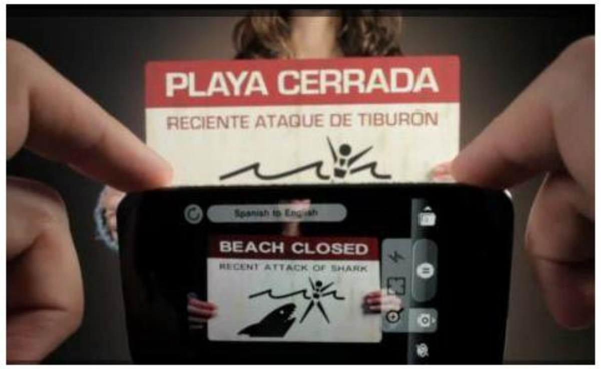
# Camera, Audio, Video and Sound

CSE 162 – Mobile Computing

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

# An interesting camera app: Word Lens Feature of Google Translate

- Word Lens: translates text/signs in foreign Language in real time
- Example use case: tourist can understand signs, restaurant menus
- Uses Optical Character Recognition technology
- Google bought company in 2014, now part of Google Translate



[ Original Word Lens App ]

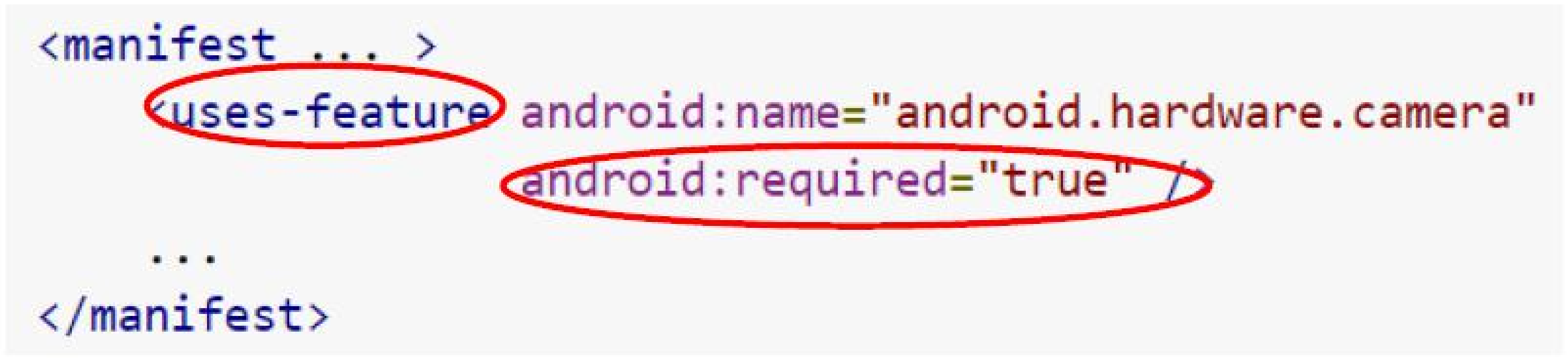
[ Word Lens as part of Google Translate ]

## Taking Pictures in an Android App

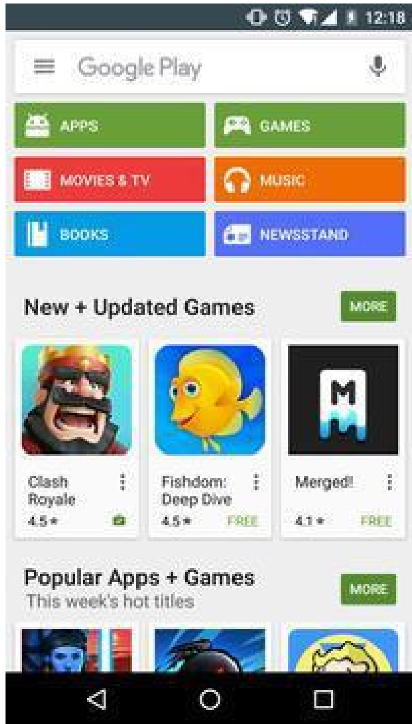
- How to take photos from your app using Android Camera app
- 4 Steps:
  - Request the camera feature
  - Take a Photo with the Camera App
  - Get the Thumbnail
  - Save the Full-size Photo

## 1. Request the Smartphone Camera Feature

- If your app takes pictures using the phone’s Camera, you can allow only devices with a camera find your app while searching Google Play Store
- How?
- Make the following declaration in AndroidManifest.xml



```
<manifest ... >
    <uses-feature android:name="android.hardware.camera"
                  android:required="true" />
    ...
</manifest>
```



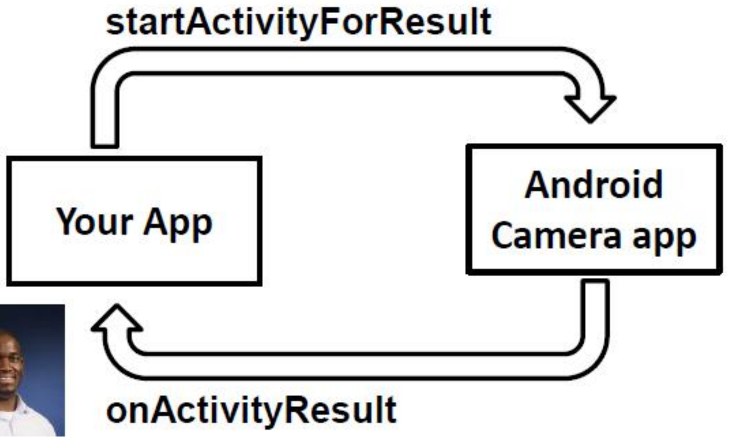
## 1. Request the Smartphone Camera Feature (continued)

