report

July 9, 2017

1 Report for AIND CV - Mimic Me

Bryan Travis Smith

This report summarizes my work on completing the Mimic Me project for Udacity's AI NanoDegree.

1.1 Display Feature Points

To implement this method, I just iterated through the face features from the face object passed to this function. For each one I added a circle to the canvas at the location of the feature point.

```
var ctx = canvas.getContext('2d');

for (var id in face.featurePoints) {
  var featurePoint = face.featurePoints[id];
  ctx.beginPath();
  ctx.arc(featurePoint.x, featurePoint.y, 2, 0, 2 * Math.PI);
  ctx.stroke();
}
```

1.2 Show Dominant Emoji

The draw emoji function is done by taking the dominantEmoji feature from the face object, and displaying it on the canvas. The feature points are returned in a fixed order, so I set the position at a fix distance from the first feature point.

```
ctx.fillText(
    face.emojis.dominantEmoji,     //emoji unicode
    (face.featurePoints[0].x+200), // x-position
    (face.featurePoints[0].y-100)  // y-position
);
```

1.3 Implement Mimic Me!

The game I implemented is very simple:

1. Iterate through the list of emoji's suppored by Affectiva
2. Allow 5 sections for the player to make a face that 'matches' the emoji

1. Increase the score if there is a match

This game is built with a fuction that makes a javascript timer that calls a call back function repeated for a fix number of times, then calls a final callback when the timer is finished.

```
function set_game_interval(callback, complete_callback, delay, repetitions) {
    var self = this;
    var x = 0;
    var intervalID = null

    this.start = function(){
        x = 0;
        intervalID = window.setInterval(function () {

           callback();

           if (++x === repetitions) {
               window.clearInterval(intervalID);
               complete_callback()
           }
        }, delay);
    }

    this.stop = function(){
         window.clearInterval(intervalID);
    }
}
```

This functional allows the game to change the emoji every 5 seconds, then clean up the game when the game is over.

At the end of each time period I use a function 'update_game'

```
function update_game(){
    emoji_index++;    //update global index of game
    matched_current_target = false  //reset the global matched emoji flag
    show_emoji_by_index(emoji_index)  //show the next emoji
    increase_total()  //increase the game total by 1
}
```

The matching is done inside the 'onImageResultsSuccess' event listener. When faces are detected the following code block is also called:

```
    if(
        toUnicode(
            faces[0].emojis.dominantEmoji
        ) == display_emojis[emoji_index] && // display_emojis are shuffled
        !matched_current_target
    ){
        matched_current_target = true
```

```
        increase_score()
    }

    update_score()  // updates the displayed score
```

If there is a match of the current emoji that hasn't been matched yet, the match flag is set to true and the game score is updated.

When the game is over, the emoji is replaced with the text "Done".

In [ ]: