CSE 162 Mobile Computing

Lab 7 Face Detection

Hua Huang
Department of Computer Science and Engineering
University of California, Merced, CA

Submission policy

- Put your name in the app package when creating the app
  - ucmerced.first_name_last_name_ID.cse162.face_detection
- Submit the code on CatCourse
  - Three separate files
  - AndroidManifest.xml, main_activity.xml, MainActivity.java

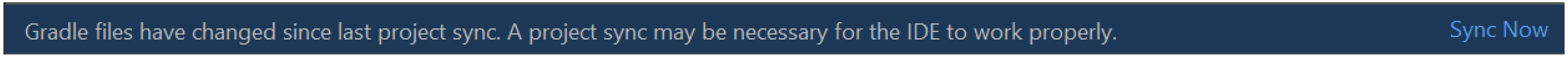
Feature

- Display an image
- Use ML Kit to find the face
- Highlight the face in the image

Setup the dependency

- In app/build.gradle
- implementation 'com.google.android.gms:play-services-mlkit-face-detection:16.1.5’
- **Click sync now**

```
dependencies {
    implementation 'androidx.appcompat:appcompat:1.2.0'
    implementation 'com.google.android.material:material:1.3.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.0.4'
    testImplementation 'junit:junit:4.+'
    androidTestImplementation 'androidx.test.ext:junit:1.1.2'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.3.0'
    implementation 'com.google.android.gms:play-services-mlkit-face-detection:16.1.5'
}
```


Gradle files have changed since last project sync. A project sync may be necessary for the IDE to work properly.

In the manifest.xml

- configure the app to automatically download the model to the device after the app is installed

```
<application ...>
...
    <meta-data
        android:name="com.google.mlkit.vision.DEPENDENCIES"
        android:value="face" />
    <!-- To use multiple models: android:value="face,model2,model3" -->
</application>
```

Prepare for the image detection

- Variables

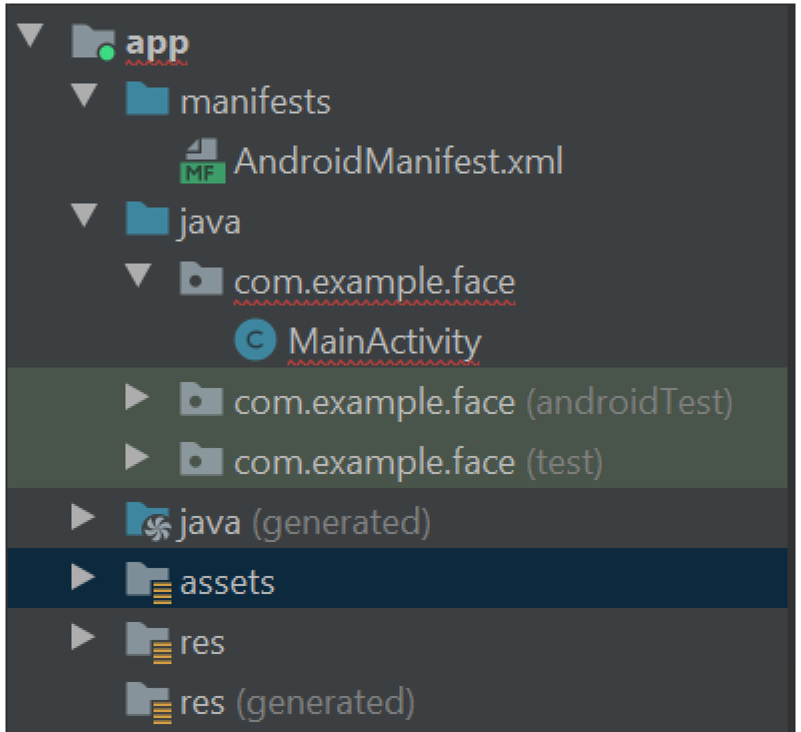
```
ImageView iw;
Canvas canvas;
Bitmap mutableBitmap;
```

