Applications of Image and Video Processing: Final Report

Dogan Aykas

18 December 2020

1 Introduction

Since the course is related to image and video processing applications, I have chosen the topic of Computer Vision Based Mouse. The project idea stemmed from the fact that such an application would incorporate different aspects of image and video processing: applying filters to images, object tracking, object recognition. In addition to the variety of related topics, the application aspect of the project, which is to create a virtual mouse, is also a significant part of the project. In order to create a virtual mouse that can be controlled through a camera source, it is critical to assign different movements/gestures to different functionalities of a mouse, such as moving the cursor, clicking left and right. Since there is no generic layout to follow for such an implementation, the project had room for improvements and upgrades and trying different structures.

2 Related Work

2.1 Existing State of Art

Computer Vision based applications for PCs is a fairly common topic that can be easily searched in the web. There is a standard framework of designing a computer vision based mouse, in which the main purpose of the application can be summarized as controlling the mouse using only webcam. The methodology of these popular projects usually depends on use of external tools which is usually colored adhesive tape or a default structure in the frames captured to provide convenience for mouse-like functionality.

2.1.1 Colored Tape Method

The reason for the use of external tools in this method is that the distinct colors of these tapes help the application to easily recognize themselves during object tracking [1]. Specifically, most of the related work functions as follows:

- Two different colored tapes are glued to two fingers of a hand, specifically on the index and thumb for convenience.
- One color is assigned to be tracked and used as the cursor of the mouse. This is encoded in the script, beforehand.
- The second color is used for the clicking action.
- When the script is run, the user sees the capture taken by the webcam. This process is not essential in the original intend of the project, but it can help the user to realize what the webcam is capturing so that the program functions properly.
  - Whenever the assigned colors that correspond to worn tapes related to moving the cursor are tracked, the script navigates the position of the hand to direct the mouse movement.
  - Similarly, when ever the clicking colored band is tracked, the script commands the computer to click, as expected.

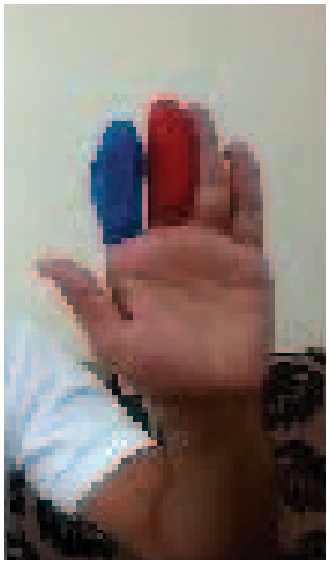

Figure 1: Example view for the colored tape method

2.1.2 Integrated Region of Interest Approach

While this is another technique of designing a computer vision based mouse that I have encountered, the project did not have the title of Integrated ROI Approach, I thought it would sound appropriate with respect to the content that the course has provided for us. This method heavily depends on the visual aid of the webcam capture [4], as explained in its rough framework below:

- The webcam capture is displayed in the screen of the user. This capture has an integrated ROI that is basically a 3x3 matrix that looks pretty much like the background of Tick-Tack-Toe.

- Each cell of the matrix has a certain function that if an object is detected within its frames, it performs the role assigned.
  - The 3x3 matrix acts similar to using arrow keys through the numpad of a computer. Specifically, the cell in the middle corresponds to clicking and the cells adjacent and diagonal to it are assigned to move the cursor through their direction, intuitive from their position.

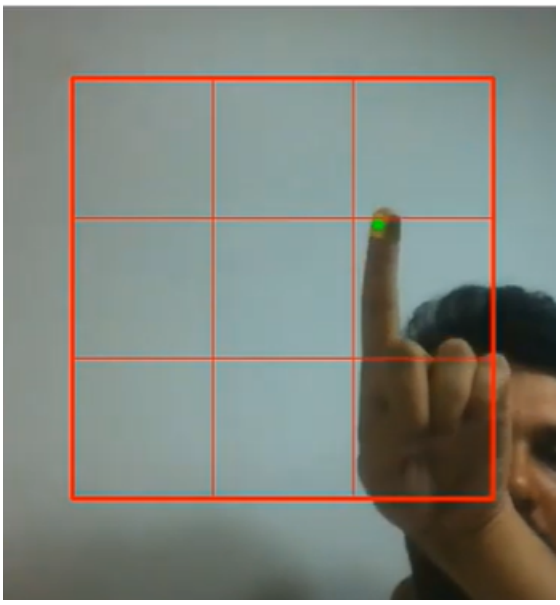

Figure 2: Example view for integrated ROI method. The method used in this picture is actually a combination with colored tape method.

