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]

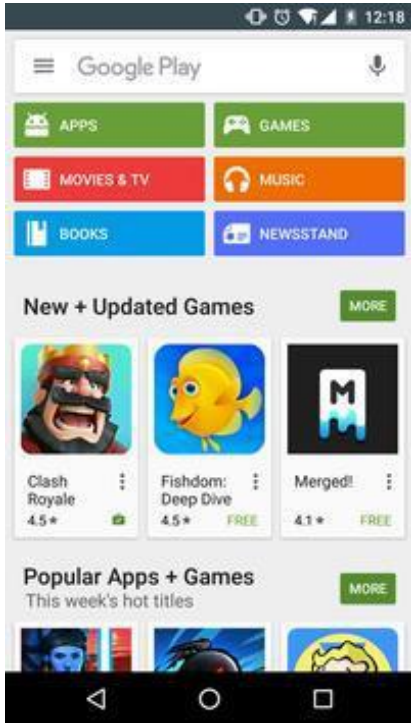
Taking Pictures in an Android App

- 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>
```



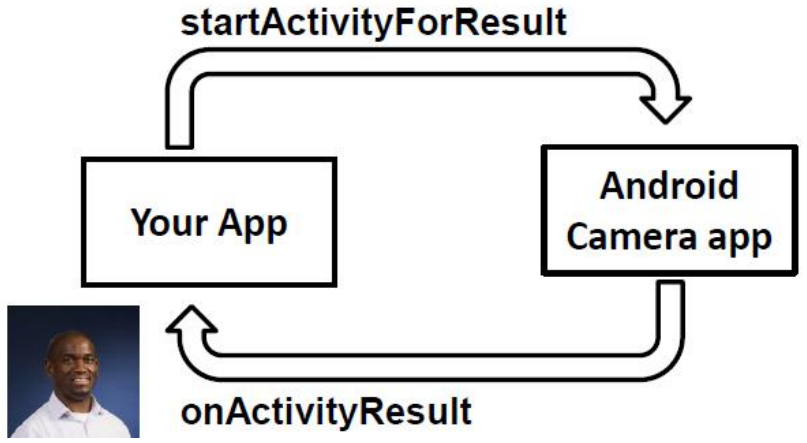
1. Request the Smartphone Camera Feature (continued)

- 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

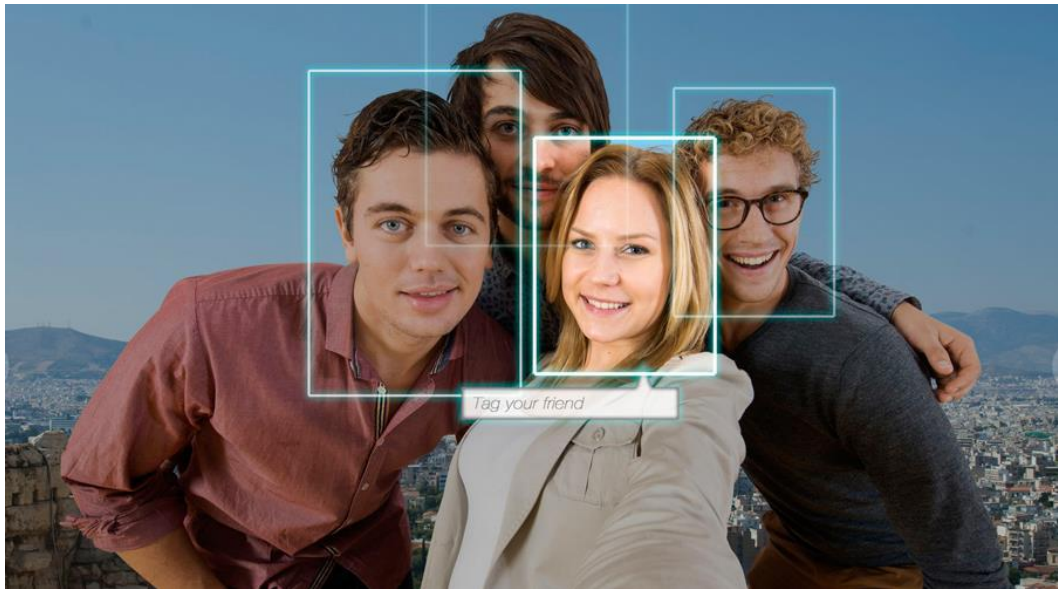
Take a picture

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

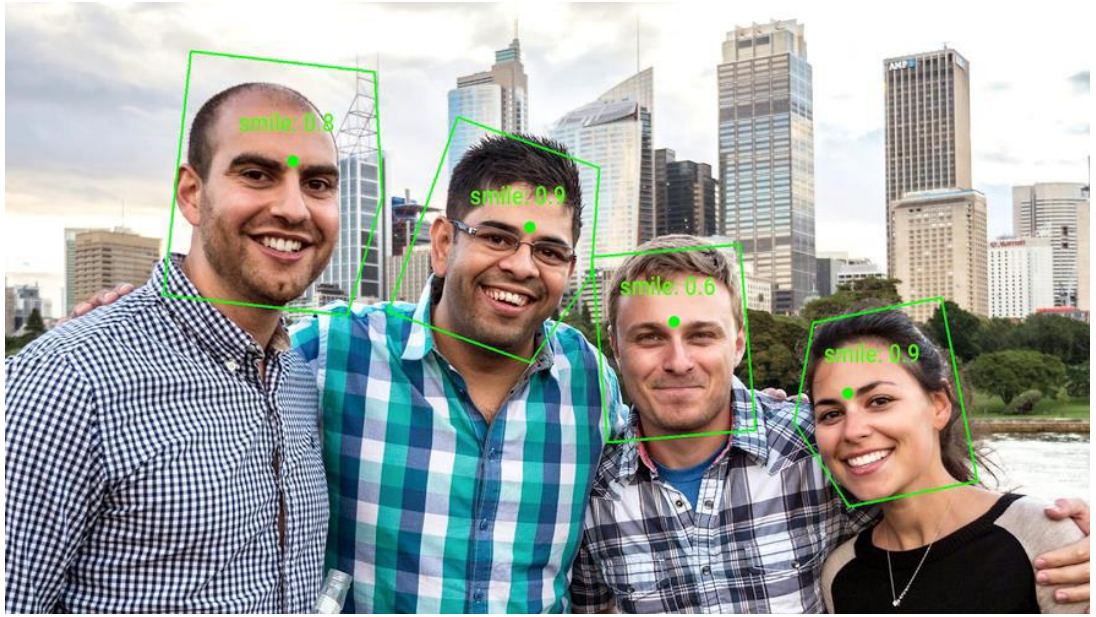
FindFace App

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

Face Detection

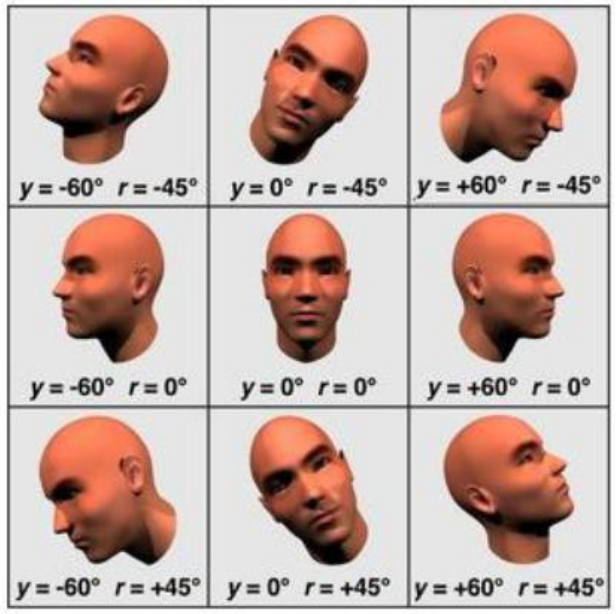
Mobile Vision API

- **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

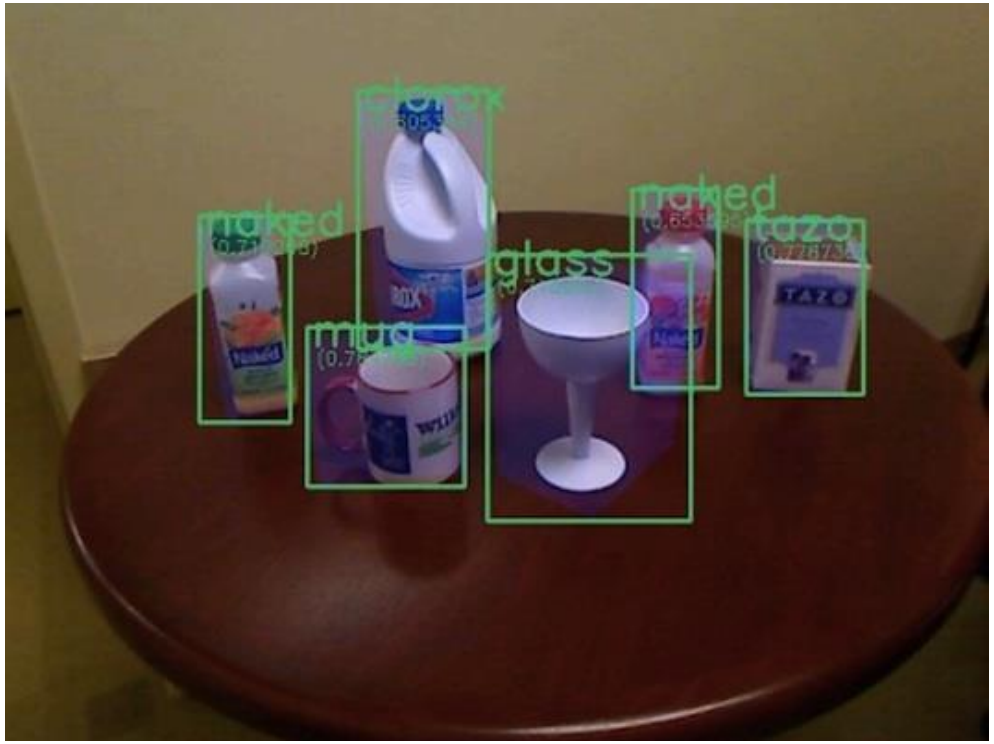
Euler Y angle	detectable landmarks
< -36 degrees	left eye, left mouth, left ear, nose base, left cheek
-36 degrees to -12 degrees	left mouth, nose base, bottom mouth, right eye, left eye, left cheek, left ear tip
-12 degrees to 12 degrees	right eye, left eye, nose base, left cheek, right cheek, left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

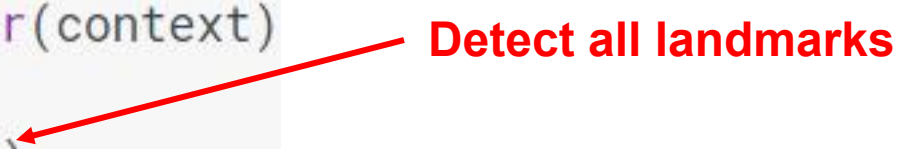
- **Face detection:** Special case of object-class detection
- **Object-class detection task:** find locations and sizes of all objects in an image that belong to a given class.
  - E.g: bottles, cups, pedestrians, and cars
- **Object matching:** Objects in picture compared to objects in database of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



- **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

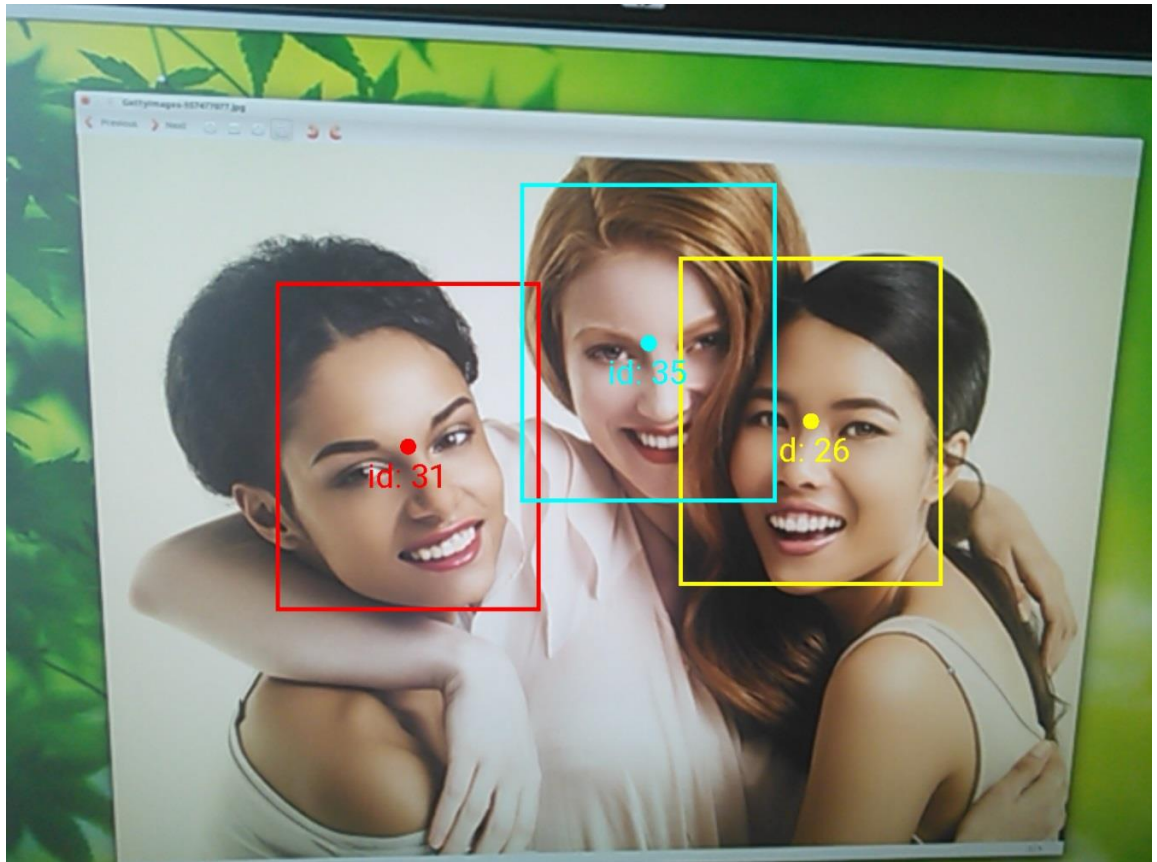
```
if (!detector.isOperational()) {
    // ...
}
```

- Releasing Face detector (frees up resources)

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

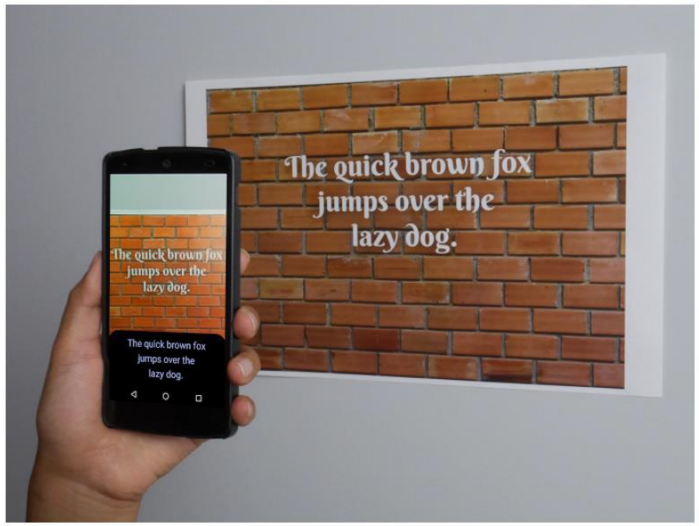
Detect & Track Multiple Faces in Video

- 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

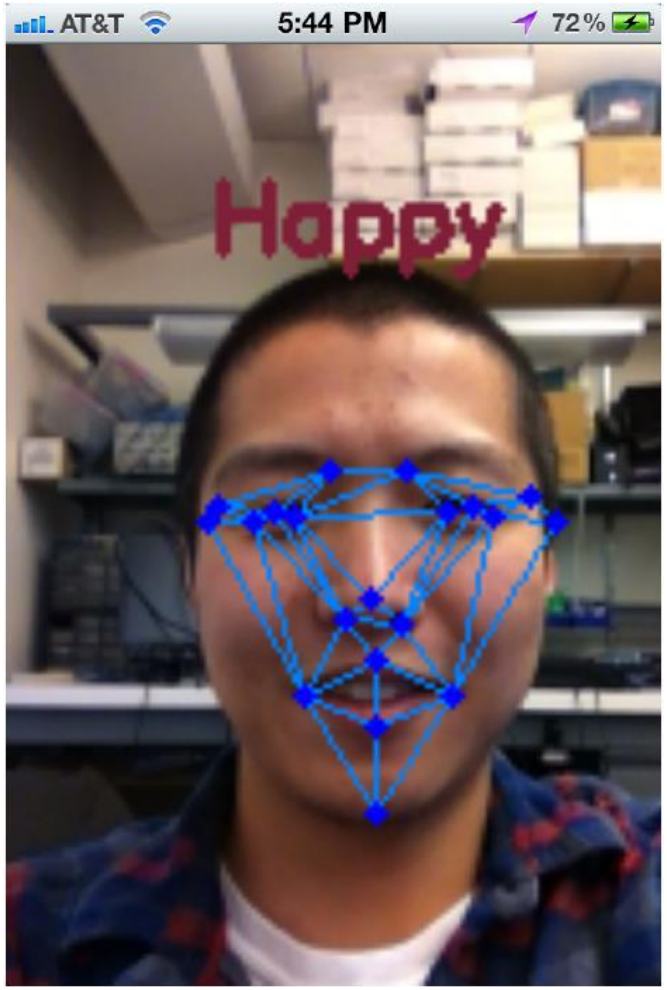


Face Interpretation

- 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.

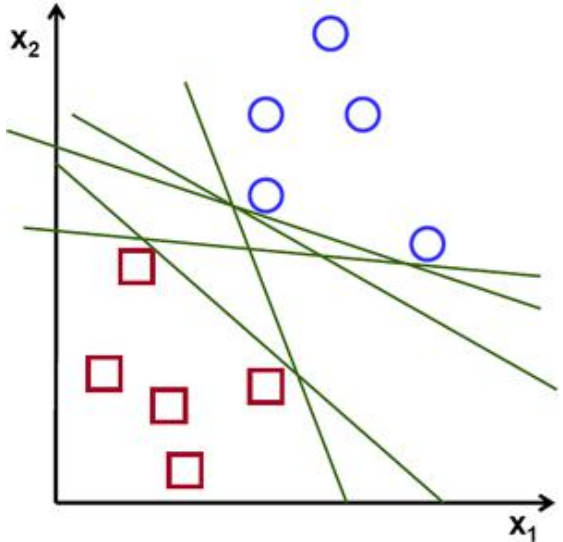
Visage Face Interpretation Engine

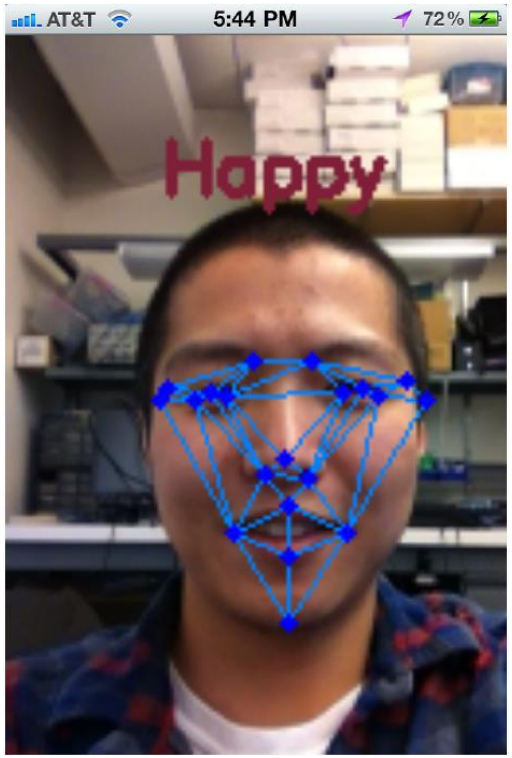
- Real-time face interpretation engine for smart phones
- Tracking user's 3D head orientation + facial expression
- Facial expression, affect, emotion
  - angry, disgust, fear, happy, neutral, sad, surprise
- Use? Can be used in Mood Profiler app



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

Camera, Audio, Video and Sound (Continued)

CSE 162 – Mobile Computing

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

Media (audio and video) Recording in Android

MediaRecorder overview

- The Android multimedia framework includes support for capturing and encoding a variety of common audio and video formats.

Requesting permission to record audio

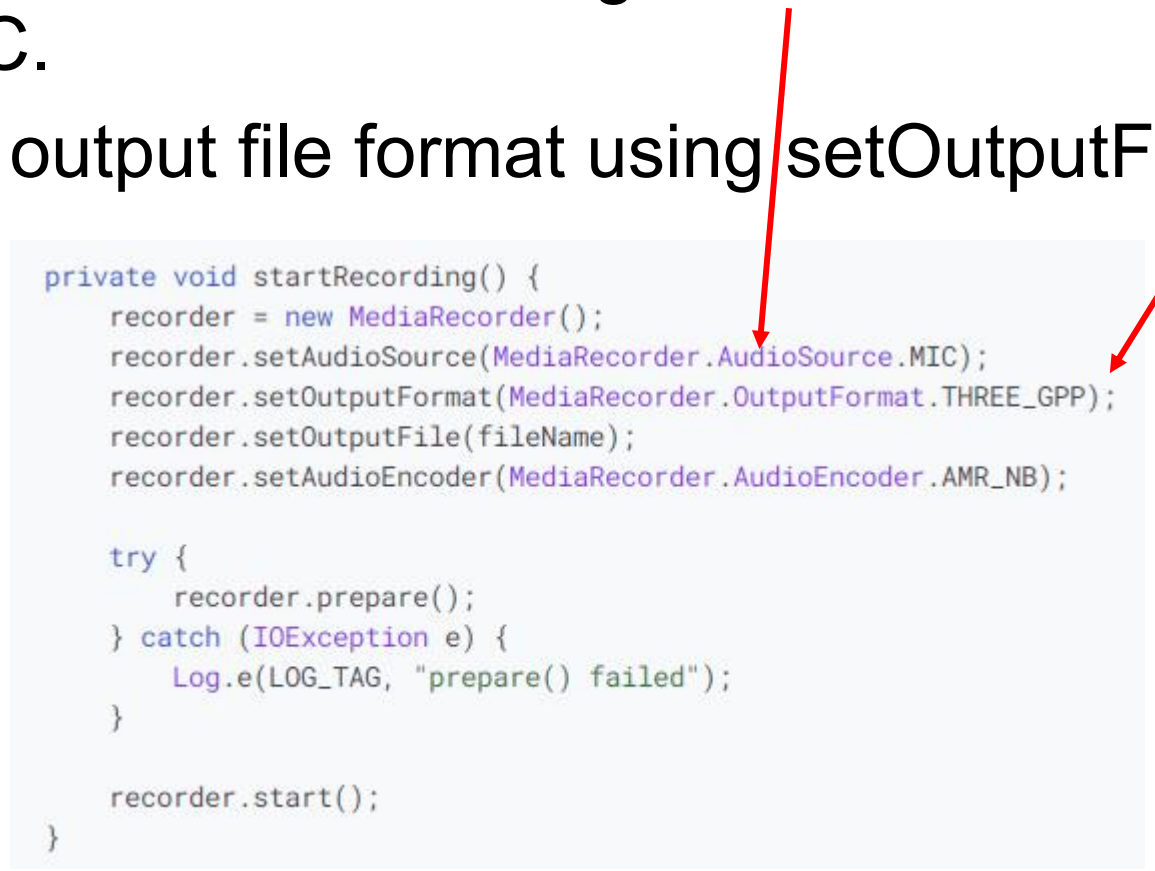
- To be able to record, your app must tell the user that it will access the device's audio input

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

- RECORD_AUDIO is considered a "dangerous" permission because it may pose a risk to the user's privacy.
- Starting with Android 6.0 (API level 23) an app that uses a dangerous permission must ask the user for approval at run time.
  - After the user has granted permission, the app should remember and not ask again.

Creating and running a MediaRecorder

- Set the audio source using setAudioSource(). You'll probably use MIC.
- Set the output file format using setOutputFormat()



