

Report For Computer Vision Concentration Project 1

Task 1: Display Feature Points

```
// 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');

  // TODO: Set the stroke and/or fill style you want for each feature point marker
  // See: https://developer.mozilla.org/en-US/docs/Web/API/CanvasRenderingContext2D#Fill\_and\_stroke\_styles
  ctx.strokeStyle = 'rgba(255,255,255,0.9)';
  ctx.lineWidth = 1;
  ctx.font = '15px serif';

  // Loop over each feature point in the face
  for (var id in face.featurePoints) {
    var featurePoint = face.featurePoints[id];

    // TODO: Draw feature point, e.g. as a circle using ctx.arc()
    // See: https://developer.mozilla.org/en-US/docs/Web/API/CanvasRenderingContext2D/arc
    ctx.beginPath();
    ctx.arc(featurePoint.x, featurePoint.y, 2, 0, 2 * Math.PI);
    ctx.stroke();
  }
}
```

Task 2: Show Dominant Emoji

Implemented Euclidean distance formula to calculate distance between two diagonally opposite feature points of the face. Emoji font size is varied pending on this distance.

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

  // TODO: Set the font and style you want for the emoji
  // Varying font size with face size
  var dist = ((face.featurePoints[10].x-face.featurePoints[5].x)**2 + (face.featurePoints[10].y-face.featurePoints[5].y)**2) ** 0.5;
  if (dist>130) {
    ctx.font = '55px serif';
  } else if (dist>100) {
    ctx.font = '45px serif';
  } else {
    ctx.font = '35px serif';
  }

  // TODO: Draw it using ctx.strokeText() or fillText()
  // See: https://developer.mozilla.org/en-US/docs/Web/API/CanvasRenderingContext2D/fillText
  // TIP: Pick a particular feature point as an anchor so that the emoji sticks to your face
  ctx.fillText(face.emojis.dominantEmoji, face.featurePoints[10].x+20, face.featurePoints[10].y+10);
}
```

Task 3: Implement Mimic Me!

1. Declaring functions and variables for the game

```
var score = 0;
var total = 0;
var target = null;

function startGame() {
  // Reset game variables
  score = 0;
  total = 0;
  setScore(score, total);
  target = setRandomEmoji();
}

function setRandomEmoji() {
  var randIdx = Math.floor(Math.random() * (reducedEmojis.length-1));
  var targetEmoji = reducedEmojis[randIdx];
  setTargetEmoji(targetEmoji);
  console.log("New Target: " + targetEmoji);
  incrementTotal();
  return targetEmoji;
}

// Draw the dominant emoji on the image
function compareWithFace(face, target) {
  detectedEmoji = face.emojis.dominantEmoji;
  if (toUnicode(detectedEmoji) == target) {
    return true;
  }
  return false;
}

function incrementScore() {
  score++;
  setScore(score, total);
}

function incrementTotal() {
  total++;
  setScore(score, total);
}
```

2. Reducing set of emoji to one's that can be recognized reliably

```
// Reduced set of emojis
var reducedEmojis = [ 128528, 9786, 128515, 128527, 128521, 128535, 128545, 128561 ];
```

3. Starting game and comparing of target face with dominant emoji from API

```
// Reset button
function onReset() {
  log('#logs', "Reset button pressed");
  if (detector && detector.isRunning) {
    detector.reset();
  }
  $('#results').html(""); // clear out results
  $('#logs').html(""); // clear out previous log

  // TODO(optional): You can restart the game as well
  startGame();
};

// Add a callback to notify when the detector is initialized and ready for running
detector.addListener("onInitializeSuccess", function() {
  log('#logs', "The detector reports initialized");
  //Display canvas instead of video feed because we want to draw the feature points on it
  $("#face_video_canvas").css("display", "block");
  $("#face_video").css("display", "none");

  // TODO(optional): Call a function to initialize the game, if needed
  startGame();
});

// Reduced set of emojis
var reducedEmojis = [ 128528, 9786, 128515, 128527, 128521, 128535, 128545, 128561 ];
// probabilities for different expressions, emotions and appearance metrics
detector.addListener("onImageResultsSuccess", function(faces, image, timestamp) {
  var canvas = $('#face_video_canvas')[0];
  if (!canvas)
    return;

  // Report how many faces were found
  $('#results').html("");
  log('#results', "Timestamp: " + timestamp.toFixed(2));
  log('#results', "Number of faces found: " + faces.length);
  if (faces.length > 0) {
    // Report desired metrics
    log('#results', "Appearance: " + JSON.stringify(faces[0].appearance));
    log('#results', "Emotions: " + JSON.stringify(faces[0].emotions, function(key, val) {
      return val.toFixed ? Number(val.toFixed(0)) : val;
    }));
    log('#results', "Expressions: " + JSON.stringify(faces[0].expressions, function(key, val) {
      return val.toFixed ? Number(val.toFixed(0)) : val;
    }));
    log('#results', "Emoji: " + faces[0].emojis.dominantEmoji);

    // Call functions to draw feature points and dominant emoji (for the first face only)
    drawFeaturePoints(canvas, image, faces[0]);
    drawEmoji(canvas, image, faces[0]);

    // TODO: Call your function to run the game (define it first!)
    if (compareWithFace(faces[0], target)) {
      incrementScore();
      target = setRandomEmoji();
    }
  }
});
```

Implementation Screenshots:

Task 1 and 2:



Task 3:

