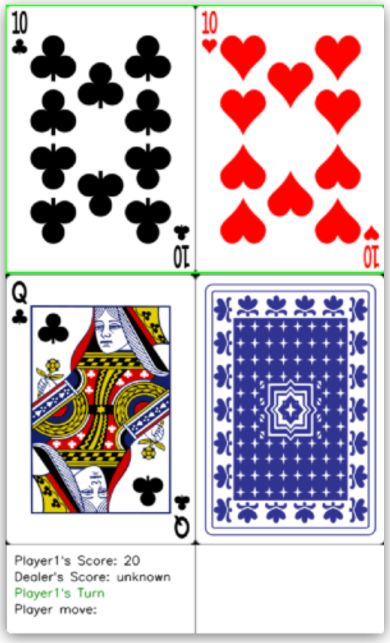
Real-time BlackJack Hand-signal Interface

Visual Interfaces
Min Kim, jk4449
May 1, 2023

OVERVIEW

Summary of the Final System

The final system is a 1-player Blackjack game that uses the user's hand gesture as the input. Once the user starts the game, they are asked to make a hand pose for "double down" and "split", which will be saved in the system. Then, the game begins. Game play of blackjack is shown via computer screen, and every time the user has to make a decision, and the camera will turn on and the system will recognize the user's decision, and continue the game depending on that input.

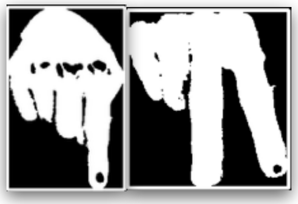


```
Player1's Score: 20
Dealer's Score: unknown
Player1's Turn
Player move:
```

Final System's Interface

Gestures that the Interface Recognize

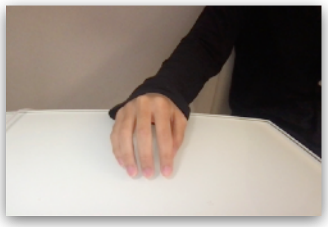
- Dynamic Gesture
  - Hit: Double tap (recognized by the vertical movement of the hand)
  - Stand: hand waves horizontally twice over the table. (recognized by the horizontal movement of the hand)
- Static Gesture
  - Double Down: make number 1 with the finger, and place it in the table. (recognized by the hand pose and lack of movement of the hand)
  - Split: make a V shape with fingers by separating index finger and middle finger, and place it in the table. (recognized by the hand pose and lack of movement of the hand)



Left: Double Down
Right: Split

Domain Engineering

The system works in most settings where there is sufficient light, background is stationary, and the color of the table and the background is not too similar to the user's skin color. One way to ensure that the system works well is to turn off the lights in the room, block natural lights, and only have one light source (ex. lamp) that shines the light at the user's hand. User can also check if the system is working well. When first starting the system, it will show the binary image that is processed by the computer, and the user can adjust the background and lights to make sure that the binary image shows and tracks the hand clearly before starting the game play.



Ideal Capture

Packages Used

I used OpenCV package to access the computer's camera, detect skin, draw contour, compare the contour to check if the real-time hand matches the key pose, etc. I used numpy package to handle the blackjack display. Specifically, I used numpy.concatenate as I had to stack images of cards on top of each other. I used random package to shuffle the deck in the blackjack game, and the time package to delay actions in blackjack game to allow the users to follow the game in a speed that they would be used to in the real-life game.

METHOD & THRESHOLDS: STATIC GESTURES

Double Down Recognition Function

This is the function used to recognize “Double Down”

Accept if all 4 criteria are met:
1. Currently identified hand contour is closest to an example of “double down” contour.
2. The difference between the current hand contour and that example is below a threshold (key_gesture_threshold).
3. The previous two criteria has been met for a threshold (stay_still_threshold) amount of time.
4. User’s hand has not been moving for a threshold (stay_still_threshold) amount of time.

Split Recognition Function

This is the function used to recognize “Split”. It is very similar to “Double Down”. The two gesture uses a different key_gesture_threshold.

