Computer Vision Final Project Report

Attempt at Tracking and Recognizing Hand Gestures in Real Time

Team Members: Javkhlan-Ochir Ganbat, Ganga Reddy Tankasala

Initial project idea

Our initial idea was to track a hand movement, specially the fingertip and recognizing various hand gestures. We did not set any constraints to our problem which caused us unexpected troubles such as different background and different illumination.

Technical Component

- A laptop with a webcam
- OpenCV library on Python
- Numpy for intensive calculation

Basic Processing outline

- Background subtraction
- Skin detection
- Locate motion
- Match the gesture
- Track the hand across the screen

Background subtraction

We tried out two background subtraction method. One is background subtraction method using mixture of Gaussians, unfortunately we were not able to implement it but OpenCV already had an implementation so we used that function.

Second method was to keep track of last k (k = 10) number of frames in the buffer. From there, we calculated the variance of each pixel and if that pixel varies more than some threshold value T (T = 2) then we consider it to be a movement or foreground. But this approach also detects any movement in the frame; therefore we run a skin threshold method on the current frame and isolate only the skins. Thresholding is calculated in HSV color space which seems to be the best color space to detect human skin. We searched through the academic articles and experiments to find the most optimal skin threshold value. Below is the common skin threshold value in HSV space.

Lower bound = [0, 40, 80] and Upper bound = [20,155,255]

Once we isolate the skin in the picture we logic AND the frame with motion and frame with skin to get the final binary image containing only the moving skin part. Here, we assumed that the only thing moving is the hand.

Studying the hsv values using -- Hand shake learning

We have employed another different approach to detect the skin part. In this process the user waves hand as a gesture in front of the camera. The consecutive m frames of images are read and the pixels are thresholded with respect to the variance.

The mean and std of the remaining skin pixels are identified in HSV frame. Either 2 sigma or 3 sigma range of values are applied as the limits when detecting the skin in real time. This process will ensure that the users hand is identified despite the differences in skin tone. This process can be extended to other physical objects as well.

The main disadvantage with this method is that shadow due to the hand in the background alters their pixels which in turn show up as pixels which are moving. This might cause the background pixels to be detected as pixels of interest.

Locate motion

Once we isolate the moving parts we need to find its location for tracking. We had two approaches for motion location. First one is the simplest approach, calculate the central moments and get the centroid of the binary image. From there we estimated the movement region using the following method.

- Calculate the distance of each pixel's x distance from the centroid
- Calculate the distance of each pixel's y distance from the centroid
- Set the upper left corner of the rectangle to coordinate that has distance from the centroid equal to 90th percentile value of x distance and y distance.
    - For example: centroid is at (50,50) and x distance's 90th percentile is 4 and y distance's 90th percentile is 6 then upper left corner is (46,44) and width is 8, height is 12

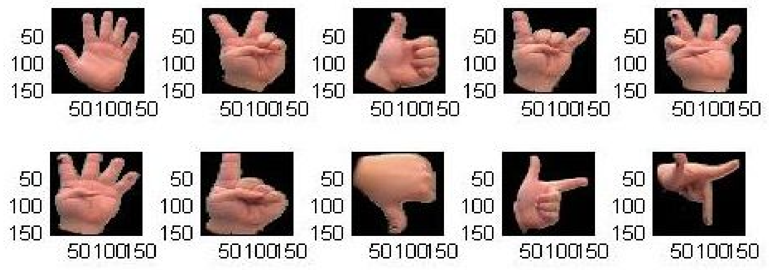
Another way is to use OpenCV function findContours to find the biggest contour. Then get the bounding box of that contour using OpenCV boundingRect() function. This approach is much more accurate than the first approach.

Matching gestures

Once we localize the motion and isolate the region from the rest of the frame, we tried two approaches to match the gesture.

First approach, we used OpenCV's findContour function to find the largest contour from the isolated image and we assumed that will be the hand. From there, we ran a spatial curvature calculation and selected curvatures above the 90th percentile. Unfortunately this approach has a lot of drawback. Due to the light fluctuations, contour of a hand cannot be always consistent and smooth therefore even tiny ripple in the contour will completely mess up the contour calculation because those ripples have the largest curvature. However this approach seems to work partially when we used background subtractor with mixture of Gaussian which generates really well defined smooth hand.

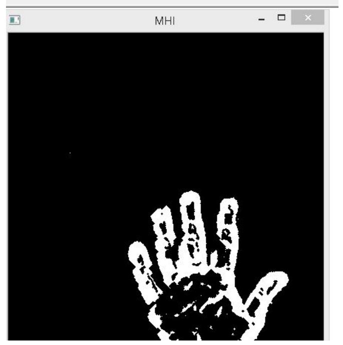
Second approach is to isolate the hand image and match them against the predefined templates using common template matching algorithm. All the predefined gestures are taken in the same environment with same lighting therefore we believe illumination in the templates do not affect the matching process. Below is the gestures we used.



We implemented Sum of Absolute Difference (SAD) and Sum of Squared Difference (SSD) for their simplicity and fast vector calculation on Numpy.