- Runtime permission is needed from the users.

```
if (ContextCompat.checkSelfPermission(getContext(),
        Manifest.permission.CAMERA) != PackageManager.PERMISSION_GRANTED) {

    if (ActivityCompat.shouldShowRequestPermissionRationale((Activity)
            getContext(), Manifest.permission.CAMERA)) {


    } else {
        ActivityCompat.requestPermissions((Activity) getContext(),
                new String[]{Manifest.permission.CAMERA},
                MY_PERMISSIONS_REQUEST_CAMERA);
    }

}
```

## 2. Capture an Image with the Camera App

- To take picture, your app needs to send **implicit Intent** requesting for a picture to be taken (i.e. action = capture an image)
- Call **startActivityForResult( )** with Camera intent since picture sent back
  - What type of intent is it? implicit or explicit?
- Potentially, multiple apps/activities can handle this/take a picture
- Check that at least 1 Activity that can handle request to take picture using **resolveActivity**



## Code to Take a Photo with the Camera App

1. Build Intent, action = capture an image

```
static final int REQUEST_IMAGE_CAPTURE = 1;

private void dispatchTakePictureIntent() {
    Intent takePictureIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
    if (takePictureIntent.resolveActivity(getPackageManager()) != null) {
        startActivityForResult(takePictureIntent, REQUEST_IMAGE_CAPTURE);
    }
}
```

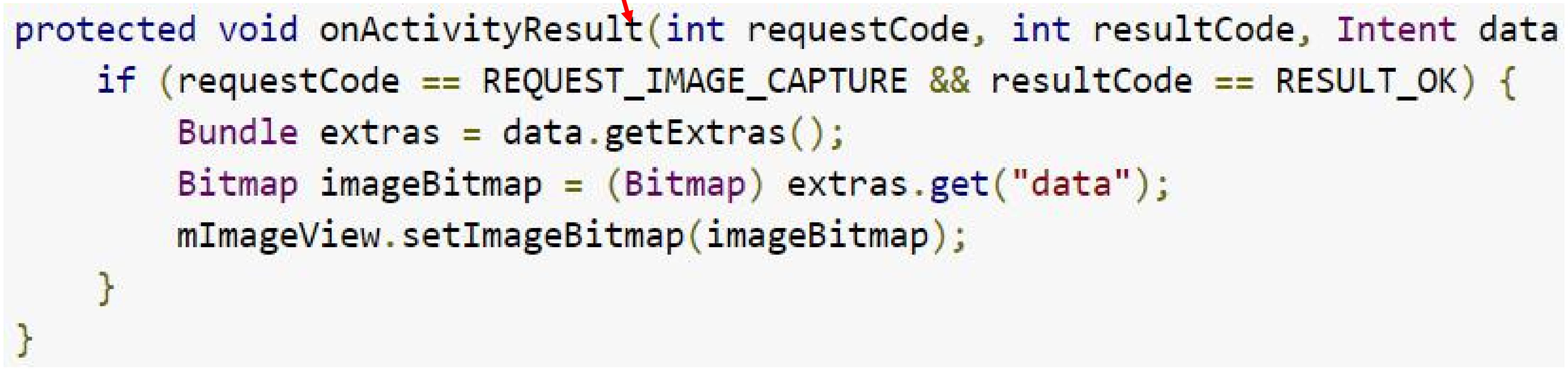
**2. Check that there's at least 1 Activity that can handle request to capture an image (Avoids app crashing if no camera app available)**

**3. Send Intent requesting an image to be captured (usually handled by Android's Camera app)**

## 3. Get the Thumbnail

- Android Camera app returns thumbnail of photo (small bitmap)
- Thumbnail bitmap returned in “extra” of **Intent** delivered to **onActivityResult( )**

**In onActivityResult( ), receive thumbnail picture sent back**



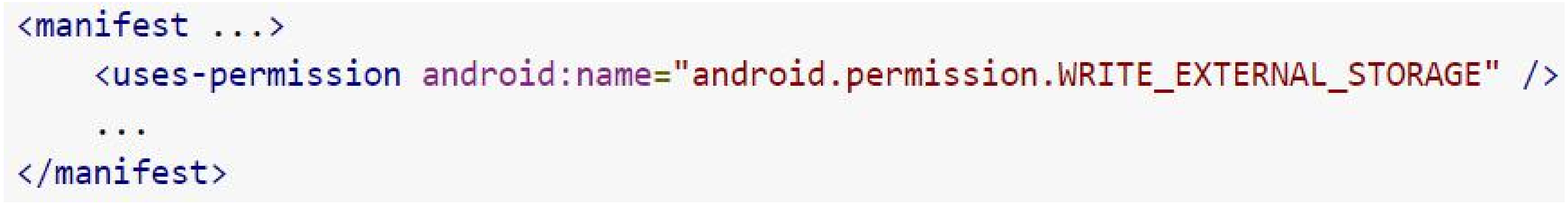
```
protected void onActivityResult(int requestCode, int resultCode, Intent data
    if (requestCode == REQUEST_IMAGE_CAPTURE && resultCode == RESULT_OK) {
        Bundle extras = data.getExtras();
        Bitmap imageBitmap = (Bitmap) extras.get("data");
        mImageView.setImageBitmap(imageBitmap);
    }
}
```

