International Journal of Research in Engineering, Science and Management
Volume-1, Issue-11, November-2018
www.ijresm.com | ISSN (Online): 2581-5792

Application and Analysis of Different Gesture Recognition Techniques Used to Create a Tic-Tac-Toe Game

Ishan Singh Malhotra
Student, Department of IB, Jashree Periwal International, New Delhi, India

***Abstract*: Gesture recognition is growing field which focuses on computers interpreting human gestures, especially those done by the hand/face via mathematical algorithms, to produce an action. Its advantages are shared by many other industries, and especially the gaming sector.**

***Keywords*: Gesture tracking, analyzing, and algorithm**

1. Introduction

A user can interact with a gaming platform without physical contact, thus building a richer bridge between the two than primitive console or keyboard based versions. However, isolating a person's movement from an image is extremely difficult if the background conditions aren't suitable. In order to address this problem, there are numerous methods by which one can track gestures. In my research, I have explored three gesture recognition techniques, which do not require large training data or machine learning algorithms, to play Tic-Tac-Toe. Three main techniques I investigated for this project are:

- Gesture Tracking through Color
- Gesture Tracking through Haar Cascades
- Gesture Tracking through Hand Contours

Similar to Glove-based-tracking, Gesture Tracking through Color involves showing a particular color (Red or Green) to play the desired move ('X' or 'O') on the Tic-Tac-Toe board. Gesture Tracking through Haar-Cascades involves direct hand feature detection by the program that's allows one to mark his or her move. Gesture Tracking through Hand Contours marks a contour of the human hand after heavy pre-processing, and uses it as a marker to track gestures. After analyzing the different methods, I came to various conclusions about the accuracy, the robustness, and the limitations of each the methods. I found that the Gesture Tracking through Color seemed to be the most consistent and robust. Although the other the techniques, though, were quite accurate when tested in required conditions, their accuracy was volatile to changes in the surrounding environment.

2. Back ground information

Haar feature-based cascade classifiers is an effective human feature detection method proposed by Paul Viola and Michael Jones. This methods works on machine learning. The program learns about human features by training with scores of images. In the algorithm, features are found based on finding Haar-like features in the image.
The machine learning program learns that certain human features correspond to certain Haar-like features.

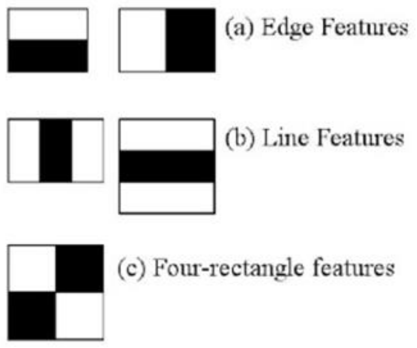


Fig. 1. Haar features

After the training is done the program learns that some features can correspond to some human features, like the eye-brows could be represent edge features and eyes could represent line features. After this, cascade files are generated, each of which have information about Haar features and what they represent as human features. These files are used to find the same human features in a real time images. As each classifier in the cascade agrees that some location has the feature it is trained for, it passes on to the next in the sequence, until the certainty of the feature raises above some threshold and it's confirmed that the particular human feature has been detected. I have used this technique to detect the human palm and fist in my project.

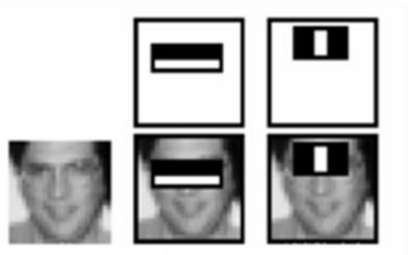


Fig. 2. Finding the eyes using Haar-features

3. Related work

In the studies of computer vision, Gesture recognition is widely used in different fields to carry out different tasks. Many companies have tried to integrate it into their gaming consoles (for example Myo Arm band) to improve gaming experience. Additionally, they have been used by medical agencies to make electronics that can be used by the disabled. Gesture Recognition has been used in the electronics, household, etc sector as well. A lot of private research has also been done in this field. Sushmita Mitra, in his paper Gesture Recognition: A Survey, yields insight into different algorithms particularly used for recognising gestures. It talks about the hidden Markov models, particle filtering and condensation, finite-state machines, optical flow, skin color, and connectionist models. The paper focuses on hand gestures in particular. Although the paper spoke about machine learning algorithms which I didn't include because of a paucity of time, I did learn about different methods with which I could preprocess the background. (Mitra, 2007). One of Gesture recognition key methods is to subtract the background. Deepjoy Das and Sarat Saharia, in their paper Implementation and Performance Evaluation of Background Subtraction Algorithms, talk about different background subtraction techniques: Frame difference, real time background subtraction and shadow detection technique and Adaptive background mixture model for real time tracking technique (Das et al, 2014). Even Suraj Pramod Patil, in his paper Techniques and Methods for Detection and Tracking of Moving Object in a Video, discusses different methods to segment background from a video frame. After weighing all these algorithms, I was able to decide which I should use in my project. (Patil, 2016) In my project, I did end up experimenting with Frame differencing and real time background subtraction to get better results.