Accept if all 4 criteria are met:
1. Currently identified hand contour is closest to an example of “split” contour.
2. The difference between the current hand contour and that example is below a threshold (key_gesture_threshold).
3. The previous two criteria has been met for a threshold (stay_still_threshold) amount of time.
4. User’s hand has not been moving for a threshold (stay_still_threshold) amount of time.

Finding the Closest Hand Contour

I noticed that the system does not recognize the gesture well if the wrist is slightly turned to a different angle, etc. Therefore, I made it so that the system records 3 contours for each gesture: one with low angle, one with medium angle, and one with high angle. All three should be facing the camera directly, and any attempted gestures must face the camera directly as well for better recognition. When finding the closest hand contour, the system would simply search through all recorded capture and if one of the three “split” happens to be the closest, it will return “split” and vice versa.

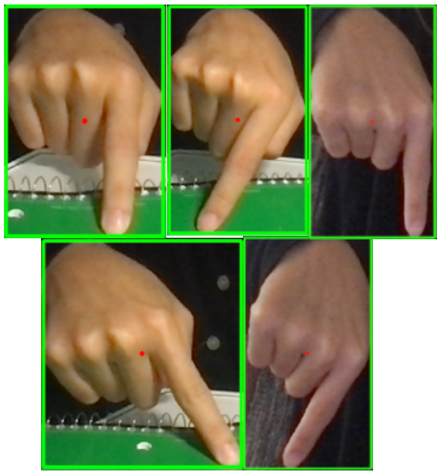
Key Gesture Thresholds

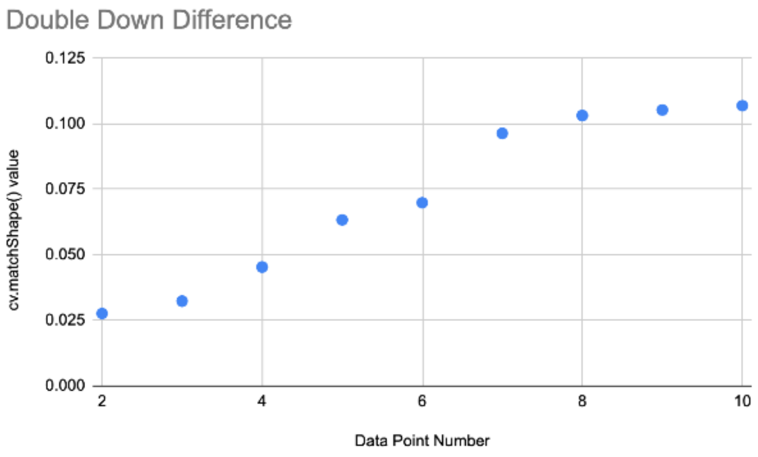
```
key_gesture_name = ["doubledown", "split"]
key_gesture_threshold = [0.12, 0.15]
```

Key gesture threshold is used in the identification of “split” and “double down”. A function (cv2.matchShape) is called to compare the difference between the real-time hand pose and key gestures that we are looking for (“split” and “double down”). If the difference is smaller than a threshold, the real-time hand pose can be identified as that key gesture. If multiple key gestures are under the threshold, the real-time hand pose is identified as the one with the smallest difference. If none of the key gestures are under the threshold, the real-time hand pose is identified as “none”.

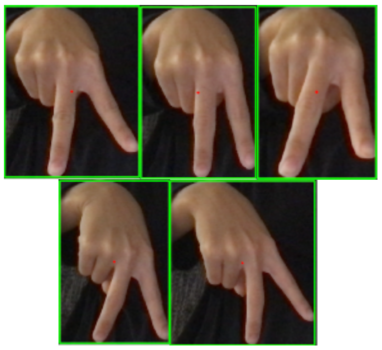
The system was first built using the same threshold for all key gestures. However, I noticed that some gestures are harder to be recognized than others. Therefore, I conducted a small survey.

