Traffic Light Recognition

Patrick Ward - 15323314

TCD Computer Science

Abstract

This program used thresholding, contour search and statistical pattern recognition to locate 25 of 30 traffic lights. The body of a traffic light box consists of a dark box surrounded by a white line. Their shape, color and contour hierarchy are also all largely homogenous. These distinguishing features that can be exploited to locate their presence and state in images. This program was evaluated using precision, recall, accuracy and DICE coefficient for each traffic light in the image.

Keywords: Thresholding, Statistical Pattern Recognition, Contour

Traffic Light Recognition

The dark colour of the traffic light box and the bright white line around its parameter allow a binary threshold operation to clearly distinguish the shape of the traffic light box from the rest of the image. The contours of the bounding box can be found using find contours. The bounding box of this shape will then match a particular aspect ratio which can be used to separate its contours from other objects in the image. To separate the traffic lights from other objects in the image with a similar aspect ratio the contours inside the bounding box are analysed to check how circular they are. If they are circular the object is taken to be a traffic light. The position of this circle used to find the state of the traffic light.

.

1 Methodology

1.1 Evaluation Metrics

This project was evaluated using the precision, recall, accuracy and DICE coefficient for each traffic light in the image. The percentage of traffic lights for which the correct state was recognised was also calculated as a measure of how well the program was able to separate traffic lights from the rest of the image and evaluate their state. Accuracy was used as a general measure of performance, although it does not take into account the number of false negatives or false positives, which can be costly when a system is attempting to located traffic lights. The precision is a measure of false positives over true positives, it evaluates how good the program is at finding a traffic light as opposed to similar objects in the image. Precision is important for evaluating the reliability of the program. The recall evaluates how good the program is at finding traffic lights generally in the image, it can show how effective the program is at locating traffic lights ignoring any false positives and is a good measure for how good the program is at actually finding traffic lights.

1.2 Binary Thresholding

Traffic light boxes are comprised of a white boundary, a black rectangle and a coloured light. When an image is converted to grayscale the white boundary pixels

and light tend to have a brightness close to 255. Conversely, the black box of the traffic light tends to have a brightness value between 0-58. For this reason, a binary thresholding if quite effective at separating traffic lights from the rest of the image. The binary threshold for this project was set to 68. This was evaluated as the best possible value for finding traffic light boxes while also excluding most other dark objects in the image.

Traffic light boxes are comprised of a white boundary, a black rectangle and a coloured light. When an image is converted to grayscale the white boundary pixels and light tend to have a brightness close to 255. Conversely, the black box of the traffic light tends to have a brightness value between 0-58. For this reason, a binary thresholding if quite effective at separating traffic lights from the rest of the image. The binary threshold for this project was set to 58. This was evaluated as the best possible value for finding traffic light boxes while also excluding most other dark objects in the image.