- In MainActivity.java onCreate()
- configure the detection options

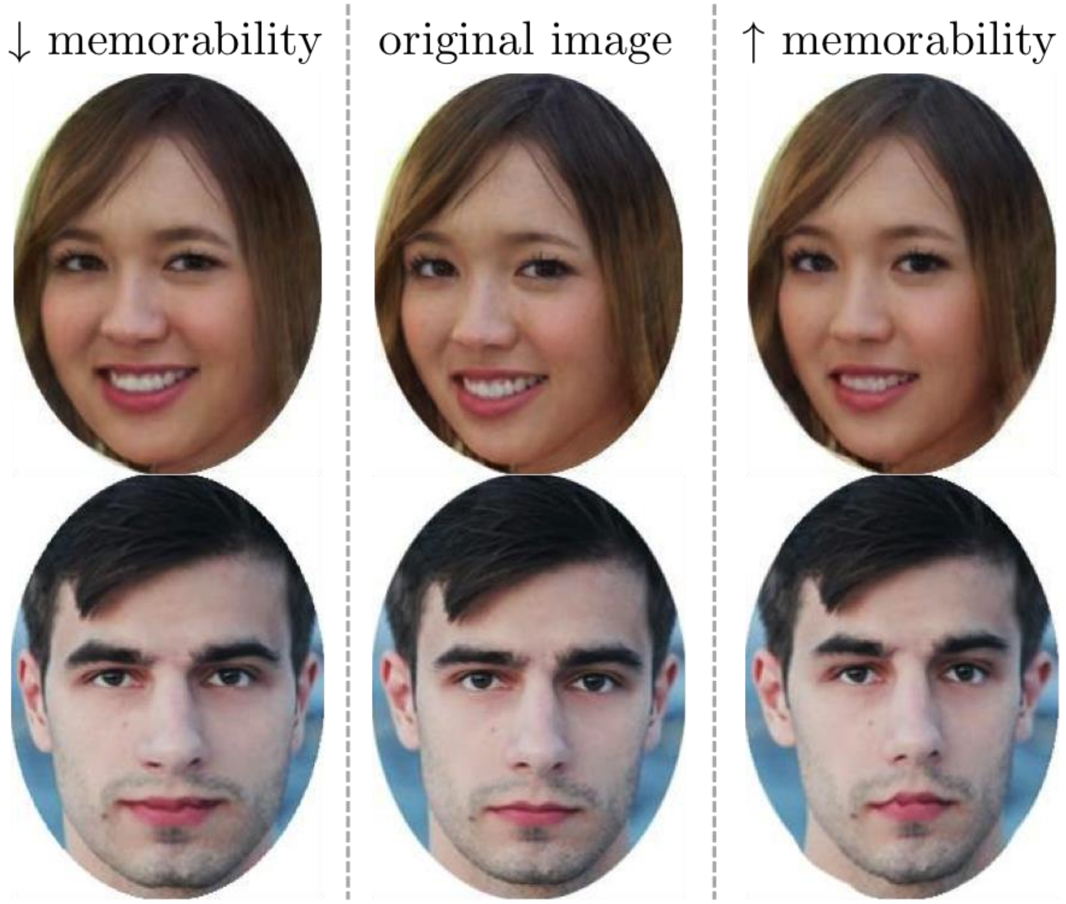
```
FaceDetectorOptions highAccuracyOpts =
        new FaceDetectorOptions.Builder()
                .setPerformanceMode(FaceDetectorOptions.PERFORMANCE_MODE_ACCURATE)
                .setLandmarkMode(FaceDetectorOptions.LANDMARK_MODE_ALL)
                .setClassificationMode(FaceDetectorOptions.CLASSIFICATION_MODE_ALL)
                .build();
```