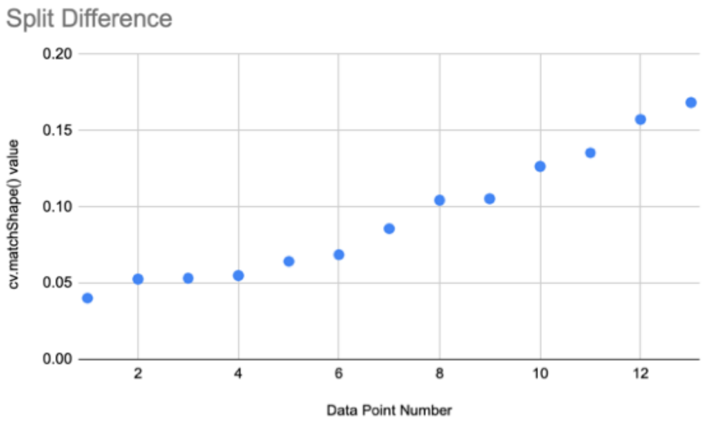
I gestured “double down” in various ways rotating my wrist, and recorded its difference to the “double down” contour. Below are the example poses of “double down”, and the result plot. As you can see in the plot, all datapoint fell below 0.12. Therefore, I decided to make the key_gesture_threshold for “double down” 0.12.





Split was more difficult to recognize. I was able to vary not only the wrist, but the distance between the two fingers. The little twists and small adjustments made the split difference higher on average. Below are pictures of sample “split” poses and the graph. Since most points fell under 0.15, I decided to make the threshold 0.15.





