REAL-TIME MORSE CODE COMMUNICATION APP

FINAL YEAR PROJECT SPRING 2014 LYU1305

Supervisor: Prof. LYU Rung Tsong Michael

Students: LUO Xin (1155026046)
ZOU Lei (1155026057)
AGENDA

Fall 2013 Review

Spring 2014 Overview

Design and Implementation

Project demo

Conclusion
FALL 2013 REVIEW
FALL 2013 REVIEW

- Separated encoding and decoding part
- Decoding with OpenCV
- Fixed Morse code frequency
- Fixed detection area
SPRING 2014 OVERVIEW

- A complete application
- Auto light source locating and tracking
- Auto code frequency detection
- Chinese supporting
IMPLEMENTATION

Apps Combination
Camera Preview & Frame Buffer
Light Source Locating & Tracking
Auto Detection
Unicode Encoding & Decoding
APPS COMBINATION

Overview

Design and Implementation

Project demo

Conclusion
APPS COMBINATION

Android Camera control

- Encoding
- Decoding

- Flash light
- Camera preview

Fall 2013 Review
Spring 2014 Overview
Design and Implementation
Project demo
Conclusion
CAMERA PREVIEW

1. Open the Camera Object
2. Modify Camera Settings
3. Create the Camera Preview
4. Start the Preview
5. Stop Preview and Release Camera

- Call `Camera.open()`
- Resolution: `setPreviewSize()`
- Frame Frequency: `setPreviewFpsRange()`
- Exposure: `setExposureCompensation()`
- Implement the `SurfaceHolder.Callback` interface.
- Call `startPreview()` to start updating the preview surface.
- Call `stopPreview()` to stop updating preview.
- Call `release()` to release the camera.
## CAMERA FRAME BUFFER

YUV420sp format

<table>
<thead>
<tr>
<th></th>
<th>Y1</th>
<th>Y2</th>
<th>Y3</th>
<th>Y4</th>
<th>Y5</th>
<th>Y6</th>
<th>Y7</th>
<th>Y8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y9</td>
<td>Y10</td>
<td>Y11</td>
<td>Y12</td>
<td>Y13</td>
<td>Y14</td>
<td>Y15</td>
<td>Y16</td>
<td></td>
</tr>
<tr>
<td>Y17</td>
<td>Y18</td>
<td>Y19</td>
<td>Y20</td>
<td>Y21</td>
<td>Y22</td>
<td>Y23</td>
<td>Y24</td>
<td></td>
</tr>
<tr>
<td>Y25</td>
<td>Y26</td>
<td>Y27</td>
<td>Y28</td>
<td>Y29</td>
<td>Y30</td>
<td>Y31</td>
<td>Y32</td>
<td></td>
</tr>
<tr>
<td>U1</td>
<td>V1</td>
<td>U2</td>
<td>V2</td>
<td>U3</td>
<td>V3</td>
<td>U4</td>
<td>V4</td>
<td></td>
</tr>
<tr>
<td>U5</td>
<td>V5</td>
<td>U6</td>
<td>V6</td>
<td>U7</td>
<td>V7</td>
<td>U8</td>
<td>V8</td>
<td></td>
</tr>
</tbody>
</table>

- **Fall 2013 Review**
- **Spring 2014 Overview**
- **Design and Implementation**
- **Project demo**
- **Conclusion**
CAMERA FRAME BUFFER

Callback Function: onPreviewFrame(byte[] data, Camera camera)
Trigger: setOneShotPreviewCallback(MainActivity.this)

YUV420sp -> RGB:
r = (1192 * y + 1634 * v);
g = (1192 * y - 400 * u - 833 * v);
b = (1192 * y + 2066 * u);

mRgb[i][j][0] = (int)(r >> 10);
mRgb[i][j][1] = (int)(g >> 10);
mRgb[i][j][2] = (int)(b >> 10);
LIGHT SOURCE LOCATING

Original version

----Depends on percentage of light ON pixels
LIGHT SOURCE LOCATING

2nd version

-----Finding light center and “cutting” the screen
LIGHT SOURCE LOCATING

2nd version

-----Finding light center and “cutting” the screen

Position of ★ is calculated by:

totalRGB = sum of all the pixels’ RGB

centerRow = (\sum_{r=0}^{totalRow} \sum_{c=0}^{totalCol} r \times RGB(r, c)) / totalRGB

centerCol = (\sum_{r=0}^{totalRow} \sum_{c=0}^{totalCol} c \times RGB(r, c)) / totalRGB
LIGHT SOURCE LOCATING

2nd version

-----Finding light center and “cutting” the screen

- Get the grid with the largest light center RGB value
- Repeat the previous process in this grid
LIGHT SOURCE LOCATING

2nd Version

----Finding light center and “cutting” the screen

- Very time consuming
- The preview frame is not continuous
LIGHT SOURCE LOCATING

3rd version

-----Comparing Grids’ light center’s RGB value

 lokalCenterRow, lokalCenterCol
LIGHT SOURCE LOCATING

3rd version

----Comparing Grids’ light center’s RGB value

- Comparing each grid’s light center’s RGB value:

  \( \text{RGB(localCenterRow, localCenterCol)} \)

- Finding the grid with the largest light center RGB value
LIGHT SOURCE LOCATING

3rd version

-----Comparing Grids’ light center’s RGB value

Set this grid to be the initial detection window
LIGHT SOURCE LOCATING

**3rd version**

----Comparing Grids’ light center’s RGB value

However...If we have two grids like this:

- Light center offset from the cluster of the light pixels
- Light center accidently to be the light pixel
LIGHT SOURCE LOCATING

Final version

----Finding local maximum RGB pixels and counting

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td></td>
<td>*</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>*</td>
<td></td>
</tr>
</tbody>
</table>

- Intra-Grid
- Inter-Grid
Final version

-----Finding local maximum RGB pixels, counting and comparing

- Intra-Grid

  - Compare RGB value pixel by pixel
  - Determine the local max RGB value
  - Count the number of pixels with local max RGB value
LIGHT SOURCE LOCATING

Final version

----Finding local maximum RGB pixels, counting and comparing

- Inter-Grid
  - Compare two grid's local max RGB value
  - If same, comparing number of those pixels
  - Record as temp global max RGB value and grid
  - If the last one's R+G+B == 765 & num >= 10, set it to be the initial detection window
LIGHT SOURCE TRACKING

After getting the initial detection window, we need to follow it in case that the camera shook accidently

- Consider the light center as the center to draw next tracking window
- Calculate the light center and draw the tracking window recursively

**Problem:** Cannot relocate the light source during decoding

**Reason:** Re-locating costs too much time ==> Preview frame is not continuous
AUTO DETECTION

Method: Made use of start signal

Sending part:
- Set start signal to be 10 times of the DOT duration

Receiving part
- Estimate the DOT duration according to the start signal length
- Decode the pattern according to the DOT duration
Unicode Representation:
Chinese Unicode ranges from U+4E00 to U+9FA5 (19968 – 17194).
UNICODE ENCODING

Check every character ch in the input message.

If (chr1>=19968&&chr1<=171941)
    result.append(“U+” + Integer.toHexString(ch));
Else
    result.append(“” + ch);

Return result

I love 中大

I love U+4e2dU+5927
Define pattern “U+hhhh” by “(U\+\(\p{XDigit}\{4\})\)”

While find the matched pattern in input string str
convert “hhhh” to Chinese character
replace “U+hhhh” to the Chinese character

Return str
Light Source Locating & Tracking

Transmission canceled & Invalid signal detection

Chinese Supporting

Bi-directional Communication
CONCLUSION

Improvement in Spring 2014

Limitations in Spring 2014

Summary in the whole year
IMPROVEMENT IN SPRING 2014

Limitations in Fall 2013:

- Separated apps
- Unchangeable transmission rate
- Non-automatic decoding
- Disturbance of environmental light
- Low accuracy under high transmission rate
- Unchangeable parameters of the environmental light, e.g. exposure value
LIMITATIONS IN SPRING 2014

- Disturbance of environmental light.

- Low accuracy under transmission frequency < 0.3s/unit.

- Cannot relocate the light source during decoding.

- Cannot determine whether the pattern “U+hhhh” is a Chinese character or not.
SUMMARY IN THE WHOLE YEAR

Real-time Morse code communication:

- Bi-directional communication
- Auto light source locating and tracking
- Auto code frequency detection
- Chinese supporting
THANK YOU!

Final Year Project Fall 2013

Supervisor: Prof. LYU Rung Tsong Michael

Students: LUO Xin (1155026046)
ZOU Lei (1155026057)