FAD Homework Session 5

Erfan Ebrahimi Bazaz

November 2020

1 pretext

All code is available at the following github:

```
https://github.com/ErfanEbrahimiBazaz/object_detection_by_active_contours_Canny_algorithm
```

2 Task 1 and 2 - active contours and Canny algorithms

In this task we need to detect all cars in a video and identify their colors.
The algorithm I applied is as follows:

1. Read the video frame.
2. Apply a blur and median blur filter on each frame to denoise by keeping the borders. This can be bilateral or Gaussian filter as well.
3. Calculate Canny algorithm of each filtered frame.
4. Calculate active contours of each Canny output.
5. Loop on all the contours on each frame and approximate the polygon of each contour.
6. Find the bounding rectangle of each polygon from the previous step.
7. Draw the bounding rectangles on each frame and look for the right type of thresholds to only get the cars.
8. Set your thresholds on contour area, bounding circle radius, bounding rectangle width and height.
9. Make a mask on each bounding rectangle and bitwise and it to the original frame.
10. Load an SVM trained classifier from previous tasks an apply it on each masked car in each frame.
11. Show the predicted color of each car via the classifier.

The result is shown in Fig. 1, 2, 3 and 4.
All frames together are shown in Fig. 5

Note: The classifier used in this activity is an SVM classifier with poly kernel with default values for C and gamma, trained on a set of about 20 to 30 cars for colors: black, white, red, blue, gray and yellow. The confusion matrix is shown in Fig. 6 and 7. The accuracy of this classifier is only 62%. In real world application this needs to be replaced. The pickle file of the classifier is available in giuthub repo.