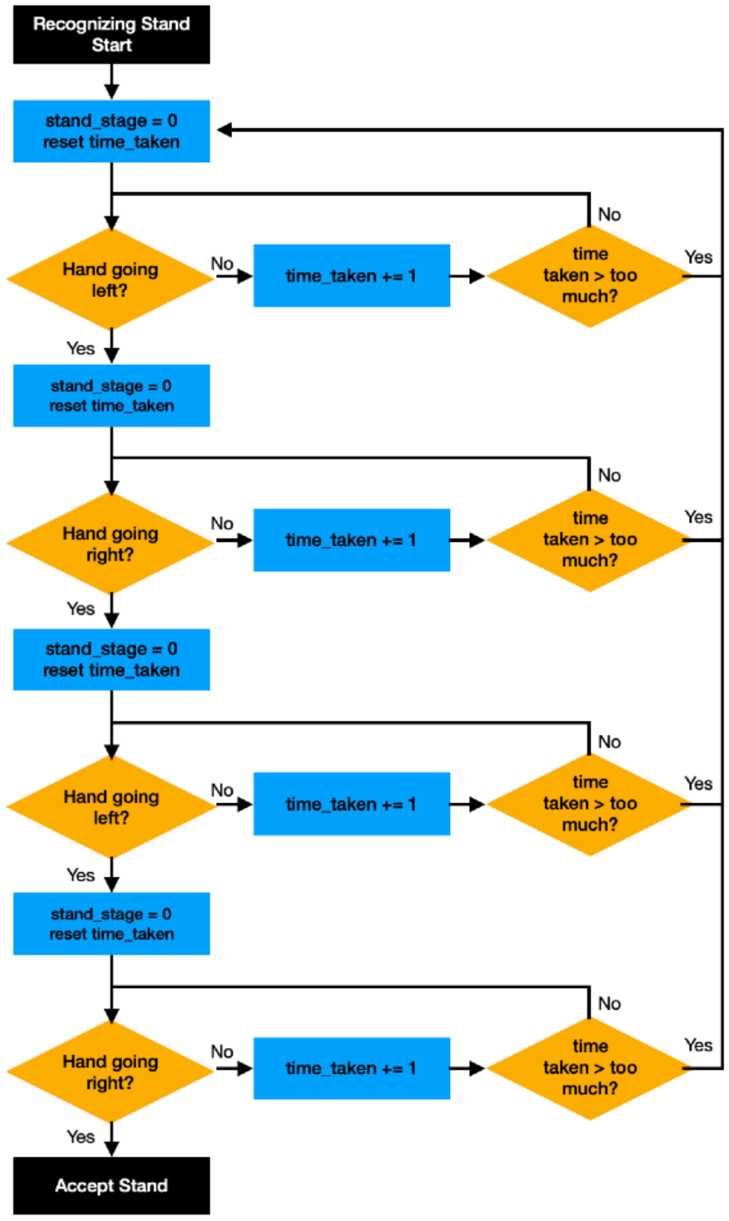
Stay Still Threshold

For a stationary gesture to be recognized, the hand pose should stay still for Stay Still Threshold amount of time. The threshold should not be too short that it recognizes a gesture that it not intended to be recognized, nor too long that it makes the user wait too much. I determined this threshold by first setting it to about 2 seconds, and gradually lowering it while using the system. I stopped lowering the threshold when the system recognized a stationary gesture when I was not completely still posing and waiting for it.

METHOD & THRESHOLDS: DYNAMIC GESTURES

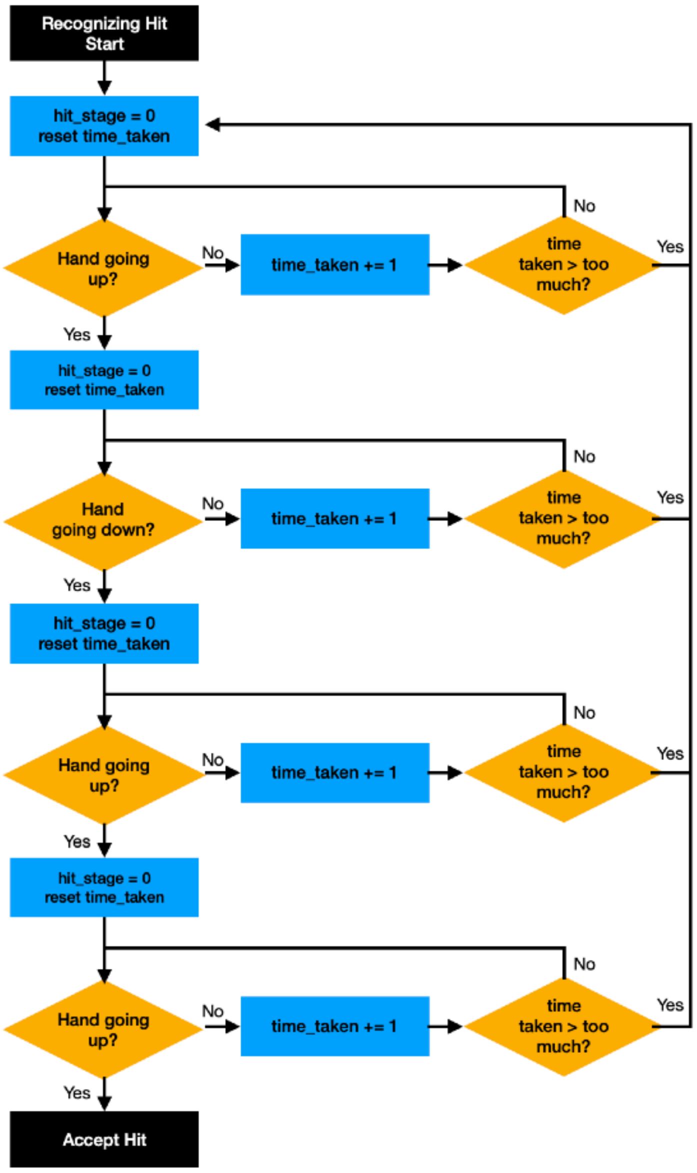
Stand Recognition Algorithm

This is a flowchart that shows the decision process of identifying the gesture “Stand”.



Hit Recognition Algorithm

This is a flowchart that shows the decision process of identifying the gesture “Hit”.



