Webcam Gesture controller

Mario Martinez, Computer Engineering

Texas Tech University

May 2019

Abstract

This paper describes the structure of a webcam gesture controller web application. The paper includes the function as well as important snippets of code for each function of the controller.,

Table of Contents

|  |  |  |
| --- | --- | --- |
| Introduction | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | 4 |
| Web layout | ------------------------------------------------------------------------ | 4 |
| Webcam panel | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | 6 |
| Wheel panel | ------------------------------------------------------------------------ | 7 |
| Trained image panel | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | 9 |
| Buttons and functions | ------------------------------------------------------------------------ | 10 |
| Conclusion | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | 11 |

List of figures

|  |  |  |
| --- | --- | --- |
| Introduction [1] | ------------------------------------------------ | 4 |
| Web layout [2] | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | 5 |
| Webcam panel [3] | ------------------------------------------------ | 5 |
| Wheel panel [4] |  | 6 |
| Code Snippet for instance(Left) mode and Global Mode (Right ) [5] | ------------------------------------------------ | 7 |
| Image of Wheel [6] | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | 7 |
| Image of Code Snippet for display wheel loop, logic, angle calculation [7] | ------------------------------------------------ | 10 |
| Example of taking left direction [9] | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | 11 |
| Image of Timed events and gesture recording [10] | ------------------------------------------------ | 12 |

**Introduction**

This system is an application that incorporates machine learning to program and map human gestures or other images to control the angle of a steering wheel. The Materialize CSS framework is used to layout the web application. The ML5 library, which is built on top of the Tensorflow library, is the learning method used for this application. ML5 classifies an image and trains the model to identify those images.The P5 library is also utilised for additional support.

The application has both manual and automatic input. In both cases, images taken from the webcam are classified to an angle in which the wheel will turn to when the webcam identifies such an image.

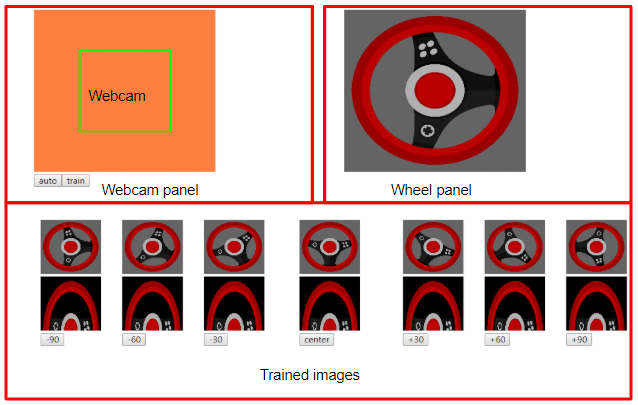
**Web layout**

Image of Web Application [1]

The webpage layout consists of three main panels, the webcam panel, the wheel panel, and the trained image panel denoted by a red rectangle.

The application utilizes Materializes grid layout in which columns exist inside rows. An unlimited amount of rows may exist however only a maximum of 12 columns may exist. To space all main panels evenly, two rows are generated. The top row contains a total of 2 columns each spanning 6 columns.





Image of Materialize Grid Example [2]

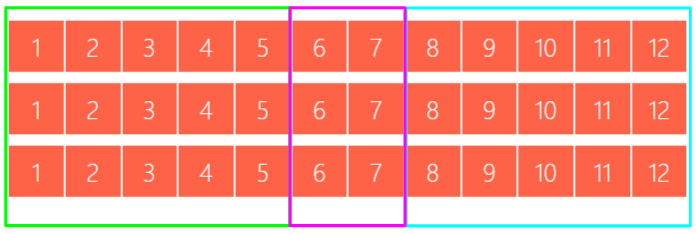
The bottom row contains 3 additional rows with three columns. These columns span 5 columns wide and a center column span of 2. 

Image of bottom row layout [3]

**Webcam panel**

The webcam panel consists of two webcam objects, one for display only and another for taking snapshots . However only the display webcam object is viewed. P5 has a global and an instance mode.

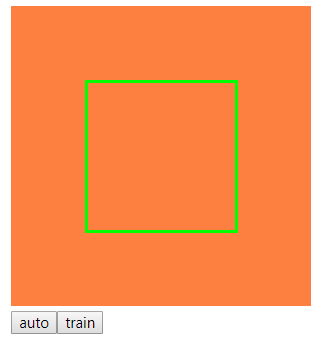
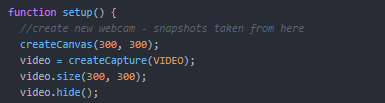
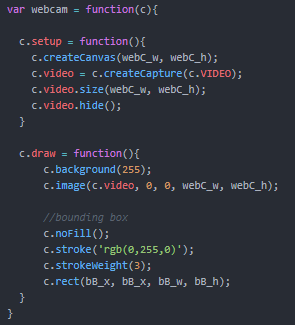


Image of Webcam Panel [4]

The webcam for display is in instance mode and the webcam for snapshots is global mode. The instance webcam contains a green rectangle where if a webcam snapshot is taken, only what's inside the green rectangle will be the snapshot. Anything outside the rectangle will not be considered in the snapshot. The instance mode webcam is what the user will view to take snap shot. When a snapshot is taken, a function crops the global webcam down to whatever is inside the rectangle. This so done so that the rectangle does not appear in the snapshot.