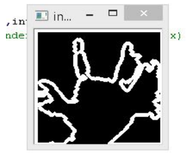
Result

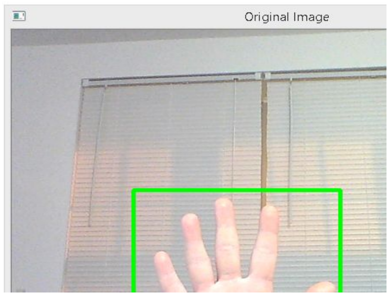
We have tried out numerous methods and even though the result were not promising but the most successful one was when we use background subtraction using temporal history and skin detection. Below is the output after we apply background subtraction. Advantage of this method is that it can detect the slightest movement and people usually do not have perfectly still hand so user seems to be holding still but we still see the hand.



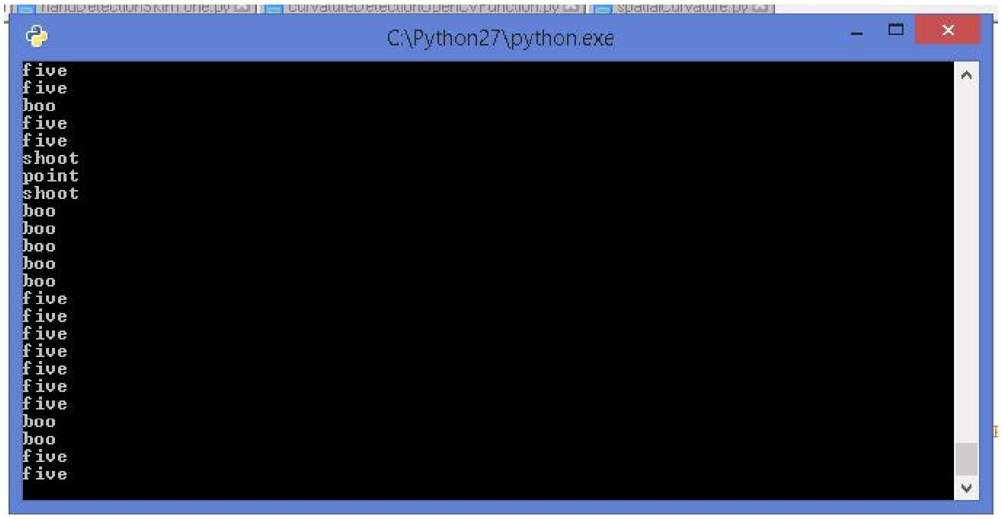
(Screen after background subtraction)

This motion area in the original image have green bounding box around it and we extract that part and apply Canny edge detector to get the general shape of hand template to be matched against the gesture images.



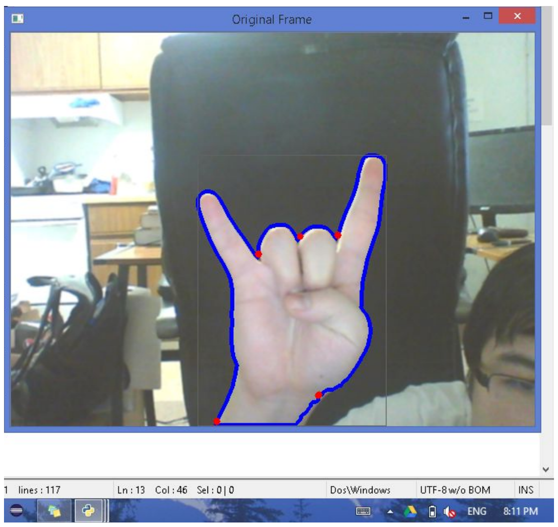


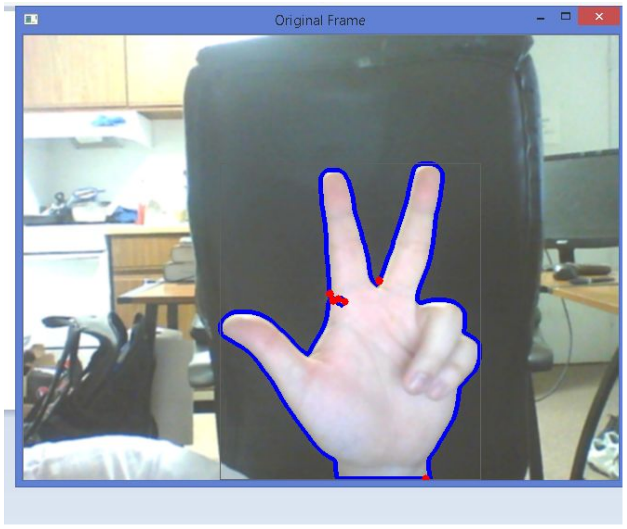
(Left: Template image after Canny Edge Right:Real time frame)
The hand in the picture shows five finger and below is the result. Even though it is not 100% accurate we can still see that it detects the gesture.(Correct gesture is five)



```
five
five
boo
five
five
shoot
point
shoot
boo
boo
boo
boo
boo
five
five
five
five
five
five
five
boo
boo
five
five
```

Our spatial curvature approach unfortunately failed due to being too sensitive to the environment. For example:
Left: Contours are very smooth so spatial curvature can be found very well.
Right: Due to small ripple, that region has really high curvature points and other important regions lacking correct curvature.