Hit/Stand Time Threshold

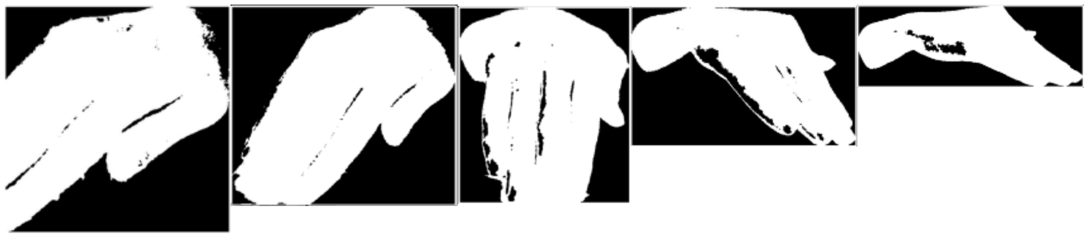
This threshold is responsible for enforcing that the up/down, left/right motion happens in a timely manner. If stand/hit stage does not go from one stage to the next within this threshold, stand/hit stage will be reset to 0. I noticed that the gesture “stand” had a lot of false positives, because the user can move their hand around unintentionally, and it had vertical movement component to it. When the user intends to and gestures “hit/stand”, they do so in a quick and tight manner (ex. stand is a quick left right left right instead of arm reaching for something in the table). This information can be used to determine hit/stand from unintentional hand movements. I started from high threshold and gradually lowered the threshold, until the system was not recognizing a true “hit” or “stand”. If this threshold is too low, there simply is not enough time for the stages to move from one to the next, hence the dynamic gestures will not be recognized.

Hand Dir Threshold

Hand direction is recognized when the user moved their hand "significantly" from their last position. I computed the movement from position A to position B by the movement in the center point of the bounding rectangle of their hand contour. That "significantly" is determined by this threshold. When I was testing the system by myself, I would move my hands precisely, so the system worked well with low hand dir threshold of 20. However, when I was testing the system, I noticed that many people struggled because while they move their hand vertically, gesturing "hit", it was often recognized as "stand" since the vertical movement of the hand usually accompanied horizontal movement. Therefore, I decided to make two different threshold for detecting vertical movement and horizontal movement, and set the horizontal hand direction threshold to 40. When I made this change, a participant who could only get the system to correctly recognize "hit" 1 out of 10 times was now able to get the system to correctly recognize "hit" 10 out of 10 times, and "stand" 10 out of 10 times.

Hand Going [Up/Down/Left/Right]?

The "hand going [up/down/left/right]?" decision function in the flow charts above utilizes the information of past record of hand directions. If the vertical direction was "up" 95% of the times in the past [hit_time threshold] records, and if the horizontal direction was not "left" nor "right" 70% of the times in the past [hit_time threshold] records, the system will say that the hand was going up. Vice versa for "down". However, for "hand going left/right", the system will not check if there was a vertical component to the movement. This is because by observing the participants, I noticed that it is very hard to not move vertically while moving horizontally, especially since the hand is off the table. Also, the shape of your hand changes during the gesture "stand" which means that there is movement of the center of the hand contour independent of the hands's movement. The hand direction is largely dependent in the center point of the hand contour, so it is very rare for even the intended "stand" to have no vertical movement component. Below is a capture of intended "stand". We can see that the center of the bounding rect moves up as the hand moves the the right.



Skin Recognition Threshold.

Last but not least, one the most important part of the algorithm is accurate hand detection. Research by Shaik, Khamar Basha, et al.[1] show that it is more efficient to work in the YCbCr color space than the RGB color space. YCbCr color space does not mix the color with intensity, and hence can have better skin detection under uneven lighting conditions. Therefore, I worked in the YCbCr color space to detect skin in this project. Research by Kolkur, S., et al.[2] has come up with a well performing threshold for human skin detection. I have used their YCbCr color space lower bound, and came up with my own YCbCr color space upper bound after trial and error. The thresholds are as follows: skin_lower = [80, 135, 85], skin_upper = [200, 200, 150].

TESTING

I tested the system with 5 participants. Each participants in the test was briefly introduced to the system, and the rules of Blackjack if they did not know before. Then, they interacted with the system until at least 10 hits and 10 stand were attempted. For each interaction, I wrote down what the participant intended and what gesture system recognized it as. Because “split” and “double down” is much rarer to use these gestures in the game naturally, those gestures were attempted afterwards separately so that a total of 10 splits and double downs were attempted.