## 4. Save Full-Sized Photo

- Android Camera app saves full-sized photo in a filename you give it
- We need phone owner’s permission to write to external storage
- Android systems have:
    - **Internal storage:** data stored here is available by only your app
    - **External storage:** available stored here is available to all apps
- Would like all apps to read pictures this app takes, so use external storage

# Save Full-Sized Photo

- Android Camera app can save full-size photo to
  - **Public external storage** (shared by all apps)
    - **getExternalStoragePublicDirectory( )**
    - Need to get permission
  - **Private storage** (Seen by only your app, deleted when your app uninstalls):
    - **getExternalFilesDir( )**
- Either way, need phone owner’s permission to write to external storage
- In AndroidManifest.xml, make the following declaration



```
<manifest ...>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    ...
</manifest>
```

```
static final int REQUEST_TAKE_PHOTO = 1;

private void dispatchTakePictureIntent() {
    Intent takePictureIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
    // Ensure that there's a camera activity to handle the intent
    if (takePictureIntent.resolveActivity(getPackageManager()) != null) {
        // Create the File where the photo should go
        File photoFile = null;
        try {
            photoFile = createImageFile();
        } catch (IOException ex) {
            // Error occurred while creating the File
            ...
        }
        // Continue only if the File was successfully created
        if (photoFile != null) {
            Uri photoURI = FileProvider.getUriForFile(this,
                                                  "com.example.android.fileprovider",
                                                  photoFile);
            takePictureIntent.putExtra(MediaStore.EXTRA_OUTPUT, photoURI);
            startActivityForResult(takePictureIntent, REQUEST_TAKE_PHOTO);
        }
    }
}
```

**Create new intent for image capture**

**Check if a camera exists**

**Create file to store full-sized image**

**Build URI location to store captured image: file//xyz**

**Put URI into intent extras**

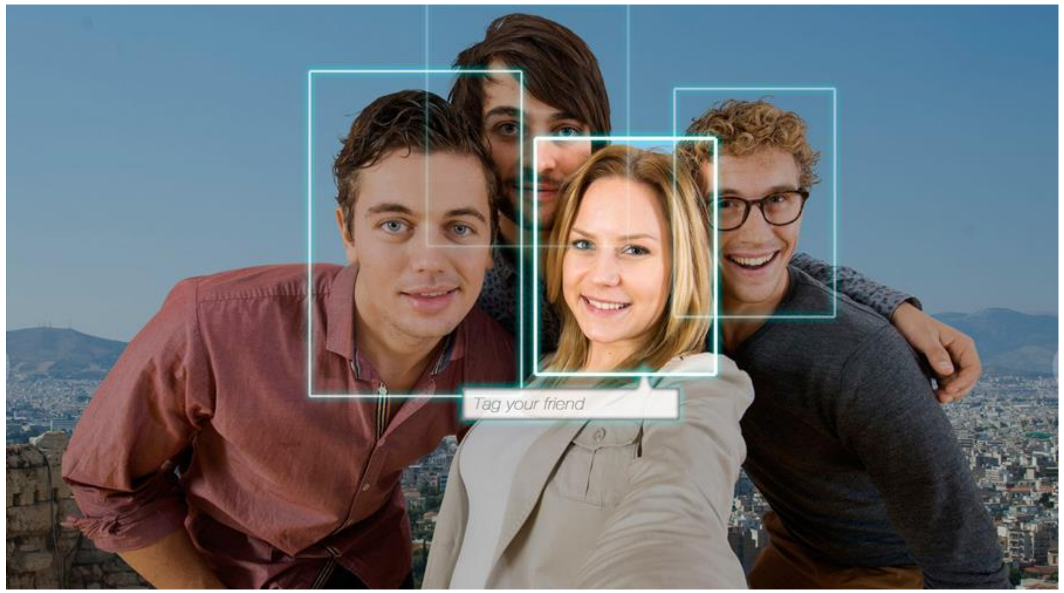
**Take a picture**

## Face Recognition

## Face Recognition

- Answers the question:

  **Who** is this person in this picture?
- Compares unknown face to database of faces (or facial attributes) with known identity
- Neural networks/deep learning now makes comparison faster



## FindFace App: Stalking on Steroids?

- See stranger you like? Take a picture
- App searches 1 billion pictures using neural networks < 1 second
- Finds person's picture, identity, link on VK (Russian Facebook)
- You can send friend Request
  - ~ 70% accurate!
- Can also upload picture of celebrity you like
- Finds 10 strangers on Facebook who look similar, can send friend request

