, and the author could not find or derive them elsewhere. Because the orientation was not saved, this implementation only detects ellipses with major axis $0^o$ and $90^o$ from the x axis. However, this limitation was considered minor in respect to the overall problem.

**Ellipse 4-tuple definition (p,q,a,b)**

- `p` = x coordinate of ellipse center
- `q` = y coordinate of ellipse center
- `a` = semimajor axis length
- `b` = semiminor axis length

### 2.2.1 Determining Ellipse Center

There are five steps to determine an ellipse's parameters from an image starting from finding the center coordinates of the ellipse to determining the semimajor axis' length (a), semiminor axis' length (b), and half the distance between the foci (c).

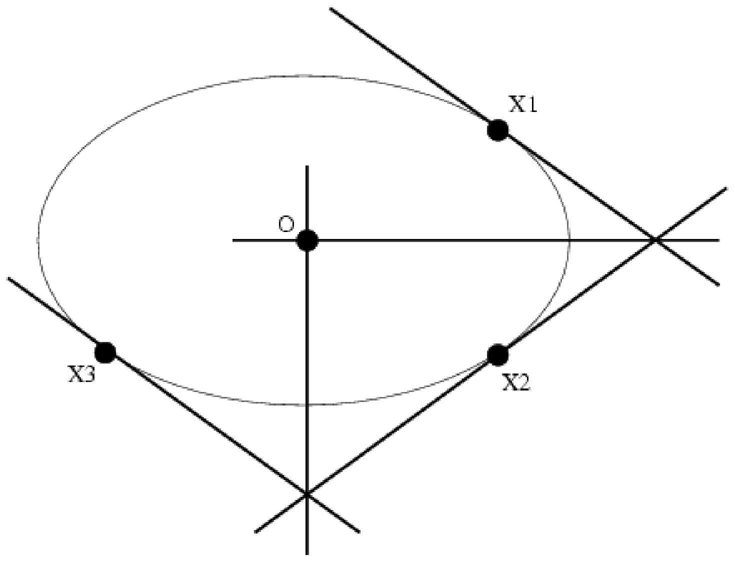

Figure 2: Determining an ellipse center, which is located at the intersection of the bisections of the three tangents to the ellipse.

**1**. Select three points, $X_1, X_2$, and $X_3$

Three points are randomly selected from the image such that each point has an equal opportunity to be chosen. Three times the number of iterations random numbers were generated from 1 to the length of the image in subindicies to form sets of three points for each iteration. A subindex is the number of a cell in a matrix and ranges from 1 to the number of cells in the matrix. This is an alternative form for specifying a matrix cell from the normal row, column form.

Only unique random numbers generated for subindicies were kept to better cover the image, because each iteration requires three random points. If, after throwing away duplicate points, there were not enough points for all iterations specified, random numbers were generated until there were enough. All numbers were kept from this second generation, even if they duplicated the first sets.

**2**. Determine the equation of the line for each point where the

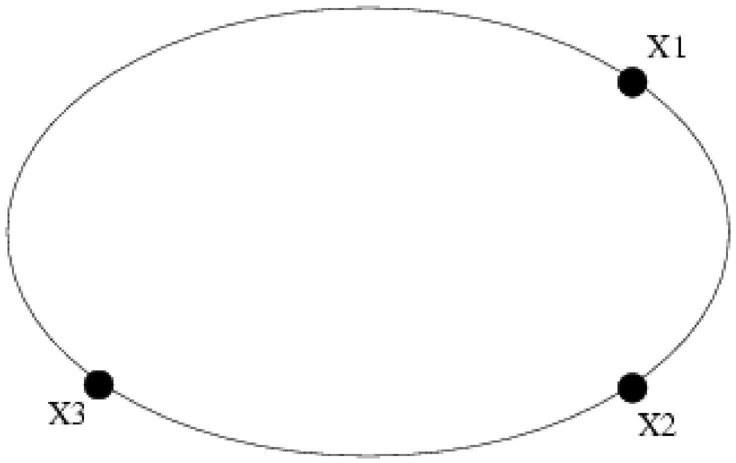

Figure 3: Selecting three points to check for an ellipse.

line's slope is the gradient at the point: $y = mx + b$. This is done by checking the pixels around the point and performing a least squares line fit to them.