Code Snippet for instance(Left) mode and Global Mode (Right ) [5]

**Wheel panel**



Image of Wheel [6]

The wheel panel is simply a seperate canvas that contains an image of a wheel. The image dimensions are perfectly square so that a rotation does not looked skewed. The image rotates to a degree when the classifier identifies a trained image to its corresponding degree.

This section of the code is set up with a variable for speed, the loaded image of the wheel, and a seperate canvas. The code works by constantly drawing the image on the canvas and applying any rotation as necessary. To rotate, a function is created that reads the variable that is mapped to what the webcam classifies, and turns the wheel accordingly using a case statement. First a the difference between the angle detected and the previous angle is set to a variable (dist). If the current angle differs from the angle detected, then the angle detected is set to the current angle and the wheel rotates to its new angle. The direction of the rotation is identified by the sign of the variable dist.

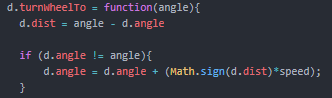
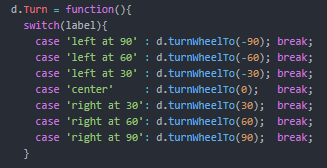
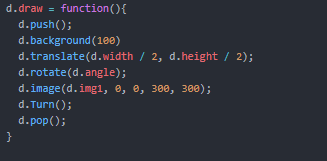


Image of Code Snippet for display wheel loop, logic, angle calculation [7]

**Trained image panel**

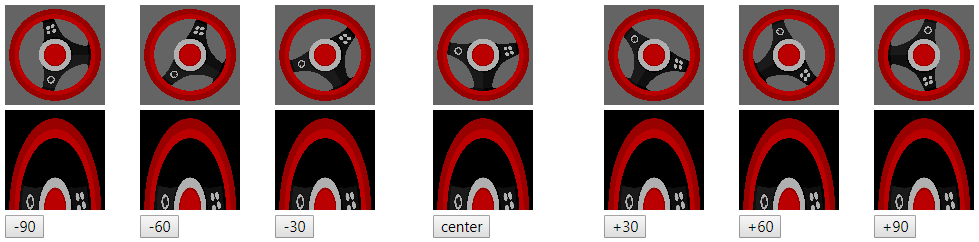


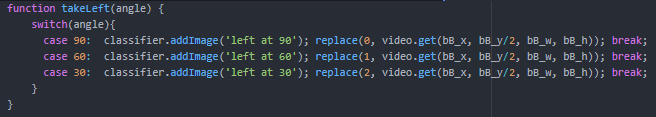
Image of Trained Panel [8]

The trained image panel simply displays the rotated wheel image and the last snapshot taken to the corresponding angle. The first row displays the wheel image at its varion angle rotations. The second row displays the last snapshot taken for the corresponding angle. The third row contains a button for each angle when pressed, will take a snapshot and map the image to the corresponding angle.

**Buttons and functions**

Nine buttons are used to operate the application. An automatic button which will count backwards from three seconds to take a snapshot for what the webcam displays for training use. A training button which will train the model with the images taken. And seven buttons ranging from -90 to 90 degrees each 30 degrees apart (-90, -60, -30, 0, 30, 60, 90). Each button manually records a gesture from the webcam and classifies the image to its corresponding angle.

To record a gesture, a function is set up that takes the angle, provided by the button, and a case statement determines which angle is to be recorded. Within each statement, the image is classified, and the image taken replaces the current image within the training panel.



Example of taking left direction [9]

When the auto button is pressed, the application enters an interval state. Every 1 second, a function (timeIt) runs that displays which gesture direction is to be taken and a countdown of how many seconds until the gesture is recorded. When the counter reaches zero, the application records the gesture and repeats for the next gesture. Once all gestures have been recorded, the interval is cleared. The timer may not be reused and the application needs to be refreshed before the timer can be used again.

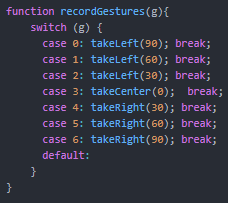
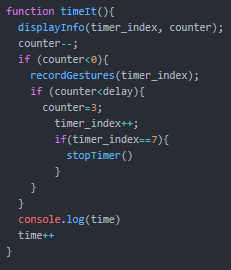


Image of Timed events and gesture recording [10]

The training button will train the model, mapping all snapshots taken to a wheel direction

All other buttons (-90, -60, -30, 0, 30, 60, 90) operate in a similar fashion. The button will classify the webcam image taken and map it to the indicated wheel direction while also displaying training loss until the model is finished training.

**Conclusion**

The application is able to take a snapshot of the webcam and map it to an angle. When the webcam detects such an image, the wheel will turn to the angle that the webcam detects. However, because the angles are too close to one another, the application determines that both an angle of 60 and 30 and very similar and therefore the wheel will alternate back and forth between the two angles. If the snapshot images are clearly different from one another, the application behaves as expected. Impractical if images trained are very similar to one another.

References

|  |  |  |
| --- | --- | --- |
| 1 | P5 Library | <https://p5js.org/libraries/> |
| 2 | Materialize Library | <https://materializecss.com/> |
| 3 | Tensorflow Library | <https://www.tensorflow.org/js/> |
| 4 | ML5 Library | <https://ml5js.org/> |
| 5 | Github | <https://github.com/mar87649/Webcam-Gesture-Controller/tree/master/Gesture_Controller> |

Appendices