Mimic Me! Report

Brian Cooley

7/10/2017

Introduction

This report describes the steps implemented to build a Mimic Me! game using Affectiva's emotion detection API. Briefly, the steps are displaying the feature points of the face, displaying the dominant emoji matching the detected emotion, and building a game mechanic. Each of these steps is discussed in the following sections.

Feature Points

The first step in building the game is to display feature points. The provided code includes a drawFeaturePoints function that is called each time a face is detected. The function accepts a javascript canvas, an image, and a face object. The face object contains an array of feature points.

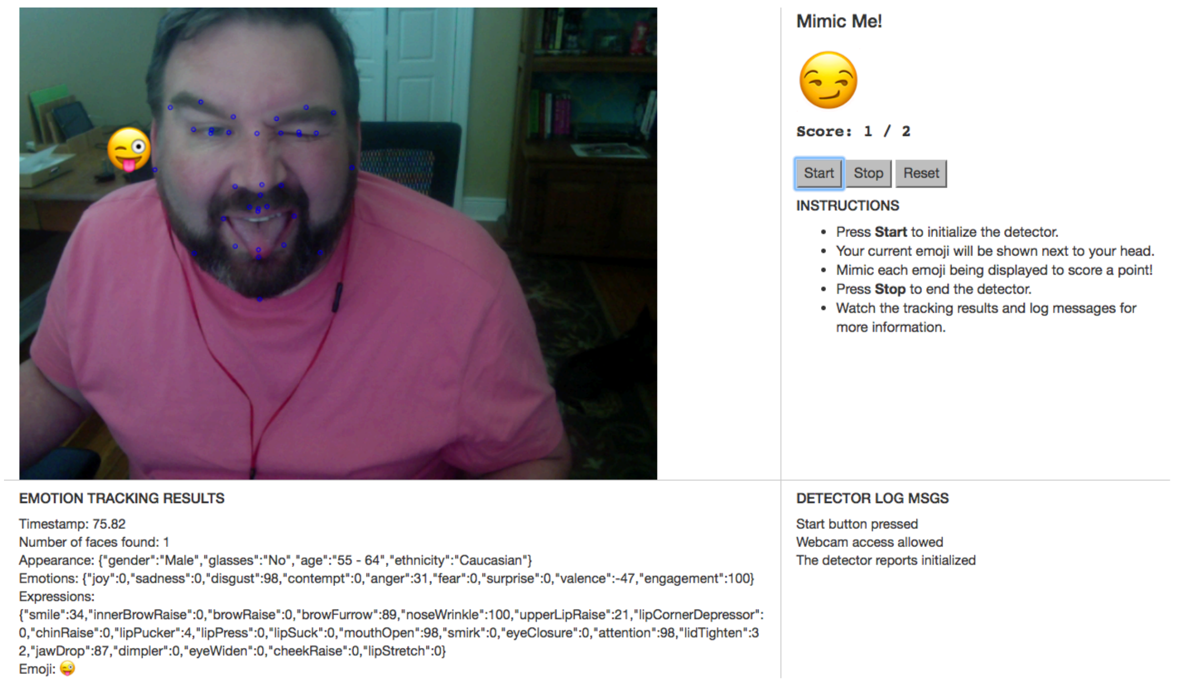
To display feature points, we get a context from the canvas and use that context to set the color of the fill and stroke to be used in any drawing commands sent to the context. I chose to draw a circle for each feature point, which is done by drawing an arc using the coordinates of the feature point, a radius, and the beginning and ending angles of 0 and 360 to draw a complete circle. I used a radius of 2

RELEVANT CODE:
```
// draw the detected facial feature points on the image
function drawFeaturePoints(canvas, img, face) {
  // Obtain a 2D context object to draw on the canvas
  var ctx = canvas.getContext('2d');

  ctx.fillStyle = 'blue';
  ctx.strokeStyle = 'blue';

  // Loop over each feature point in the face
  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();
  }
}
```

An image of the feature points is shown below. The feature points are shown by the small blue circles appearing at key locations on the face (nose, eyes, mouth, etc.)


Mimic Me!

😏

Score: 1 / 2

INSTRUCTIONS

- Press **Start** to initialize the detector.
- Your current emoji will be shown next to your head.
- Mimic each emoji being displayed to score a point!
- Press **Stop** to end the detector.
- Watch the tracking results and log messages for more information.