4. Solution

My solution is broken down into the three sections in each method:

- Background simplification,
- Gesture recognition
- Gameplay

For each of the aforementioned sections, I shall discuss how I have been able to accomplish them in the three gesture recognition methods (using Color, using Haar-Cascades, Using Contour)

Section 1-Background simplification: One has to enhance the some features of an image and depress others before it can be processed. This is done by making changes to the background. Each of the three method requires a different type of simplification because they all work differently.

A. *Method 1: Gesture tracking through color*

This method tracks gesture by two colors — red and green. Thus I am differentiating those two colors in the images. Fig. 3, shows the Flowchart of how background is pre-processed.

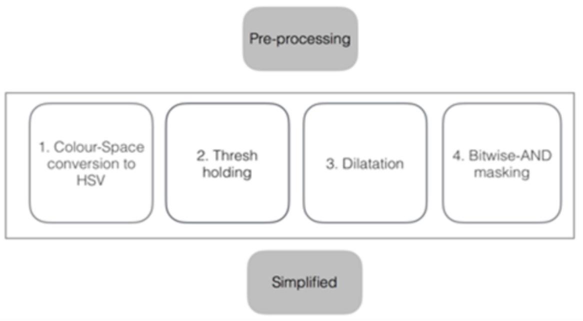


Fig. 3. Background simplification flowchart

1) *Color Space conversion to HSV*

There are numerous ways of representing color in an Image: BGR, HSV, etc. We choose to convert it into HSV (Hue, saturation, luminance) because in the BGR format, the B, G and R components of an object's color are in accordance with the amount of light hitting the object in real time; they change as the lighting changes. This makes object/color discrimination quite hard and inaccurate. Describing the image in terms of hue/lightness/Chroma seems more logical and relevant, because it is invariant to change in brightness.

2) *Thresh-holding for color extraction*

Two colors are used to mark a gesture — red and green. Thus, a range is defined for the two colors using their HSV value. Only the colors whose HSV values satisfy the range are accepted; the remaining colors are removed.

3) *Dilation*

Dilation increases the object size by filling up the holes (caused by random noise) in the image and can underscore the features. This will reduce subtle errors contribute in the accuracy.

4) *Bitwise-AND masking*

Masking creates a layer over the colored image that all allows us only the unmasked red or green portions. This is the end of the pre-processing.

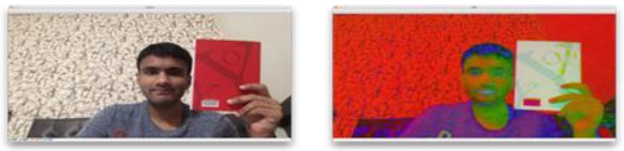


Fig. 4. Original frame and HSV color space

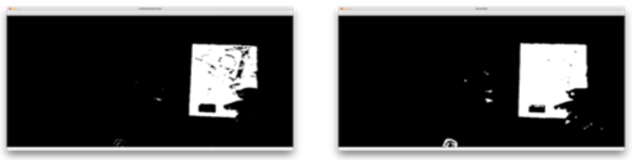


Fig. 5. Thresholded image and dilated frame

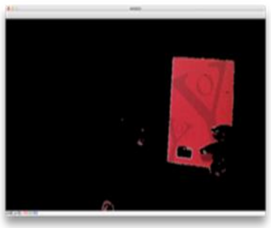


Fig. 6. Bitwise-AND Masked Frame

B. *Method 2: Haar cascades based gesture recognition.*

This method recognizes the human palm and human fist to mark gesture. Fig. 7, shows the Flowchart of how background is pre-processed.

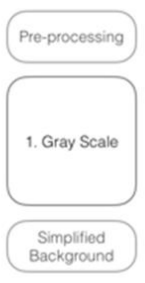


Fig. 7. Flowchart of Pre-processing

1) *Gray scaling*

To identify Haar-like features, one needs use only a single channel image. Thus gray scaling eliminates irrelevant color information, converting colored image to a gray-scale image which only uses one channel. Now the detect MultiScale() function can be applied to find feature location.

C. *Method 3: Hand contour based gesture recognition.*

This method tries to single out only the hand from the background. This method is expected to be used in a dark room, preferably where the only light shining is from the computer. Due to the heavy dependence on background environment and lighting, the following method extensively simplifies the background.

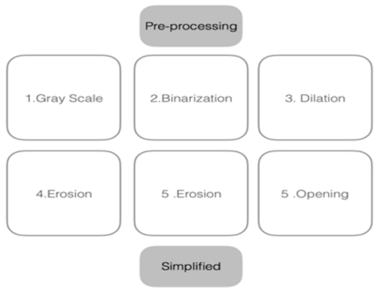


Fig. 8. Background simplification Flowchart

Gray scale: Gray scaling turns a colored image into a black and white one. This is extremely necessary because if we deal with a 3 channel image, finding contours will become harder.

Binarization: This applies a threshold to a Gray Scaled image, meaning that if a value of a pixel is greater than the threshold, it would assign it the max value(white), and otherwise it will assign zero(Black). This is really important as to find the contours of the hand, the image frame should only recognize the hand and disregard the background.

Dilation: Dilation increases the object size, filling holes in image and underscoring the features. This will reduce subtle errors caused by random noise and will contribute in the accuracy.

Erosion: It erodes away the boundaries of foreground object, keeping the foreground in white.

Blurring: Blurring is used to smoothen the image which is helpful for removing noise and high frequency content (noises, edges) from the image.

Opening: Opening is essential for noise reduction. It allows us remove false positives, so one can focus on the bigger picture, which is the hand in our case.

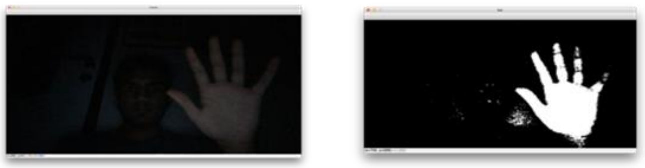


Fig. 9. Original Frame and Gray-Scaled

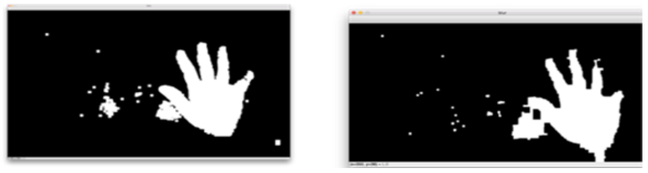


Fig. 10. Binarized Image and Blurred Image

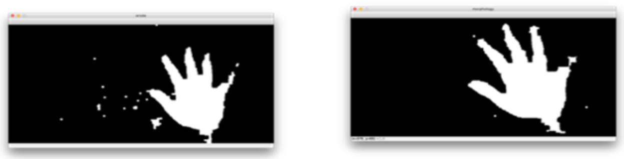


Fig. 11. Dilated and Eroded after opening flowchart of pre-processing

Section 2 - Gesture Recognition: In this section, I will discuss how gestures are recognized by the three methods, how the location of the gesture is determined, and how the move is registered and an 'X' or 'O' is finally drawn accordingly on the Tic-Tac-Toe board.

D. *Method 1: gesture tracking through color*

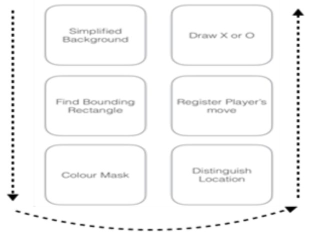


Fig. 12. Flowchart for Gesture Recognition

E. Simplified background, color mask

After the background is simplified two frames are created, one for each player, which are masked entirely expect for the portion having red or green color respectively

F. Find contours

The find Contours function is used to determine the shape and size of the colored object in the image being used to do gesture control. A rectangle is drawn around the found contour. Only the largest contour is taken into account and the other smaller rectangles are disregarded. This makes sure that other small pieces of red or green color in the image frame do not hinder the gesture recognition process.

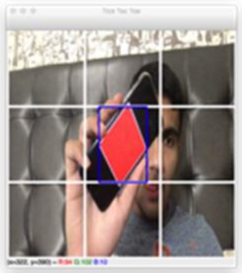


Fig. 13. Finding contours on Grid

G. Distinguish location

The Tic-tac-toe playing board is a 450 * 450 pixels square box. It has been divided into 9 further boxes. And each box is given a given a number like in the figure. The numbers are given in respect to the grid pixel value (eg, any values less than equal to 150 in both X and Y direction will correspond to block number 0, and so on). Hence, when rectangle surrounding the contour that represents the user's gesture is detected, the midpoint of it is taken. This is simply done by adding the half the width and height to the coordinate values of the top right corner.

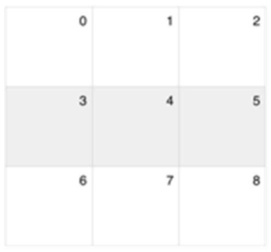


Fig. 14. Numbered Grid

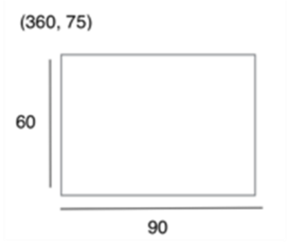
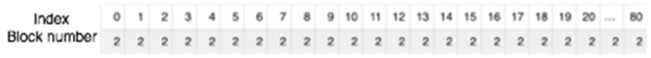


Fig. 15. Detected rectangle

(405, 105) in his case which corresponds to box 2. This is how the Location of the box is determined.