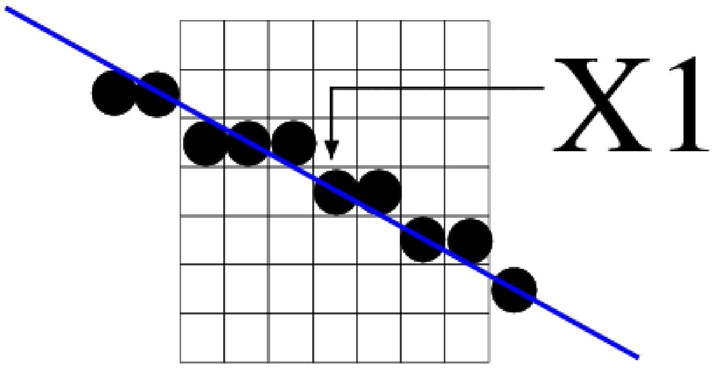

Figure 4: Determine the slope and y-intercept of the line passing through the selected point based on its neighbor pixels represented by the gridded pixels.

Determining the point's line equation is easy with MATLAB. `Roipoly` was used to select points in a seven by seven region around the point of interest. From the coordinates of these points we use the `polyfit` to find the slope $m_1$ and y-intercept $b_1$ for the point of interest.

**3**. Determine the intersection of the tangents passing through point pairs $(X_1, X_2)$ and $(X_2, X_3)$

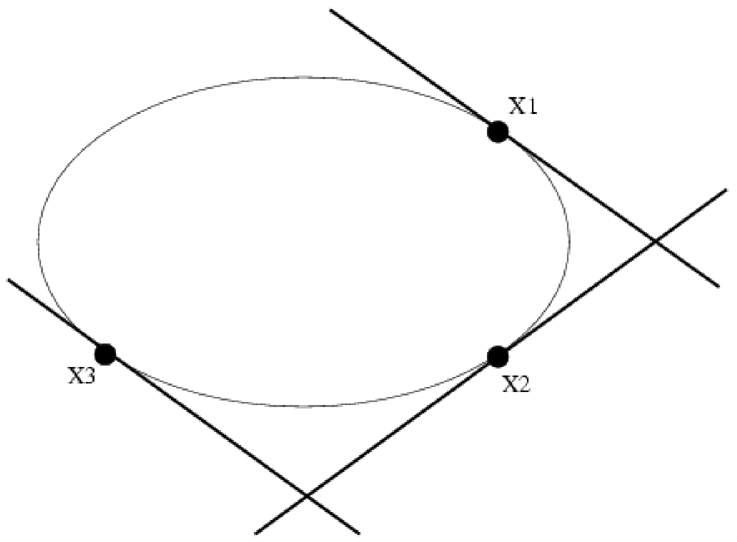

Figure 5: Tangents to the ellipse at points $X_1$,$X_2$, and $X_3$. The ellipse's center is located where the bisectors of the tangent intersections cross.

The tangent intersection points $t_{12}$ and $t_{23}$ are found by solving these systems of linear equations for the x and y coordinates:

Tangents $X_1$ and $X_2$ for $t_{12}$:

$$\begin{bmatrix} m_1x & + & b_1 & - & y & = 0 \\ m_2x & + & b_2 & - & y & = 0 \end{bmatrix}$$

Tangents $X_2$ and $X_3$ for $t_{23}$:

$$\begin{bmatrix} m_2x & + & b_2 & - & y & = 0 \\ m_3x & + & b_3 & - & y & = 0 \end{bmatrix}$$

## **4**. Calculate the bisector of the tangent intersection points. This is a line from the tangent's intersection, $t$, to the midpoint of the two points, $m$.

