Gesture-enabled Remote Control for Healthcare

Hongyang Zhao, Shuangquan Wang, Gang Zhou
College of William and Mary

Daqing Zhang
Institut Mines-Télécom

Presenter: Hongyang Zhao
Background

- Gesture recognition is widely used in healthcare
  - Manipulate healthcare device
  - Physical rehabilitation
  - Fall detection
Limitation of Current Gesture Recognition Platform

- Not comfortable to wear
- No open API
- Too expensive

Lack of reliable platform for gesture recognition and motion sensing study in healthcare
Limited work in continuous gesture recognition

- **Continuous gesture recognition**
  - Retrieve and recognize gesture from a sequence of hand movement

- **Current work**
  - Not accurate
  - Huge computational effort

Lack of effective continuous gesture recognition mechanism for resource-limit device.
Outline

- Wristband hardware platform
- Continuous hand gesture segmentation and recognition framework
- Introduction to APIs
Gemote Hardware Components

**Various sensors**
- Accelerometer
- Gyroscope
- Compass

**USB charge**
- Charge Time: 1 hour

**BLE supported**
- Rage: 40m

**Li-Ion battery**
- 3.7V, 75mAH

**Strong computational capability**
- nRF 52832 (ARM M4)

**Energy efficiency**
- Work Current: 10~20mAH
- Sleep Current: 1µAH
Gemote Hardware Features

- **Open API**
  - Open data sensing APIs to Android developers.

- **Comfortable to wear**
  - PCB: 26mm length, 25mm width.
  - Shell: 47mm length, 31mm width, and 9mm thick.

- **Affordable price**
  - $29
Application of Gemote wristband-1
Application of Gemote wristband-2
Outline

- Wristband hardware platform
- Continuous hand gesture segmentation and recognition framework
- Introduction to APIs
Define gestures that best emulate a remote controller

<table>
<thead>
<tr>
<th>Button Function</th>
<th>Gesture Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Up</td>
<td>Up</td>
</tr>
<tr>
<td>Down</td>
<td>Down</td>
</tr>
<tr>
<td>Left</td>
<td>Left</td>
</tr>
<tr>
<td>Right</td>
<td>Right</td>
</tr>
<tr>
<td>Select</td>
<td>Clockwise</td>
</tr>
<tr>
<td>Back</td>
<td>Anticlockwise</td>
</tr>
<tr>
<td>Home</td>
<td>Back&amp;Forth</td>
</tr>
</tbody>
</table>
How to retrieve and recognize seven defined gesture from a sequence of hand movements?

- Raise hand -> Left gesture -> Back & Forth gesture -> put down hand
Sequence Start/End Detection

- Lightweight threshold-based detection metric
  \[ HM = \sqrt{Gyro_x^2 + Gyro_y^2 + Gyro_z^2} \]

- Start of hand movement: \( HM > 50 \)
- End of hand movement: \( HM < 50 \) for 400ms
Within-sequence gesture separation

- Gestures start/end in positions with slow hand movement
- Static positions: positions with slow hand movement
- Static positions ⇔ slow hand movement ⇔ low HM

• Slow hand movement
• Low HM
Within-sequence gesture separation

- Valleys in HM curve are potential start/end positions of gestures.
- Apply sliding window to detect valleys of HM curve.

Segment 1~6
Merging Adjacent segments

- One gesture may lie in one segment or several adjacent segments.

- Merge adjacent segments so that each segment only contains one gesture
  - Gesture Continuity
  - Gesture Completeness
Gesture Continuity

- If two segments have similar slopes near connecting points, these segments belong to one gesture.

  - Find points before connecting point within time window 300ms as $t_a, t_b, t_c, t_d, t_e, t_f$, and points after as $t_g, t_h, t_i, t_j, t_k, t_l$.
  - Form 12 lines and find the maximum angle as $\theta_{gi}$.
  - Compute weight $w_{gi}$ as the area size of the curve $g_i$.
  - Gesture Continuity (Con)
    - $Con(t_1) = \frac{\sum(w_{gi} \cdot \theta_{gi})}{\sum w_{gi}}$
Gesture Completeness

- The defined gestures start/end in the same position
  - The sum of sensor readings should be close to 0 for a complete gesture

- Gesture Completeness (Com)
  - \( Com(t_1) = \frac{\left| \sum_{t_0}^{t_2} g_x \right| + \left| \sum_{t_0}^{t_2} g_y \right| + \left| \sum_{t_0}^{t_2} g_z \right|}{\sum_{t_0}^{t_2} |g_x| + \sum_{t_0}^{t_2} |g_y| + \sum_{t_0}^{t_2} |g_z|} \)
Con VS Com