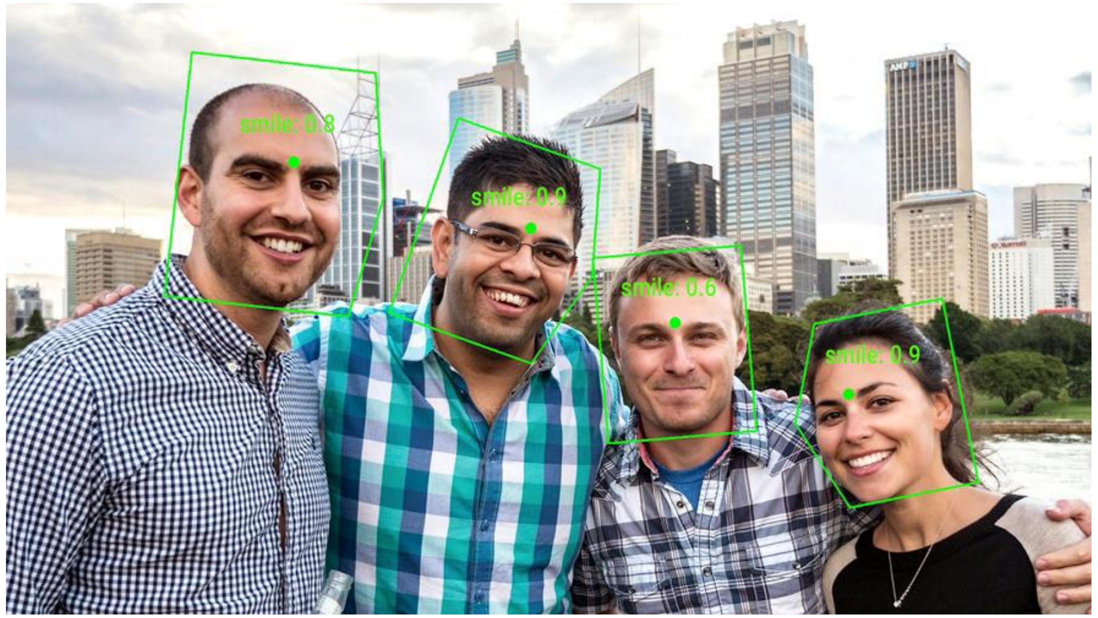
## FindFace App

- Also used in law enforcement
- Police identify criminals on watchlist

## Face Detection

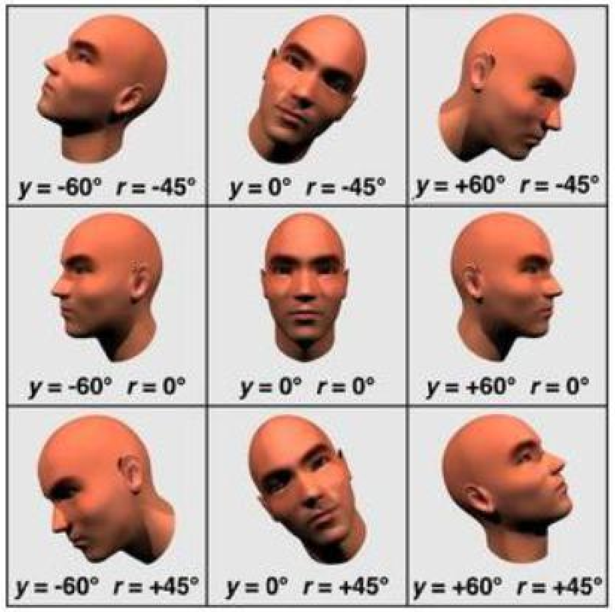
## Mobile Vision API

- **Face Detection:** Are there [any] faces in this picture?
- **How?** Locate face in photos and video and
- **Facial landmarks:** Eyes, nose and mouth
- **State of facial features:** Eyes open? Smiling?



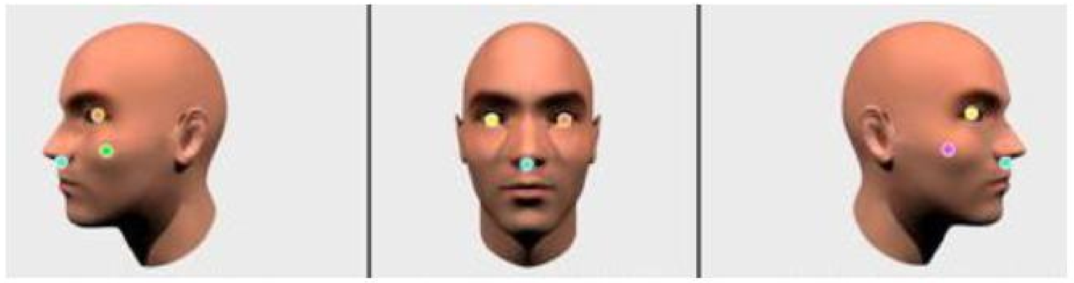
# Face Detection: Google Mobile Vision API

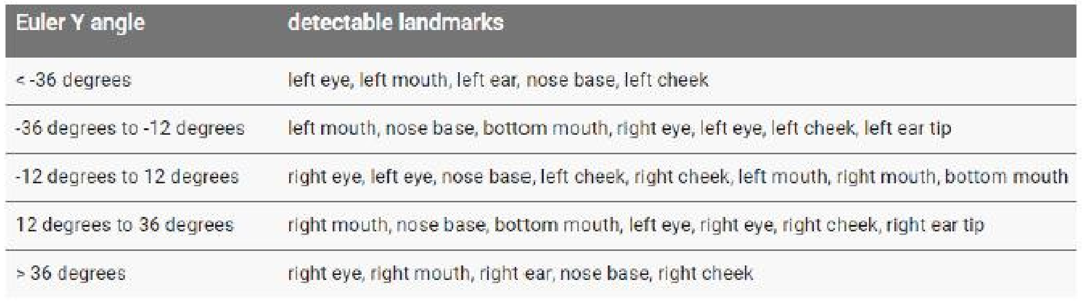
- Detects faces:
    - reported at a position, with size and orientation
    - Can be searched for landmarks (e.g. eyes and nose)