The midpoint coordinate $m_{12}$ equals half the distance from $X_1$ to $X_2$. The midpoint coordinate and bisection coordinate $t_{12}$ are used to get the bisection line equation. This is found by solving the following equation to find the slope:

$$slope = \frac{m_y - t_y}{m_x - t_x}$$

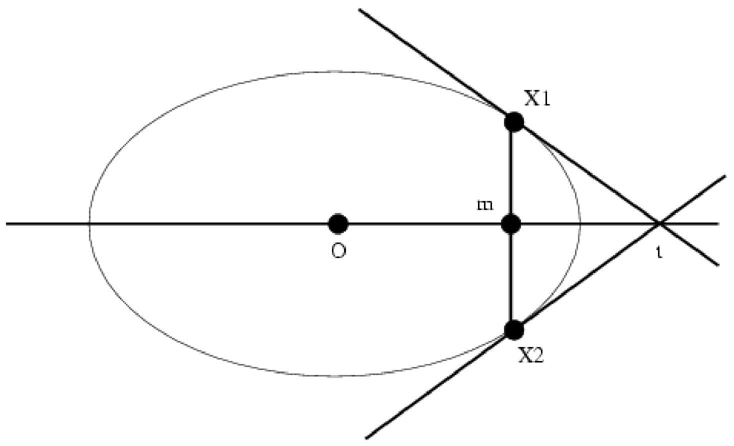

Figure 6: The line bisecting a tangent is found using the point slope line equation, the midpoint between the two points of interest, $m$, and the intersection of the points of interest's tangents, $t$.

and using the slope in the line equation to find the y-intercept:

$$b = slope * x - y = slope * t_x - t_y$$

the bisection line is then: $y = slope * x - b$

**5**. Find the bisectors intersection to give the ellipse's center, $O$

The ellipse's center is located at the intersection of the bisectors. The intersection coordinates are found using the bisectors line equations determined in step 4 in the following system of linear equations.

Ellipse center located at $(x, y)$ derived from:

$$\begin{bmatrix} m_1 x & + & b_1 & - & y & = 0 \\ m_2 x & + & b_2 & - & y & = 0 \end{bmatrix}$$

### 2.2.2 Determining semimajor (a) and semiminor axis' (b) )

Now that the ellipse's center $(p, q)$ has been determined (in the previous section) the remaining ellipse parameters:

- `a` - semimajor axis length
- `b` - semiminor axis length

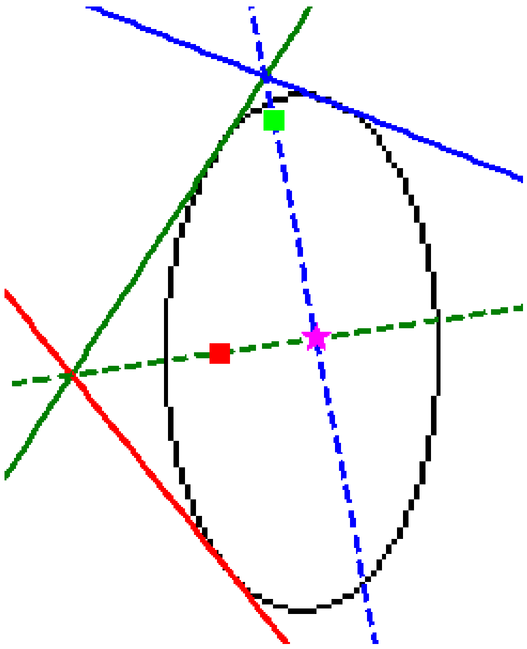

Figure 7: The tangents, bisectors, and center of ellipse found by the implemented algorithm.

can be found from the ellipse equation: $A(x-p)^2+2B(x-p)(y-q)+C(y-q)^2 = 1$ using the three points randomly selected to create three linear equations with respect to A, B, and C. First, the ellipse is translated to the origin to reduce the ellipse equation to: $Ax^2+2Bxy+Cy^2 = 1$. This is done by subtracting $p$ from $x$ and $q$ from $y$ for the three points selected in the beginning $X_1, X_2$, and $X_3$.

Once the ellipse is translated to the origin, the following system of linear equations is solved to find the coefficients A, B, and C:

$$\begin{bmatrix} Ax_1^2 & + & 2Bx_1y_1 & + & Cy_1^2 & =1 \\ Ax_2^2 & + & 2Bx_2y_2 & + & Cy_2^2 & =1 \\ Ax_3^2 & + & 2Bx_3y_3 & + & Cy_3^2 & =1 \end{bmatrix}$$

Next next solve the following equations for the semimajor axis ($a$) and semiminor axis($b$):