```
private void startRecording() {
    recorder = new MediaRecorder();
    recorder.setAudioSource(MediaRecorder.AudioSource.MIC);
    recorder.setOutputFormat(MediaRecorder.OutputFormat.THREE_GPP);
    recorder.setOutputFile(fileName);
    recorder.setAudioEncoder(MediaRecorder.AudioEncoder.AMR_NB);

    try {
        recorder.prepare();
    } catch (IOException e) {
        Log.e(LOG_TAG, "prepare() failed");
    }

    recorder.start();
}
```

Creating and running a MediaRecorder

- Set the output file name using setOutputFile(). You must specify a file descriptor that represents an actual file.
- Set the audio encoder using setAudioEncoder().
- Complete the initialization by calling prepare().
- Start the recorder.

```
private void startRecording() {
    recorder = new MediaRecorder();
    recorder.setAudioSource(MediaRecorder.AudioSource.MIC);
    recorder.setOutputFormat(MediaRecorder.OutputFormat.THREE_GPP);
    recorder.setOutputFile(fileName);
    recorder.setAudioEncoder(MediaRecorder.AudioEncoder.AMR_NB);

    try {
        recorder.prepare();
    } catch (IOException e) {
        Log.e(LOG_TAG, "prepare() failed");
    }

    recorder.start();
}
```

Stop recording

```
private void stopRecording() {
    recorder.stop();
    recorder.release();
    recorder = null;
}
```

Audio Project Ideas

- OpenAudio project, http://www.openaudio.eu/
  - Many tools, dataset available
    - OpenSMILE: Tool for extracting > 1000 audio features
    - Windowing
  - MFCC
  - Pitch
  - Statistical features, etc
- Supports popular file formats (e.g. Weka)
- OpenEAR: Toolkit for automatic speech emotion recognition
- iHeaRu-EAT Database: 30 subjects recorded speaking while eating

Speech Recognition

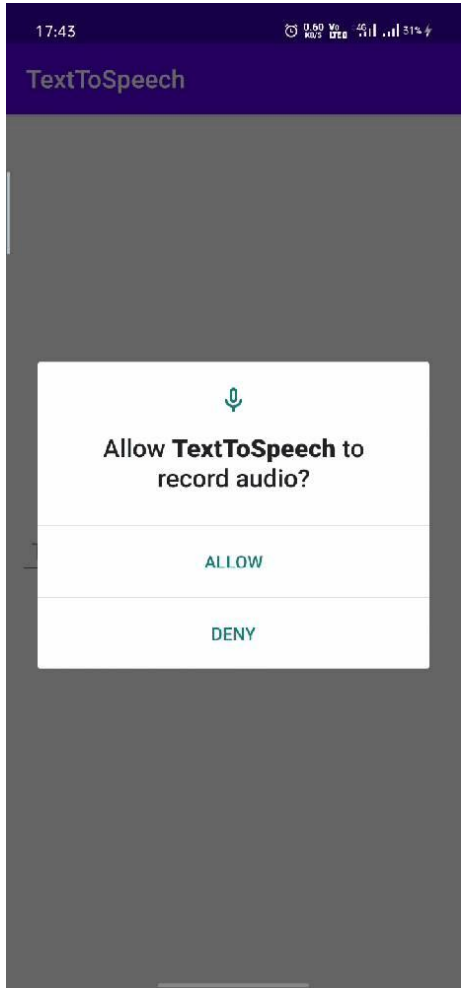
- peech recognition, also known as automatic speech recognition (ASR), computer speech recognition or speech-to-text, is a capability that enables a program to process human speech into a written format.

Speech recognition use cases

- Automotive: Speech recognizers improves driver safety by enabling voice-activated navigation systems and search capabilities in car radios.

- Technology: Virtual agents are increasingly becoming integrated within our daily lives, particularly on our mobile devices.
  - We use voice commands to access them through our smartphones, such as through Google Assistant or Apple's Siri, for tasks, such as voice search, or through our speakers, via Amazon's Alexa or Microsoft's Cortana, to play music.
- Healthcare: Doctors and nurses leverage dictation applications to capture and log patient diagnoses and treatment notes.

Build a speech recognizer in Android



The Android SpeechRecognizer Class

- This class provides access to the speech recognition service.
- The implementation of this API is to stream audio to remote servers to perform speech recognition.
  - Not suitable for continuous recognition
  - consume a significant amount of battery and bandwidth.

Usage of the SpeechRecognizer Class

- Create the class using *createSpeechRecognizer*()

```
speechRecognizer = SpeechRecognizer.createSpeechRecognizer(this);
final Intent speechRecognizerIntent = new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
speechRecognizerIntent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL,RecognizerIntent.LANGUAGE_MODEL_FREE_FORM);
speechRecognizerIntent.putExtra(RecognizerIntent.EXTRA_LANGUAGE, Locale.getDefault());
```

- Create a speechRecognizerIntent.
  - The constant **ACTION_RECOGNIZE_SPEECH** starts an activity
    - The activity will prompt the user to speak and send it through a speech recognizer.
  - ***EXTRA_LANGUAGE_MODEL***: Informs the recognizer which speech model to use
  - ***EXTRA_LANGUAGE_MODEL***: Informs the recognizer which speech model to prefer when performing ACTION_RECOGNIZE_SPEECH.
  - ***EXTRA_LANGUAGE****:* Optional IETF language tag (as defined by BCP 47), for example, “en-US”.

```
1  speechRecognizer = SpeechRecognizer.createSpeechRecognizer(this);
2  final Intent speechRecognizerIntent = new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
3  speechRecognizerIntent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL,RecognizerIntent.LANGUAGE_MODEL_FREE_FORM);
4  speechRecognizerIntent.putExtra(RecognizerIntent.EXTRA_LANGUAGE, Locale.getDefault());
```

Create a speechRecognitionListener class

```
speechRecognizer.setRecognitionListener(new RecognitionListener() {
    @Override
    public void onBeginningOfSpeech() {
        editText.setText("");
        editText.setHint("Listening...");
    }
    @Override
    public void onResults(Bundle bundle) {
        micButton.setImageResource(R.drawable.ic_mic_black_off);
        ArrayList<String> data = bundle.getStringArrayList(SpeechRecognizer.RESULTS_RECOGNITION);
            editText.setText(data.get(0));
    }
});
```

- We get an ArrayList of String as a result
- We can use it to parse a command, or input texts.