Orientation

Landmarks





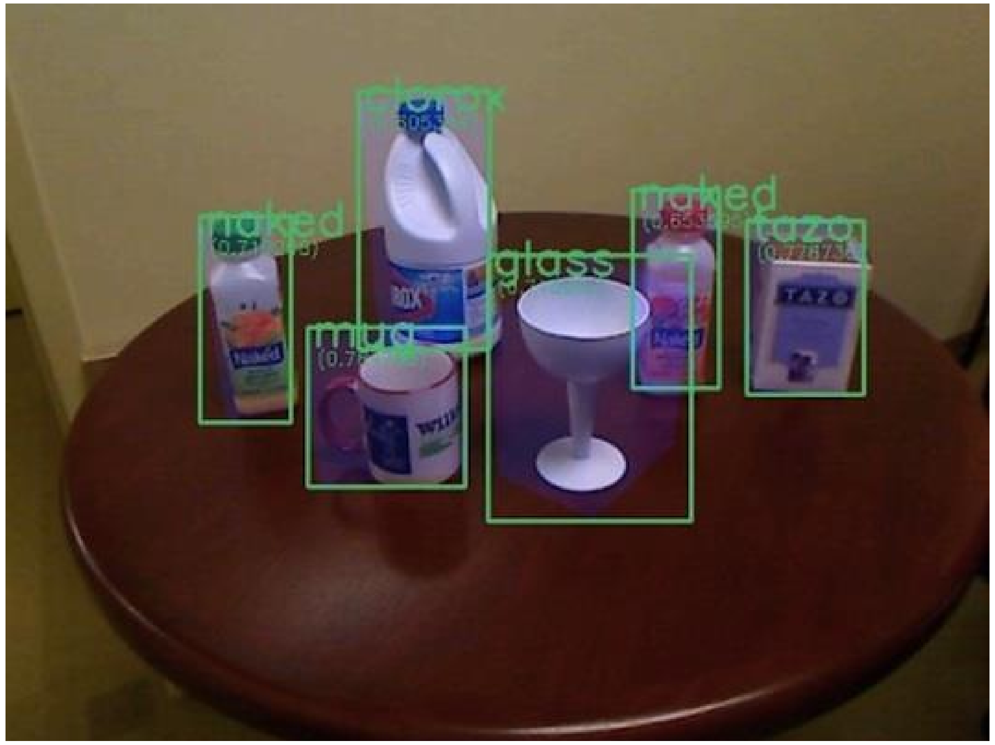
| Euler Y angle | detectable landmarks |
|---|---|
| < -36 degrees | left eye, left mouth, left ear, nose base, left cheek |
| -36 degrees to -12 degrees | left mouth, nose base, bottom mouth, right eye, left eye, left cheek, left ear tip |
| -12 degrees to 12 degrees | right eye, left eye, nose base, left cheek, right cheek, left mouth, right mouth, bottom mouth |
| 12 degrees to 36 degrees | right mouth, nose base, bottom mouth, left eye, right eye, right cheek, right ear tip |
| > 36 degrees | right eye, right mouth, right ear, nose base, right cheek |

## Google Mobile Vision API

- Mobile Vision API also does:
  - **Face tracking:** detects faces in consecutive video frames
  - **Classification:** Eyes open? Face smiling?
- Classification:
  - Determines whether a certain facial characteristic is present
  - API currently supports 2 classifications: eye open, smiling
  - Results expressed as a confidence that a facial characteristic is present
    - Confidence > 0.7 means facial characteristic is present
    - E.g. > 0.7 confidence means it’s likely person is smiling
- Mobile vision API does face **detection** but NOT **recognition**

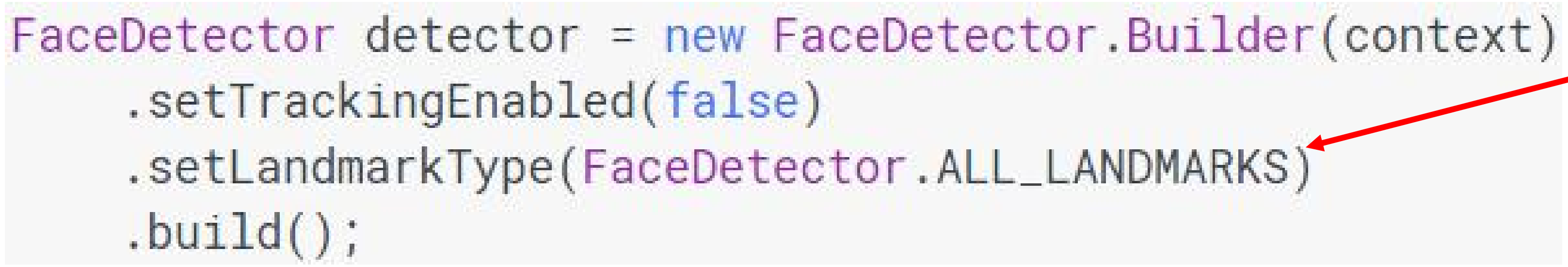
## Face Detection

- **Face detection:** Special case of object-class detection
- **Object-class detection task:** find locations and sizes of all objects in an image that belong to a given class.
    - E.g: bottles, cups, pedestrians, and cars
- **Object matching:** Objects in picture compared to objects in database of labelled pictures