Figure 1: Original video frame

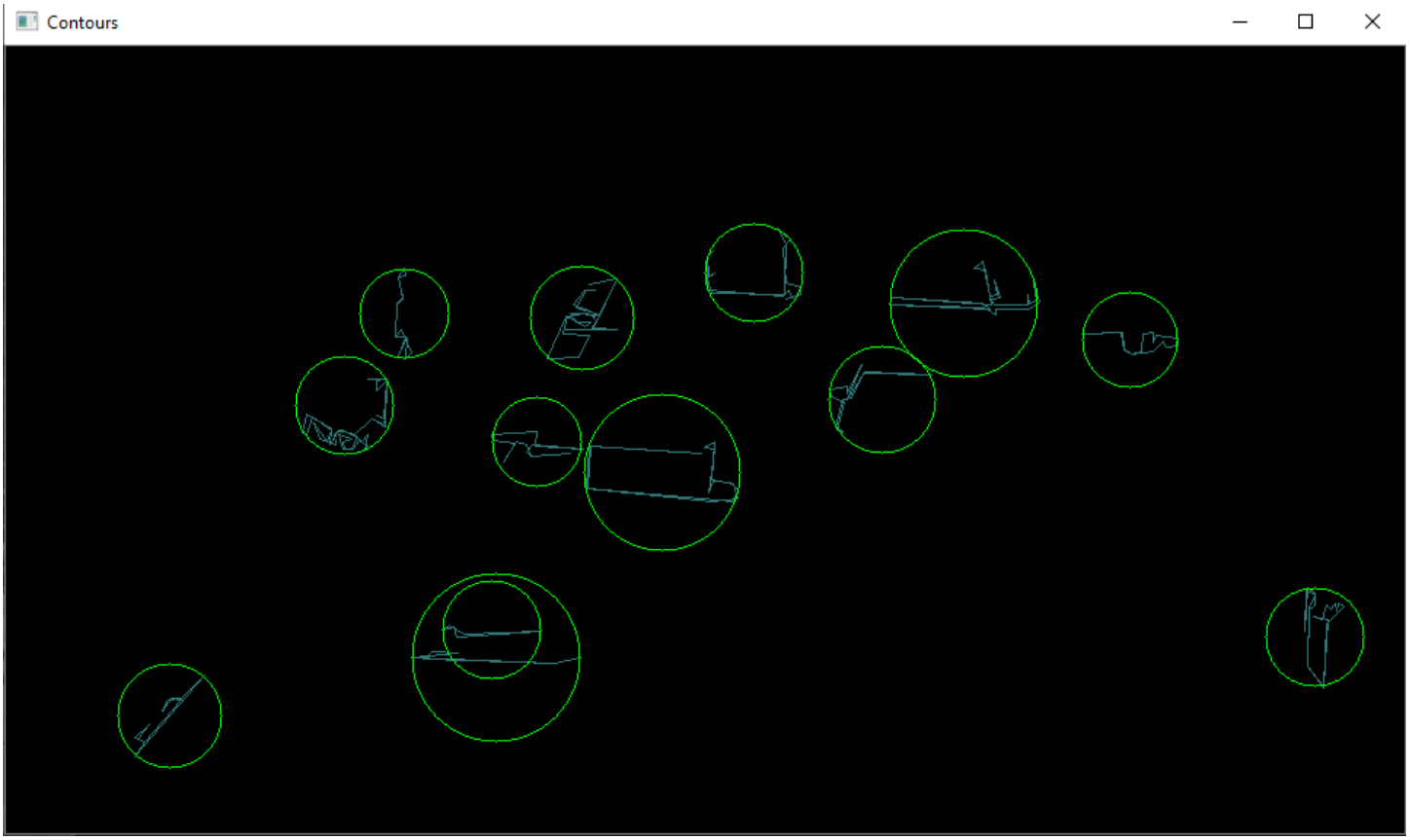

Figure 2: Detected contours

Figure 3: Masked cars on each frame