```
public void onResults(Bundle bundle) {
    micButton.setImageResource(R.drawable.ic_mic_black_off);
    ArrayList<String> data = bundle.getStringArrayList(SpeechRecognizer.RESULTS_RECOGNITION);
        editText.setText(data.get(0));
}
```

Speech Analytics

Voice Based Analytics

- Voice can be analyzed, lots of useful information extracted
  - Who is talking? (Speaker identification)
  - How many social interactions a person has a day
  - Emotion of person while speaking
  - Anxiety, depression, intoxication, of person, etc.
- For speech recognition, voice analytics used to:
  - Discard useless information (background noise, etc)
  - Identify and extract linguistic content

Mel Frequency Cepstral Coefficients (MFCCs)

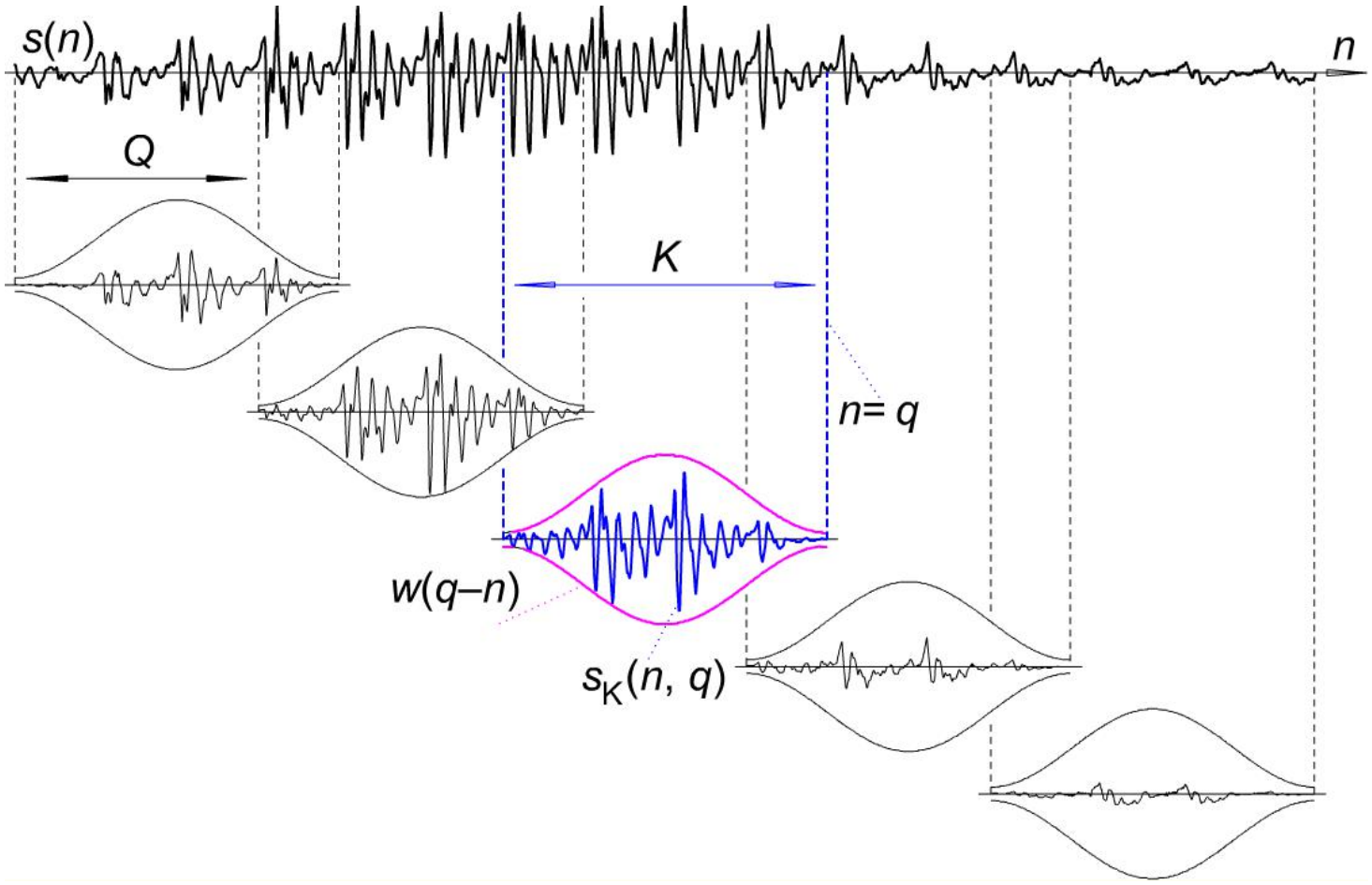
- MFCCs widely used in speech and speaker recognition for representing envelope of **power spectrum of voice**
- **Power spectrum?** Amount of power at various frequencies
- Roughly
  - Male voice: low frequency (bass)
  - Female voice: high frequency (treble)
- Popular approach in Speech recognition
  - MFCC features + Hidden Markov Model (HMM) classifiers

MFCC Steps: Overview

1. Frame the signal into short frames.
2. For each frame calculate the periodogram estimate of the power spectrum.
3. Apply the me lfilter bank to the power spectra, sum the energy in each filter.
4. Take the logarithm of all filter bank energies.
5. Take the DCT of the log filter bank energies.
6. Keep DCT coefficients 2-13, discard the rest.

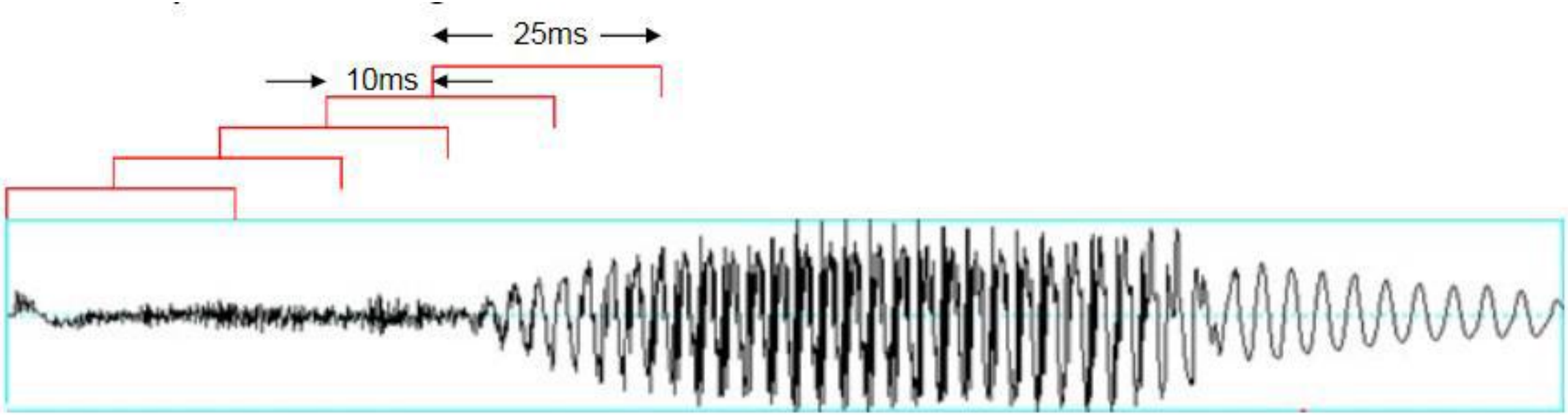
Step 1: Windowing

- Audio is continuously changing.
- Break into short, overlapping segments (20-40 milliseconds)
- Can assume audio does not change in short window



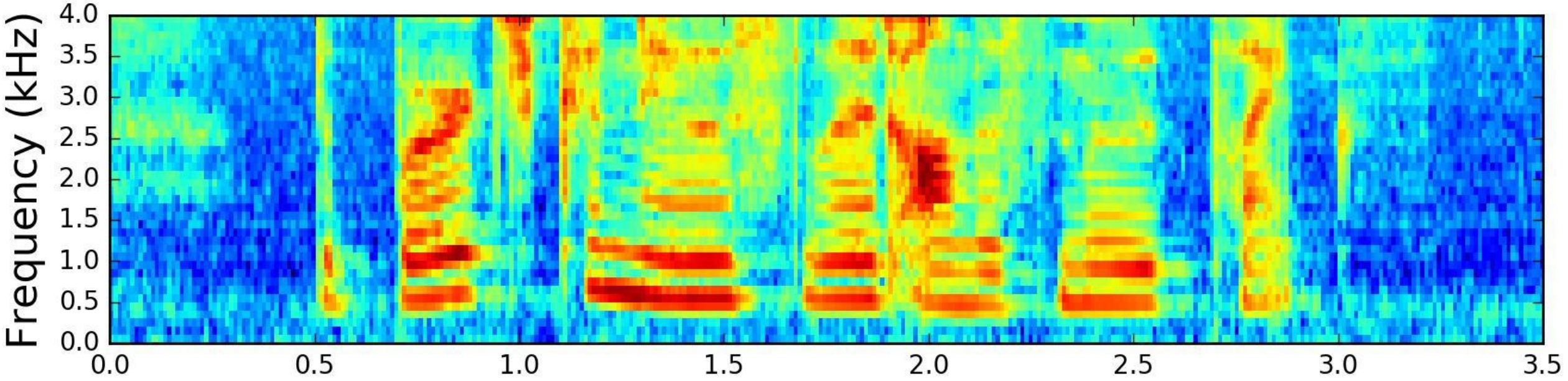
Step 1: Windowing

- Essentially, break into smaller overlapping frames
- Need to select frame length (e.g. 25 ms), shift (e.g. 10 ms)



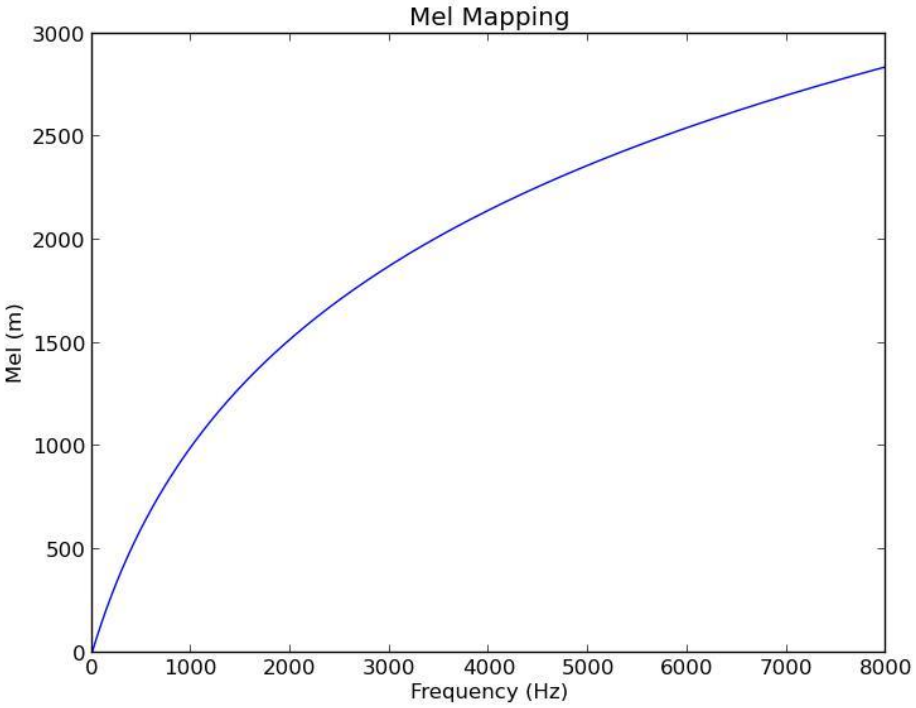
Step 2: Calculate Power Spectrum of each Frame

- Cochlea (**kaa**·klee·uh), part of human ear, vibrates at different parts depending on sound frequency
- Power spectrum Periodogram similarly identifies frequencies present in each frame



Background: Mel Scale

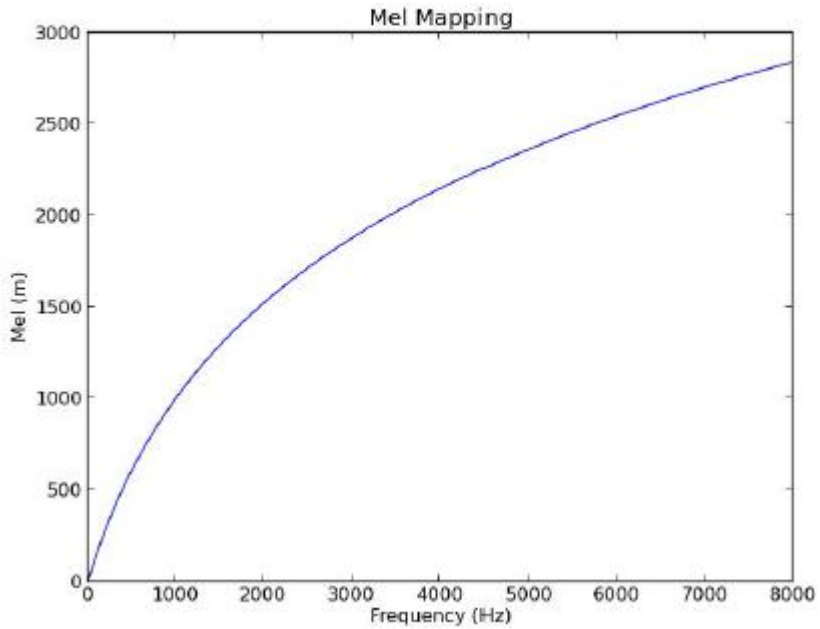
- Transforms speech attributes (frequency, tone, pitch) on non-linear scale based on human perception of voice
- Result: non-linear amplification, MFCC features that mirror human perception
  - E.g. humans good at perceiving small change at low frequency than at high frequency



Step 3: Apply Mel FilterBank

Non-linear conversion from frequency to Mel Space

$$M(f) = 1125 \ln(1 + f/700) \qquad (1)$$



Step 4: Apply Logarithm of Mel Filterbank

- Take log of filterbank energies at each frequency
- This step makes output mimic human hearing better
- We don’t hear loudness on a linear scale
- Changes in loud noises may not sound different

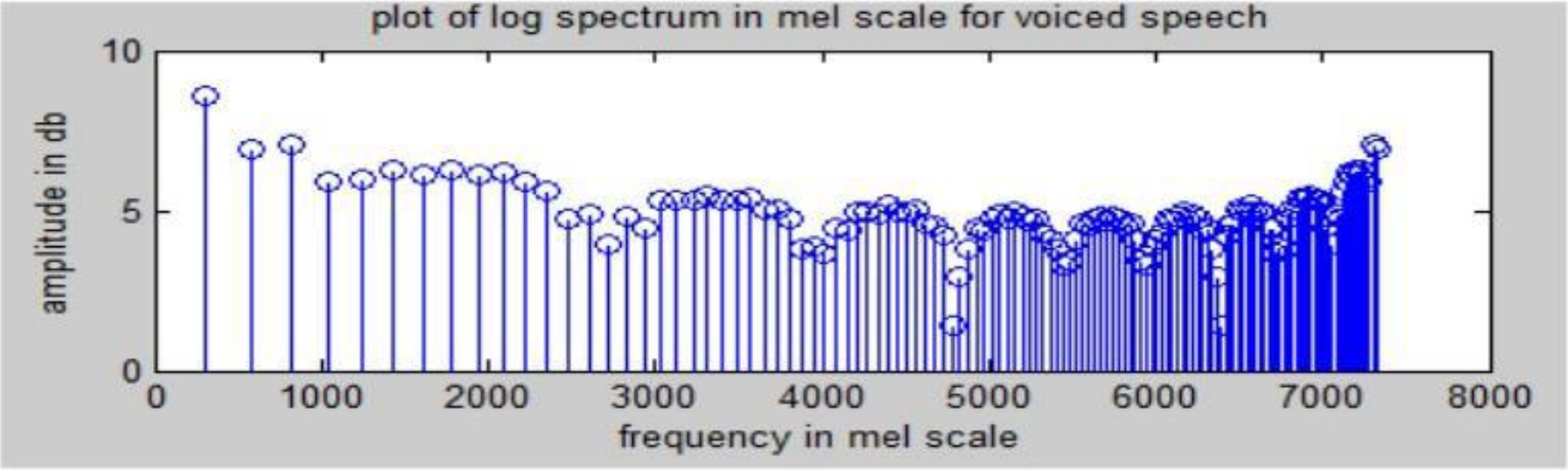


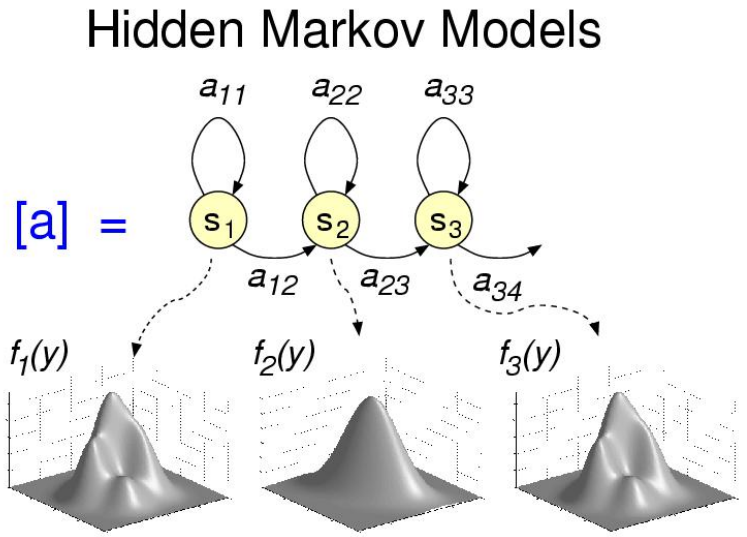
Fig.7. Spectrum of voiced speech

Step 5: DCT of log filterbank:

- There are correlations between signals at different frequencies
- Discrete Cosine Transform (DCT) extracts most useful and independent features
- Final result: 39-element acoustic vector used in speech processing algorithms

Speech Classification

- Human speech can be broken into phonemes
- Example of phoneme is /k/ in the words (**c**at, s**ch**ool, s**k**ill)
- Classic Speech recognition tries to recognize sequence of phonemes in a word
- Typically uses Hidden Markov Model (HMM)
  - Recognizes letters, then words, then sentences
  - Like a state machine that strings together sequence of sounds recognized



Emotion Detection

Definitions

- Affect
  - Broad range of feelings
  - Can be either emotions or moods
- Emotion
  - Brief, intense feelings (anger, fear, sadness, etc)
  - Directed at someone or something
- Mood
  - Less intense, not directed at a specific stimulus
  - Lasts longer (hours (4?) or days)

Physiological Measurement of Emotion

- **Biological arousal:** heart rate, respiration, perspiration, temperature, muscle tension
- **Expressions:** facial expression, gesture, posture, voice intonation, breathing noise

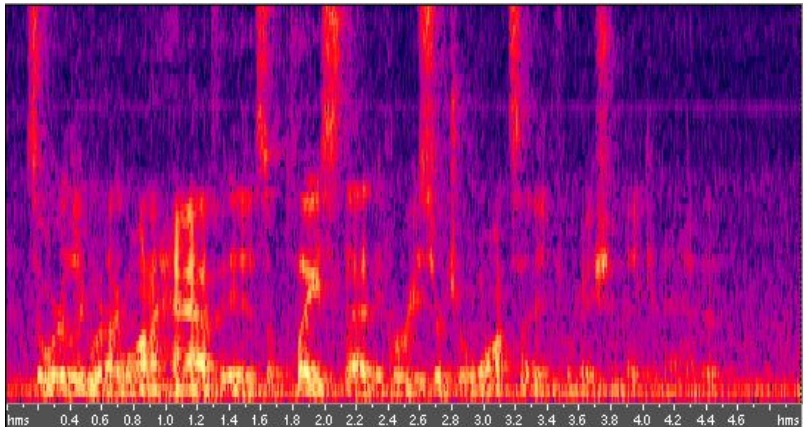
Emotion	Physiological Response
Anger	Increased heart rate, blood vessels bulge, constriction
Fear	Pale, sweaty, clammy palms
Sad	Tears, crying
Disgust	Salivate, drool
Happiness	Tightness in chest, goosebumps

Audio Features for Emotion Detection

- MFCC widely used for analysis of speech content, Automatic Speaker Recognition (ASR)
  - Who is speaking?
- Other audio features exist to capture sound characteristics/dynamics (prosody)
  - Useful in detecting emotion in speech
- **Pitch:** the frequency of a sound wave. E.g.
  - Sudden increase in pitch => Anger
  - Low variance of pitch => Sadness

Audio Features for Emotion Detection

- **Intensity:** Energy of speech, intensity. E.g.
  - Angry speech: sharp rise in energy
  - Sad speech: low intensity
- **Temporal features:**
  - Speech rate, voice activity (e.g. pauses)
  - E.g. Sad speech: slower, more pauses



- **Other emotion features:** Voice quality, spectrogram, statistical measures

Uses of Affect Detection
E.g. Using Voice on Smartphone

- Audio processing (especiallyto detect affect, mental health) can revolutionize healthcare
  - Detection of mental health issues automatically from patientsvoice
  - Population-level (e.g campus wide) mental health screening
  - Continuous, passive stress monitoring
  - Suggest interventions: breathing exercises, play relaxing music
  - Monitoring social interactions, recognize conversations (number and duration per day/week, etc)

Voice Analytics Example: SpeakerSense (Lu et al)

- Identifies speaker, who conversation is with

Voice Analytics Example: StressSense (Lu et al)

- Detected stress in speaker's voice
- Features: MFCC, pitch, speaking rate
- Classification using GMM
- Accuracy: indoors (81%), outdoors (76%)

Voice Analytics Example: Mental Illness Diagnosis

- What if depressed patient lies to psychiatrist, says “I’m doing great”
- Mental health (e.g. depression) detectable from voice, can be used to detect lying patient
- Doctors pay attention to speech aspects when examining patients
- E.g. depressed people have slower responses, more pauses, monotonic responses and poor articulation

Category	Patterns
Rate of speech	slow, rapid
Flow of speech	hesitant, long pauses, stuttering
Intensity of speech	loud, soft
Clarity	clear, slurred
Liveliness	pressured, monotonous, explosive
Quality	verbose, scant

Detection of COVID from Respiratory sounds

- large-scale crowdsourced dataset of respiratory sounds collected to aid diagnosis of COVID-19.
- Coughs and breathing to understand how discernible COVID-19 sounds are from those in asthma or healthy controls.
- Simple binary machine learning classifier is able toclassify correctly healthy and COVID-19 sounds.
- Were able to distinguish
  - User who had COVID-19 + cough vs healthy user with a cough
  - Users who had COVID-19 + cough vs. Users with asthma and a cough.

Playing Audio and Video in Android

MediaPlayer

- Android Classes used to play sound and video
  - **MediaPlayer:** Plays sound and video
  - **AudioManager:** plays only audio

- Any Android app can create instance of/use MediaPlayerAPIs to integrate video/audio playback functionality

- MediaPlayercan fetch, decode and play audio or video from:
  - Audio/video files stored in app's resource folders (e.g. **res/raw/**folder)
  - External URLs (over the Internet)

MediaPlayer

- MediaPlayer supports:
  - **Streaming network protocols:** RTSP, HTTP streaming
  - **Media Formats:**
    - Audio (MP3, AAC, MIDI, etc),
    - Image (JPEG, GIF, PNG, BMP, etc)
    - Video (MPEG-4, H.263, H.264, H.265 AVC, etc)
- 4 major functions of a Media Player
  - **User interface**, user interaction
  - Handle **Transmission errors**: retransmissions, interleaving
  - **Decompress** audio
  - **Eliminate jitter:** Playback buffer (Pre-download 10-15 secs of music)

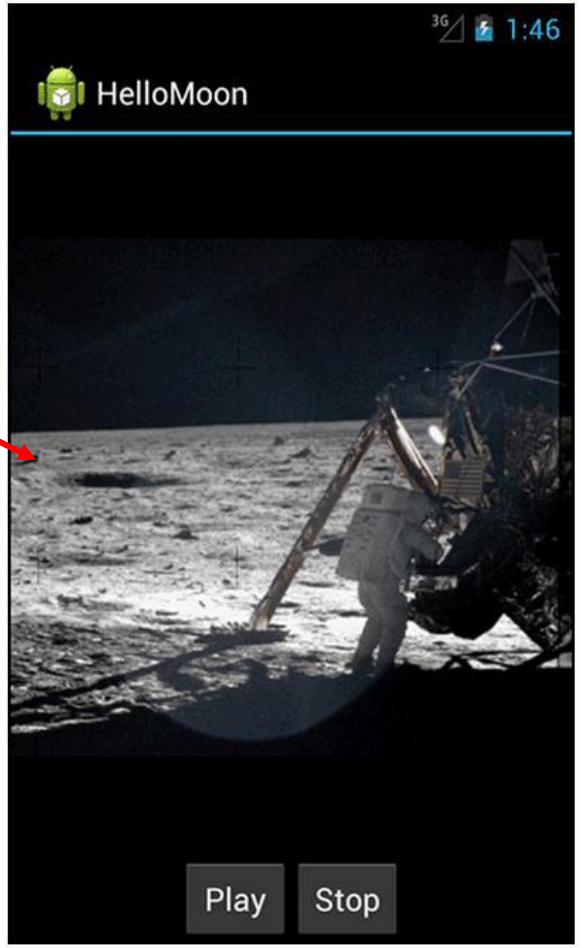
Example: Playing Audio File using MediaPlayer

- Use **MediaPlayer**to play audio file

Resources

- Put image **armstrong_on_moon.jpg** in **res/drawable/**folders
- Place audio file to be played back (**one_small_step.wav**) in **res/raw** folder
- Create **strings.xml** file for app
  - Play, Stop, Image description..

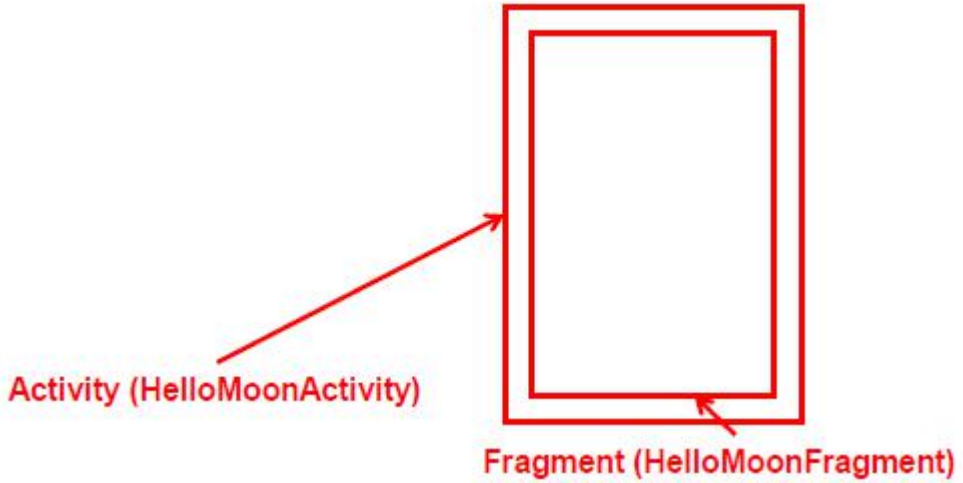
```
<?xml version="1.0" encoding="utf-8"?>
<resources>