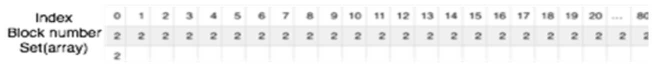
H. Move registration

Move registration refers to getting an idea that the player is positive of his intent of doing the particular gesture and that it not one done by mistake. To understand it one can think of a computer mouse and its buttons. Only when the user presses the buttons on the mouse are his actions confirmed; they are not confirmed by simply dragging the mouse cursor around. Incorporating this function is essential as it will remove random gesture errors (for example, moving the color from box 0 to 3 would register the move on all three boxes). To add this feature in gestures I have exploited the functions of an array and a set. The location of rectangle is found in every camera frame and is appended in an array. The length of that is taken out. And if the length is equal to one, the move is registered. This means at the player has to gesture (show the color in this case) for a given time in order for his move to be counted. Then the array is converted into a set which removes all identical values (leaving the original).



Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	...	80
Block number	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2

Fig. 16. Visual of array



Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	...	80
Block number	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
Set(array)	2																						

Fig. 17. Visual of array & set

I. Drawing X or O

After the location is found, a corresponding X or O depending on the color is drawn. This will further be elaborated in the Game play section of the paper.

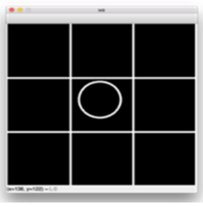


Fig. 18. Tic-Tac-Toe Board with marked move

J. Method 2: Gesture tracking through haar cascades

In this method, gesture detection is done by recognizing the palm and fist of the user. This is done by finding Haar-Like-Features (discussed in the background section of the paper) in the camera frame that correspond to the palm and the fist. The Palm corresponds to the 'X' and the first to 'O'. The cascade location and move registration is identical to how it is done in method described earlier (Gesture Tracking through color.

K. Cascade location

The program is fed with plenty XML cascade files of a fist and palm. The detect Multi Scale () function returns a rectangle corresponding to the Fist or Palm found. The center of the rectangle is calculated (see previous) and the block number is found (see previous).

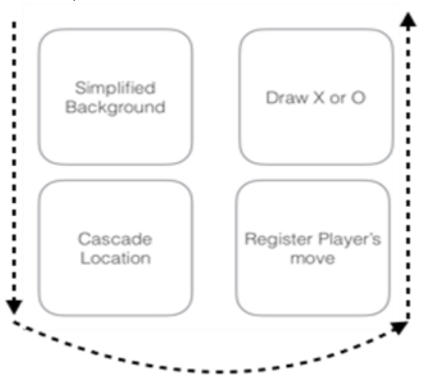


Fig. 19. Flowchart for gesture recognition

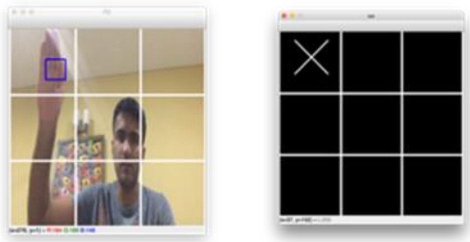


Fig. 20. Palm gesture and corresponding output

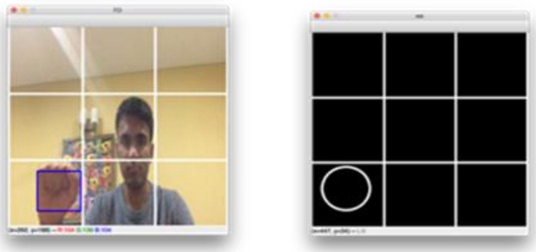


Fig. 21. Fist gesture and corresponding output

Rest of the process is similar to the method color based gesture recognition described earlier.

L. Method 3: Gesture recognition through hand contours.

In this method, gestures are tracked by drawing a contour around the hand. The location of the contour is used to mark the location of the gesture. This method doesn't not have separate gestures for ‘X’ and ‘O’. It simply assigns the move alternatively. The steps of how the location is estimated, the move is registered and how the move is placed is identical to method 1 discussed earlier.

M. Simplified background

Background is simplified as described earlier and the resulting image is as shown.

N. Contour detection

A contour of the resulting image is found (a contour is a curve joining continuous points in the image that have the same light intensity).

O. Convex hull

After the contour is detected, it allows us to make a convex hull around the hand. With it we define a region of interest and inside it use the contours to find a Bounding Rect. The benefit of defining a region of interest is to mini mise chances of picking up random noise.