Unfortunately we were not able to fix this issue and therefore was not able to implement the gesture matching part using spatial curvature.

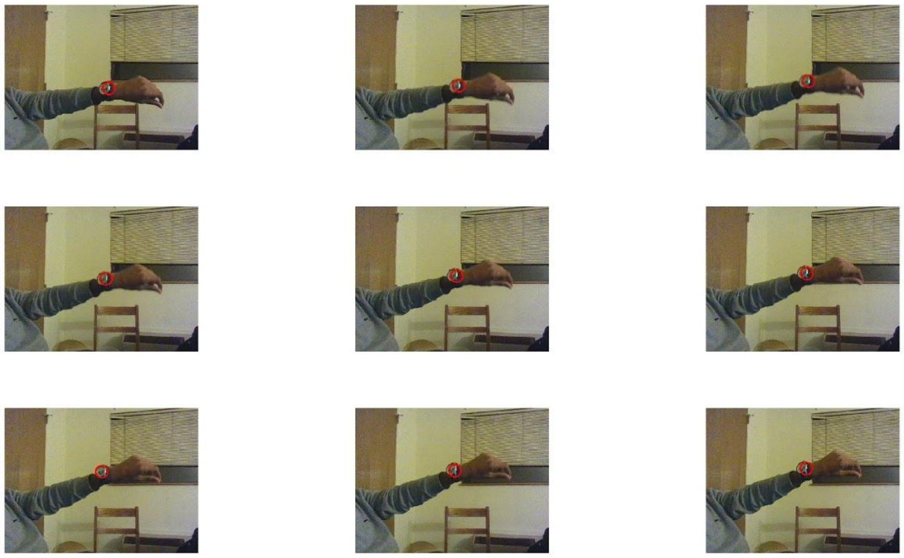
Hand Tracking

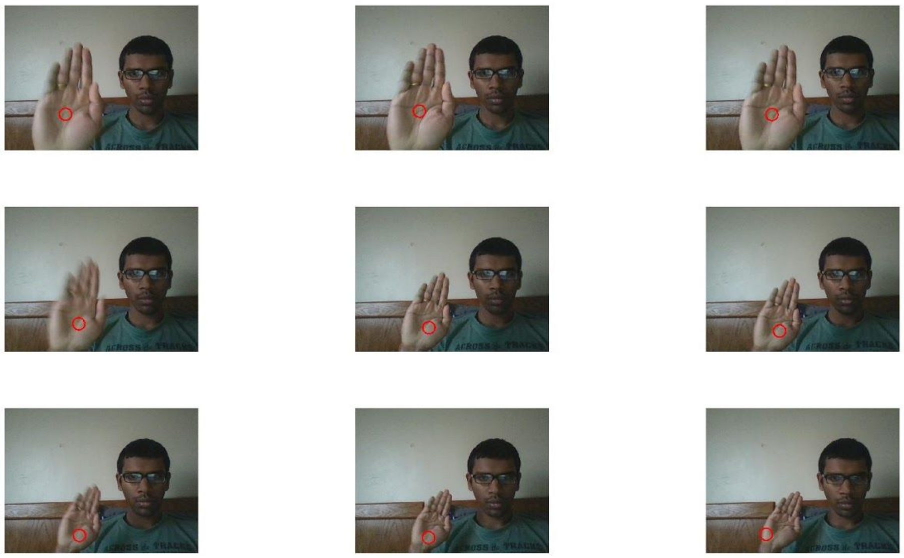
Tracking the entire hand across the screen was simple since we already located its centroid. But we also tried to track it with MeanShift algorithm; we converted our MeanShift algorithm in MATLAB into python and used it, instead of OpenCV's native implementation.

Mean shift tracking

We have implemented mean shift tracking , but we have faced the below problems

1) Fast hand movements -- the marker would loose the location of hand
2) Most of the hands color spectrum was similar, though it can track the hand, any particular part of hand like fingers , etc can't be tracked.
3) Computationally expensive compared to the above method.





When Mean shift was implemented for tracking the hand watch , it was very effective. But when the center of hand was tracked , it oscillated randomly around the hand (due to no variation in the pixels). When applied around fingers, it couldn’t track it effectively.

Scope of future work

Hand gesture and motion can be tracked at a very cheap computational cost using the similitude momennts and other statistics. We intend to work on exploring the range of values of similitude moments of gestures. As well as implementing the usual approach people employed in gesture tracking using convexity defects. Applying machine learning models on top of the motion will be our other interest to work on

Conclusion

Here we tried different approaches in hand gesture recognition. We want to have as much freedom therefore not restricting lighting source and partially restricting background. Even though the idea was very simple soon turned into a very complex problem. However our approach runs in real time and have certain degree of freedom.