   <string name="app_name">HelloMoon</string>
   <string name="hello_world">Hello world!</string>
   <string name="menu_settings">Settings</string>
   <string name="hellomoon_play">Play</string>
   <string name="hellomoon_stop">Stop</string>
   <string name="hellomoon_description">Neil Armstrong stepping
            onto the moon</string>

</resources>
```



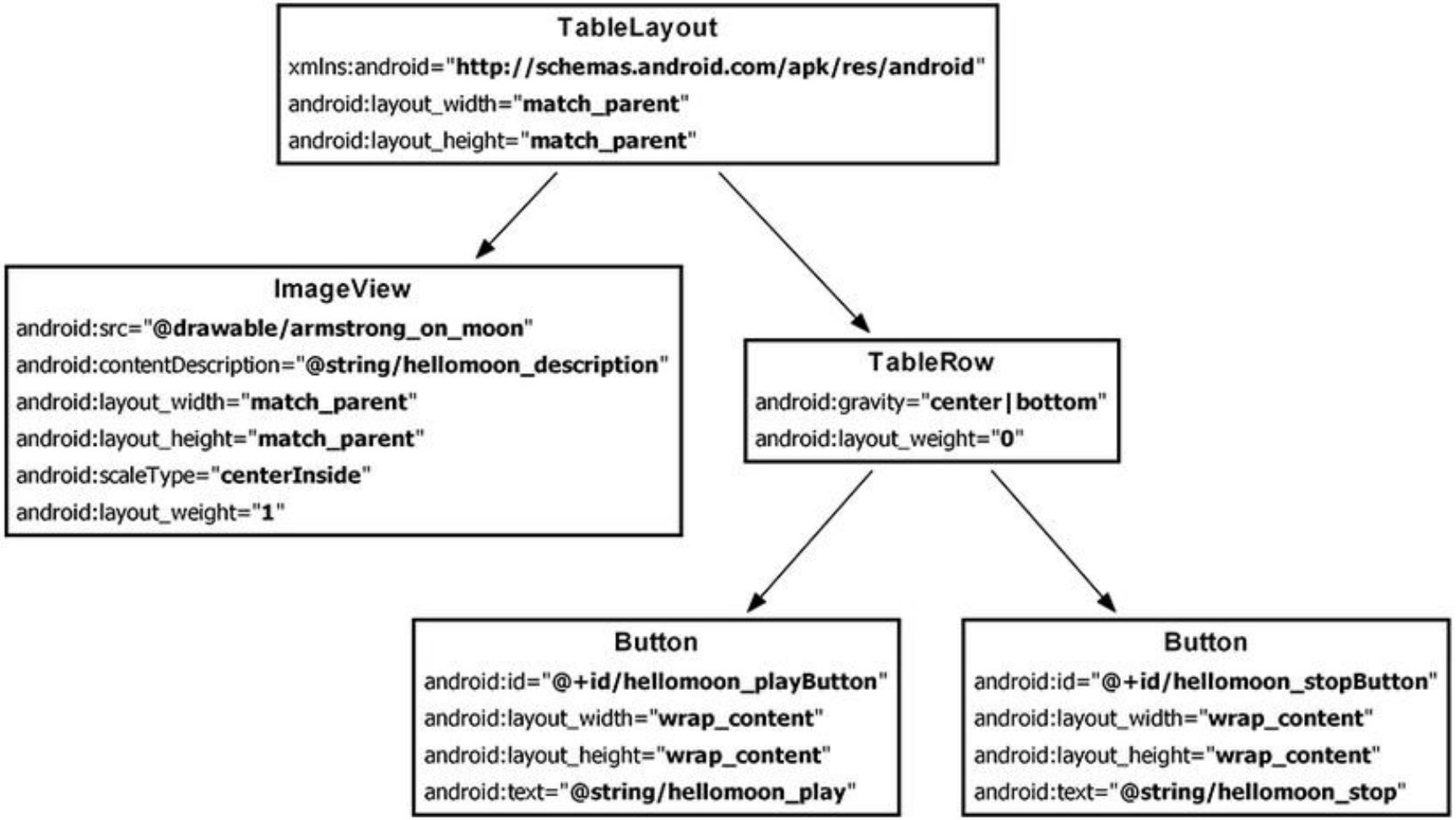
The UI

- 1 activity (**HelloMoonActivity**) that hosts **HelloMoonFragment**
- **AudioPlayer**class will be created to encapsulate **MediaPlayer**

- First set up the rest of the app:
  - Define fragment's XML layout
  - Create fragment java class
  - Modify the activity (java) and its XML layout to host the fragment



Defining the Layout for HelloMoonFragment



Define XML for HelloMoon UI (fragment_hello_moon.xml)

Creating a Layout Fragment

- **Layout fragment:** Add fragments to hosting Activity's XML file
- Create activity's XML layout (**activity_hello_moon.xml**)
- **Activity's** XML layout file contains/hosts fragment

```
<?xml version="1.0" encoding="utf-8"?>
<fragment xmlns:android="http://schemas.android.com/apk/res/android"
  android:id="@+id/helloMoonFragment"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:name="com.bignerdranch.android.hellomoon.HelloMoonFragment">

</fragment>
```

Using Media Player:

Step 1: Request Permission in AndroidManifest or Place video/audio files in res/raw

- If streaming video/audio over Internet (network-based content), request network access permission in AndroidManifest.xml:

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

Set up HelloMoonFragment.java

```
public class HelloMoonFragment extends Fragment {

    private Button mPlayButton;
    private Button mStopButton;

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup parent,
            Bundle savedInstanceState) {
        View v = inflater.inflate(R.layout.fragment_hello_moon, parent, false);

        mPlayButton = (Button)v.findViewById(R.id.hellomoon_playButton);
        mStopButton = (Button)v.findViewById(R.id.hellomoon_stopButton);

        return v;
    }
}
```

Get handle to Start, Stop buttons

- If playing back local file stored on user's smartphone, put video/audio files in **res/raw** folder

Step 2: Create MediaPlayer Object, Start Player

- To play audio file saved in app's **res/raw/** directory
- **Note:** Audio file opened by create (e.g. one_small_step.mp3) must be encoded in one of supported media formats

Create AudioPlayer Class encapsulates MediaPlayer

```
public class AudioPlayer {

    private MediaPlayer mPlayer;

    public void stop() {
        if (mPlayer != null) {
            mPlayer.release();
            mPlayer = null;
        }
    }

    public void play(Context c) {
        mPlayer = MediaPlayer.create(c, R.raw.one_small_step);
        mPlayer.start();
    }
}
```

- **Releasing the MediaPlayer**

- MediaPlayer can consume valuable system resources
- When done, call **release( )** to free up system resources
- In **onStop( )** or **onDestroy( )** methods, call

```
mediaPlayer.release();
mediaPlayer = null;
```

- **MediaPlayer in a Service:** Can play media (e.g. music) in background while app is not running
  - Start MediaPlayer as service

Hook up Play and Stop Buttons

```
public class HelloMoonFragment extends Fragment {
    private AudioPlayer mPlayer = new AudioPlayer();
    private Button mPlayButton;
    private Button mStopButton;

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup parent,
            Bundle savedInstanceState) {
        View v = inflater.inflate(R.layout.fragment_hello_moon, parent, false);

        mPlayButton = (Button)v.findViewById(R.id.hellomoon_playButton);
        mPlayButton.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                mPlayer.play(getActivity());
            }
        });

        mStopButton = (Button)v.findViewById(R.id.hellomoon_stopButton);
        mStopButton.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                mPlayer.stop();
            }
        });
        return v;
    }
```

Extra: stream audio from internet

- To play audio from remote URL via HTTP streaming over the Internet

```
String url = "http://........"; // your URL here
MediaPlayer mediaPlayer = new MediaPlayer();
mediaPlayer.setAudioStreamType(AudioManager.STREAM_MUSIC);
mediaPlayer.setDataSource(url);
mediaPlayer.prepare(); // might take long! (for buffering, etc)
mediaPlayer.start();
```

Multimedia Networking: Basic Concepts

Multimedia networking: 3 application types

- Multimedia refers to audio and video. 3 types

*1.streaming, stored*audio, video
  - *streaming:* transmit in batches, begin playout before downloading entire file
  - e.g., YouTube, Netflix, Hulu
  - Streaming Protocol used (e.g. Real Time Streaming Protocol (RTSP), HTTP streaming protocol (DASH))

2.streaming live audio, video
  - e.g., live sporting event

3.conversational voice/video over IP
  - Requires minimal delays due to interactive nature of human conversations
  - e.g., Skype, RTP/SIP protocols

Live Streaming

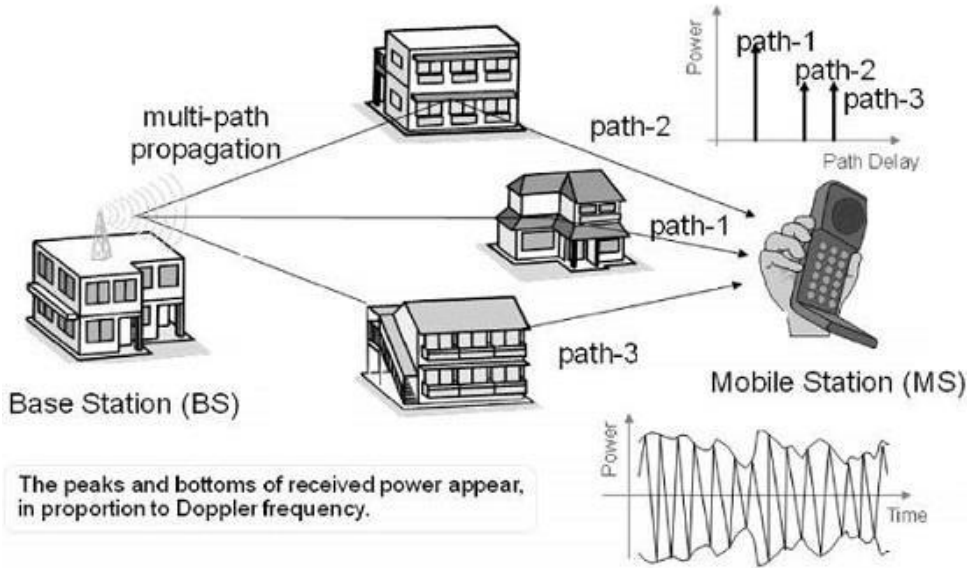
- Live streaming extremely popular now (E.g. going Live on Facebook)
- A person can share their experiences with friends
- Popular live streaming apps include Facebook, Periscope
- Also possible on devices such as Go Pro
- Uses RTMP (real time protocol by Adobe), or other 3 rd party APIs

Facebook Live

Live GoPro

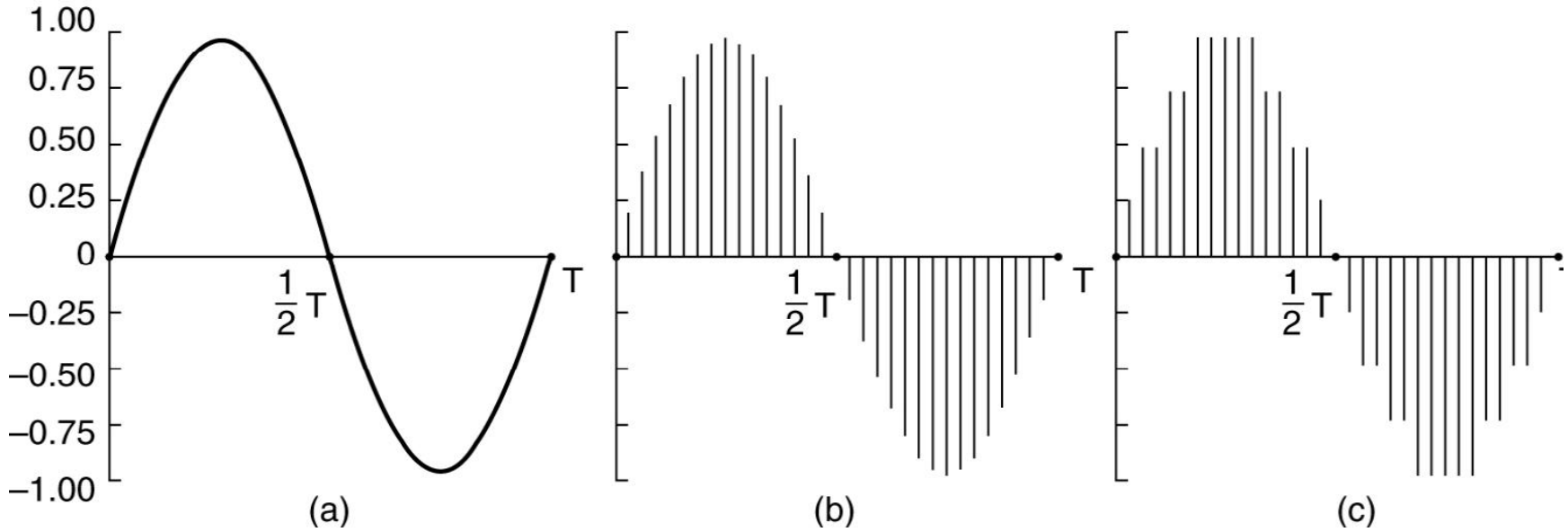
Live Streaming Bandwidth Issues

- On WiFi, bandwidth is adequate, high quality video possible
- Cellular links:
  - Low bandwidth,
  - Variable bandwidth (multi-path fading)
    - Even when standing still
  - Optimized for download not upload
- Video quality increasing faster than cellular bandwidths
  - Ultra HD, 4k cameras makes it worse, now available on many smartphones



Digital Audio

- Sender converts audio from analog waveform to digital signal
- E.g PCM uses 8-bit samples 8000 times per sec
- Receiver converts digital signal back into audio waveform



Audio Compression

- Audio CDs:
  - 44,100 samples/second
  - Uncompressed audio, requires 1.4Mbps to transmit real-time
- Audio compression reduces transmission bandwidth required
  - E.g. MP3 (MPEG audio layer 3) compresses audio down to 96 kbps

Video Encoding

- **Digital image:** array of <R,G,B> pixels
- **Video:** sequence of images
- **Redundancy:** Consecutive frames mostly same (1/30 secs apart)
- **Video coding (e.g. MPEG):**use redundancy *within*and *between*images to decrease # bits used to encode video
  - **Spatial**(within image)
  - **Temporal**(from 1 image to next)

spatial coding example: instead of sending *N* values of same color (all purple), send only two values: color value (*purple*) *and number of times repeated* (N)

frame *i*

temporal coding example: instead of sending complete frame at i+1, send only differences from frame i

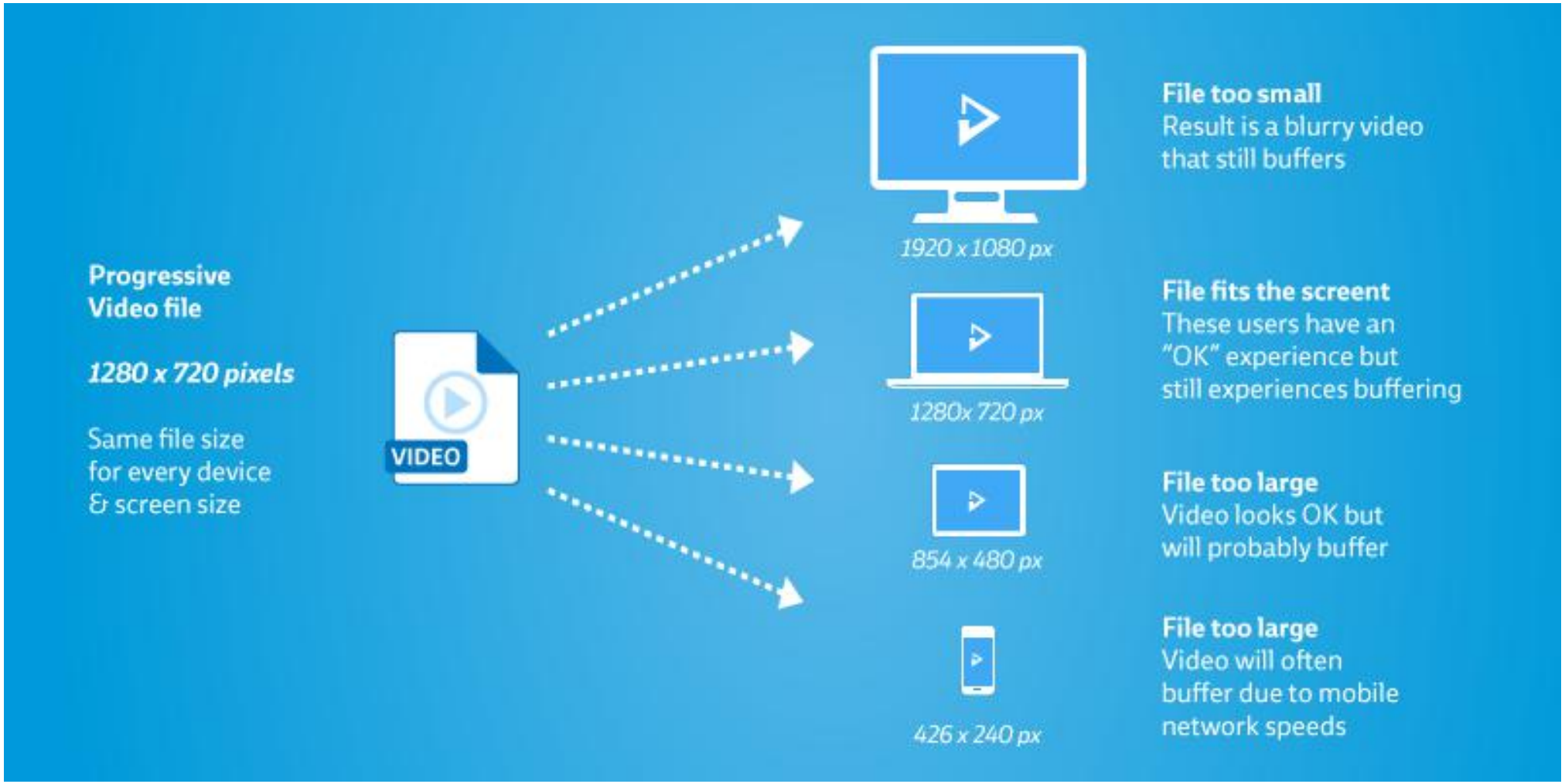
frame *i+1*

Adaptive Video Streaming

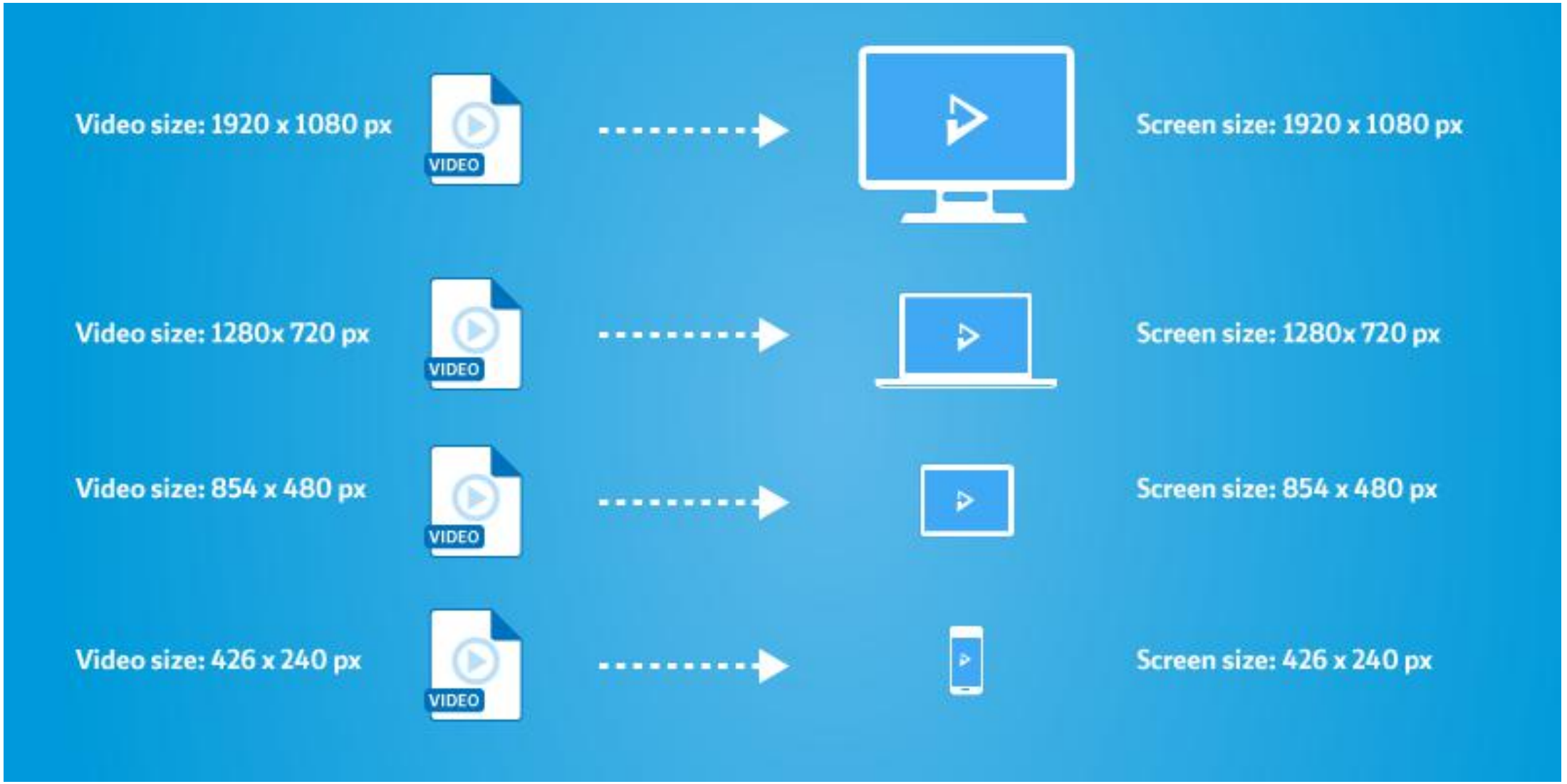
Adaptive Video Streaming

- E.g., Dynamic Adaptive Streaming over HTTP (DASH) in Youtube
- Motivation: one size does not fit all

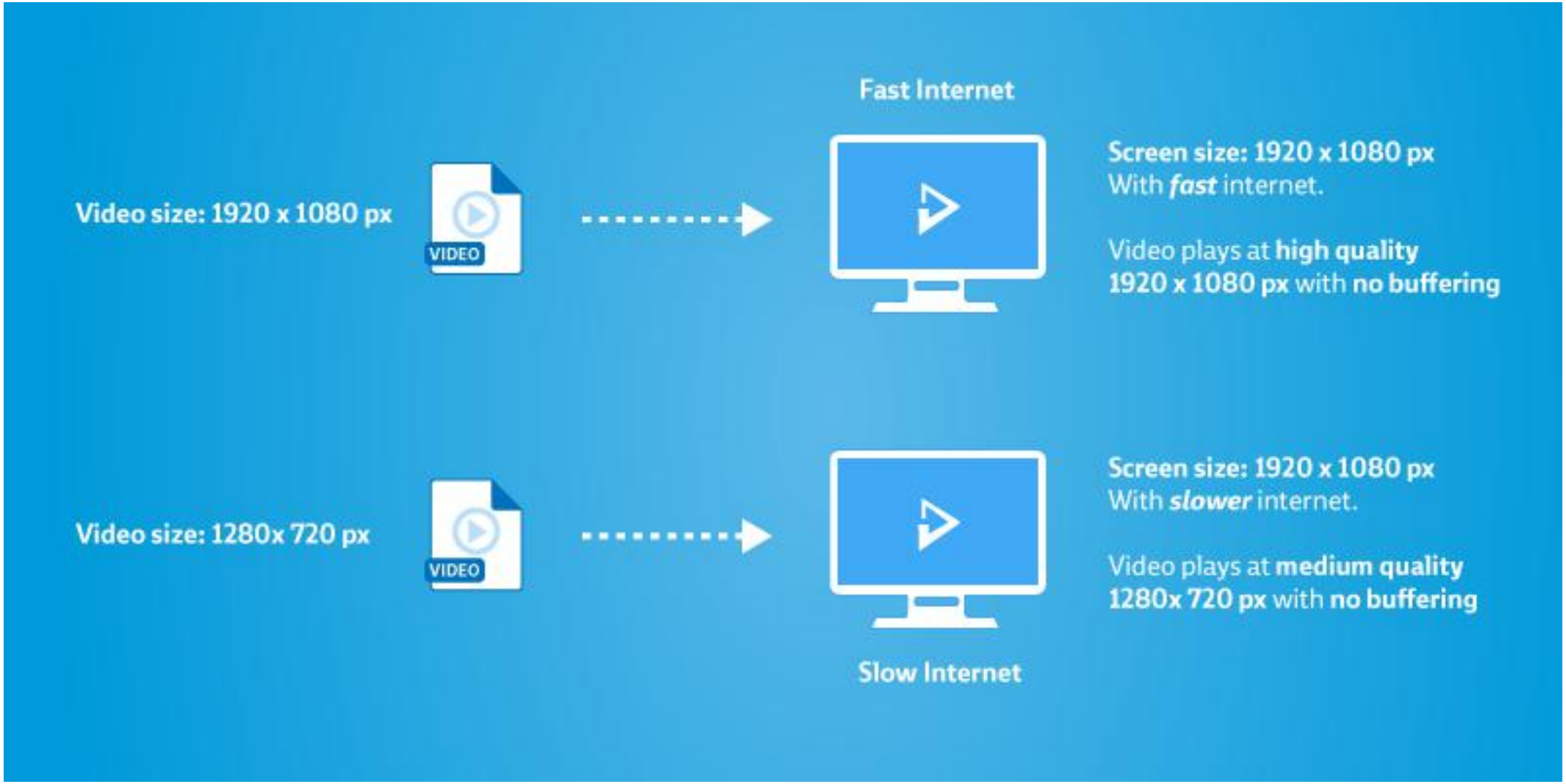
- Baseline: a progressive video stream is simply one single video file being streamed over the internet, then stretch to different screens.



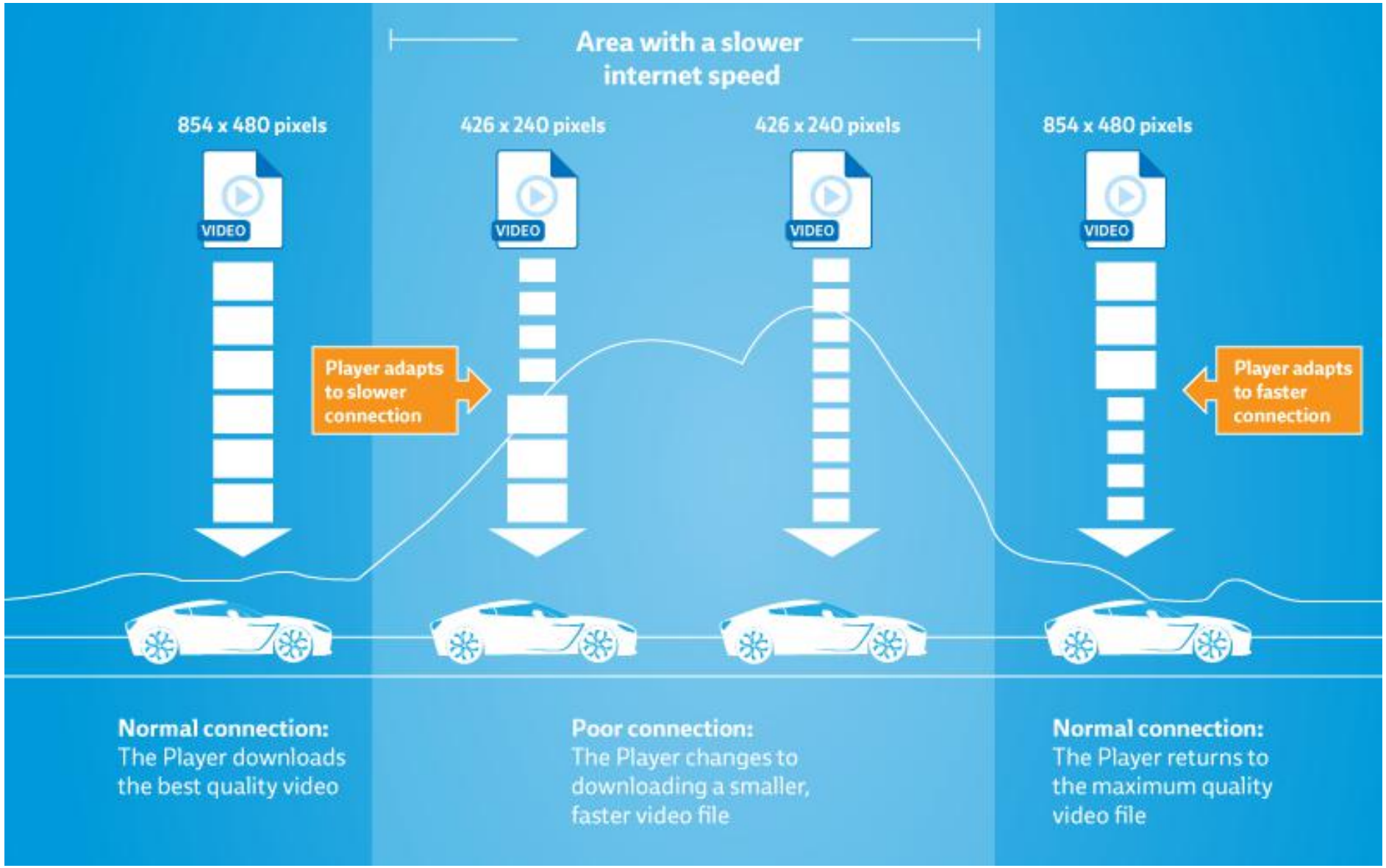
- Adaptive streaming: allows the video provider to create a different video for each of the screen sizes



- Adjust the video size based on network connection quality



- The biggest strength: adaptive bitrate



Security, Privacy, and Ethics in Mobile Development

CSE 162 – Mobile Computing

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

Ethical Issues in Mobile Development

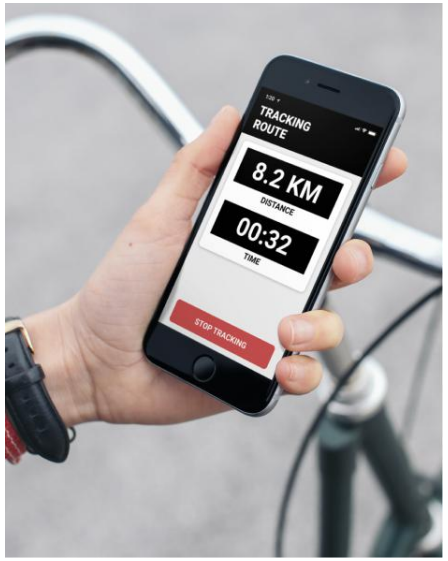
What sensitive information can be collected by mobile systems?

- Video
- Sounds
- GPS
- Locations
- User’s self reported information
- Many more!

- All with one unified API, and cheap

Examples of Legitimate Apps

- Assess pedestrian or bike friendliness of neighborhoods.
- Use the location awareness to understand the user's exposure to air pollution as they move around.
- Use phones to snap, tag and upload photos of community events



Ethical Issues in Mobile Sensing

- Privacy: control over personal data
- Consent: Informed permission
- Equity: fairness in how individuals are treated
- Social forgetting: purposeful discarding of information about individuals in order to enable forgiveness, recovery, or a clean slate.

Privacy

- Mobile systems can gather significant amounts of data about the users.
- However, the data can be subpoenaed, or be demanded by U.S. authorities without warrant
- Unauthorized sharing or data theft can occur at a variety of places
- Complicated end-user licensing agreements may lead users to give away broad rights to share their data in return for services.

Recap: Privacy Preserving in Crowdsensing

- Approaches to protect the data
- Examples:
  - **Anonymization**
  - **Encryption**
  - **Obfuscation**: Adding distracting or misleading data to a log or profile.

Considerations: Privacy vs Functionality

- Commerce can suffer from strong privacy rights, as there is less information for both producers and consumers in the marketplace
  - Example electronic payments, and ant-money-laundering

- Truthfulness, openness, and accountability can suffer at the hands of strict privacy protections
  - Example: new algorithms for mobile sensing that allow users to replace sensitive location data with believable but fake data

Consent

- Consent is a value central to data policies in the United States. A critical component of respect, beneficence and justice is informed consent.

Challenges in Consent

- Consent in mobile apps is complicated by relying on ubiquitous devices.
  - Opting out of the mobile phone network is not a realistic option.
  - In 2011, Apple and Android were storing location data over and beyond what users were notified of and consented to
- Financial interest can conflict with consents
  - Use data to produce targeted advertising, sell valuable behavioral data to third parties, or use location to hone price or product discrimination
- Secondary, unforeseen purposes of data use
  - motion data can infer Parkinson’s disease

Soft Surveillance

- A technique used by agents of power, such as governments, to collect seemingly voluntary but actually mandatary data.
  - Example: searches to enter planes
  - Example: Withheld Social Security benefits if people do not “voluntarily” submit personal information
- Mobile sensing systems can easily become soft surveillance systems
  - Everywhere, all time presence
  - People can be involved by simply agreeing to data collection

Equity

- Fairness and justice in how individuals are treated
- If powerful institutions gather data from relatively less powerful individuals, mobile sensing could tilt towards control and increased surveillance
- Alternatively, distributed sensing and analysis could shape technologies of care or even empowerment.
- Besides, the availability of mobile phones enables systematic data collection with radically lower cost, which enables data-driven decision-making to small institutions and community groups

Social Forgetting

- Purposeful discarding of information to enable forgiveness and a clean slate

- Mobile sensing can create a record of people's movements, habits, and routines that persists
  - A subject of both celebration and concern

Pros and Cons of persistent records

- Pros
  - Augment human memory. Can improve healthcare, and enable memory bank to relive past events.
- Cons
  - Unintended loss of fresh start
  - Increased surveillance

Solution

- The “right to be forgotten”.
  - A combination of policies and technologies that allow for the gradual decay of digital data.

More info

- EU General Data Protection Regulation
- California Consumer Privacy Act

Privacy Policy

Creating a Privacy Policy

- A privacy policy is a document created to go with a product (app, website, etc.) that describes how the product and company behind it will do the following with a customer or client's data:
  - Gather
  - Use
  - Disclose
  - Manage

Creating a Privacy Policy

- Ask yourself some questions:
  - What data is collected?
  - How it is collected?
  - What you will/can do with it?
  - What will happen to it after X amount of time?
  - Is it anonymous?
  - Are there ads?
  - Is the data shared with another organization?
  - … and more…

You need a privacy policy because...

- You are collecting personal data
- You are using a third-party service
- Government regulations
- App Store regulations
- Risk alienating customers
- Open to lawsuits

What's in a policy?

- **Information** - what personal information is being collected on the site
- **Choice** - what options the customer has about how/whether her data is collected and used
- **Access** - how a customer can see what data has been collected and change/correct it if necessary

What’s in a policy?

- **Security** - state how any data that is collected is stored/protected
- **Redress** - what customer can do if privacy policy is not met
- **Updates** - how policy changes will be communicated

Example Policies

- Google: https://www.google.com/policies/privacy/
- Apple: http://www.apple.com/legal/privacy/en-ww/
- Facebook: https://www.facebook.com/policy.php
- Twitter: https://twitter.com/privacy?lang=en

Example Policies

- Note that these are mainly in “regular, plain English!”
- Movement away from “legalese”
- Some privacy policies were automatically processed

What does a privacy policy get you?

- Disclosure of what’s going on
- A level of trust with developer
- Meeting requirements from publishers / government agencies

Example

- Google Analytics is one of the most popular digital analytics software.
  - Allows you to analyze details about the visitors on your website and design strategy to improve business
- If you've enabled any Google Analytics Advertising features, you are required to notify your users:
  - What features you've implemented.
  - How you and third-party vendors use first-party
  - How visitors can opt-out of the Google Analytics Advertising

https://support.google.com/analytics/answer/2700409?hl=en

Beyond Policies

- Writing down what you do is good…
- … following it is even better
- Remember: privacy is not security
- The privacy policy says what you are collecting and what you plan to do
- And absence of this does not mean you shouldn’t protect data you collect!

Wearables (1)

CSE 162 – Mobile Computing

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

In this lecture

- What is wearable computing
- Wearable sensing technologies
- Challenges in wearable computing

- The term “wearable computing” is really built around any device that is attached or worn in some way

- So, the possibilities here are vast
  - Smartwatch
  - Continuous health monitoring
  - Brain-computer interface
  - More

Everyday Realistic Wearable Technology

- What most people think about is watches and/or bracelets
- Smart Watches
  - Apple Watch - https://www.apple.com/watch/
  - Android Wear - https://www.android.com/wear/
  - Some proprietary watches
- Personal Fitness Devices
  - Fitbit - https://www.fitbit.com/home
  - Other devices

What is a wearable computer?

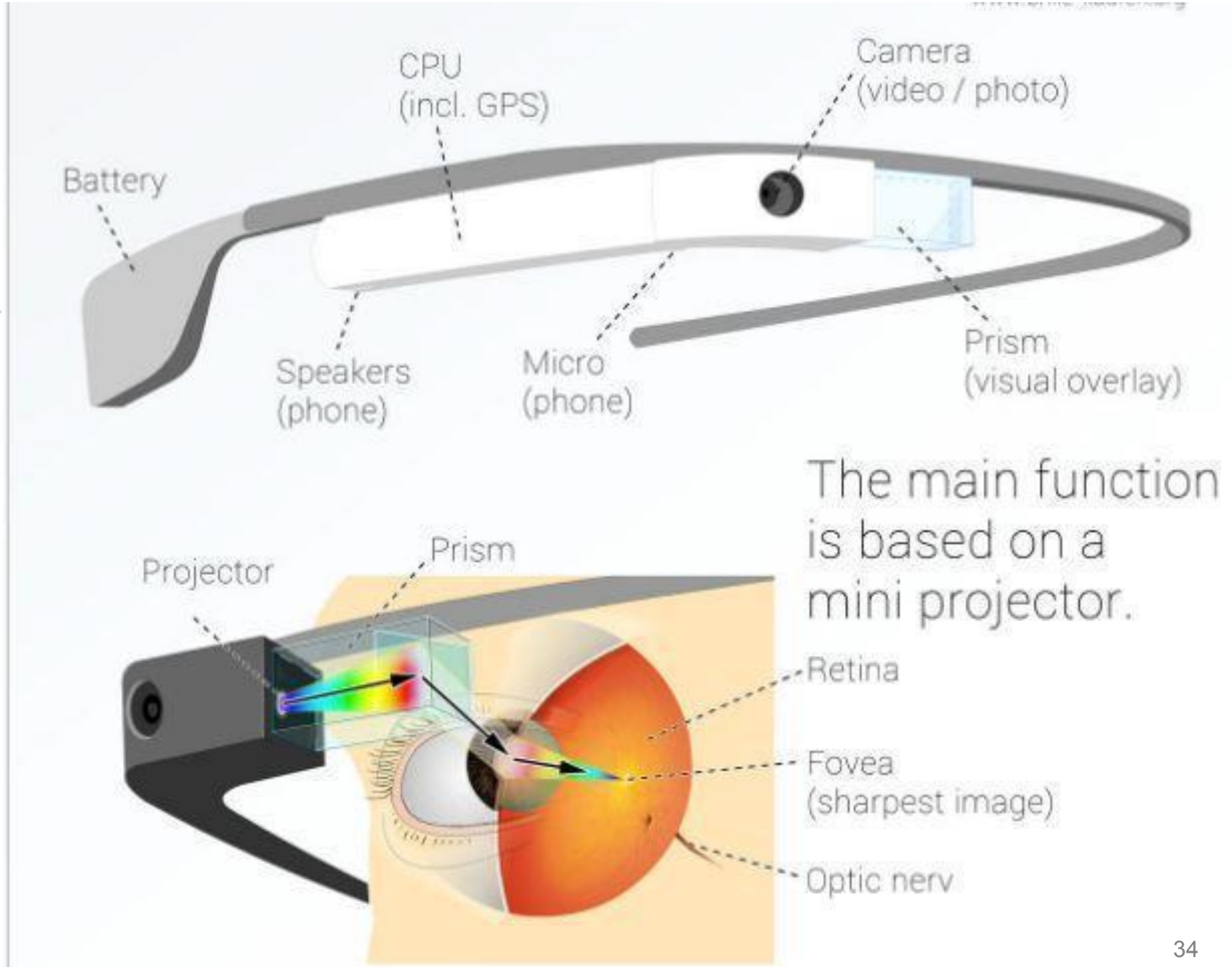
- A computer that is
  - Portable while operational
  - Enables hands-free/hands-limited use
  - Able to get the user's attention
  - Is always on, acting on behalf of the user
  - Able to sense the user's current context

Rhodes, B. J. (1997). The wearable remembrance agent: A system for augmented memory. Personal Technologies, 1(4), 218-224.

Wearable Devices

Smart glasses

- Head Mounted Display (HMD)
- Head Up Display (HUD)



View Through Google Glass

- Always available peripheral information display
  - Combining computing, communications and content capture

Smart watches

Foundation
Mechanical wrist watches established

1880

Imagination
Couldn't a watch do much more?

1940

Electronification
Quartz watches revolutionize the industry

1970

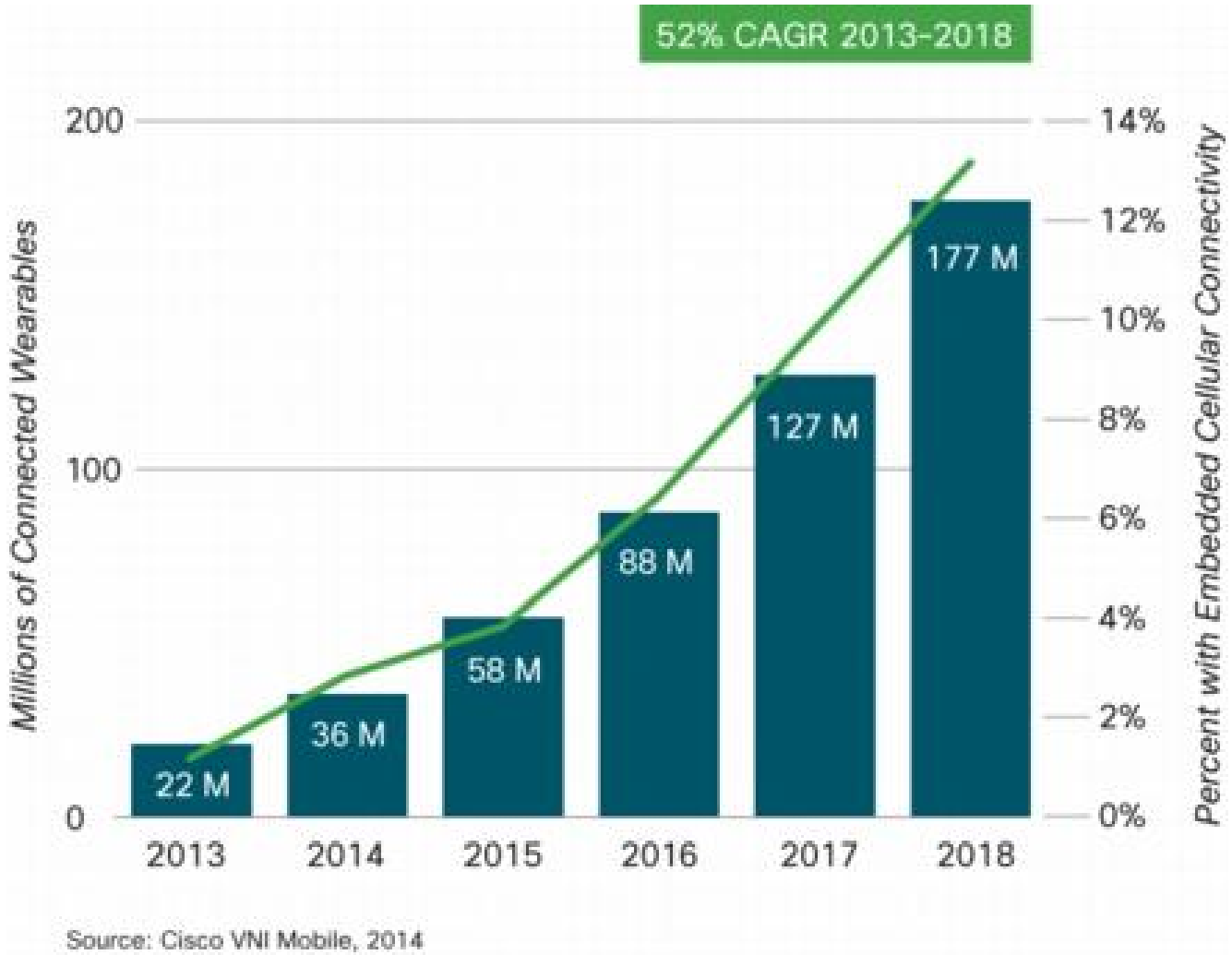
Experimentation
1st smartwatches, but no commercial success

1990

Commercialization
Smartwatches start to sell

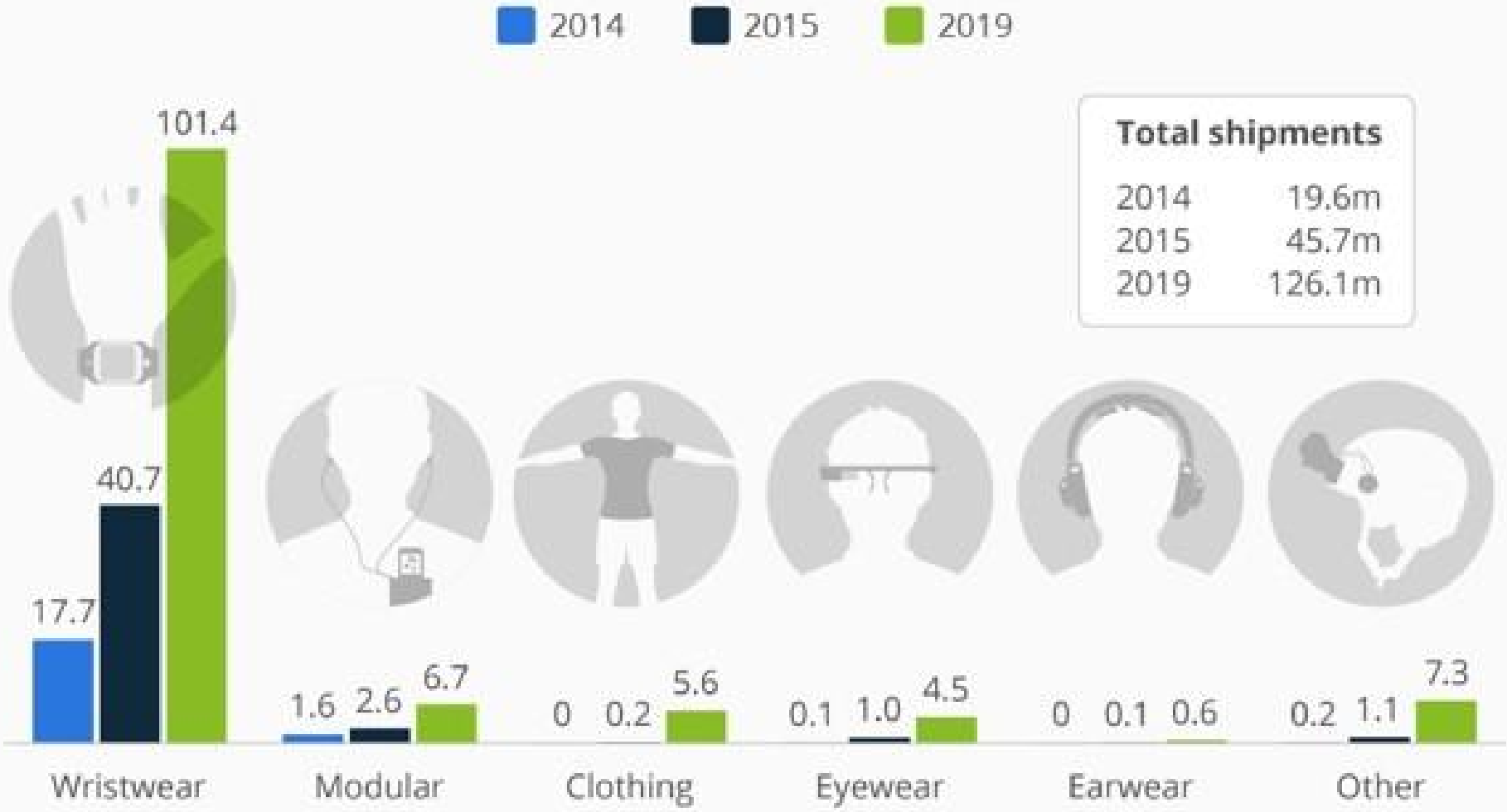
2010

Number of Smartwatches Shipped



The Predicted Wearables Boom Is All About The Wrist

Worldwide wearable device shipment forecast (in million units)



@StatistaCharts Source: IDC

statista

Smart earplugs

- An emerging computing platform
- New features
  - Augmented acoustic reality
  - Real-time translation
  - Monitor biometrics
  - Fitness coaching
  - Biometric identification

Intra-oral sensing

- “Sensor-Embedded Teeth for Oral Activity”

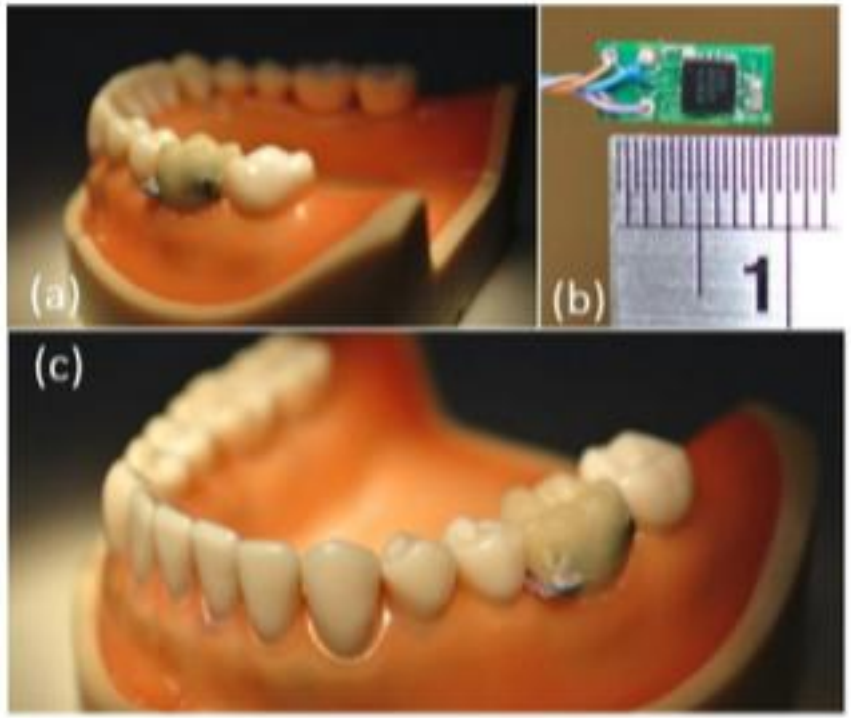


Figure 1. The breakout board with (b) tri-axial accelerometer and (a)(c) sensor embedded denture.

Why discuss this in mobile?

- They are, by definition, mobile devices
- They often run versions of the operating systems we are already working with (Android Wear ⇔ Android)

- The wearable devices often work best when paired with a phone or other mobile device
- Many wearable apps come as part of a pair with a phone version

Ideal Attributes

- Persist and provide constant access to information services.
  - Everyday and continuous use.
  - Wearable can interact with the user at any given time.
  - The user can access the wearable quickly and with little effort.

- Sense and model context.
  - The wearable can observe and model the user's environment, physical and mental state.

- Adapt interaction modalities based on the user's context.
  - The wearable should adapt its input and output modalities automatically to those that are most appropriate and socially graceful at the time.

Why use wearable computers?

- Some people wear too many computers.
  - PDA, cellular phone, pager, laptop, electronic translator, and a calculator.
  - Mp3 player, audio digitizers, digital camera.
- These devices all contain very similar components.
  - Microprocessor, memory, screen, keyboard, battery, and in some cases, a wireless modem.
  - The main distinctions between these devices are the interface and the application software.
- Wearable computers could exploit the commonality in components to eliminate cost, weight and redundancy.

Mediate interactions

- Wearable computers will help provide a consistent interface to computationally augmented objects in the physical world.
  - Example—Gesture Pendant.
  - One gesture could provide an intuitive command for many devices.

Aid communication

- The wearable can also assist in human-to-human communication.
- Wearable computers can also help manage interruption in the user’s daily life.

Augment reality

- Augmented reality overlays information-rich virtual realities onto the physical world.
- In a sense, augmented reality is a combination of the application domains described previously.

Wearable App Development

Android Wear

- Released in 2014
- Android Wear devices were initially designed to be second screens, but can be used independently in more situations
- Like Android itself, Wear can be adapted to many different devices
- Uses the same UI theme that the rest of Android uses

Use Cases

- Telling time
- OK Google
- Notifications and quick replies
- Phone app control (like music, for example)
- Fitness
- Basic functions (alarms, stopwatch, etc.)
- Mobile Payment

When do you want a watch app?

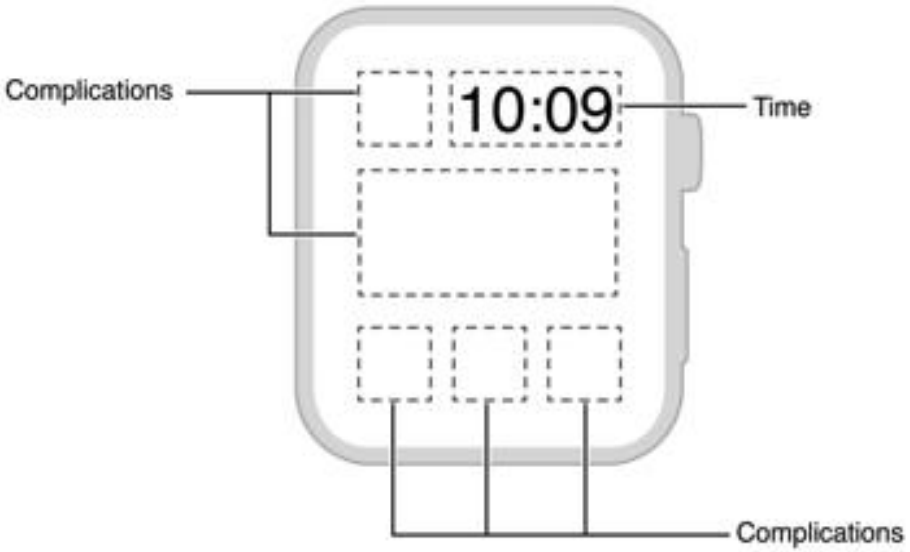
- Good Watch Apps
  - Buy Me a Pie (grocery list)
  - Seven Minute Workout (works with sensors)
  - Due (reminders)
  - Google Maps (directions on your wrist!)
  - Pandora (control your music)
  - Shazaam (what is that song?)
  - Weather Underground (quick weather forecast)

When do you want a watch app?

- Odd Watch Apps
  - Chipotle?
  - FlightRadar?
  - Fandango?
  - AAA?
  - Amazon?

Questions to Consider

- UI concerns regarding touch size and screen size become even more problematic...



Alert

SideBySideButtonsAlert

ActionSheet

Questions to consider

- Would a watch app add anything to my full app?
  - What additional sensing can the watch provide?
  - Can the information be shown in a very small format?
  - Are there simple controls to the app that could be added to a watch?
- What type of interaction do you want the user to have?

Energy Efficiency in Mobile Computing

Heat dissipation in wearables

- Close proximity of the wearables to the human body
  - Airflow is low, limiting cooling
  - Strict limitation on temperature
- Careful design of software necessary to help avoid many heat generation crises.

United States
CONSUMER
PRODUCT SAFETY
COMMISSION

MENU

Fitbit Recalls Ionic Smartwatches Due to Burn Hazard; One Million Sold in the U.S.

Share:

- 115 reports of the watch's battery overheating in the US, and 59 internationally.
- There were 78 reports of burn injuries in the US, including two reports of third-degree burns, and four reports of second-degree burns.
- 40 burns were reported internationally.

Problem: Battery Power is Scarce

- Battery energy is most constraining resource on mobile device
- Most resources (CPU, RAM, WiFi speed, etc) increasing exponentially *except* battery energy (ref. Starner, IEEE Pervasive Computing, Dec 2003)

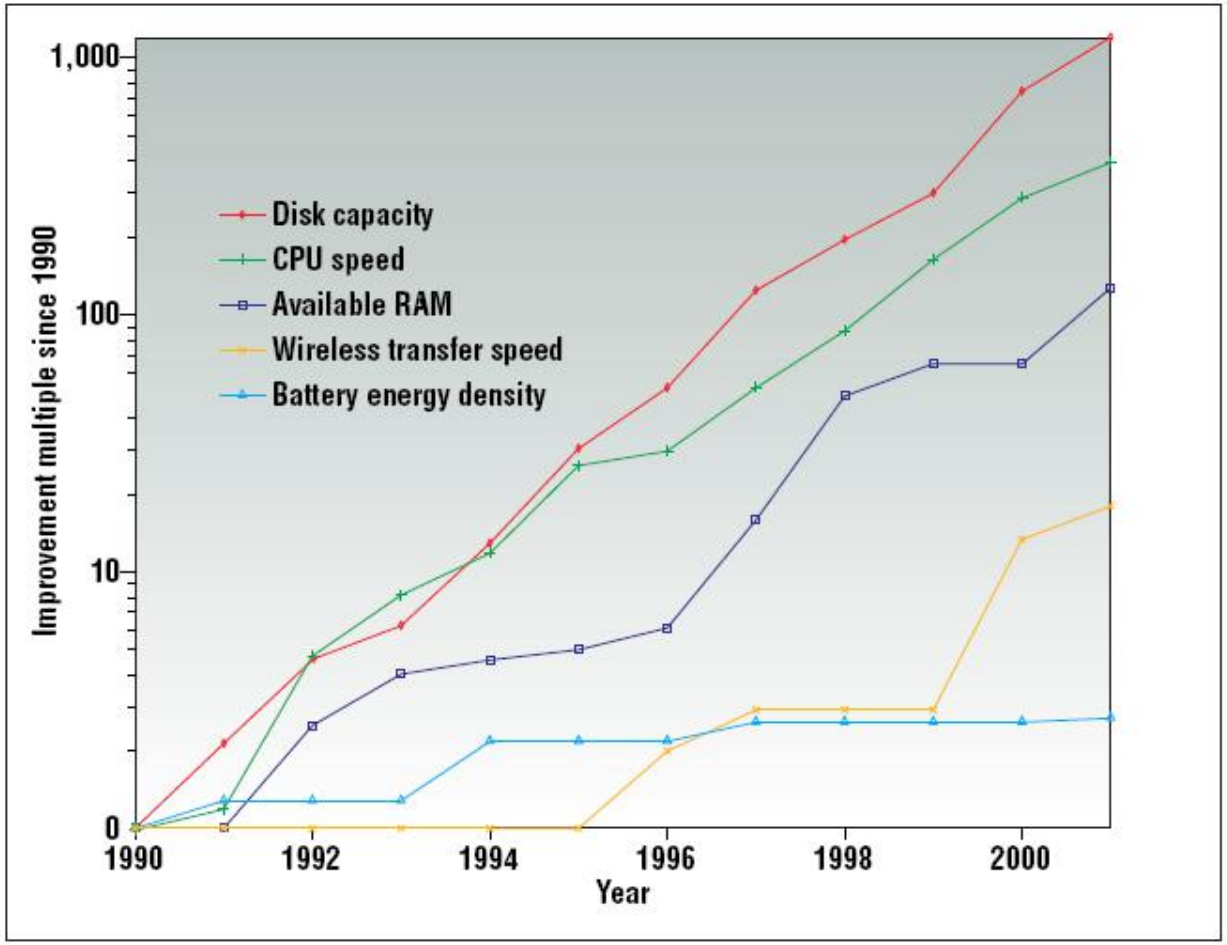


Figure 1. Improvements in laptop technology from 1990–2001.

Ideal wearables (Recap)

- Persist and provide constant access to information services.
- Sense and model context.
- Adapt interaction modalities based on the user's context.

Ideal wearables: challenges

- Persist and provide constant access to information services.
  - Challenge: supply the energy to support all-time access
- Sense and model context.
  - Challenge: power enough sensors
- Adapt interaction modalities based on the user's context.
  - Challenge: power sufficient computing capacity for context learning

Summary: battery capacity continues to bottleneck the ideal wearables

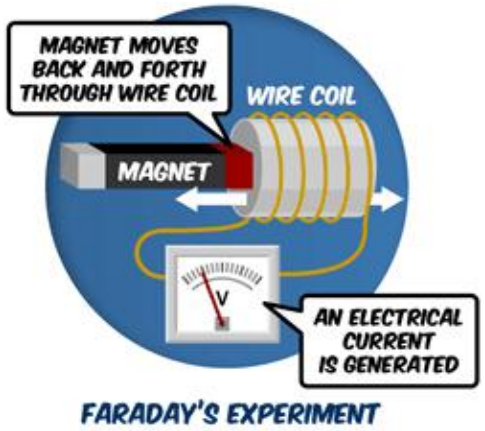
Energy Harvesting for Wearable Systems

Energy Harvesting

- What is it?

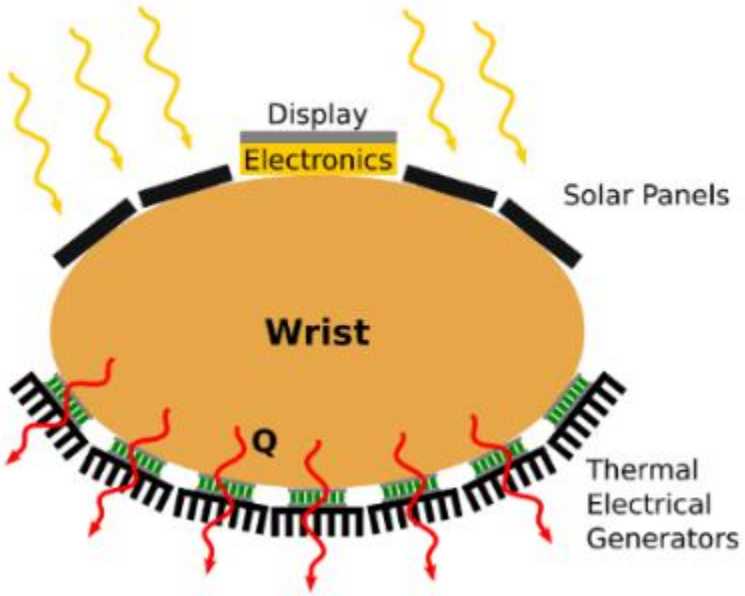
[3]

- Macro scale- wind and solar
- Micro scale- electromagnetic and piezoelectricity



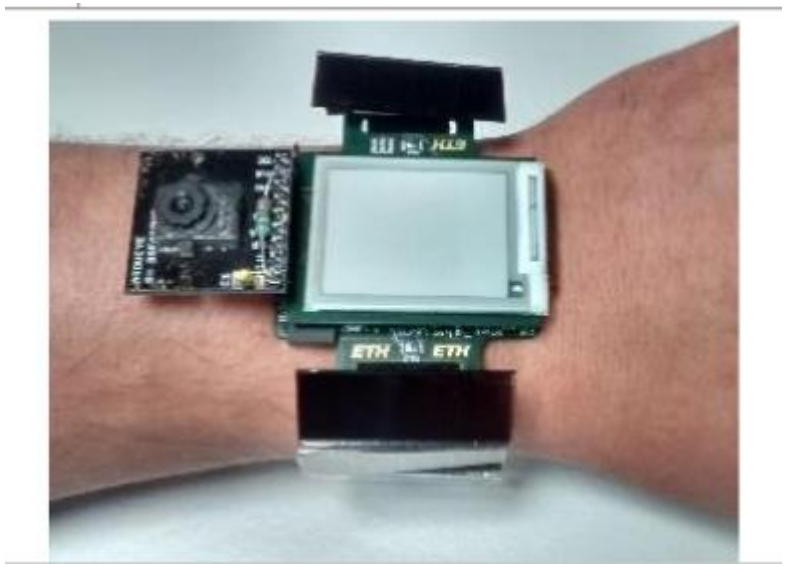
Thermal Electrical Generator

Havest body heat and convert it to electricity



Texas Instruments Innovation Challenge: Europe Design Contest 2015

Self-Sustainable Smart Wearable Device with Energy Harvesting for Context Recognition Applications



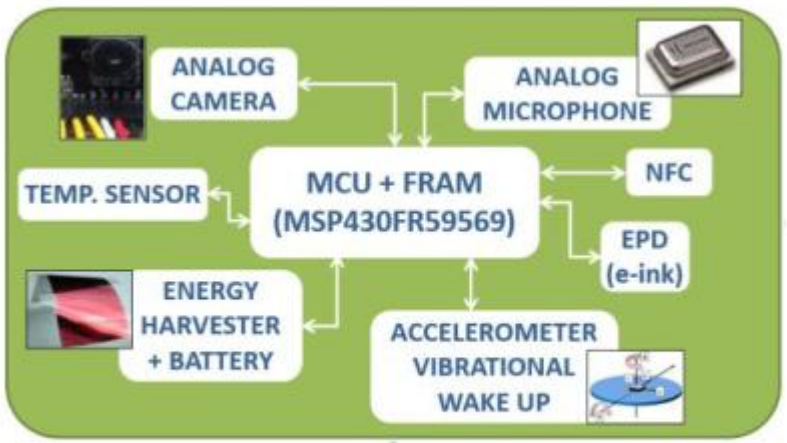
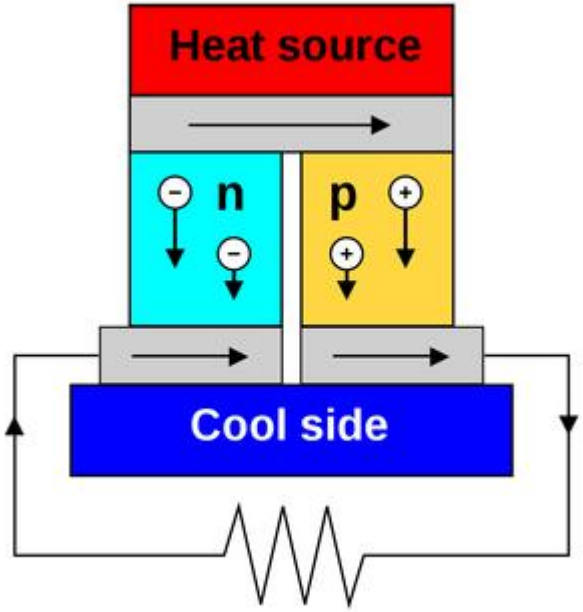


Fig. 2: Block diagram of the smartwatch

- A thin conductive material takes advantage of the temperature difference between its two sides to produce electricity.
- This is known as the Seebeck effect.



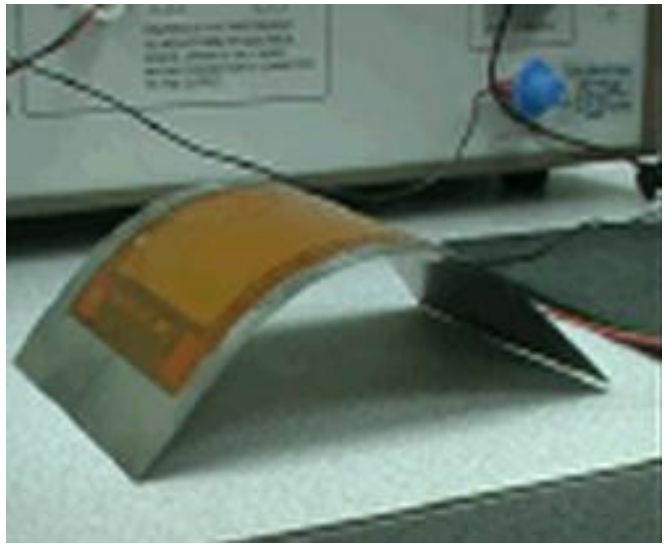
- If a thermoelectric device is located on skin, it will produce power as long as the ambient air is at a temperature that is lower than the skin.
- A device that is one square centimeter in area can yield up to 30 microwatts.