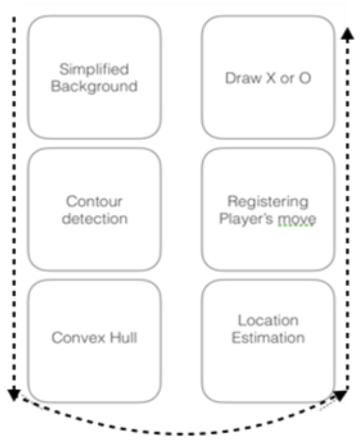


Fig. 22. Flowchart for gesture recognition

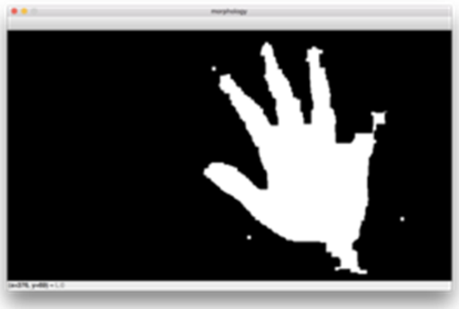


Fig. 23. Image with simplified background

P. Location estimation

Initially, I tried finding the location of the rectangle similar to how I did in the previous methods — by finding the mid-value of the rectangle detected. However, I spotted a very major problem. In this method, if the player tries to place his move to the corners, he/she has to move his forearm. As the forearm is also brought closer to the screen, contour detected tends to get larger and thus the center tends to stay where it is even though the palm has moved. This way, the player can’t place his move properly. To eliminate this, the location is tracked only using the mid value of the top side of the rectangle. This allows the player place his move even with his forearm coming in the camera frame.

Q. Rest of the steps are as described in Method 1.

Section 3 - Game Play: Every game is bound by some rules. And every game, depending on the type of interface it uses has to have a platform that makes it comfortable for the users to use. This section will talk about how I have integrated different validations and usability measures in my version of Gesture Tic-Tac-Toe. In the all the three techniques, the game play method is identical; hence I shall talk about them in go only. The aforementioned are some essential features in my Tic-Tac-Toe game.

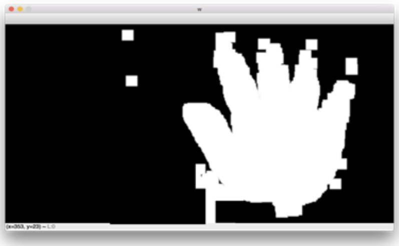


Fig. 24. Contour detection

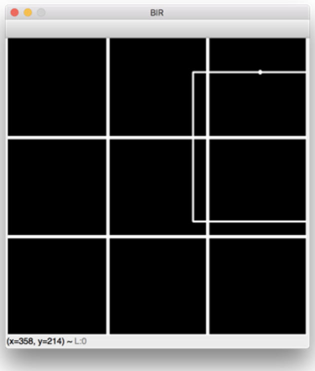


Fig. 25. Rectangle detection

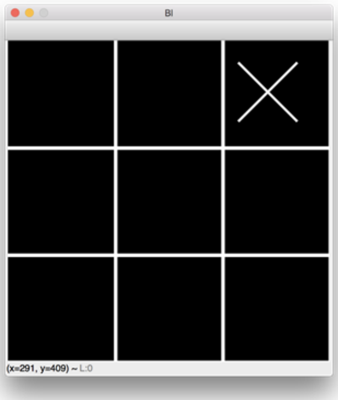


Fig. 26. Move drawn

Fig. 27. Game play features

R. X&O location

The location where the move has to be placed is not solely dependent on the exact location where gesture is found. In the every small box, the move will always be placed right in the centre even if the gesture is not recorded in the centre. In every box, the mid value is already defined. Whenever a move has to be placed, it is placed according to the mid value of that box.

S. Reference screen

A reference screen with the 9 boxed is also shown with a marker pointing to the gesture location. This allows a player to understand where his/her gestures are on the Tic-Tac-Toe grid.

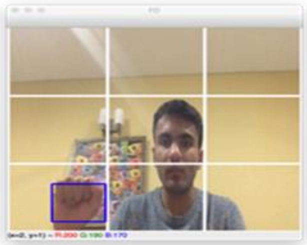


Fig. 28. Reference grid

T. Winner declaration

An integer value is assigned to Xs and Os and an array for 9 indexes is maintained. As a move is input, the corresponding value in the array is given the unique integer of the X (1) or the O (0). After this, a list of winning cases are initialized using the array. For example, in this array [1, 1, 1, 0, _, 0, 0, _,_], X will be declares the winner as the array shows that the top row of the Tic-Tac-Toe board is filled with X. When the more such cases fit, the loop breaks and the winner is declared.

U. Validating illegal moves

There aren’t many illegal moves possible. The only ones are one player can’t move twice and one player can’t overwrite on the other players moves. The former is accomplished using a Boolean flag that only allows one person at a time. The latter is done by using the array declared in the winner declaration portion. A move is only counted when the corresponding box number index in the array holds no value. This allows only one input per box.

5. Analysis

In this section I will talk about the analysis of the three methods I have used in my gesture Tic-Tac-Toe. I have analyzed the methods based on the following aspects

- Usability
- Accuracy
- Robustness
- Computational Speed

A. Usability

Usability is defined as the degree to which a user can adapt to using a particular technique/method. As all the algorithms use different logics, their usability also differs in various aspects.