$semimajoraxis(a) = \sqrt{|A^{-1}|}$

seminor (b) = $\sqrt{|C^{-1}|}$

### 2.2.3 Verifying the Ellipse Exists in the Image

Even though at this point the ellipse parameters $(p, q, a, b, c)$ were found it is possible the ellipse does not exist in the image. Two checks occur to verify the ellipse exists. First, because the ellipse is defined by the general equation for a conic section:

$Ax^2 + Bxy + Cy^2 + Dx + Ey + F = 0$

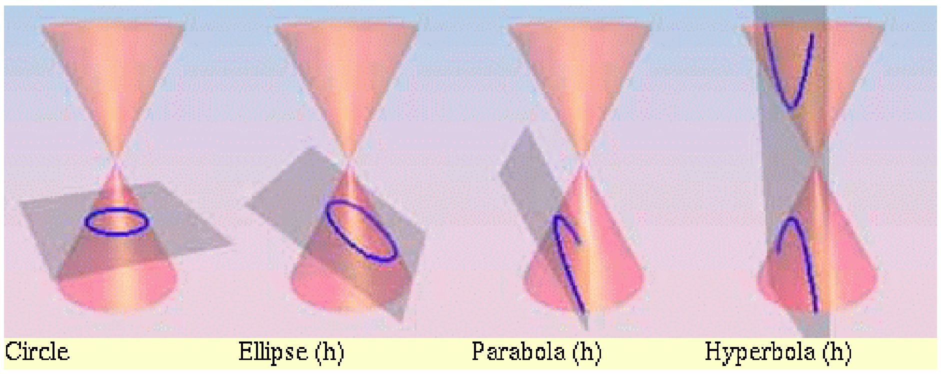

Figure 8: Example of different two-dimensional shapes derived from passing a plane through a conic section [1].

The sign of $4AC - B^2$ determines the type of conic section [1]:

|  |  |
|---|---|
| $> 0$ | Ellipse or Circle |
| $= 0$ | Parabola |
| $< 0$ | Hyperbola |

If the sign is positive then it is an ellipse. Even though the ellipse equation is satisfied as we see from Figure 9 it is possible the ellipse does not have enough pixels in the image.

To determine if the ellipse exists in the image the equation of the ellipse is used to generate points in the image on the perimeter to the ellipse. The number of points generated is equal to the circumference of the ellipse, which is found with the equation: $\pi * semimajor\_axis * semiminor\_axis$. These points are used to generate a mask of the ellipse, which is 'anded' with the image. The number of pixels in the new image are counted and divided by the circumference of the ellipse. This yields a ratio of pixels to circumference. If the ratio is greater than a threshold specified by the user the ellipse exists in the image.

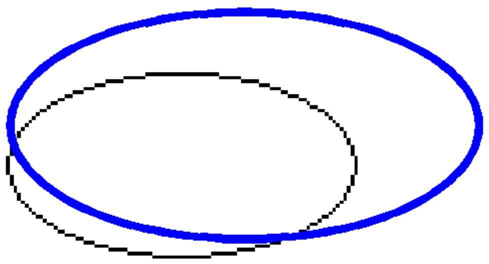

Figure 9: The black ellipse is in the image. The blue ellipse is verified by the ellipse equation, however, it does not actually exist in the image because its ratio of pixels to circumference is to low.

| p | q | a | b | score |
|---|---|---|---|---|
| 98.8937 | 99.5075 | 25.9742 | 47.2827 | 3.0000 |
| 100.5589 | 99.2206 | 25.5182 | 37.1271 | 2.0000 |
| 105.9815 | 82.0521 | 28.4240 | 56.8710 | 1.0000 |
| 115.0860 | 86.8599 | 38.8704 | 58.1393 | 1.0000 |
| 109.0266 | 102.1437 | 45.7310 | 43.1455 | 1.0000 |
| 103.0161 | 95.9755 | 20.9283 | 51.6607 | 1.0000 |
| 89.7838 | 122.2568 | 11.4692 | 19.6179 | 1.0000 |

Figure 10: Example accumulator.

## 2.3 Accumulating

At this stage the ellipse's parameters were found and it was verified to exist in the image. Now the ellipse is added to the accumulator.

The accumulator stores the $(p, q, a, b, score)$ of an ellipse. The half distance between the foci, $c$, is not stored because it is not needed to generate ellipse points. Ellipse points are generated by solving the following equations for $\phi = 0 \; to \; 2 * \pi$:

$$x = a * cos(\phi)$$
$$y = b * sin(\phi)$$

The number of points generated are equal to the number of values used between $[0 \text{ and } 2 * \pi]$, in this algorithm the number of values generated is equal to the circumference of the ellipse.