- If these generators are located side by side, the amount of power being harvested is increased.

Piezoelectricity

- Deform to get voltage

[4]

Apply voltage to deform



[5]

Piezoelectricity

- What is it?
- Occurs in polycrystalline materials
  - (For example: Quartz, lead-zirconate-titanate, Rochelle salt, etc.)

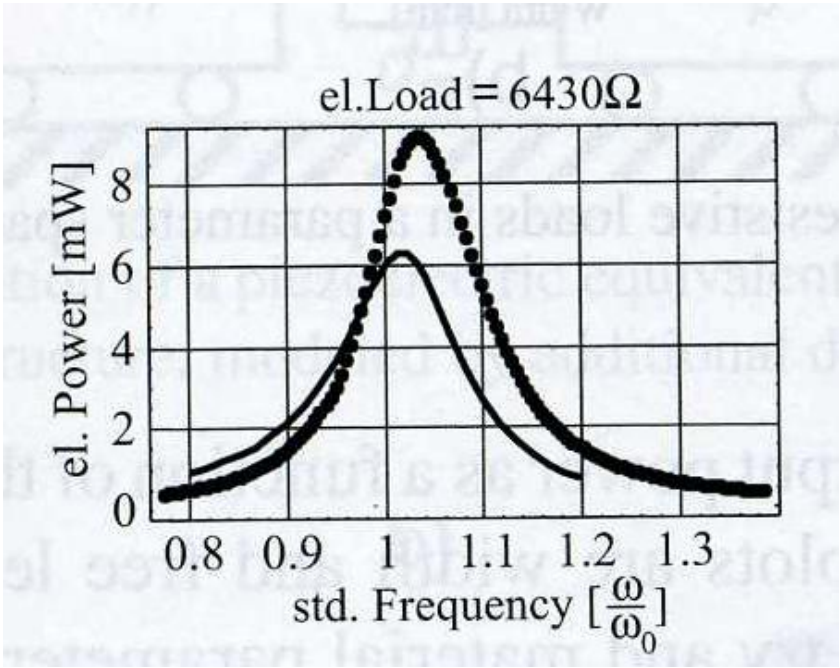
[4]

Piezoelectricity- Output

Output always different

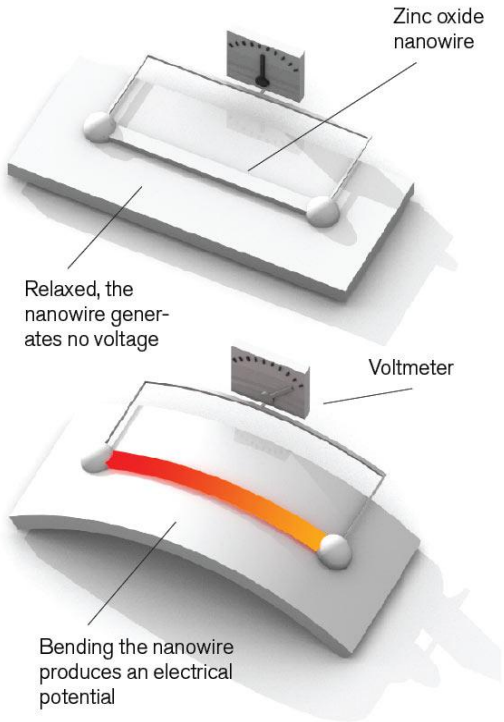
High voltage, low current

Sample output:

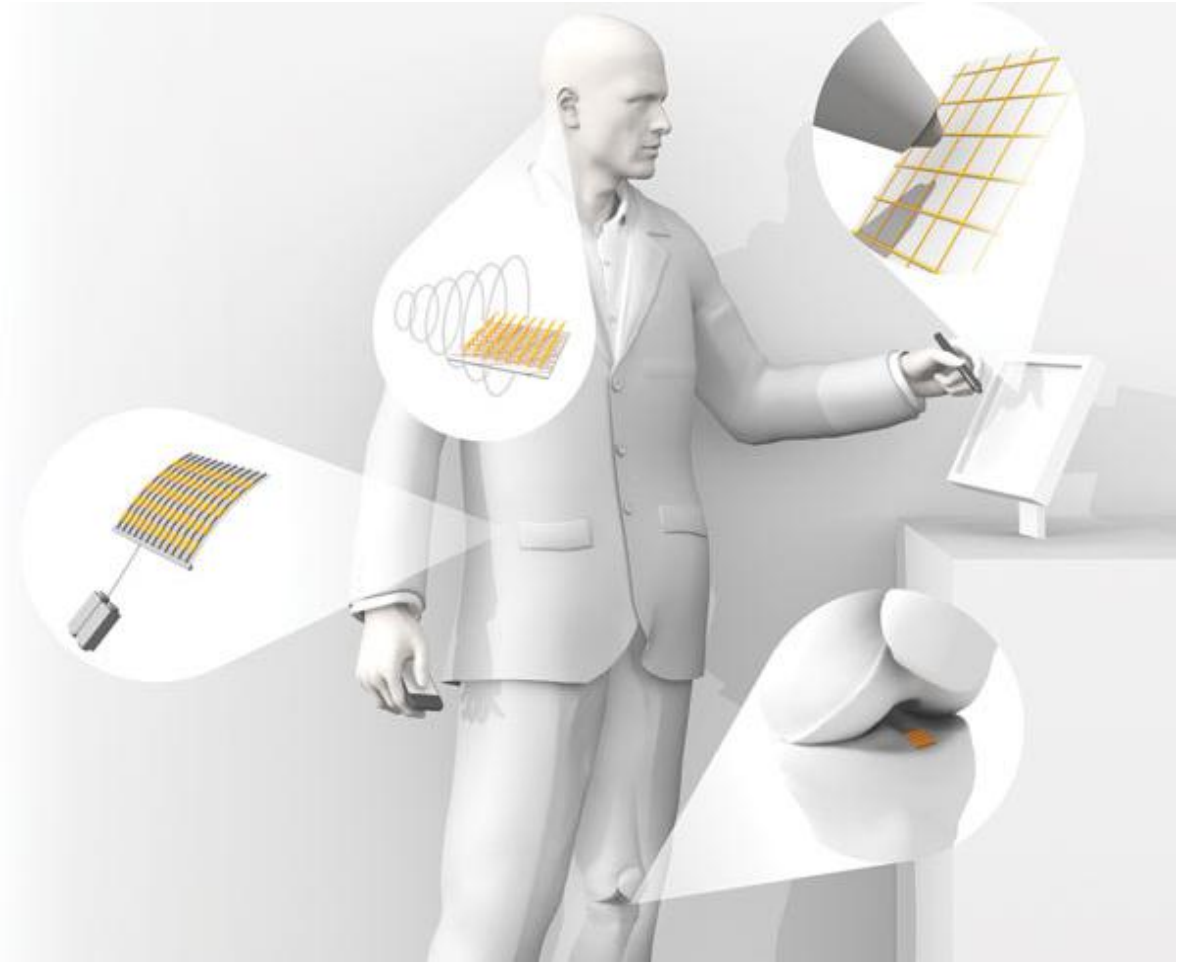


[6]

Future Outlook



[7]



Power Management in Android

Android Power Management Design

- Apps and services must request CPU resource with ‘wake locks’ through the Android application framework to keep power on, otherwise Android will shut down the CPU
- Android PM uses wake locks and time out mechanism to switch state of system power, so that system power consumption is managed.

Wakelock

- By default, Android tries to put the system into a sleep or better a suspend mode as soon as possible
- Some applications need to assure that the screen stays on or the CPU stays awake to react quickly to inputs
  - To do so, we use wakelock
  - Without active wake locks, the CPU will be turned off
  - Partial wakelocks can turn off screen or keyboards.

Android Wakelock Architecture

- Wakelocks are power-managing software mechanisms that make sure that your Android device doesn’t go into deep sleep
- Because a given application needs to use your system resources.
- Class PowerManager.WakeLock is mechanism to indicate that the application requires the device to stay on.

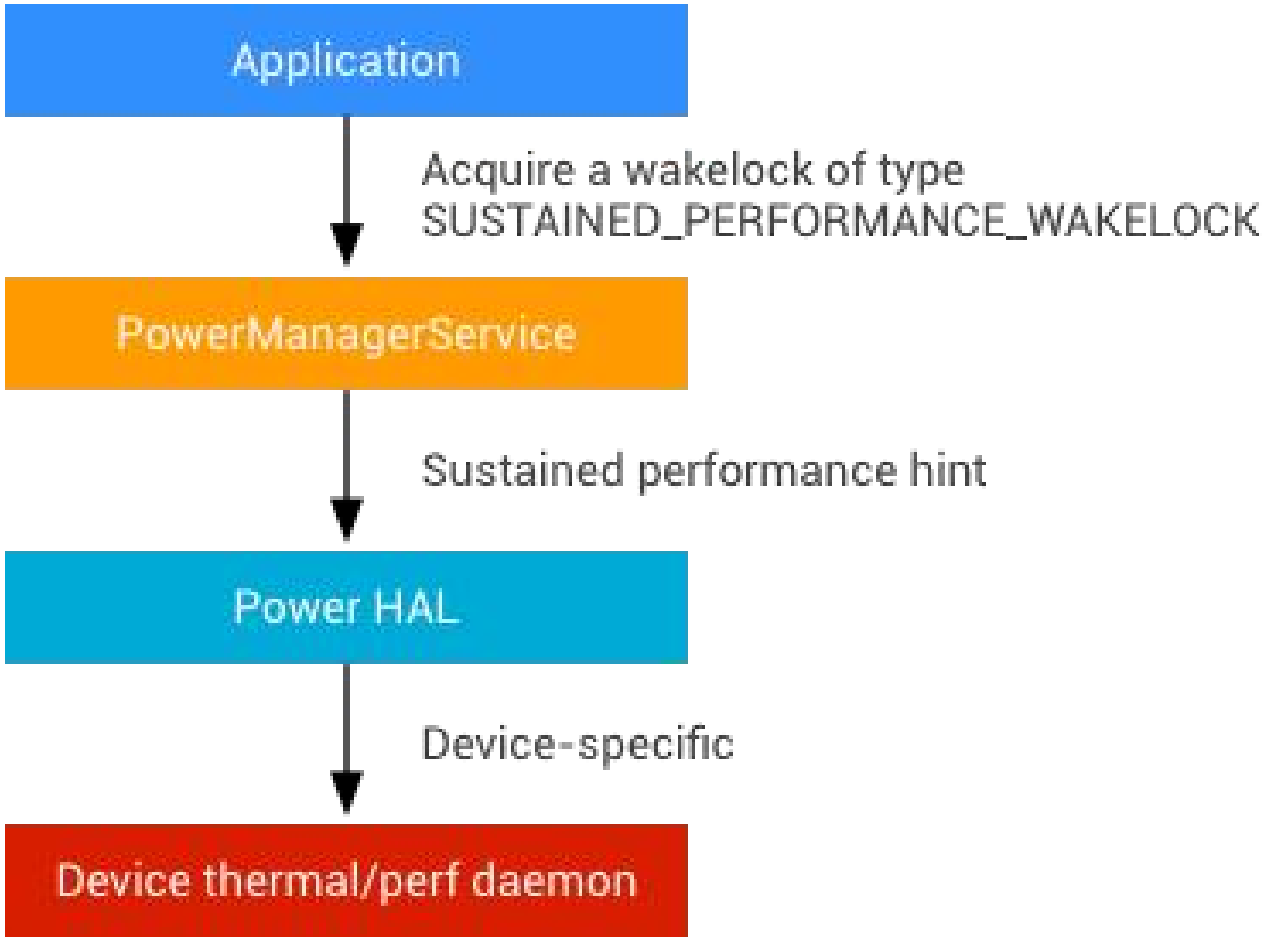
Wakelock Levels

Experiment with it in your upcoming camera programming lab

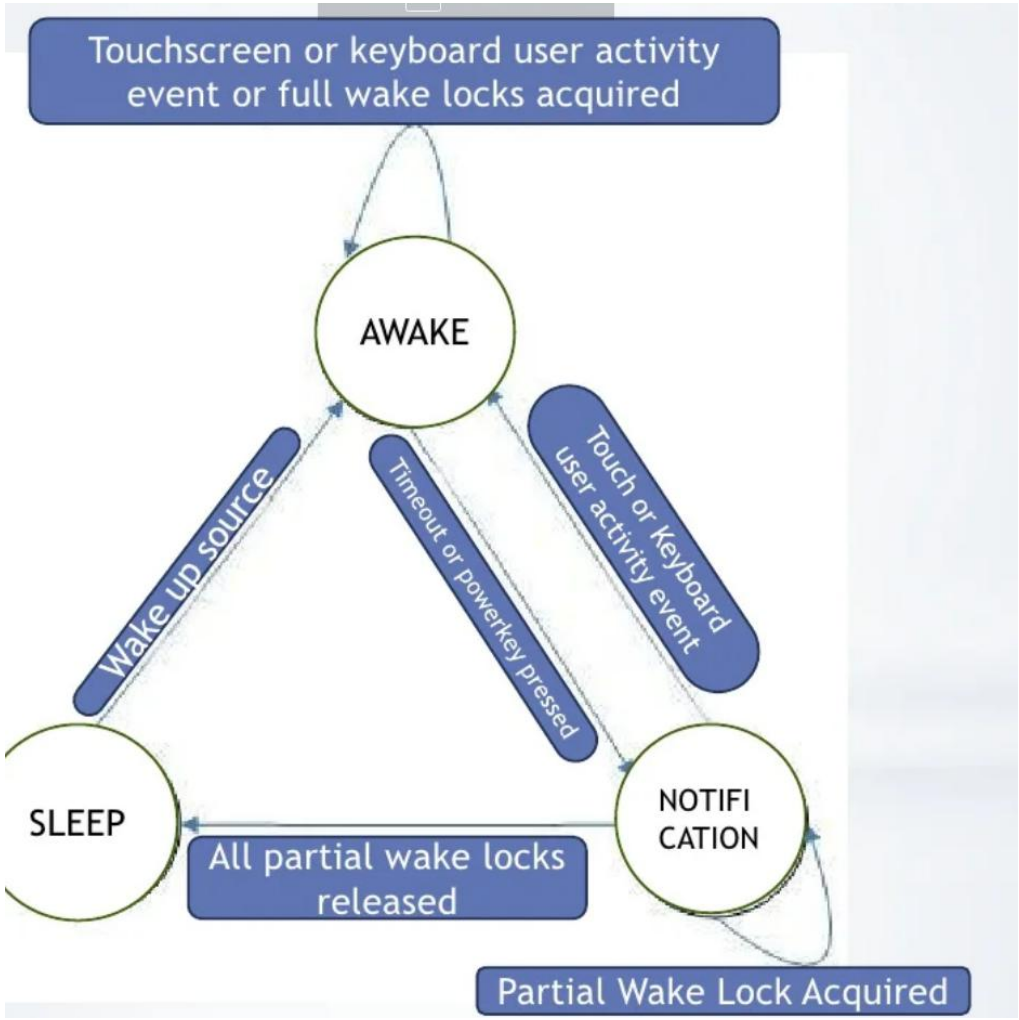
Flag Value	CPU	Screen	Keyboard
PARTIAL_WAKE_LOCK	On	Off	Off
SCREEN_DIM_WAKE_LOCK	On	Dim	Off
SCREEN_BRIGHT_WAKE_LOCK	On	Bright	Off
FULL_WAKE_LOCK	On	Bright	Bright

Android PM Design

The PowerManager class can control the power state of the device

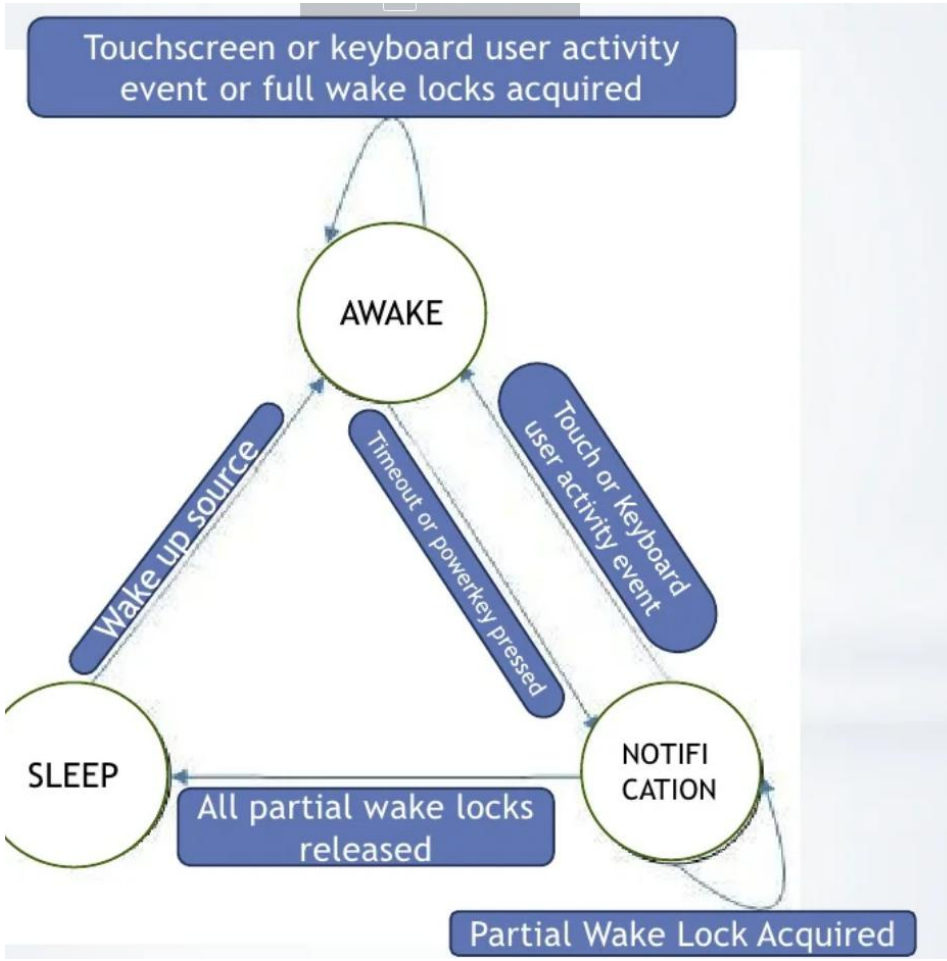


States for Android Power Management



States for Android Power Management

- When an application acquires full wake lock or an UI activity event occurs, the machine enters ‘wake’ state
- If timeout happens, the machine enters ‘notification’ state
  - If partial wake locks are acquired, it will remain in ‘Notification’
  - If all partial locks are released, the machine go into ‘sleep’



System Sleep

- API to bring device to sleep when we press the power button
- Require DEVICE_POWER permission
- Can only be called in system process context by verifying UID and PID
- When the power button is pressed, an API goToSleep() is called in the PowerManager
- goToSleep() will force release all wake locks

Software Techniques for Battery Conservation in Mobile Devices

- Three things to help
  - Make apps *Lazy First*
  - Take advantage of platform features
  - Use tools to identify battery draining components

Lazy First

- **Reduce**
  - Are there redundant operations your app can cut out?
- **Defer**
  - Does an app need to perform an action right away?
- **Coalesce**
  - Can work be batched, instead of putting the device into an active state many times?

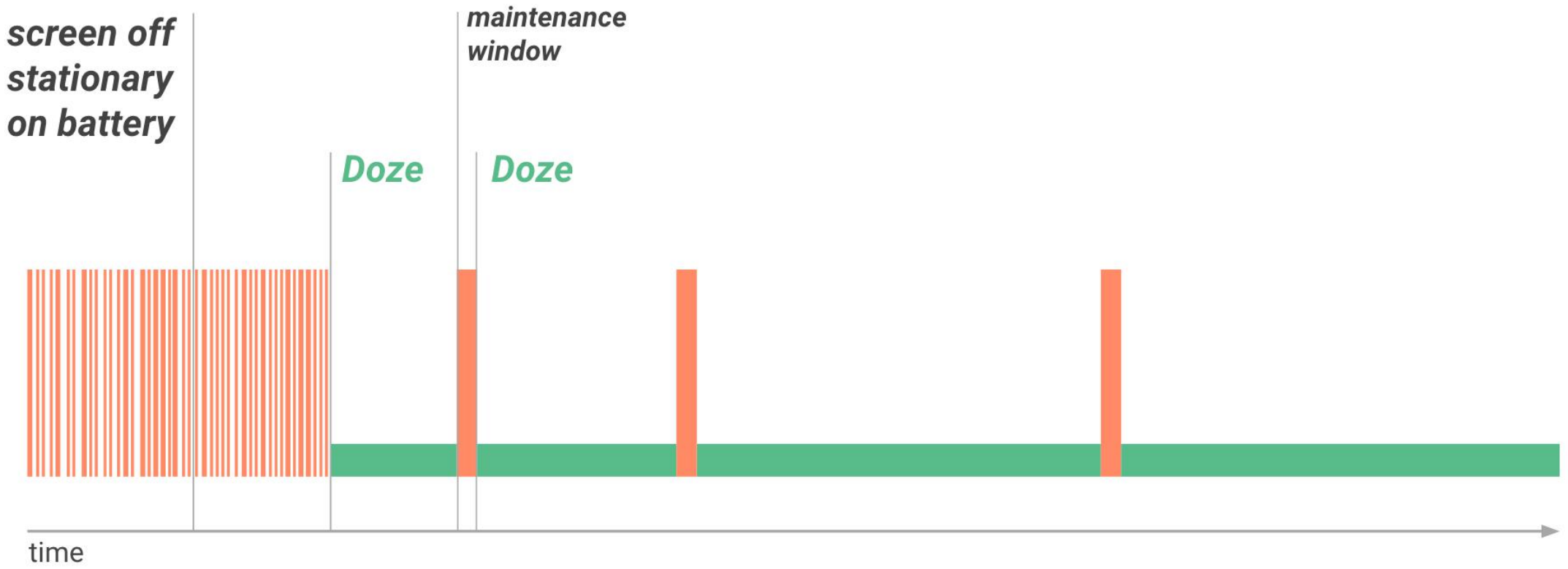
- Three things to help
  - Make apps *Lazy First*
  - **Take advantage of platform features**
  - Use tools to identify battery draining components

Android platform features

- **Doze** and **App Standby**:
  - Two power saving features
  - Manage how apps behave when a device is not connected to a power source
- Doze
  - Defers background CPU and network activities when the device is unused for a long time
- App Standby
  - Defers network activity when user has not recently interacted

Doze Mode

- Starts when a device is **unplugged** and **stationary** for **a period of time**, with the **screen off**
- Periodically, the system exits Doze to let apps complete their deferred activities, such as running pending syncs, jobs, alarms, and network access



Doze Mode

- Apps on your phone will have no network access, the system will ignore “wakelocks” when apps try to keep the device from going to sleep, and no background tasks will be allowed to run.
  - High-priority push messages will still show up.
  - So for example, a Hangouts message will appear on a device that’s in Doze mode.

Android App Standby

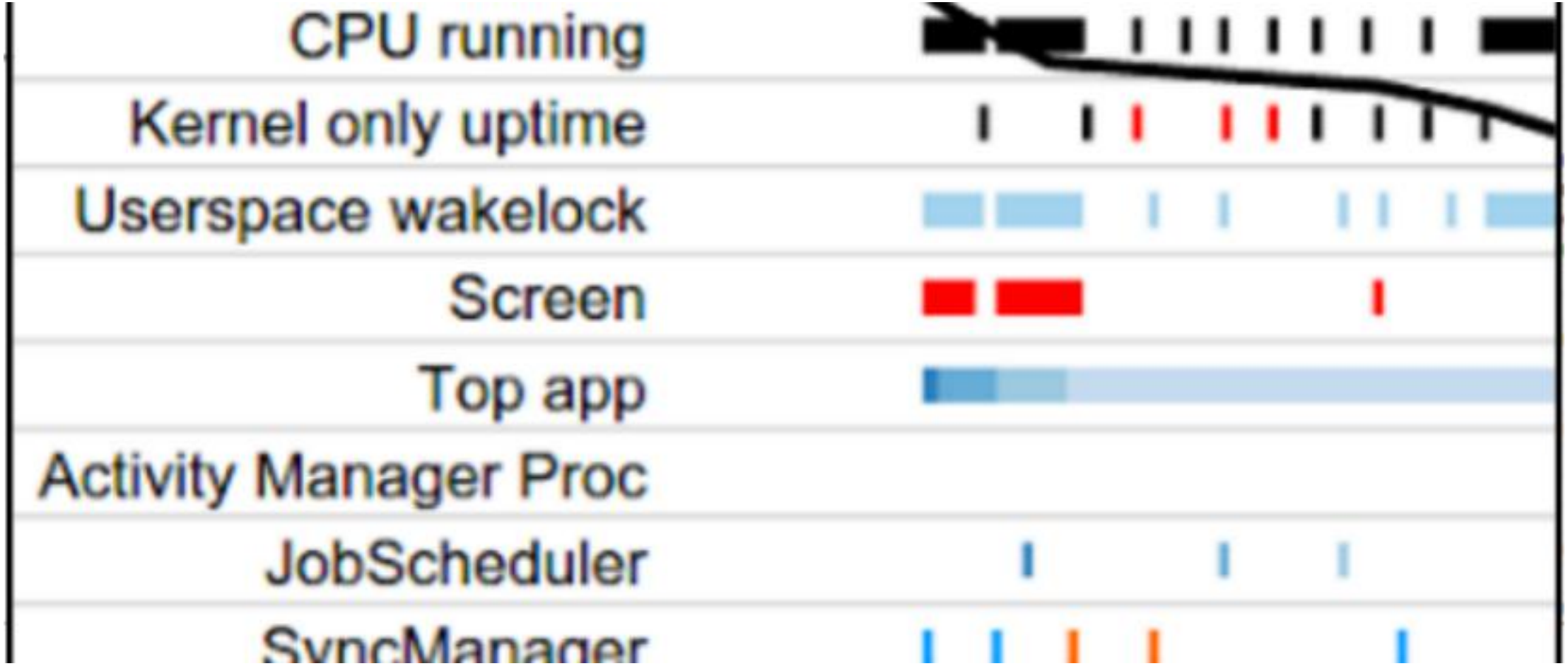
- The system puts an app into Standby mode and defers its network access if for a certain period of time,
  - the user has not explicitly launched the app, and
  - the app doesn’t have a foreground process (activity), and
  - the app doesn’t generate notifications, and
  - the app is not an active device admin app.
- When the phone is plugged to power, apps are released from standby state
- If the device is idle for a long period of time, idle apps access network around once a day

- Three things to help
  - Make apps *Lazy First*
  - Take advantage of platform features
  - **Use tools to identify battery draining components**

Battery Historian: A tool to analyze power use in Android

provides a system-wide visualization of various app and system behaviors, along with their correlation against battery consumption over time.

A closeup look



Battery historian: App-specific view

- Provides tables for the following data:
  - The app’s estimated power use on the device.
  - Network information.
  - Wakelocks.
  - Services.
  - Process info.

Display the ranking of app powers

Sort apps by

Device estimated power use

com.curlytail.pugpower (Uid: 10372)

Tables

- System Stats
  - Aggregated Checkin Stats
  - Sorted by Device estimated power use
  - Device's Power Estimates
  - Userspace Wakelocks
  - SyncManager Syncs
  - Mobile Radio Activity Per App

Seems suspicious

Device's Power Estimates:

Show 10 entries

Ranking	Name
0	OVERCOUNTED
1	ANDROID_SYSTEM
2	ROOT
3	SYSTEM_UI
4	CELL
5	com.fungames.pokerific
6	com.android.chrome
7	SCREEN
8	com.curlytail.pugpower

Other cases where Battery Historian can help

- Examples:
  - Firing wakeup alarms overly frequently (every 10 seconds or less).
  - Continuously holding a GPS lock.
  - Scheduling jobs every 30 seconds or less.
  - Scheduling syncs every 30 seconds or less.
  - Using the cellular radio more frequently than you expect.