B. *Method 1: Gesture recognition through color*

Additional requirements: As the name suggests, this method requires the users to use particular colors for the computer to recognize gesture. A player need a greenish shade for 'X' and a reddish shade for 'O'. The following is the range of the hue with a player can register his/her move. As no bodily feature possesses such hues, a person will have to arrange for the color by himself/herself. They can wear this colour's glove or can show a colored card to mark the gesture. The player will have to make sure the background or his/her own clothing is not of the colors in the range.

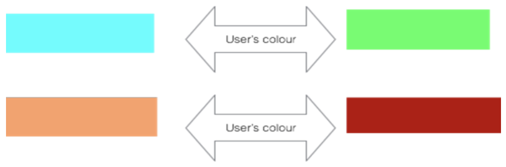


Fig. 29. Range of color values

C. *Environmental requirements:*

No particular lighting is needed; only normal room lighting. The colours should be visible. The best approach would using a mobile with a picture of that color (I got the color from https://coolors.co). This way, one can adjust the lighting of the mobile screen and make sure that the color visible to the camera is of the particular hue.

D. *User's position*

The user should be located at a distance where the colored screen/card fits inside only one box of the tic-tac-toe board, otherwise it might lead to double entry.

E. *Method 2: Using Haar Cascades to mark gesture*

No additional items are required here as this methods track human features.

F. *Environmental requirements*

Although this method tracks human gestures, its recognition can be affected slightly by the ambient light. Hence, one needs to use this method in a lit room, where preferably light is not behind the user.

G. *User's position*

The user should be located at a distance where hand fits inside only one box of the tic-tac-toe board, otherwise it might lead to double entry.

H. *Method 3: Using hand contours to mark gesture*

No additional items are required here as this methods track human the motion of the human hand.

I. *Environmental requirements*

This method is heavily volatile to change in background lighting. One must use this method in a dark room, preferably completely dark with only the laptop light illuminating the hand. While using this method, a completely white screen is also displayed that functions to torch up the front hand.

J. *User's position*

The user should be located near the white screen of the computer so that only the hand can be illuminated. One can adjust the screen's brightness to make sure only the hand is being scanned not the rest of the body. In this method the user does not have to make sure that the hand is in the only one box; they must be near the screen as gesture is marked only from the top portion of the contour.

K. *Overall comment*

After analyzing the prerequisites of each of the method, I can say that the method using the Haar Cascades is most user friendly as it has least requirements.

L. *Accuracy 5.2*

For each of the methods, I have determined the accuracy. To do so, I with a couple of my friends tabulated results for 150 Tic-Tac-Toe games (50 per each) for all three methods.

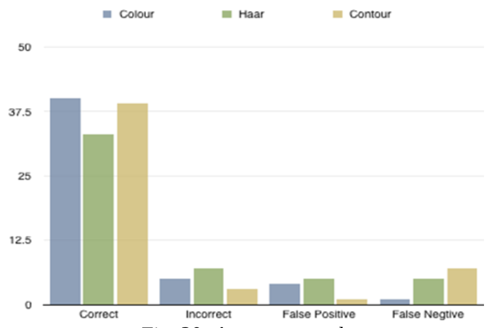


Fig. 30. Accuracy graph

Correct refers to correct gesture recognition. Incorrect means X and O were not correctly distinguished. False positive means gesture was detected but no gesture was present. False negative means no gesture was detected when gesture was done.

M. *Overall analysis*

From the above graph we can say that the color based detection and contour based are almost equally accurate. Haar cascades based recognition is not very accurate because it is very much dependent on the training data file that it is provided with. To increase its accuracy, I will have to search for more trained data. The incorrect in color based tracking probably could have come from different lighting. Probably, the angle at which the color source was held wasn't projecting the color to the camera properly. The incorrect in contour based tracking could have come from contours other than of the hand being detected. This could be adjusted by decreasing or increasing the computer screen's brightness.

N. *Robustness 5.3*

Here I have tested the three methods to the extremes. I managed change some parameters to see which methods tend to falter when, and which tend to be robust.

O. Method 1: gesture recognition through Haar-Cascades

Distance: I shall explore the distance till where the gesture can be recorded. Detecting the 'O' (gesture is fist) is relatively hard as compared to the 'X' (gesture is hand). This is because the Haar-features are less prominent in the fist and hence with increase in distance they get obscured.

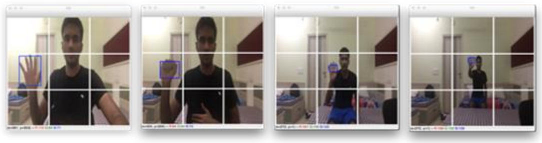


Fig. 31. Results in different distance

P. Ambient Light

I have explored the robustness with changing room lighting with Haar-Cascade method. The distance in this exploration is fixed to 90 cm.

Lighting is quintessential here, especially because the trained data is not very extensive. Therefore, with decrease in light will reduce accuracy

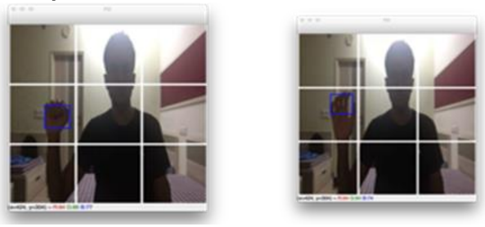


Fig. 32. Results in different distance

Orientation: Showing the gesture in different manners will test its robustness. I have tested for various orientations for the Haar-cascade method.

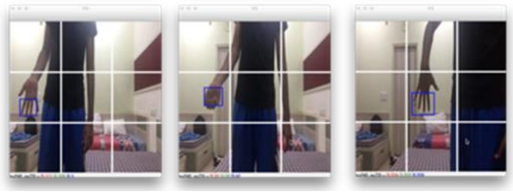


Fig. 33. Inverted and turned around gesture

Q. Method 2: Gesture recognition through Hand contours

Distance: I tested robustness for the method using Hand contours on different distances

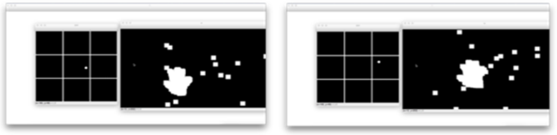


Fig. 34. Successful Marker location till 150m

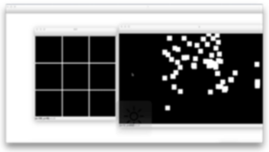


Fig. 35. Unsuccessful Marker location at 250m

R. Lighting

I tested the robustness for the method using Hand contours on different ambient lighting. As extensive thresholding is done in this method, it only works when there is no ambient lighting in the room.

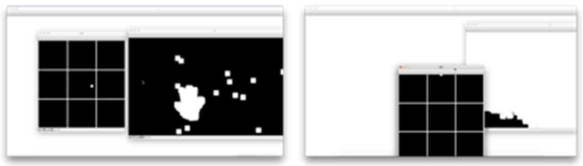


Fig. 36. Effect of distance on contour method

Contour method is not affected by the orientation of the hand as it's not searching for features. Even a inverted or turned around hand would work identically. Hence, I chose not to experiment on those parameters.

S. Method 3: Gesture recognition through Color

Distance: I tested robustness for the method using Color on different distances

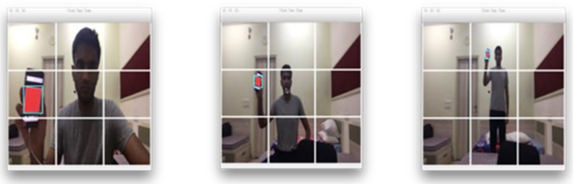


Fig. 37. Successful result with Red color

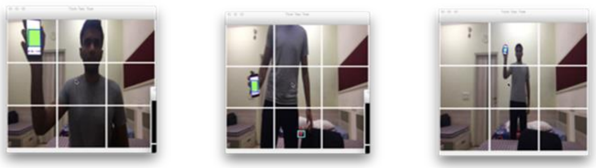


Fig. 38. Unsuccessful results with green color

The color based method is unaffected to change in orientation and brightness till the time the correct color value is being fed, hence I chose not to experiment on those parameters.

Overall Analysis: From the above results, I have come to a conclusion that the color-based Gesture recognition was most robust. It doesn't depend on lighting or orientation and can used from a long distance. Haar-Based is more robust than Color-Based as it relatively accurate results with varying distance and ambient light.

T. Computational Speed 5.4

I have found about the computational speed using a qualitative method. I have found the rate at which frames are produced in each of the methods.

Above is a table of the rate at which the frames are processed in the methods. One can observe that the method using Color and Contour to track gestures almost process the frames at almost the same rate — approximately .025 seconds. The contour method is slightly larger than the colour based probably because of the heavy amount background simplification going

Table 1
Showing effect of varying distance on Haar cascades method

Distance(cm)	Always correct detection for X	Always correct detection for O	Sometimes correct detection for X	Sometimes correct detection for O	No detection for X	No detection for O	Contour Area-Hand Ratio for Hand Approx.	Contour Area-Hand Ratio for Fist
0-90	TRUE	TRUE					1	1
90-150	TRUE					TRUE	0.5	0
150-250			TRUE			TRUE	0.4	0

Table 2
Showing effect of varying ambient light on Haar Cascades method

Ambient Light	Always correct detection for X	Always correct detection for O	Sometimes correct detection for X	Sometimes correct detection for O	No detection for X	No detection for O	Contour Area-Hand Ratio for Hand Approx.	Contour Area-Hand Ratio for Fist
Well lit	TRUE	TRUE					1	1
Dim	TRUE			TRUE			0.5	1
Dark (only computer)					TRUE	TRUE	0	0