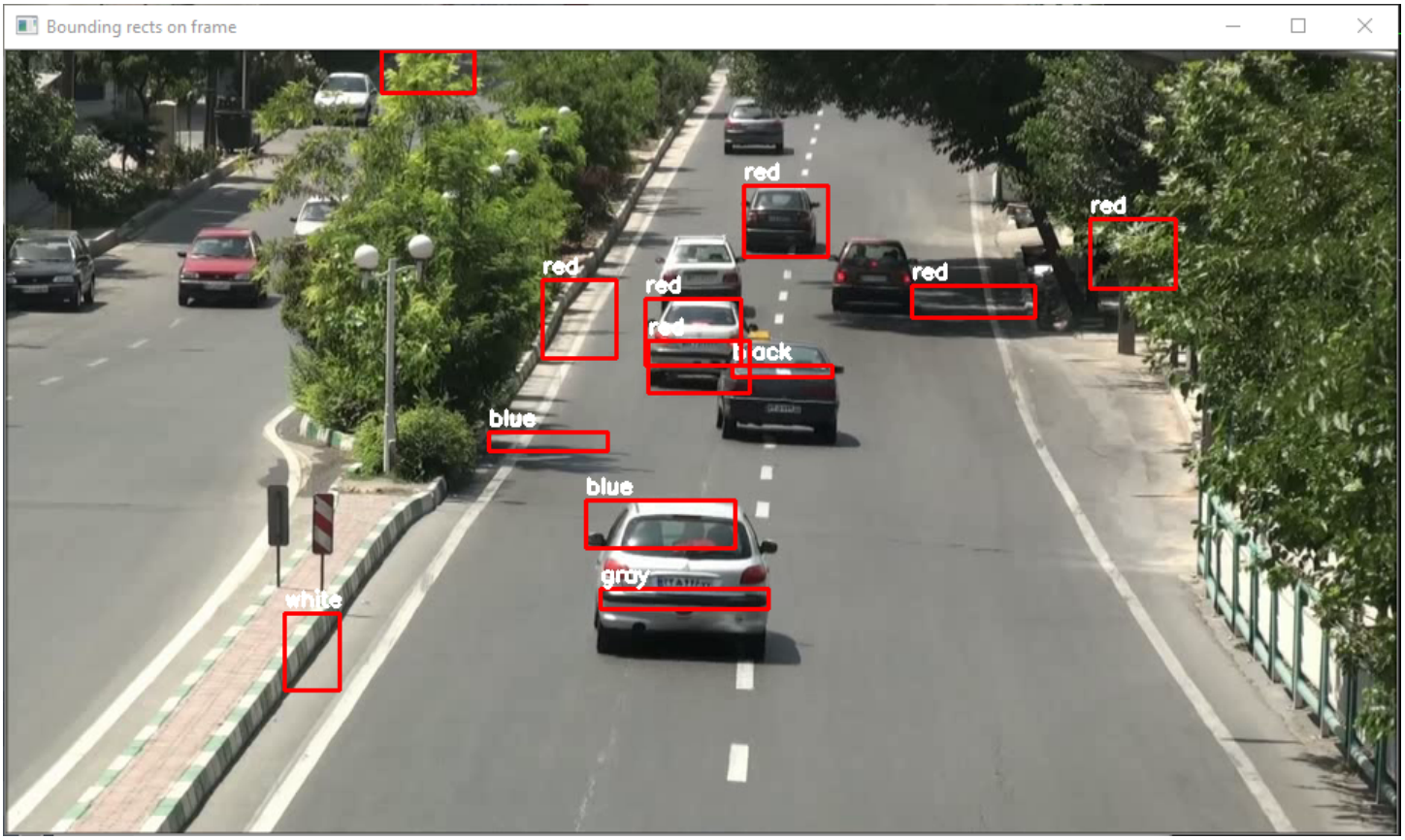

Figure 4: Predicted colors applied real time

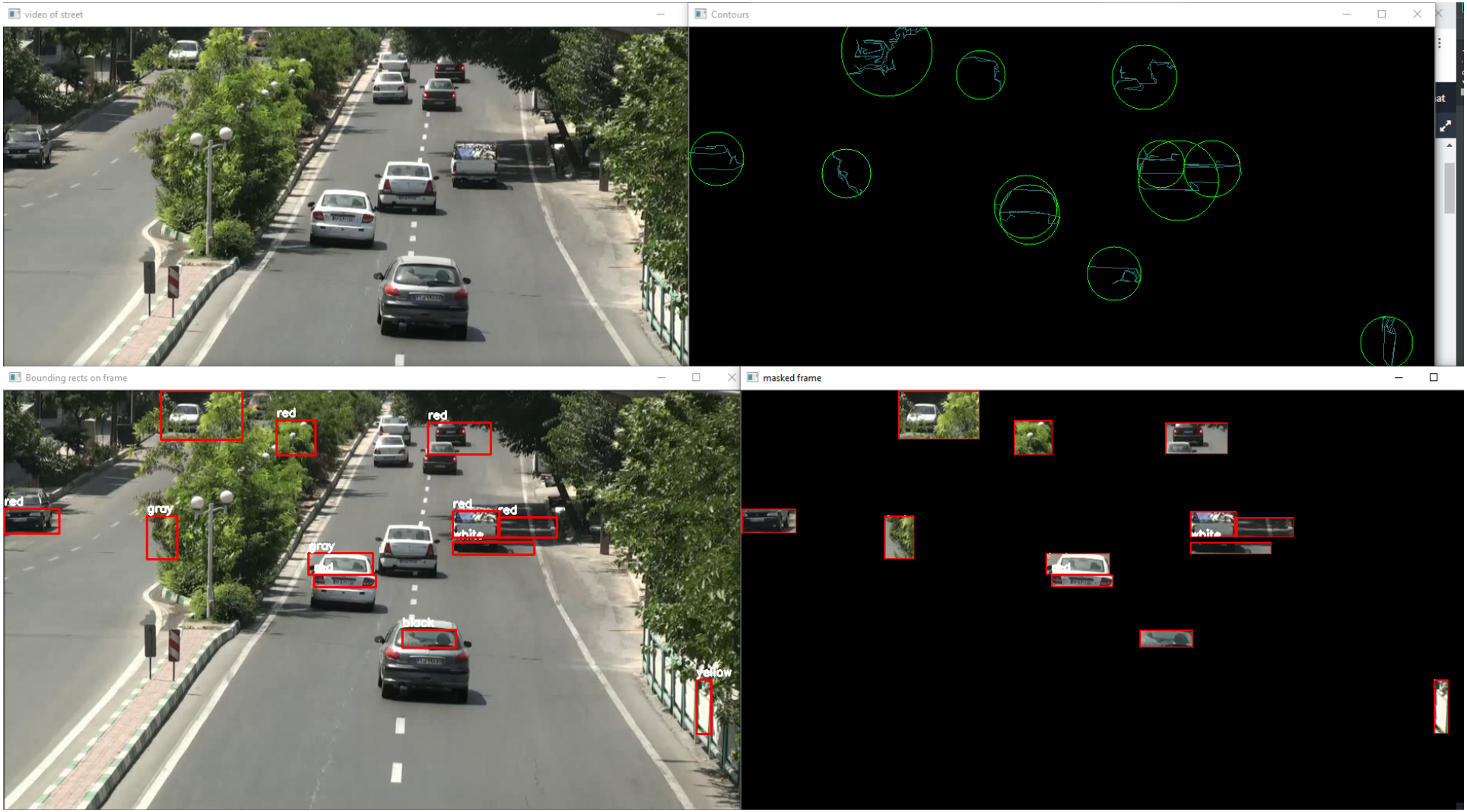

Figure 5: Side by side frame, contours, masks and predicted colors

