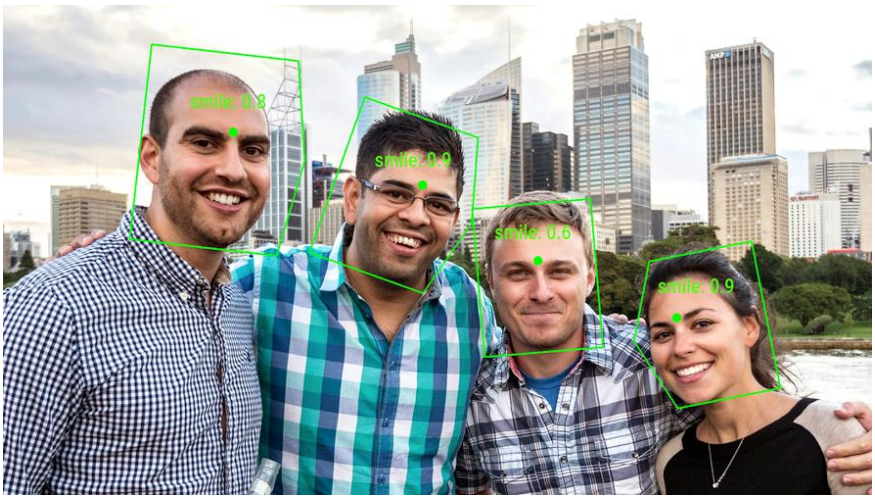
Facial Interpreter

Deadline: 16/11/2019 - 23:59:59

In this homework, you need to create an app “Facial Interpreter”.

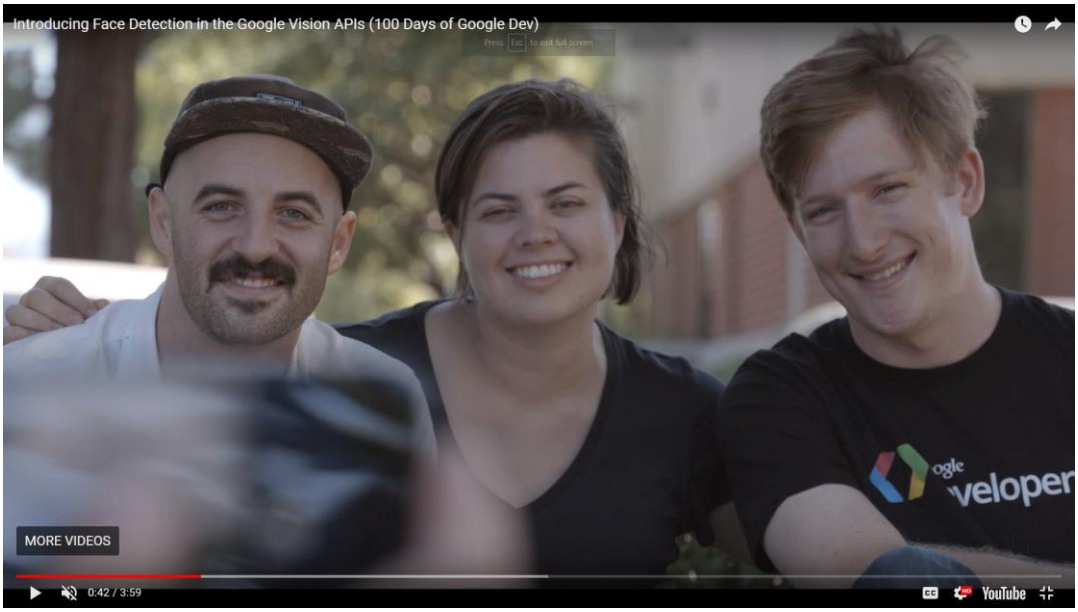
Background

In lab3, we had tried **camera API** and build a camera preview mode. Indeed, once you got image, you can do a lot of things. Google motion vision team had already build a set of FACE API, the quality is guarantee and it comes with other fancy alternatives, you are going to learn how to use it.



The Face API finds human faces in photos, videos, or live streams. It also finds and tracks positions of facial landmarks such as the eyes, nose, and mouth.

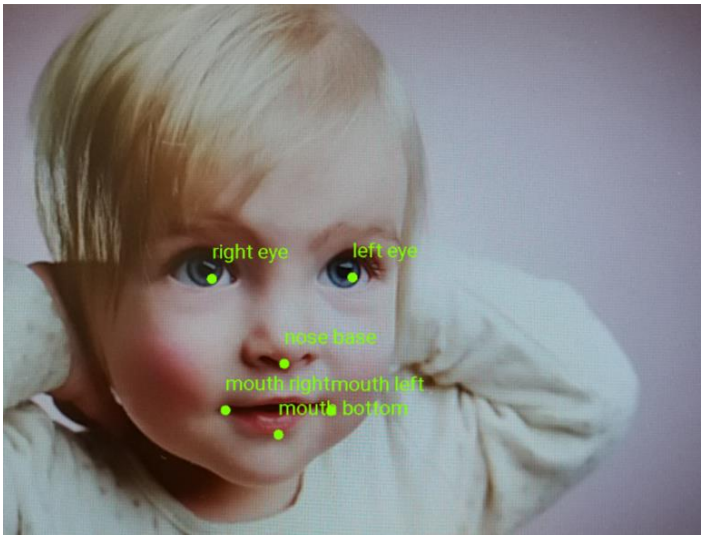
Watch this for more introduction



https://youtu.be/qu1RjE_2zhQ

Requirement (Both A+B):

A) Modify the app so that it identifies the following for any tracked face:
(left eye, right eye, nose base, mouth left, mouth bottom, mouth right)



B) Please answer the following question in a text file.
After you extract the face landmarks, what kind of APP can you develop and how? Please elaborate.

Hand-in method:

Pack the android (java) project with the answer of the question(above) into one single zip file, upload the file to Google Drive, then send it as a Drive attachment . The email is eesm5060ust@gmail.com.

This work can be hand-in group of two. (Please write down student name and IDs in the email)

There will be no notification for receiving your work. If there is no bounce-back email or non-

delivery report returned, it is assumed your work sent successfully.

For the email title, please use "[EESM5060_2019] Assignment 3"

Hints:

1) Some resource files had been provided(assign3_resource.zip). Use what you learn in this course and make a buildable project. Use Android Studio and start a project.

2) Use ece.course.myapplication as package name can save you a lot of work

3) If you find the Snackbar cannot be resolved, make sure the build.gradle(Module:app) includes all the dependencies as in *Tips.png*

4) To add landmarks on face, read the code in the next page.

01) Open **FaceTracker.java** and modify onUpdate() as shown below. The call to update() will momentarily cause a build error while you are in the process of modifying the app to use the **FaceData** model and you will fix it soon.

```
@Override
public void onUpdate(FaceDetector.Detections detectionResults, Face face) {
  mOverlay.add(mFaceGraphic);

  // Get face dimensions.
  mFaceData.setPosition(face.getPosition());
  mFaceData.setWidth(face.getWidth());
  mFaceData.setHeight(face.getHeight());

  // Get the positions of facial landmarks.
  updatePreviousLandmarkPositions(face);
  mFaceData.setLeftEyePosition(getLandmarkPosition(face, Landmark.LEFT_EYE));
  mFaceData.setRightEyePosition(getLandmarkPosition(face, Landmark.RIGHT_EYE));
  mFaceData.setMouthBottomPosition(getLandmarkPosition(face, Landmark.LEFT_CHEEK));
  mFaceData.setMouthBottomPosition(getLandmarkPosition(face,
Landmark.RIGHT_CHEEK));
  mFaceData.setNoseBasePosition(getLandmarkPosition(face, Landmark.NOSE_BASE));
  mFaceData.setMouthBottomPosition(getLandmarkPosition(face, Landmark.LEFT_EAR));
  mFaceData.setMouthBottomPosition(getLandmarkPosition(face,
Landmark.LEFT_EAR_TIP));
  mFaceData.setMouthBottomPosition(getLandmarkPosition(face, Landmark.RIGHT_EAR));
  mFaceData.setMouthBottomPosition(getLandmarkPosition(face,
Landmark.RIGHT_EAR_TIP));
  mFaceData.setMouthLeftPosition(getLandmarkPosition(face, Landmark.LEFT_MOUTH));
  mFaceData.setMouthBottomPosition(getLandmarkPosition(face,
Landmark.BOTTOM_MOUTH));
  mFaceData.setMouthRightPosition(getLandmarkPosition(face, Landmark.RIGHT_MOUTH));

  mFaceGraphic.update(mFaceData);
}
```