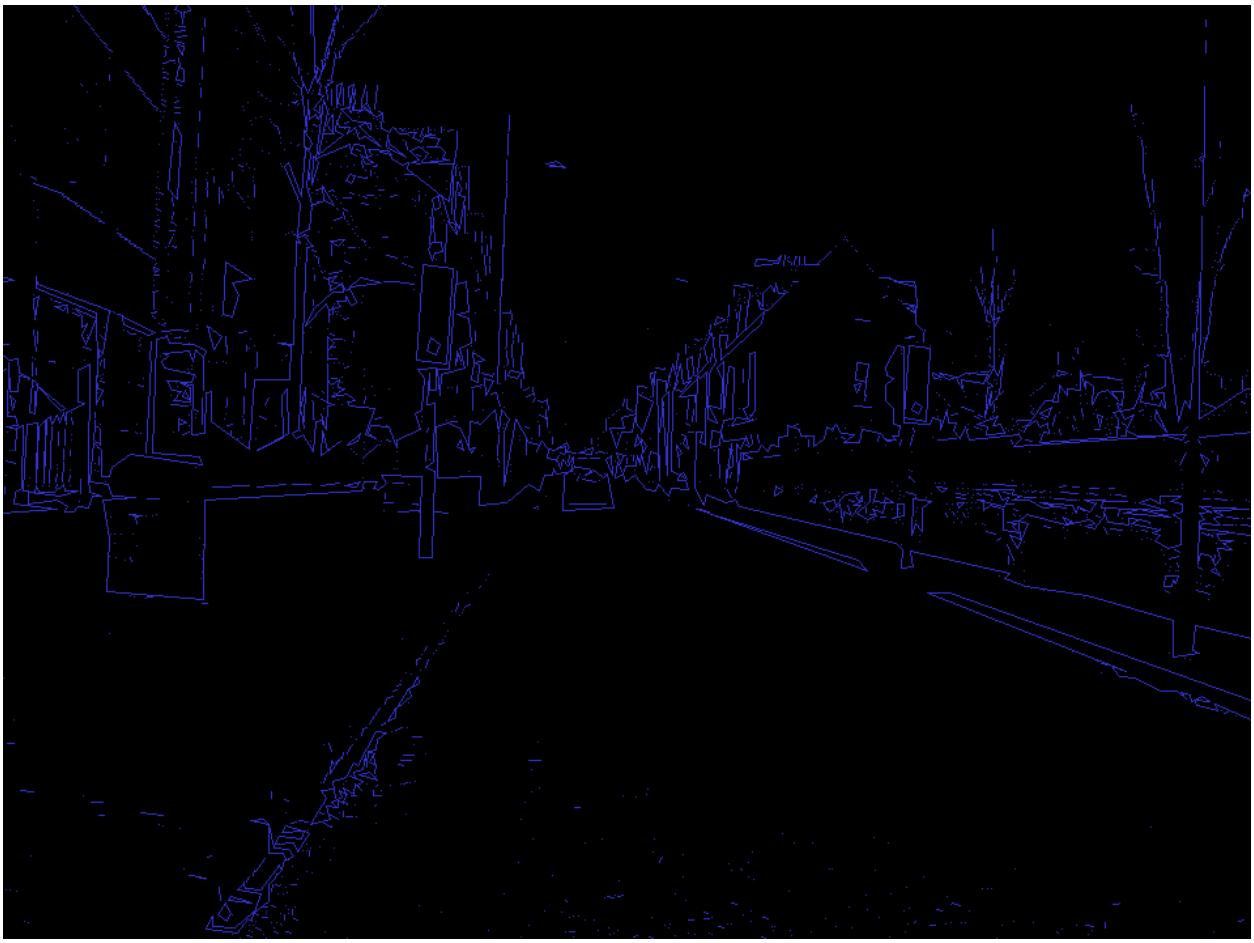
Binary thresholding works evaluate each pixel in the image. If the brightness value for the pixel is above the binary threshold the value is set to the maximum brightness value and given a value of one. If the pixel is below the threshold it is set to the minimum brightness value and given a value of zero. This separates very

dark regions of the image. Since the find contours algorithm search for white, objects on a black background the image is then inverted

1.3 Finding Contours

Using this binary image the program then applies the opencv findCoutour() algorithm to find the contours in the image. The contour algorithm checks if pixels directly above, beside or diagonal to a particular pixel are of a different value to the pixel being considered. If a pixel is of a different value of the pixel being considered is found to be on a contour. Adjacent contour points are stored in vectors, these vectors make up shapes. When polygons are applied to these shapes we can see a contour image (see figure 1). Since the white traffic lights are clearly distinguished from the black background the outlines of the lights can be found using this method.

Figure 1: Contour Image



2.5 Evaluating Contours

To find the contours which represent traffic light boxes the aspect ratio and size of the contours minimum bounding box is analysed. Each contour is considered separately. If the bounding box has an aspect ratio between .25 and .8 and an area of between 520 and 12,0000 then it is potentially a traffic light box. The ratio was selected to allow for wide and narrow traffic lights. The minimum and maximum area were selected to reduce the number of false positives that occur as a result of

very small or very large objects in the image that could not represent traffic light boxes.