Table 3
Showing effect of different orientation on Haar Cascades method

Gesture Orientation	Always correct detection for X	Always correct detection for O	Sometimes correct detection for X	Sometimes correct detection for O	No detection for X	No detection for O
Normal	TRUE	TRUE				
Side ways			TRUE			TRUE
Inverted	TRUE			TRUE	TRUE	TRUE
Turned around	TRUE					TRUE

Table 4
Showing effect of Distance on Hand Contours method

Distance (cm)	Marker Always in correct position	Marker sometimes in correct position	No marker detection	Marker recognized on all box positions	Marker not recognized on all box positions
0-90	TRUE			TRUE	
90-150	TRUE				TRUE
150-250			TRUE		TRUE

Table 5
Showing effect of ambient light on Contour method

Ambient Light	Marker Always in correct position	Marker sometimes in correct position	No marker detection	Marker recognized on all box positions	Marker not recognized on all box positions
Lit			TRUE		TRUE
Dull			TRUE		TRUE
Completely dark(only computer lighting)	TRUE			TRUE	

Table 6
Showing effect of distance on colour-tracking method

Distance(cm)	Always correct detection for Red colour	Always correct detection for Green colour	Sometimes correct detection for Red colour	Sometimes correct detection for Green colour	No detection for Red colour	No detection for Green colour
0-90	TRUE	TRUE				
90-150	TRUE	TRUE				
150-250	TRUE	TRUE				

in it. We can also see that the method using Haar Cascades is significantly larger —almost double .05 seconds. This increase is because this method is based on Viola Jones algorithm which uses an enormously large pre-trained dataset to find Haar features in the camera frame.

Table 7
Different frames per second

COLOUR (seconds)	CONTOUR (seconds)	HAAR (seconds)
0.02229499	0.0264748	0.06747438
0.02515602	0.0380949	0.06948805
0.01929187	0.0335919	0.03191994
0.02008700	0.0196397	0.05644374
0.02057385	0.0248169	0.06057218
0.02227807	0.0268049	0.06638791
0.02196502	0.0381385	0.05387115
0.02110435	0.0240799	0.03118416
0.02096986	0.0203901	0.05716996
0.02951789	0.0336377	0.04880881
0.02122087	0.0177302	0.03966189
0.02190098	0.0498828	0.06122803
0.02137780	0.0398378	0.05238318
0.02181386	0.0325198	0.06860327
0.02792501	0.053212	0.061161
0.01956391	0.021484	0.065491
AVERAGE	AVERAGE	AVERAGE
0.02100243	0.0294315	0.05246168

6. Conclusion

After the testing all three methods, I learnt a lot about the different techniques. I can say that color based tracking and contour based tracking were increasingly efficient and accurate. They were robust and tolerant to changes as compared to the Haar-Cascade method. However, using those two methods have a tradeoff -- they have limited usability. The Haar-Cascade method, though only accurate to a short distance from the computer and relatively intolerant to changes in the ambient lighting, is definitely more user friendly as it requires no external equipment or ambient conditions.

7. Discussion

In my project, I used different techniques to create a gesture controlled tic-tac-toe without use of any large training procedure. Although the methods I devised gave relatively a positive result, there are a few limitations, like effect of ambient light, requirement of additional material, etc. I would like to explore other machine learning methods, like the Hidden Markov Model, Particle Filtering and Condensation Algorithms, etc. to increase the quality of my game. In the future, I can train the program to analyze particular gestures like snapping, pointing, etc to add a new dimension to the game. The techniques I've used here can have have other applications as well. They can be used to substitute the computer mouse, to other games, to make applications more mobile for people who suffer from physical or mental illnesses (e.g, A keyboard for individuals with Parkinson's disease or a Sign-Language to text/audio converter for those who are mute)

.

8. Acknowledgement

Thank you to Professor Susan Fox (Ph. D.) for mentoring me in Computer vision and guiding me through this research project.

References

[1] Rautaray, S. S. (2015, January 7). Vision based hand gesture recognition for human computer interaction: A survey. Retrieved July 25, 2018, from https://link.springer.com/article/10.1007/s10462-012-9356-9
[2] Mitra, S. (may 2007). Gesture Recognition: A Survey. IEEE Transactions on Systems, Man, and Cybernetics—Part C: Applications And Reviews, vol. 37, no. 3).
[3] Das, D. (2014). Implementation and Performance Evaluation of Background Subtraction Algorithms. International Journal on Computational Sciences & Applications (IJCSA), Vol. 4, no. 2), April 2014.
[4] Patil, S. P. (2016). Techniques and Methods for Detection and Tracking of Moving Object in a Video. Nternational Journal of Innovative Research in Computer and Communication Engineering, Vol. 4 (5), 2016.
[5] Meena, S. (2011). A Study on Hand Gesture Recognition Technique. Department of Electronics and Communication Engineering National Institute of Technology, 0-98.
[6] Open CV Tutorial https://docs.opencv.org/2.4/doc/tutorials/tutorials.html
[7] Szeliski, R. (2010). Computer Vision: Algorithms and Applications. http://szeliski.org/Book/