Note that you're now passing a **FaceData** instance to **FaceGraphic's update** method instead of the **Face** instance that the **on Update** method receives. This allows you to specify the face information passed to FaceTracker, which in turn lets you use some math trickery based on the last known locations of facial landmarks when the faces are moving too quickly to approximate their current locations

02) Now open **FaceGraphic.java**. It's now receiving a **FaceData** value instead of a **Face** value from **FaceTracker**, you need to change a key instance variable declaration from:

private volatile Face mFace; to private volatile FaceData mFaceData;

03) Modify update()

```
void update(FaceData faceData) {
  mFaceData = faceData;
  postInvalidate(); // Trigger a redraw of the graphic (i.e. cause draw() to be called).
}
```

04) Update draw() to draw dots over the landmarks of any tracked face, and identifying text over those dots:

```
@Override
public void draw(Canvas canvas) {
  final float DOT_RADIUS = 3.0f;
  final float TEXT_OFFSET_Y = -30.0f;

  // Confirm that the face and its features are still visible before drawing any graphics over it.
  if (mFaceData == null) {
    return;
  }

  // 1
  PointF detectPosition = mFaceData.getPosition();
  PointF detectLeftEyePosition = mFaceData.getLeftEyePosition();
  PointF detectRightEyePosition = mFaceData.getRightEyePosition();
  PointF detectNoseBasePosition = mFaceData.getNoseBasePosition();
  PointF detectMouthLeftPosition = mFaceData.getMouthLeftPosition();
  PointF detectMouthBottomPosition = mFaceData.getMouthBottomPosition();
  PointF detectMouthRightPosition = mFaceData.getMouthRightPosition();
  if ((detectPosition == null) ||
      (detectLeftEyePosition == null) ||
      (detectRightEyePosition == null) ||
      (detectNoseBasePosition == null) ||
      (detectMouthLeftPosition == null) ||
      (detectMouthBottomPosition == null) ||
      (detectMouthRightPosition == null)) {
    return;
  }

  // 2
  float leftEyeX = translateX(detectLeftEyePosition.x);
  float leftEyeY = translateY(detectLeftEyePosition.y);
  canvas.drawCircle(leftEyeX, leftEyeY, DOT_RADIUS, mHintOutlinePaint);
  canvas.drawText("left eye", leftEyeX, leftEyeY + TEXT_OFFSET_Y, mHintTextPaint);

  float rightEyeX = translateX(detectRightEyePosition.x);
  float rightEyeY = translateY(detectRightEyePosition.y);
  canvas.drawCircle(rightEyeX, rightEyeY, DOT_RADIUS, mHintOutlinePaint);
  canvas.drawText("right eye", rightEyeX, rightEyeY + TEXT_OFFSET_Y, mHintTextPaint);

  float noseBaseX = translateX(detectNoseBasePosition.x);
  float noseBaseY = translateY(detectNoseBasePosition.y);
  canvas.drawCircle(noseBaseX, noseBaseY, DOT_RADIUS, mHintOutlinePaint);
  canvas.drawText("nose base", noseBaseX, noseBaseY + TEXT_OFFSET_Y, mHintTextPaint);

  float mouthLeftX = translateX(detectMouthLeftPosition.x);
  float mouthLeftY = translateY(detectMouthLeftPosition.y);
  canvas.drawCircle(mouthLeftX, mouthLeftY, DOT_RADIUS, mHintOutlinePaint);
  canvas.drawText("mouth left", mouthLeftX, mouthLeftY + TEXT_OFFSET_Y, mHintTextPaint);

  float mouthRightX = translateX(detectMouthRightPosition.x);
  float mouthRightY = translateY(detectMouthRightPosition.y);
  canvas.drawCircle(mouthRightX, mouthRightY, DOT_RADIUS, mHintOutlinePaint);
  canvas.drawText("mouth right", mouthRightX, mouthRightY + TEXT_OFFSET_Y,
mHintTextPaint);

  float mouthBottomX = translateX(detectMouthBottomPosition.x);
  float mouthBottomY = translateY(detectMouthBottomPosition.y);
  canvas.drawCircle(mouthBottomX, mouthBottomY, DOT_RADIUS, mHintOutlinePaint);
  canvas.drawText("mouth bottom", mouthBottomX, mouthBottomY + TEXT_OFFSET_Y,
mHintTextPaint);
}
```