Figure 2: Bounding Boxes Around Contours with No Constraints

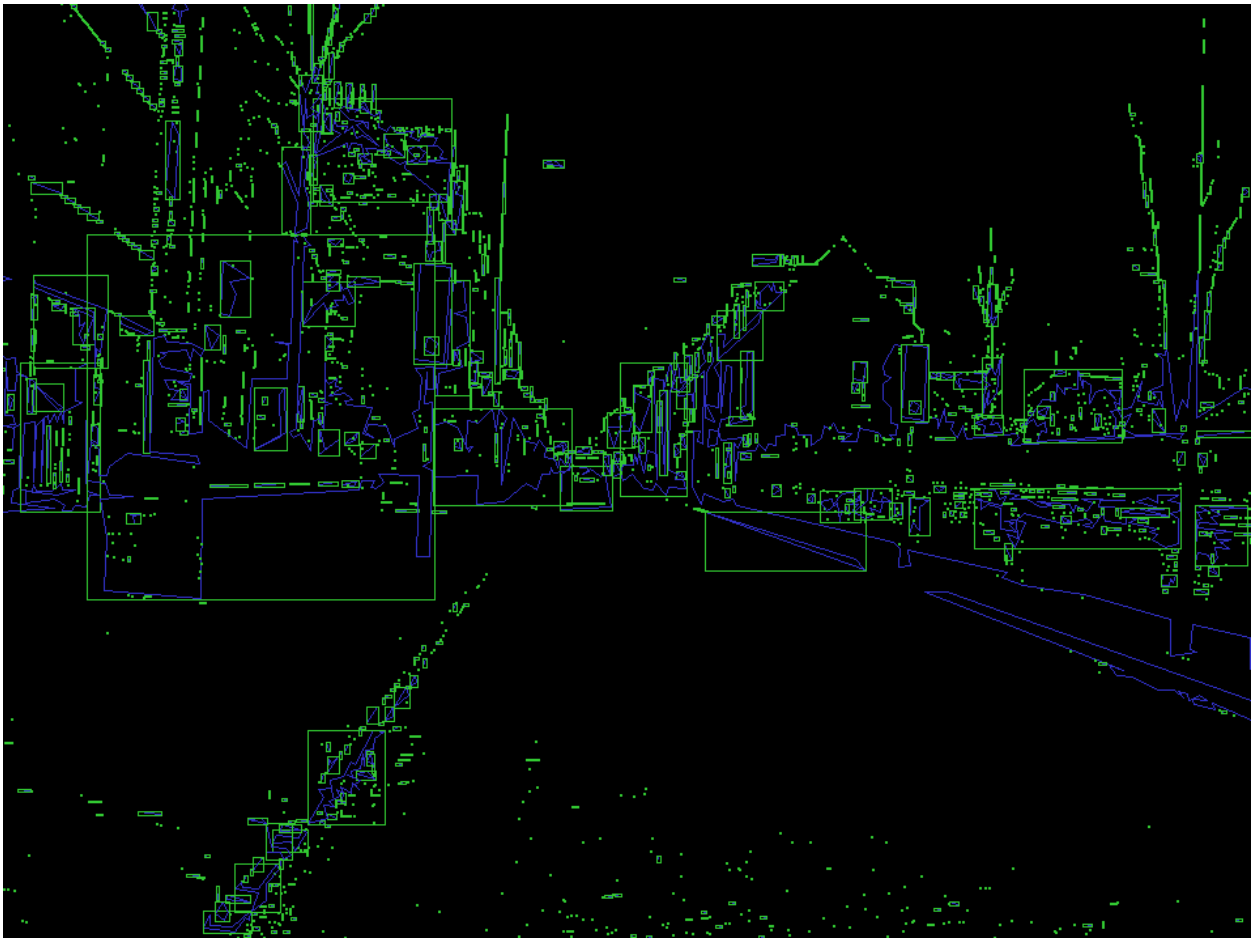


Figure 3: Bounding Boxes with Area Constraints

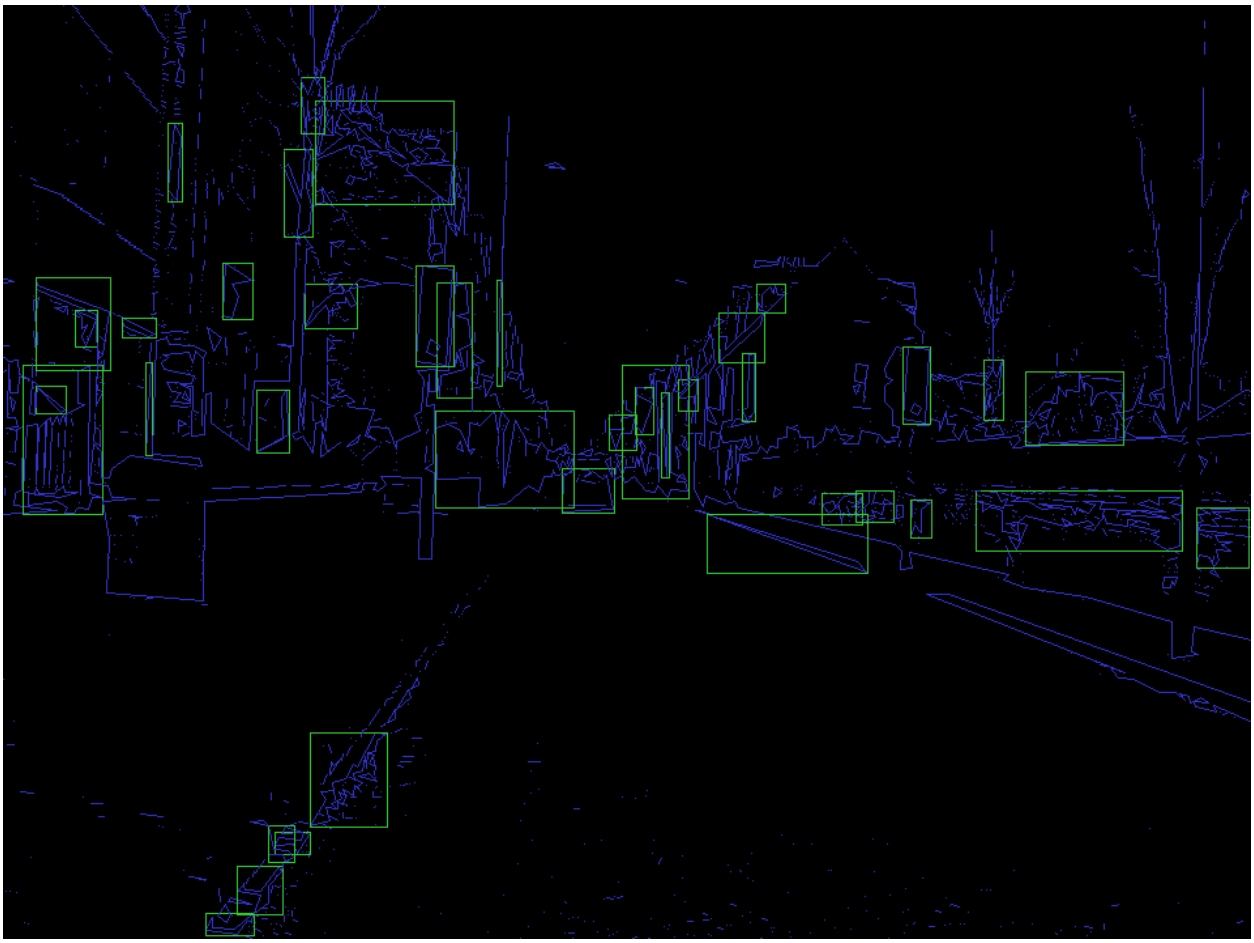
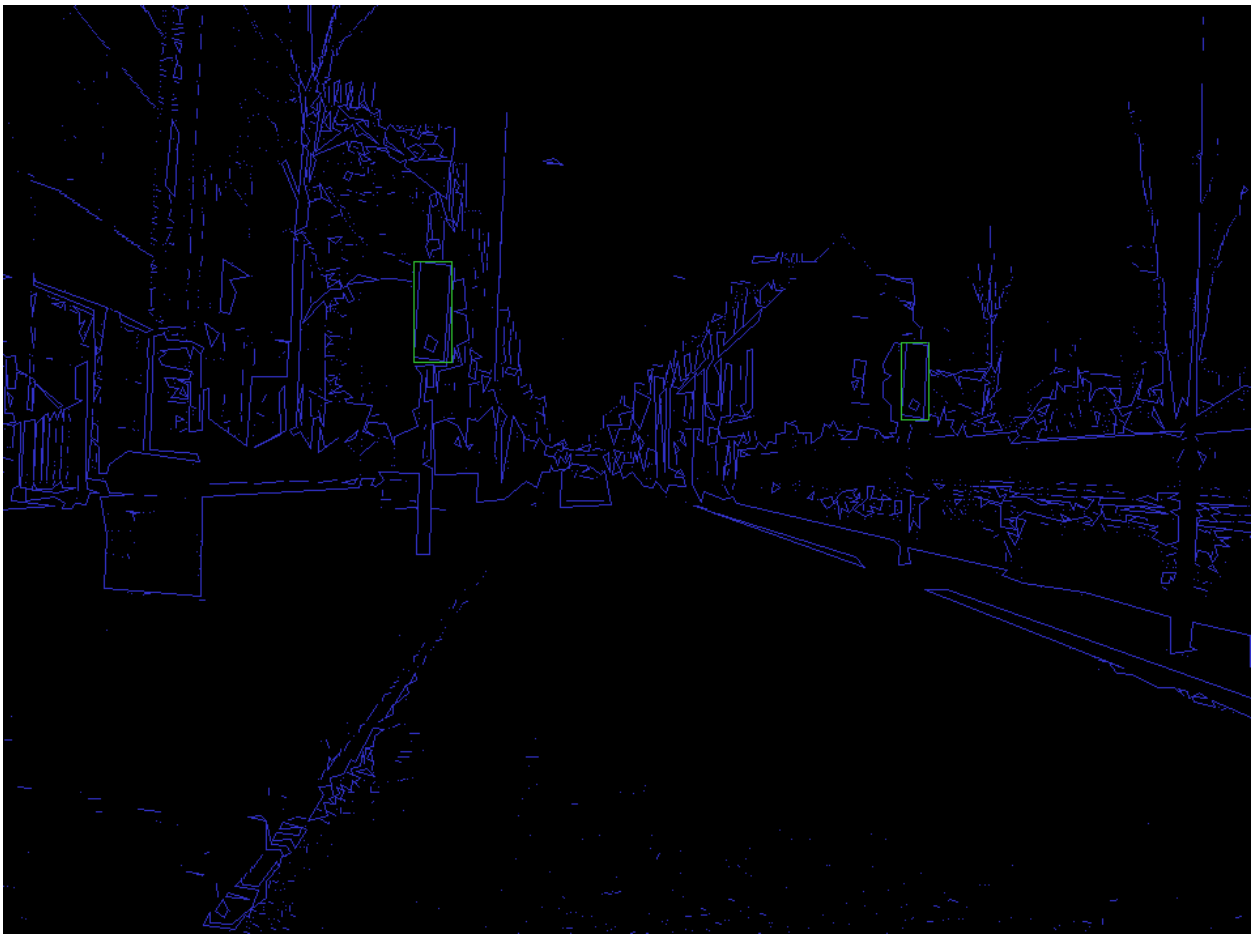