2.2 Motivation for the Methods Used

The two methods mentioned in the previous section are the reason why I chose to work on this topic. Since the aim of the project is not to replicate existing methods, but understand and possibly add our own work, I focused on ideas how to work on this topic, using intuition that the course curriculum provided me. The two negative aspects I have deducted from these two projects are:

- Use of external tools for the colored tape method
- Dependence on a maintaining webcam capture due to the integrated ROI, which users require to use the application

The colored tape method requires external materials that are these colored tapes. These colors have to be included in the script beforehand, which hinders the dynamic use of the application. Furthermore, the application may perform poorly on different light settings since the range of the pixel values is pre-coded. An alternative method may include an addition which helps the application to get the histogram of the pixel values corresponding

to an ROI. This may help with the stability of the performance when the same prop is used in different lighting. However, this addition does not change the dependence of external tools, having distinct colors to be recognized easier through the camera.

The integrated ROI method takes away from the aspect that the main purpose for these applications is to navigate mouses through hand gestures captured by a webcam. Because the structure for this method requires the constant visual of the integrated ROI, this takes away from the ergonomics of the application. Thus, the capture has to be portrayed in the screen the whole time, limiting the amount of space to click in the hypothetical space of a screen.

Considering the two negative aspects I have observed, I chose to design a computer vision based mouse that depends on hand gestures. I intended to make a project that has the similar purpose of functioning as a computer mouse but not dependent on use of external tools. Thus, I have focused on two main topics which are virtual mouse and hand tracking applications. The hand tracking part offers a solution to function as the virtual mouse, which is combined with live hand recognition of how many fingers are being shown. The model does not require any training since convexity defects which allows the application to detect how many fingers are being shown is a live tracking method. Before digging into the methodology, the semantics of the hand recognition part is as follows:

- 2 fingers correspond to default cursor movement
  - Enables cursor moving mode, by tracking of the hand
  - Cursor is moved freely after the cursor gesture is recognized
- 3 fingers corresponds to "down" key for scrolling down
- 4 fingers right click
- 5 fingers left click
- Unrecognized gestures can be used for aligning the hand to the camera

Once these gestures are recognized, the script runs the function they are assigned to, which is mentioned in the next chapter.

3 Methodology

3.1 Data Preprocessing

Since this script of the project functions as a live-tracking application, it does not include a specific content for data preprocessing. Intuitively, flipping the camera and converting the instantaneous images from the capture into grayscale. The flipping sounds like an insignificant step since webcams and front cameras of smartphones already handle that, to provide the users of a mirror-like experience. However, it is included in the python script since OpenCV library functions in this way.

3.2 Model Structure

The python script appended includes both the virtual mouse and hand tracking aspects of the project. The hand tracking part is the main focus and the essential aspect that corresponds with the course content, so I will initially introduce the hand tracking application model before giving the entire picture of what is happening when the script is run.

- Conversion to grayscale
- Gaussian blur
  - Noise is removed
- Threshold
  - Active and inactive pixels
- Contours
  - Edges of the hand
  - Outer rectangle
- Convexity Defect
  - Retrieved according to defects and their angles
  - Used to calculate which gesture is shown
  - Cursor
- Gesture view
  - Helpful user interface

3.2.1 Conversion to Grayscale

This is a pretty straightforward first step common to many image processing applications that the images of the capture are converted to grayscale for further steps. I will continue referring to the capture as image since the script run through a while loop in which images of the capture are processed.

3.2.2 Gaussian Blur

As discussed in the first lecture of this course, Gaussian blurs are filters aimed to remove noise in images that may be related to a Gaussian noise. Applying a Gaussian masks helps in removing this Gaussian noise but the image becomes blurry as a result. However, removing this noise, which my computer's webcam definitely has due to its low quality, is significant, since any webcam is prone to having this noise. The blurring is not that significant for the further steps of our model since the idea is to detect the number of fingers being shown into the camera.

3.2.3 Threshold

Following the Gaussian mask, the image is binarized with respect to a certain threshold. During this process, the script uses Otsu's method that is incorporated within the OpenCV library. After this step, the image is binarized in terms of its pixel values in which the pixels are either active or not, resulting in a black and white image.

3.2.4 Contours

Using the binarized image, the contours of the hand that corresponds to the edges are received, with the aid of OpenCV, again. The problem is that the script may detect several contours that belong to other objects that is not the hand intended to be shown. To solve this problem, the contour with the maximum area is selected to be processed. This steps presume the fact that the webcam capture should be in an angle such that the background should be empty for the best performance. The retrieved contour that belong to a hand is used for getting the bounding box of the contour.