## Creating the Face Detector

- In app’s **onCreate** method, create face detector



```
FaceDetector detector = new FaceDetector.Builder(context)
        .setTrackingEnabled(false)
        .setLandmarkType(FaceDetector.ALL_LANDMARKS)
        .build();
```

**Detect all landmarks**

- **detector**is base class for implementing specific detectors. E.g. face detector, bar code detector
- Tracking finds same points in multiple frames (continuous)
- Detection works best in single images when **trackingEnabled** is false

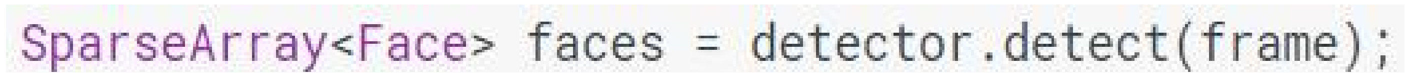
## Detecting Faces and Facial Landmarks

- Create Frame (image data, dimensions) instance from bitmap supplied

```
Frame frame = new Frame.Builder().setBitmap(bitmap).build();
```

## Detecting Faces and Facial Landmarks

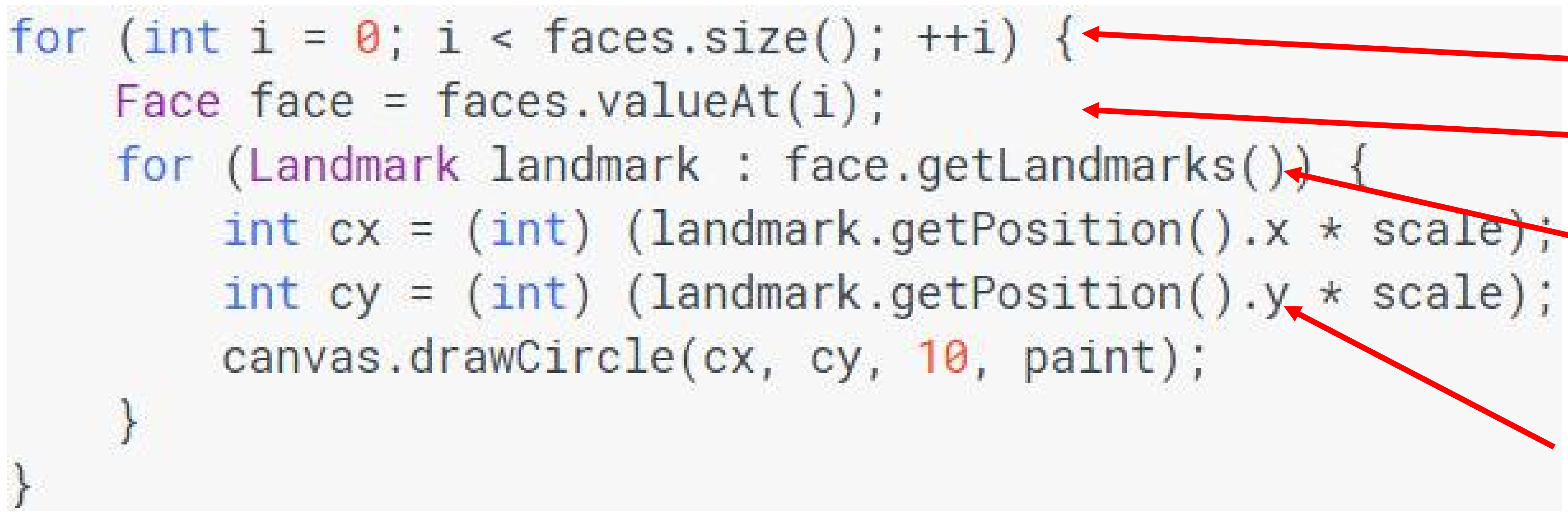
- Call detector synchronously with frame to detect faces



```
SparseArray<Face> faces = detector.detect(frame);
```

# Detecting Faces and Facial Landmarks

- Detector takes **Frame** as input, outputs array of **Faces** detected
- **Face** is a single detected human face in image or video
- Iterate over array of faces, landmarks for each face, and draw the result based on each landmark's position



```
for (int i = 0; i < faces.size(); ++i) {
    Face face = faces.valueAt(i);
    for (Landmark landmark : face.getLandmarks()) {
        int cx = (int) (landmark.getPosition().x * scale);
        int cy = (int) (landmark.getPosition().y * scale);
        canvas.drawCircle(cx, cy, 10, paint);
    }
}
```

**Iterate through face array**

**Get face at position i in Face array**

**Return list of face landmarks, e.g., eyes, nose**

**Return landmark's xy position, where 0,0 is image's upper left corner**

## Other Stuff

- To count faces detected, call **faces.size( )**. E.g.

```
TextView faceCountView = (TextView) findViewById(R.id.face_count);
faceCountView.setText(faces.size() + " faces detected");
```