EMOTION TRACKING RESULTS

```
Timestamp: 75.82
Number of faces found: 1
Appearance: {"gender":"Male","glasses":"No","age":"55 - 64","ethnicity":"Caucasian"}
Emotions: {"joy":0,"sadness":0,"disgust":98,"contempt":0,"anger":31,"fear":0,"surprise":0,"valence":-47,"engagement":100}
Expressions:
{"smile":34,"innerBrowRaise":0,"browRaise":0,"browFurrow":89,"noseWrinkle":100,"upperLipRaise":21,"lipCornerDepressor":
0,"chinRaise":0,"lipPucker":4,"lipPress":0,"lipSuck":0,"mouthOpen":98,"smirk":0,"eyeClosure":0,"attention":98,"lidTighten":3
2,"jawDrop":87,"dimpler":0,"eyeWiden":0,"cheekRaise":0,"lipStretch":0}
Emoji: 😜
```

DETECTOR LOG MSGS

```
Start button pressed
Webcam access allowed
The detector reports initialized
```

FIGURE 1: FACIAL FEATURE POINTS INDICATED BY BLUE CIRCLES

Dominant Emoji

The second step in the implementation of the game was to display the dominant emoji. A function drawEmoji accepting a canvas, an image, and a face was provided. To draw the emoji, we get a context from the canvas, use it to set the font, then call fillText on the context. The final call to fillText takes the emoji code, provided to us by the face object via the dominantEmoji property of the face object's emojis object. It also takes a location, which we can use to position the emoji near the face. I found that offsetting the location of the first item in the face object's featurePoints array by 50 pixels in the x direction proved to be a good location for the dimensions obtained with my camera and my location in the room. A more complex implementation that uses, say, the interocular distance may provide more consistent positioning outside the facial frame, if desired.

RELEVANT CODE:
```
function drawEmoji(canvas, img, face) {
  // Obtain a 2D context object to draw on the canvas
  var ctx = canvas.getContext('2d');

   ctx.font = '48px serif';

   ctx.fillText(face.emojis.dominantEmoji, face.featurePoints[0].x - 50,
face.featurePoints[0].y);
}
```

Mimic Game

Showing a random emoji

In building out the game, the first task is to show a random emoji to the player. An array of possible emojis is provided in the template code. To pick a random emoji, we simply generate a random number with the length of the emoji array.

RELEVANT CODE:

```
function generateTargetEmoji() {
    var index = Math.round(Math.random() * emojis.length);
    var emoji = emojis[index];
    return emoji;
}
```

You can see the target emoji shown in the figure. It appears in the upper left corner below the heading "Mimic Me!"

Matching the dominant emoji and target emoji

The second step is to check whether the dominant emoji detected from the player's expression matches the randomly generated target emoji. When we run the game, we check to see if such a match has occurred.

RELEVANT CODE:

```
function runGame(detectedEmoji) {
    if (!isDetecting || startTime == null) return;

    var now = new Date();
    var millisElapsed = now.getTime() - startTime.getTime();

    // If a minute has elapsed since the last target was set, go to the
    // next target Emoji
    if (millisElapsed > 60000) {
        total++;
        updateScore();
        return;
    }

    if (detectedEmoji === targetEmoji) {
        score = score + 1;
        total = total + 1;
        updateScore();
    }
}
```

The visual indication of matching the target is the incrementing of the score. You can see this in the figure where the score is 1/2, indicating that the player has matched one image and timed out on one image.

Showing a new emoji when a match is made

Finally, we need to show a new emoji when the player matches the target emoji. Also, a timeout has been added to make the game interesting. The player is given 1 minute to match the target emoji before moving to the next emoji.

RELEVANT CODE:

```
function updateScore() {
    setScore(score, total);
    targetEmoji = generateTargetEmoji();
    setTargetEmoji(targetEmoji);
    startTime = new Date();
}
```

When a match is made, the visual indication is the changing of the target emoji. In the figure, you can see that the target emoji is now a smirk. This screen capture was taken just after matching the winking, tongue emoji. Also notice that the score is 1/2, indicating that one of the target emojis that had shown during the game was not matched within one minute.