3.2.5 Convexity Defect

All the previous steps are done with relation to this particular step which includes convexity defects. The original contour and the bounding box, which is frequently referred as convexhull, are used as inputs. Using these two, the empty parts in the convexhull that is not covered by the contour of the hand are observed. The amount of convex defects that are present is the number of fingers shown minus one. Figure is explanatory of this concept, since depicting full computation of how this works is a little complicated which may take over the main focus.

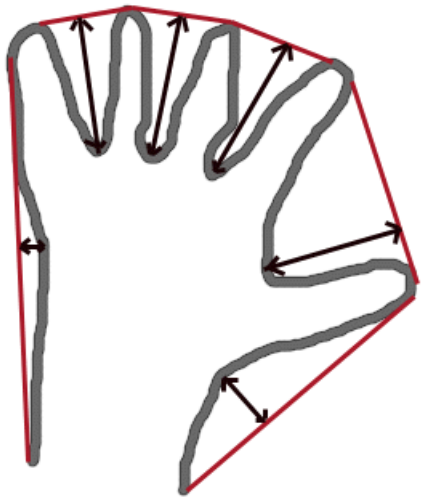

Figure 3: Visual description of convexity defects. The gray lines is the contour and the red outer box is the convex hull.

3.2.6 Gesture View

The gesture view is not a crucial but helpful extension which served as a testing method during the experiments. I have left it out since it is also a helpful visual tool to guide the user and describe how many fingers are detected. All the steps including the gesture view is processed in a while loop in the script. Once a certain number of fingers shown is

recognized, the scripts executes the function according to the semantic explanation in the motivation chapter.

4 Experimental Results and Performance of the Model

The live tracking and recognition focus on the project do not follow the traditional framework of image classifiers in which, the performance of the neural network is described through the training of the model and certain evaluations. While there are certain papers on ways to measure the performance's of object tracking, I could not implement them since the design of the model was the main focus of this project due to its object tracking orientation. Details regarding the performance of the model will be mentioned in the Discussion chapter. Below are the depictions of the several views from the python script related to this project, each labeled with the corresponding process being done.

Figure 4: Gaussian Blur

Figure 5: Binarized Image

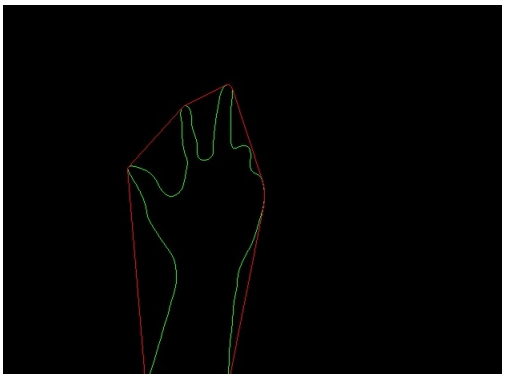

Figure 6: Contours and Convex Hull

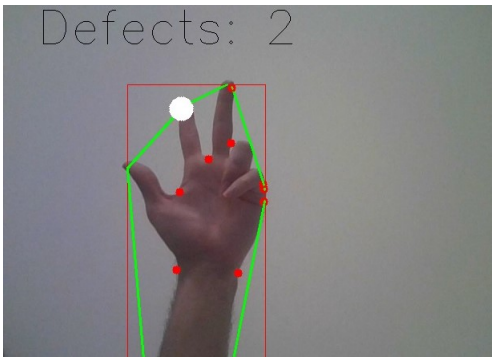

Figure 7: Gesture View

5 Discussion

It is mentioned in the model chapter that the application requires a clean background to work properly. The workflow depends on the contour of the hand and its bounding box and any image introduced can be understood as noise. This can be observed inspecting figure.

5.1 Background Removal Attempt

Having a clear background requirement is limiting, and this restricts the usage of the application. As suggested by the instructor of the course, I tried on implementing background removal algorithms. There are several papers on the effect of background removal for object tracking [3]. For this, I tried two approaches that were in parallel with the course content. However, the implementation of the background removal approaches led to poor performance for the hand recognition and tracking. The background removal was successful in terms of eliminating the background that acts like noise for the images but this also took away from the sensitivity of the application towards what it sees. Thus, the background removal techniques I have searched for and implemented are not present in the script appended. Still, I have included it in the report since I spent a significant amount