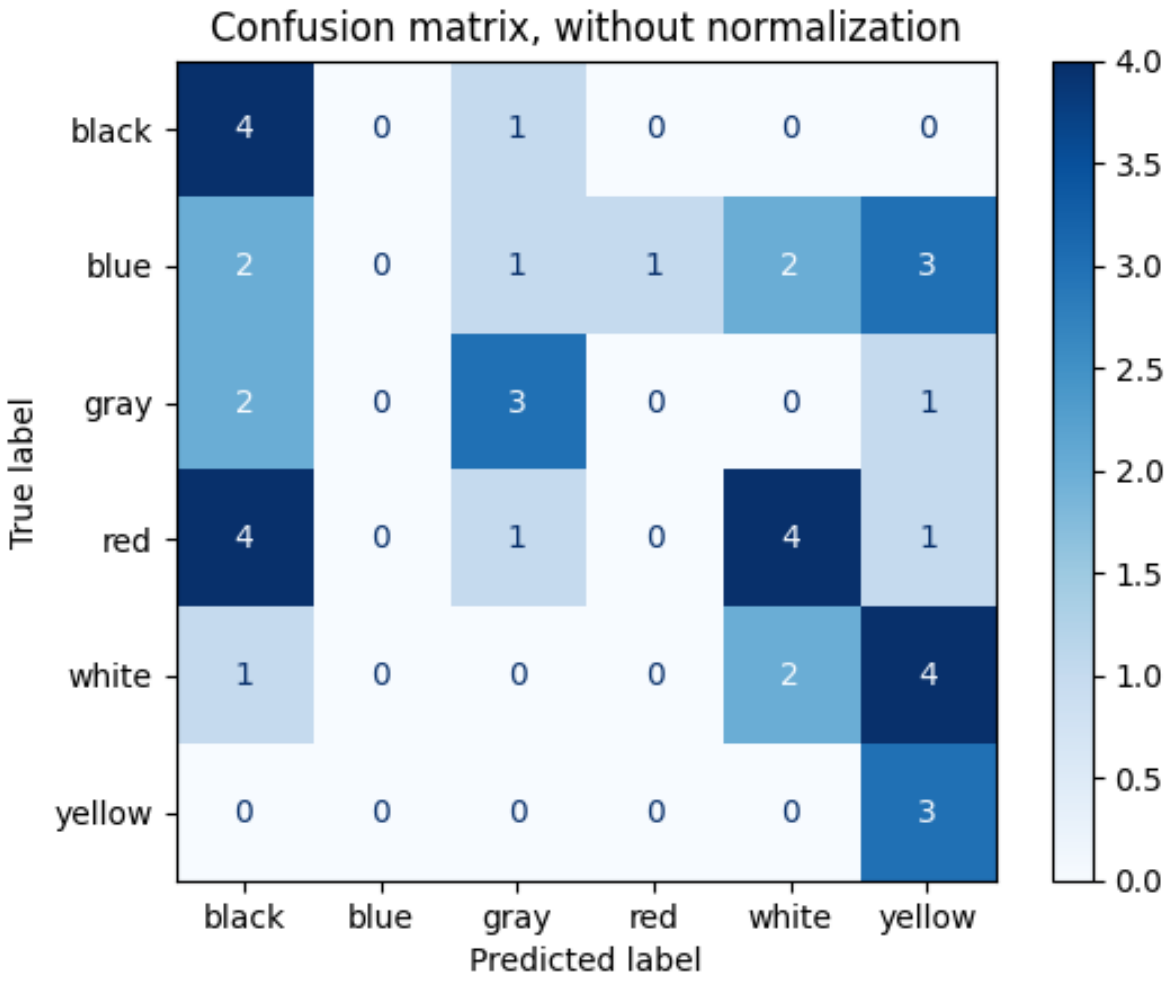

Figure 6: Confusion matrix of the SVM classifier with poly kernel without normalization, score = 62%

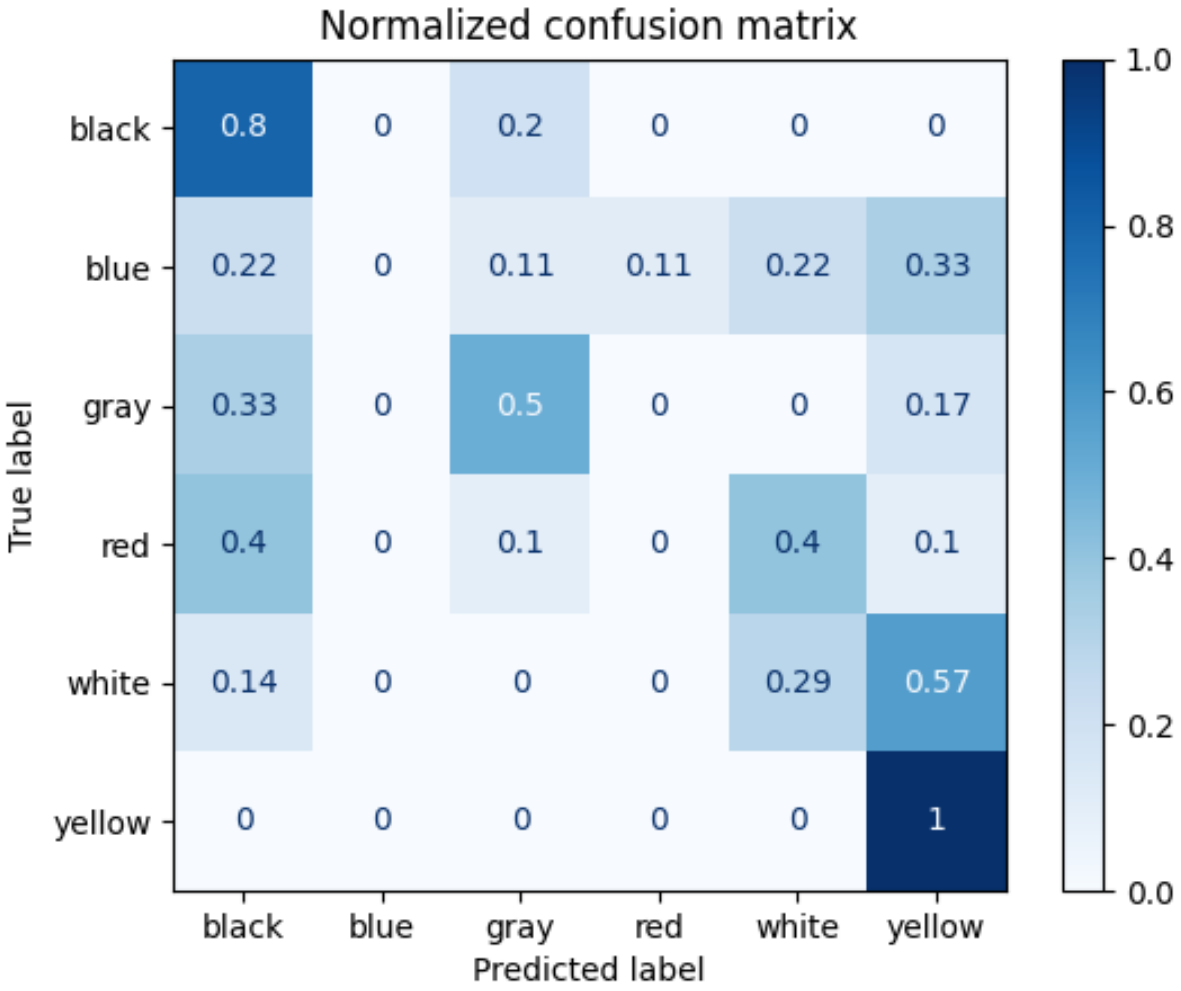

Figure 7: Confusion matrix of the SVM classifier with poly kernel with normalization, score = 62%

3 Task 3 - detecting finger prints

In this task we need to detect fingers print by using filtering, Canny algorithm and active contours.

I applied a wide range of algorithms varying from median blur, bilateral and Gaussian blur to denoise and get bigger contours by applying Canny and active contours. However, none of them could really get it properly.

Some of the results are shown in Fig. 8 to 13. It needs to be noted that applying filters to get coarse-grained features, can prevent any contour detection altogether. This can be seen in Fig. 11 and 12.

In some cases like 13 the algorithm has worked quite well.