- Querying Face detector’s status

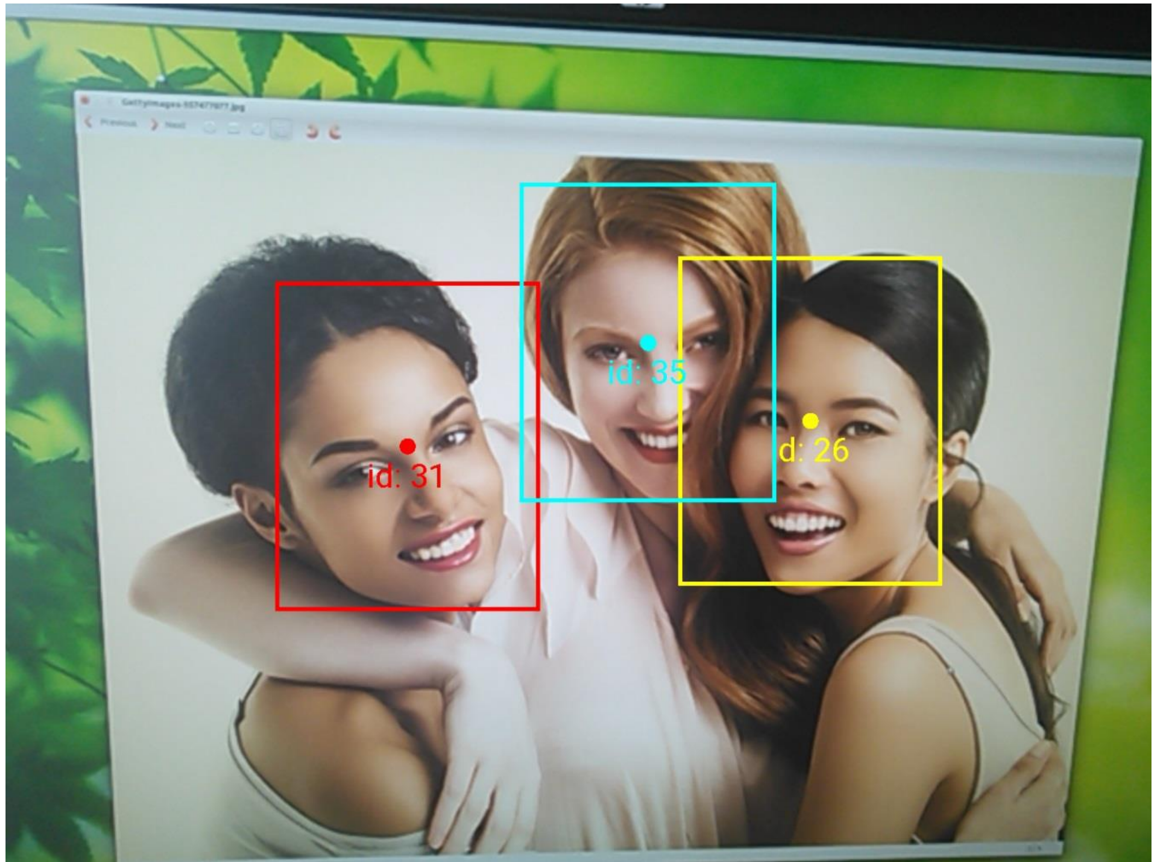
```
if (!detector.isOperational()) {
    // ...
}
```

- Releasing Face detector (frees up resources)

```
detector.release();
```

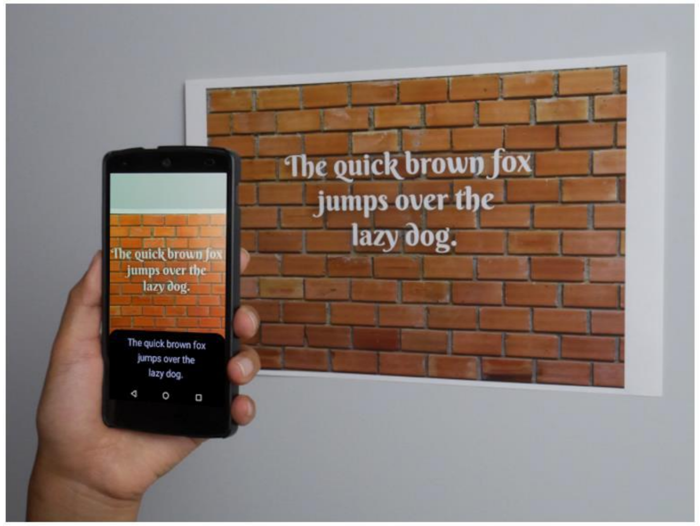
# Detect & Track Multiple Faces in Video

- Can also track multiple faces in image sequences/video, draw rectangle round each one