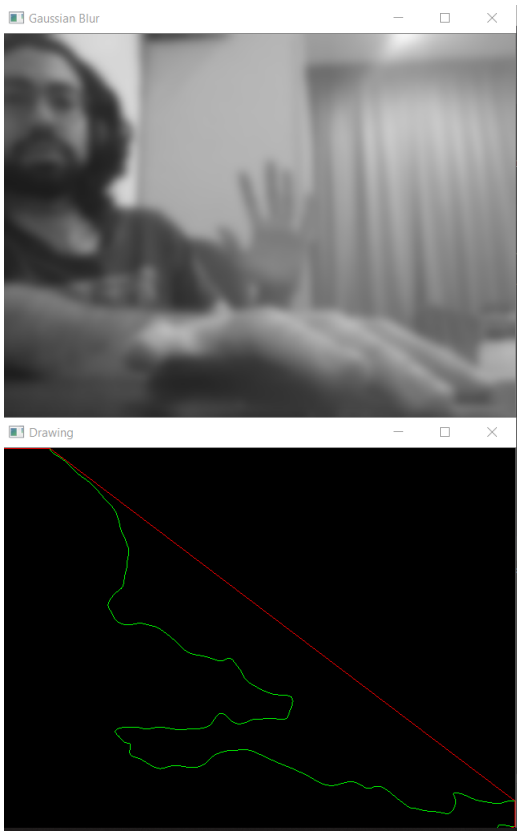

Figure 8: Effect of noise seen through Gaussian blur and contour view

of time working on them.

Two methods of background removal were applied that served a role of preprocessing. The first method applied was to eliminate the background of the webcam capture in an interactive way. For this to happen, the application is started, and a certain button is assigned as the role of background removal. This requires the steadiness of the webcam after the background removal is done so moving the camera after this step was not appropriate.

The second method I have looked into was very intuitive in which it used the histogram of hand pixels, again with an interactive approach. There are several implementations of computer vision that use this, where several small ROIs are integrated temporarily to the capture. The user is supposed to align the hand with these ROIs and the pixel values retrieved are used for background removal, thus making this method invariant to lighting conditions of the environment of the webcam. This method is quite useful for removing the background but I observed some loss of sensitivity for the recognition of the different gestures of hands, which is the main reason I discarded this part from the code.

5.2 Ergonomics

The workflow I follow is heavily related to core concept of object tracking. Some of the ergonomic issues came up towards the end of the project which I want to briefly mention as sub sections.

5.2.1 Cursor

The limitation of the cursor movement is related to the structure of the webcam and I suspect this may greatly influence the performance on other devices. Furthermore, the convenience of mouse is too much to deny when compared to the virtual mouse created for the project. Frequently, during cursor movement, the webcam's range of capture is small compared to the hypothetical space that correspond to the screen in which the cursor can freely roam. Therefore, unrecognized gestures such as showing the hand in fist form is intended to provide the user to align the hand with the camera after it is off view.

5.2.2 Clicking

Clicking is a problematic part to virtually assign to a computer vision based application that functions through action recognition. Since the script is built around a while loop, the result of showing the gesture of clicking left continues unless the recognition is lost. This results the application to indefinitely click within the time frame it is observed. Therefore, I had the intuition to keep the clicking gestures short enough yet still allowing the application to recognize.

5.2.3 Limited Gestures

Since a hand includes five fingers, this allows the convexity defect algorithm to calculate up to 4 defects. Showing a single finger is not captured as a defect so that option is the same as showing no fingers. In addition to basic functionalities of a mouse like moving the cursor and clicking, a gesture has to be unassigned in order to deal with the alignment of the cursor with respect to the camera.

5.3 Potential Incorporation of Neural Networks

The model does not include any neural networks, yet I believe there is room for such improvements. There are example projects for object recognition in which a network is trained for object recognition and the network is used for live recognition [2]. Neural networks can give much better results than the convexity defect algorithm used for the model in this project. However, there could be issues of speed since there could be some delay due to the many connections of any neural network. Alternatively, methods like YOLO which are quite fast models that do not require the convolutional structure of CNNs. The dataset needed for such training can be problematic to find, since the training examples may not correspond with the images retrieved from the webcam.

6 Conclusion

The focus of the project is to combine the virtual mouse idea with methods applied in hand tracking. Default projects include use of external tools that may not be present to users all the time and this is the intention of combining the two topics. The model used for hand tracking and recognition has many methods including filtering, histograms, binarization which we encountered partly in the Autonomous Driving and Object Recognition. While it is difficult to measure the performance of tracking methods, enhancements that are similar to YOLOs are possible due to their fast computation speed. There appears some ergonomic issues regarding the ergonomic use of the application, similar enhancements can be made such as a better implemented background removal approach that does not hinder the performance of the virtual mouse.

References

[1] @Codacus. Part 2: Simple gesture recognition to create virtual mouse | using opencv and python (tutorial), 2017.

[2] @Codacus. Contextual attention for hand detection in the wild, 2019.

[3] Shireen Y. Elhabian, Khaled M. El-Sayed, and Sumaya H. Ahmed. Moving object detection in spatial domain using background removal techniques - state-of-art. *Recent Patents on Computer Science*, 1(1):32–54, 2010.

[4] @Skyfi Labs. Computer vision based mouse - skyfi labs online project-based course, 2018.