Figure 8: Finger print together with its Canny, contours, and bounding rectangles on the finger print

Figure 9: Finger print together with its Canny, contours, and bounding rectangles on the finger print

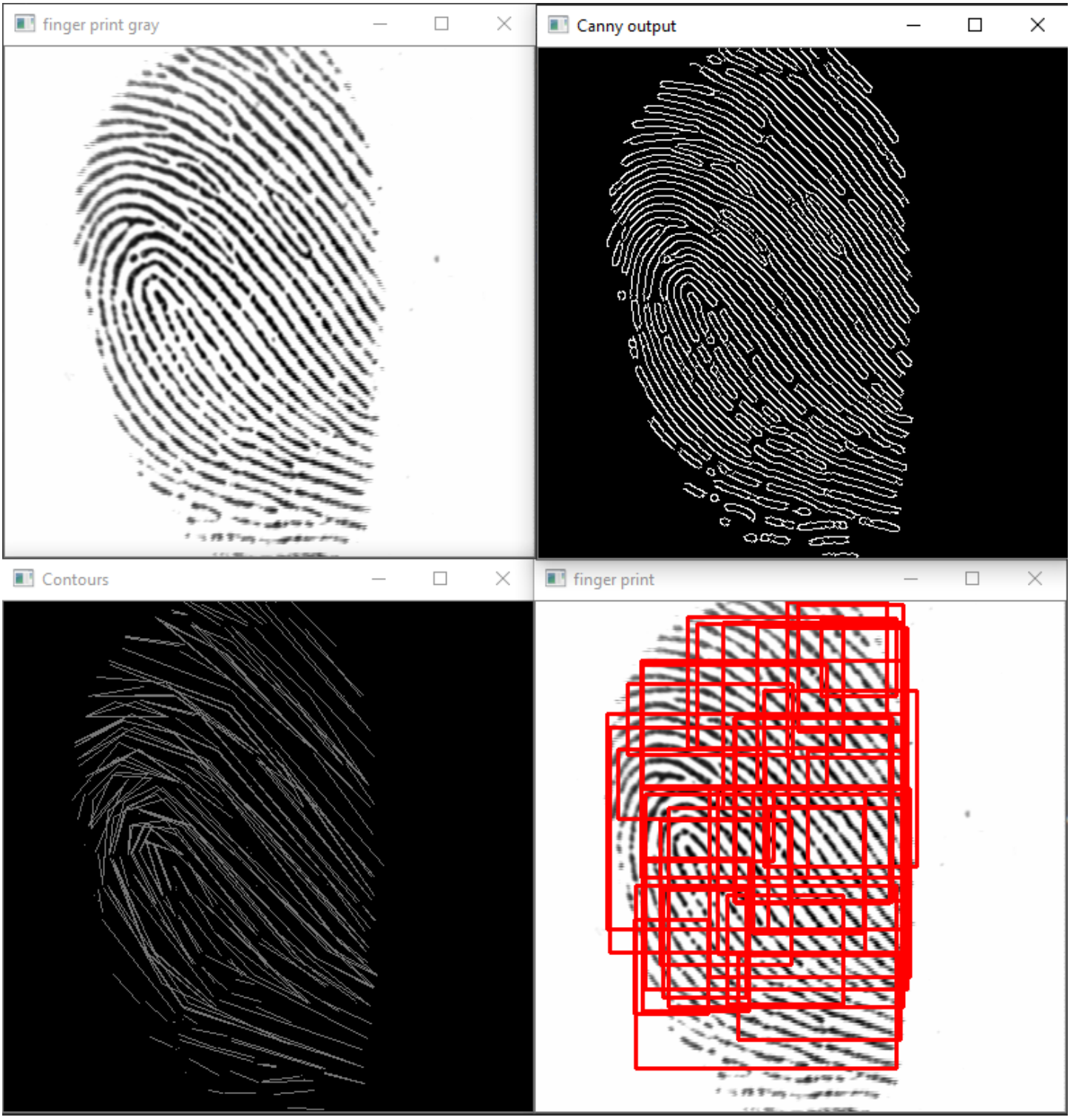

Figure 10: Finger print together with its Canny, contours, and bounding rectangles on the finger print.

Figure 11: Finger print together with its Canny, contours, and bounding rectangles on the finger print/ Filtering has hindered any active contour identification

Figure 12: Finger print together with its Canny, contours, and bounding rectangles on the finger print. Filtering has hindered any active contour identification

Figure 13: Finger print together with its Canny, contours, and bounding rectangles on the finger print - an example of good contour identification.