Natural Human-Computer Interaction with Kinect and Leap Motion

Presented by: Xietong LU
Email: u5431681@anu.edu.au
Supervisor: Yuchao DAI
Overview

- Kinect vs. Leap Motion
  - Advantages
  - Disadvantages
- Motivation
- Implementation
  - PHASE 1: Develop Leap Motion Application
  - PHASE 2: Develop Kinect Hand Detector
  - PHASE 3: Combine Leap Motion & Kinect
- Conclusion
Kinect vs. Leap Motion

Kinect

- RGB camera and depth camera
- + Large range: 500mm - 3000mm (near mode)
- + Skeletal Tracking
- - No built-in hand detection function
- - invisible when too close

Skeletal tracking
[cited from Microsoft.com]
Kinect vs. Leap Motion
Kinect vs. Leap Motion

- Kinect
  - RGB camera and depth camera
  - + Skeletal Tracking
  - - No built-in hand detection function
  - - invisible when too close

Skeletal tracking [cited from Microsoft.com]
Kinect vs. Leap Motion

- Leap Motion
  - device for hand recognition
  - + Reliable data in some cases
  - + Built-in gesture recognition function
  - - Small region
  - - Post limitation
Kinect vs. Leap Motion

- Leap Motion
  - Post limitation
Motivation

- Combine Kinect and Leap Motion
  - More accurate hand detection
  - Support more gestures
  - Enhance user experience in HCI
Implementation

Development and Learning

3 phases

PHASE 1: Develop a gallery application using Leap Motion
PHASE 2: Develop a hand detector using Kinect
PHASE 3: Combine Leap Motion & Kinect and visualize the result
Implementation - PHASE 1

- A gallery application
  - Web-based

- Fully controlled by Leap Motion
  - Leap Motion’s gesture recognition function
  - Finite State Machine

- Demonstration
  - Video Link
Implementation - PHASE 2

- Hand detection application
  - Palms and fingertips

- Using Kinect
  - No built-in hand detection functions
Implementation - PHASE 2

- **Workflow**
  - **Pre-process**
    - RGB Frame
    - Depth Filtering
    - Skin Colour Filtering
    - Find Contours
    - Contours Size Filtering
  - **Depth Calibration**
    - Use skeletal data to find depth threshold
    - DBSCAN - density-based clustering
Implementation - PHASE 2

- **Workflow**
  - Palms
    - Maximum inscribed circle
  - Fingertips
    - Convexity defects
    - k-curvatures algorithm
    - Distance to palm centre

- **Demonstration**
  - In next phase
Implementation - PHASE 2

- **Workflow**
  - Palms
    - Maximum inscribed circle
  - Fingertips
    - Convexity defects
    - k-curvatures algorithm
    - Distance to palm centre

- **Demonstration**
  - In next phase
Implementation - PHASE 3

- Combine Kinect and Leap Motion
  - Position in 2 coordinate systems
  - Both have errors - impossible to match exactly

- Horn’s Method [1]
  - A closed-form solution
  - Find $R$, $s$ and $r_0$
  - Minimize the sum of squares of errors
  - $r_{\text{transformed}} = sR(r_{\text{LeapMotion}}) + r_0$

- Demonstration
Conclusion

- An incomplete work
  - Gesture recognizer

- A fair fundamental work
  - OOP, well structured

- Gain
  - Involved a number of external libraries
    - Leap Motion API, Kinect API, OpenCV, PCL(point cloud library), ...
  - Different Algorithms in visions
  - Apply knowledge from data mining
  - HCI(human computer interaction)
Q&A
Reference