Accuracy and Precision

	Hit	Stand	Split	Double Down
Accuracy	0.96	1	0.7	0.72
Precision	0.89	0.74	0.9	0.92

Above is a chart for accuracy and precision for each gestures. The system recognizes hit very well, as it has high accuracy and precision. Stand has high accuracy but low precision: the system over recognizes it. Split and double down both have lower accuracy but higher precision.

To talk about the low precision of “stand”: one thing to note is that the user will rarely gesture split or double down, since those decisions can only be made in a specific situation (e.g. when the user is dealt a pair, when the user wants to double the bet). User will mostly gesture hit and stand. Therefore, although stand has lower precision, it enables the system to recognize Stand quickly and easily, which is beneficial to the users. Unfortunately, the table above does not show the amount of time it took for the system to recognize the user’s input. By observation, stand was recognized instantly, followed by hit, which was recognized within a reasonable time frame. There were multiple incidents where the participant was gesturing “split” but the system did not recognize it. Therefore, the participant started to move their hand around to attempt the gesture in a different angle, and resulted in the system recognizing “stand”.

Perceived Ease Of Use (PEOU)

Question	**Response (out of 7)**
Learning to operate the system would be easy for me	6.6
I would find it easy to get the system to do what I want it to do	5.8
My interaction with the system would be clear and understandable	6.8
I would find the system to be flexible to interact with	6
It would be easy for me to become skillful at using the system	6.4
I would find the system easy to use	7

The participants filled out PEOU survey after they were done with the testing. Following is the result with the average value for each questions. The response is in the scale of 1 to 7, where 1 is unlikely and 7 is likely. It is understandable that many participants did not find it completely easy to get the system to do what they wanted to do, as the precision and accuracy was not perfect. I think the project was a success because most participants found the system very easy to learn and easy to understand, and all participants thought that it was absolutely easy to use. Although the precision and accuracy was not perfect, the goal of the project is to make an enjoyable and interesting game for the user, so I have fulfilled my goal.

REFLECTION

From this project, I learned the importance of threshold and how to test for them and to set them properly. It was very motivating when changing a threshold suddenly made a gesture recognizable. I debugged with all values that would be compared to the thresholds visible, so that I would have a sense of where the thresholds should be. I also learned the importance of testing. When I was testing the system myself, everything worked well and all gestures were recognized almost perfectly because I have done the gestures so many times, and I know exactly how these gestures are recognized. However, when the participants tested the system, I was able to realize a lot of flaws in the system that I could not see before because the participant did the movements in a way that is different from mine, but still within a boundary from the gesture's definition. For example, I described to the participant that they had to stay still after gesturing "split". However, most participants thought they should stay still or less amount of time then I build the system to wait for. As such, in this project, a lot of progress was made between the testings when I realized the "crowd consensus" can be different from my opinion.

I also learned a lot from trying to build a system that has to make a decision in real-time. This pushed me to be simple in my approach, so that the final product does not run too slow. I originally thought of implementing media pipe, but did not use so because it would slow down the system so much. In order to make the system run without a glitch, I had to sketch out and think through different course of actions and come up with a way that would minimize computation and comparisons. Also, I had to come up with a way to implement the decision process flowchart inside a constantly repeating loop, which was challenging and rewarding.

REFERENCES

[1] Shaik, Khamar Basha, et al. "Comparative Study of Skin Color Detection and Segmentation in HSV and Ycbcr Color Space." *Procedia Computer Science*, vol. 57, 2015, pp. 41–48., https://doi.org/10.1016/j.procs.2015.07.362.
[2] Kolkur, S., et al. "Human Skin Detection Using RGB, HSV and YCbCr Color Models." Proceedings of the International Conference on Communication and Signal Processing 2016 (ICCASP 2016), 2017, pp. 324–332., https://doi.org/10.2991/iccasp-16.2017.51.