Figure 4: Bounding Boxes with All Constraints



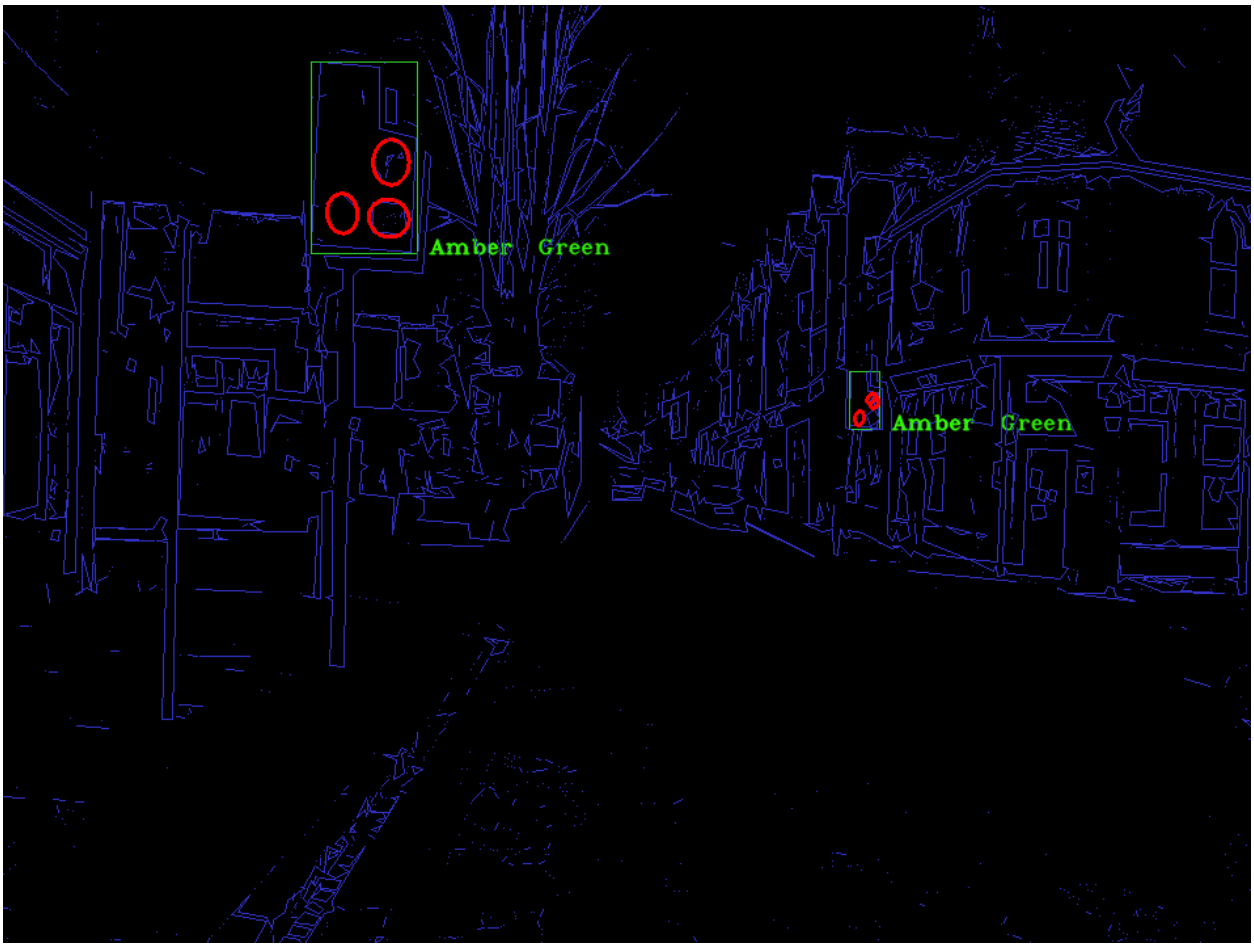
1.4 Evaluating Contents of Contours

Once a contour is found to have the proper aspect ratio the contents of the contour are evaluated. Using the contour hierarchy vector, the program cycles through each of the first children of the traffic box contour. For each contour, it draws an ellipse around the contour. The eccentricity of an ellipse is a measure of its circularity which ranges from 0 for a perfect circle to 1 for what is almost a line. Traffic lights tended to get a value of .85 or above for eccentricity. The program excludes any

potential traffic light boxes that do not have a circular object with at least an eccentricity of .85 or above inside of them. This produces the results seen in figure 6.

Figure 5: Ellipses with No Constraints

Figure 6: Ellipses with Constraints



Since many of the false positives were not actually a similar shape to a traffic light but just produced a similar bounding box on account of their outermost points. These could be removed when considering the overlap between the bounding box and the actual contour.

The state of the traffic lights was found using the position of the circular hole in the traffic light box. Circles within 10 pixels of the centre were classified as amber

lights. Circles above ten pixels of the centre were classified as red and circles below 10 pixels of the centre as Green lights.

2 Results and Discussion

2.1 Results

The procedure outlined here produced 25 true positives, 4 false positives and 5 false negatives. It recognised 19 traffic lights correctly. This equates to a precision of 86.2% , a recall of 83%, an accuracy of 88%, and a DICE ratio of 85%. It correctly found the correct state of 63% traffic lights. See images 1 - 20.

2.2 Discussion

This program was not able to distinguish between signs on a traffic light box and the light in the box. These results could be improved using colour recognition for the traffic lights. Colour and region recognition such as mean shift segmentation could also allow the program to find the traffic lights which it missed. The missed lights were largely due to a trade-off between raising the threshold for the binary image. If the threshold was to high the contours would lost on some images, while if it was too low the same would happen on others.

Figure 7: Processed Image 1

Figure 8: Processed Image 2

Figure 9: Processed Image 3

Figure 10: Processed Image 4

Figure 11: Processed Image 5

Figure 12: Processed Image 6

Figure 13: Processed Image 7

Figure 14: Processed Image 8

Figure 15: Processed Image 9

Figure 16: Processed Image 10

Figure 17: Processed Image 11

Figure 18: Processed Image 12

Figure 19: Processed Image 13

Figure 20: Processed Image 14