Prepare the image for processing

- Create the folder: app->right click->new->directory->src/main/assets
- Open the folder: assets->right click->show in explorer
- Place the image file into the path: app/src/main/assets



sample image



Display the results

- In activity_main.xml, use imageview

```
 <ImageView
       android:id="@+id/image_view"
       android:layout_width="match_parent"
       android:layout_height="match_parent"
       android:padding="5dp"
       android:layout_margin="10dp"
/>
```

In MainActivity.java

- read the image
- Convert the image to the appropriate format using InputImage object

```
Bitmap bm=getBitmapFromAssets("faces.png");
InputImage image = InputImage.fromBitmap(bm, 0);
```

getBitmapFromAssets

```
private Bitmap getBitmapFromAssets(String fileName){

    AssetManager am = getAssets();
    InputStream is = null;
    try{

       is = am.open(fileName);
    }catch(IOException e){
       e.printStackTrace();
    }
    Bitmap bitmap = BitmapFactory.decodeStream(is);
    return bitmap;
  }
```

Get an instance of FaceDetector

```
FaceDetector detector = FaceDetection.getClient(highAccuracyOpts);
```

Use Canvas to draw the detection box

- in onCreate(), create a copy of the face image to draw upon

```
mutableBitmap = bm.copy(Bitmap.Config.ARGB_8888, true);
canvas=new Canvas(mutableBitmap);
```

In onCreate()

- display the image

```
iw= (ImageView)findViewById(R.id.image_view);
iw.setImageBitmap(mutableBitmap);
```

Process the image

```
Task<List<Face>> result =
        detector.process(image)
            .addOnSuccessListener(
                new OnSuccessListener<List<Face>>() {
                    @Override
                    public void onSuccess(List<Face> faces) {
                        // Task completed sucessfully
                        // ...
                    }
                })
            .addOnFailureListener(
                new OnFailureListener() {
                    @Override
                    public void onFailure(@NonNull Exception e) {
                        // Task failed with an exception
                        // ...
                    }
                });
```

Get the detection result

- If the face detection operation succeeds, a list of Face objects are passed to the success listener.
- Each Face object represents a face that was detected in the image.
- For each face, you can get its bounding coordinates in the input image, as well as any other information you configured the face detector to find.
- In this lab, we want to plot a rectangle for each face.

Many facial features can be detected. We focus on foundingbox of the face

```
Rect bounds = face.getBoundingBox();
float rotY = face.getHeadEulerAngleY();  // Head is rotated to the right rotY degrees
float rotZ = face.getHeadEulerAngleZ();  // Head is tilted sideways rotZ degrees
// If landmark detection was enabled (mouth, ears, eyes, cheeks, and
// nose available):
FaceLandmark leftEar = face.getLandmark(FaceLandmark.LEFT_EAR);
if (leftEar != null) {
   PointF leftEarPos = leftEar.getPosition();
}
// If contour detection was enabled:
List<PointF> leftEyeContour =
      face.getContour(FaceContour.LEFT_EYE).getPoints();
List<PointF> upperLipBottomContour =
      face.getContour(FaceContour.UPPER_LIP_BOTTOM).getPoints();
```

```
for (Face face : faces) {
   Rect bounds = face.getBoundingBox();
}
```

- When the face detection is successful, use the canvas to draw the detection boxes.

```
Paint paint= new Paint();
paint.setAntiAlias(true);
paint.setColor(Color.RED);
paint.setStyle(Paint.Style.STROKE);
paint.setStrokeWidth(8);

canvas.drawRect(bounds,paint);

iw= (ImageView)findViewById(R.id.image_view);
iw.setImageBitmap(mutableBitmap);
```

Note

- If the face is not detected and there are errors, try rebuild/rerun it several times
- E/Vision: Error loading module com.google.android.gms.vision.face optional module true: gu: No acceptable module found

Extra credit

- Implement the feature to track face in real time
  - Display a video. It can be a prerecorded video or it can be a real time camera video preview
  - Process the image frame by frame, and draw the bounding boxes on the faces.