## Mobile Vision API: Other Functionality

- Barcode scanner
- Optical Character Recognition (OCR): Recognize text

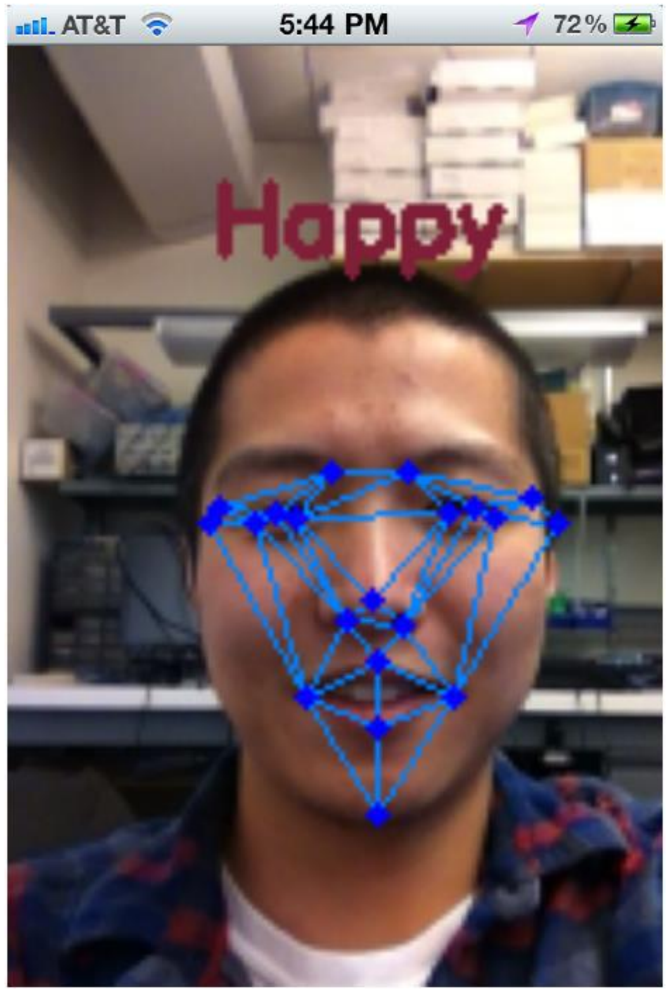

The quick brown fox jumps over the lazy dog.
The quick brown fox jumps over the lazy dog.
The quick brown fox jumps over the lazy dog.

## Face Interpretation

- Reference:
- Yang, Xiaochao, et al. "Visage: A face interpretation engine for smartphone  applications." *Mobile Computing, Applications, and Services Conference*.  Springer Berlin Heidelberg, 2012. 149-168.

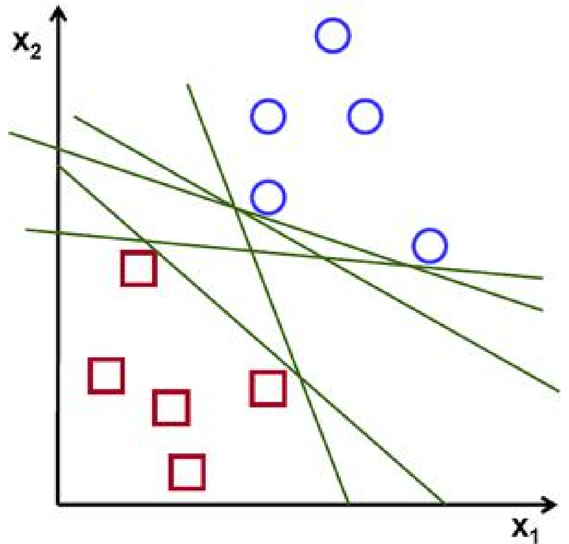
# Visage Face Interpretation Engine

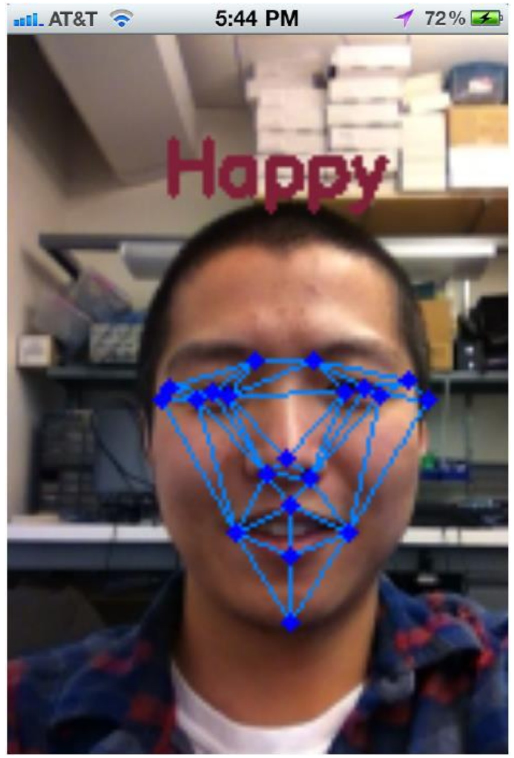
- Real-time face interpretation engine for smart phones
- Tracking user's 3D head orientation + facial expression
- Facial expression, affect, emotion
  - angry, disgust, fear, happy, neutral, sad, surprise
- Use? Can be used in Mood Profiler app



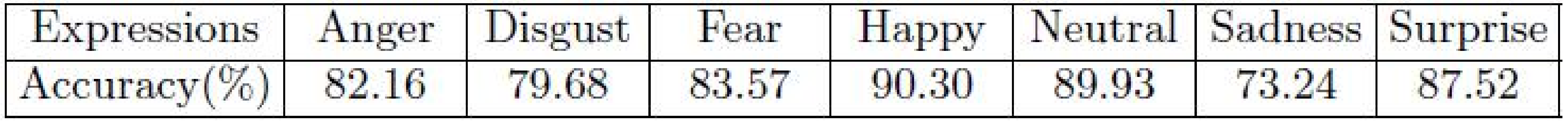
## Facial Expression Inference

- Active appearance model
  - Describes 2D image as triangular mesh of landmark points
- 7 expression classes: angry, disgust, fear, happy, neutral, sad, surprise
- Extract triangle shape, texture features
- Classify features using Machine learning





## Classification accuracy



| Expressions | Anger | Disgust | Fear | Happy | Neutral | Sadness | Surprise |
|---|---|---|---|---|---|---|---|
| Accuracy(%) | 82.16 | 79.68 | 83.57 | 90.30 | 89.93 | 73.24 | 87.52 |

**Table 4.** Facial expression classification accuracy using the JAFFE dataset