Below is an example accumulator. The best ellipse has a score of 3.0, is centered at (98.9,99.5), has semimajor axis of length 25.97 and semiminor axis length 47.3. Figure 11 shows the best ellipse found over the original image.

The following three steps occur to accumulate a new ellipse's center coordinates $(p, q)$, semimajor axis $(a)$, and semiminor axis $(b)$.

1. For all $(p_i, q_i, a_i, b_i)$ ellipses in the accumulator test:

   - If the distance between the new ellipse center is within a threshold. $\sqrt{(p_i - p)^2 + (q_i - q)^2} >$ distance_threshold
   - $|a_i - a| >$ semimajor_axis_threshold.
   - $|b_i - b| >$ semiminor_axis_threshold.

2. For any ellipse in the accumulator where the above conditions hold, perform a weighted average between each of the ellipse parameters (use the score as the weight) and replace the ellipse in the accumulator with the new weighted ellipse, then increase the score for this ellipse by one.

   Example weighted average of semimajor axis length:

$$\frac{a_i * score + a}{score + 1}$$

3. If there are no ellipses in the accumulator that satisfy this condition place the new ellipse in the accumulator with a score of 1.

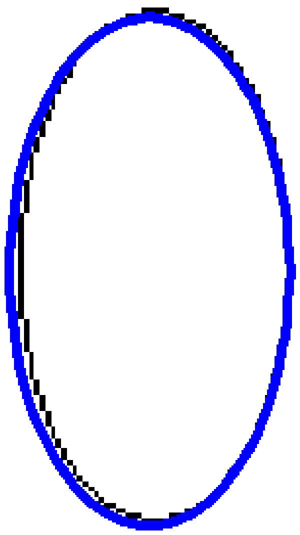

Figure 11: Best ellipse with score 3.0 from the example accumulator shown in blue over the black ellipse image.

## 2.4 Storing Best Ellipses and Repeating

After the `for` loop to accumulate ellipses completes the algorithm finds the best ellipses in the accumulator and stores them in a matrix of the same form as the accumulator $(p, q, a, b, score)$. Ellipses are added to the best ellipses matrix the same way they are stored in the accumulator described in the previous section.

Each ellipse is compared to the new ellipse and if they are similar they are weight averaged together based on their scores. This prevents duplicate ellipses from occurring in the best ellipse table if they are found during different epochs.

When an ellipse is placed into the best ellipse matrix it is removed from the image to increase the likelihood other ellipses in the image will be found. Figure 18 shows an example of the found ellipses removed from the image.

Once all the best ellipses are added to the matrix and removed from the image the accumulator is cleared for the next epoch and the process repeats.

## 3 Results

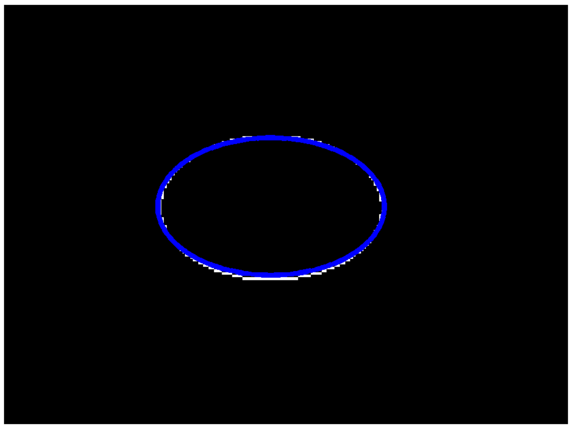

Figure 12: Ellipse found in blue with score seven after 5 epochs 10 iterations per epoch.

Figure 12 illustrates the result of running the program on an ellipse with orientation $0^o$ to the x-axis. The program ran for 5 epochs at 10 iterations per epoch. The best ellipse found scored seven. The ellipse does fit perfectly on the ellipse in the image, the non-overlapping area is a product of stretching the image in MATLAB.

To determine if the ellipse program discriminates against non-ellipse objects the ellipse program was run with non-ellipse object images. Figure 13 exemplifies