- 100 continuous gestures: 177 connecting points
  - Blue stars: connecting points that separate two gestures
  - Red circles: connecting points that are inside gestures
- Merge two adjacent segments if Con < 40 degree and Com < 0.2
Merging Adjacent Segments

- After merging, each segment contains exactly one gesture.
Hand Gesture Recognition

- **Features Extraction**
  - Raw, first-derivative and the integral of acceleration data and gyroscope data

- **Classification**
  - Hidden Markov Model

- **Accuracy**
  - Segmentation: 98.8%
  - Recognition: 95.7%
  - Overall: 94.6%
Outline

- Wristband hardware platform
- Continuous hand gesture segmentation and recognition framework
- Introduction to APIs
UG wristband Open API

- API definition (three classes)
  - UGManager
    - Used to scan UG devices
  - UGDevice
    - Used to connect to certain UG device and collect sensor data from it.
  - UGProfile
    - Used to represent the current status of a UG device.
Open API for UG wristband

- **UGManager**
  - `Void startScan(ScanCallback cb)`
  - `Void stopScan()`
  - `Interface ScanCallback{
      void onScanCallback(UGDevice device);
    }`
Open API for UG wristband

- **UGDevice**
  - Void Connect (StatusChangeCallback cb)
  - Void Disconnect()
  - Void startDataSensing (DataAvailableCallback cb, int rate)
  - Void stopDataSensing()
  - Void setLED(Byte[] ledMask)
  - Void getBatteryLevel()
  - String getAddress()
  - Interface StatusChangeCallback{
      void onStatusChange (UGDevice device, int status);
  }
  - Interface DataAvailableCallback{
      void onSensorDataAvailable (UGDevice device, float[] data);
      void onBatteryLvlAvailable (UGDevice device, int data);
  }
Open API for UG wristband

- **UGProfile**
  - public static final int `STATUS_DISCONNECTED` = 0;
  - public static final int `STATUS_CONNECTED` = 1;
  - public static final int `STATUS_DATA_SENSING_ON` = 2;
  - public static final int `STATUS_DATA_SENSING_OFF` = 3;
Steps to use UG APIs

- Import packages
- Use UGManager class to scan UG wristbands
- Use UGDevice class to connect to UG wristbands
- Use UGDevice class to read data from UG wristbands
Import packages

- import com.ultigesture.ug.UGDevice;
- import com.ultigesture.ug.UGManager;
- import com.ultigesture.ug.UGProfile;
Use UGManager class to scan UG wristbands

- Create an object of UGManager class
- Call `startScan()` method to scan nearby UG wristbands
  - `UGManager mUGManager = new UGManager(this);`
  - `mUGManager.startScan(mScanCallback);`
Use UGDevice class to connect to UG wristbands

- Implement **callback** functions to receive scan results.
- Call **connect()** method to connect to a UG wristband

```java
private UGManager.ScanCallback mScanCallback = new UGManager.ScanCallback(){
    @Override
    public void onScanCallback(UGDevice device) {
        device.connect(mConnectionStateChangedCallback);
    }
};
```
Use UGDevice class to read data from UG wristbands.

- Implement **callback** functions to receive the status of a UG wristband.
- Check if a UG wristband is connected.
- Call `startDataSensing()` method to collect sensor data from a UG wristband.

```java
private UGDevice.StatusChangeCallback mConnectionStateChangeCallback =
    new UGDevice.StatusChangeCallback()
    {
        @Override
        public void onStatusChange(UGDevice device, int status) {
            if (status == UGProfile.STATUS_CONNECTED){
                device.startDataSensing(mDataAvailableCallback, 100);
            }
        }
    };
```

**Sampling interval: 100ms**
Implement *callback* functions to receive data from UG wristbands.

```java
private UGDevice.DataAvailableCallback mDataAvailableCallback =
    new UGDevice.DataAvailableCallback(){
        @Override
        public void onSensorDataAvailable(UGDevice ugDevice, float[] data) {
            // data[0]: AccX  data[1]: AccY  data[2]: AccZ
            // data[3]: GyroX data[4]: GyroY data[5]: GyroZ
            // data[6]: MagX  data[7]: MagY  data[8]: MagZ
        }
    }

    @Override
    public void onBatteryLvlAvailable(UGDevice device, int data) {
        // data: battery level
    }
};
```
Demo-1
Demo-2
Conclusion

- Wristband hardware platform
  - Comfortable to wear
  - Open API
  - Affordable price

- Continuous hand gesture segmentation and recognition framework
  - Lightweight
  - Accurate

- Introduction to APIs
Question?