Demo Abstract: TouchAble - A Camera-Based Multitouch System

Lin-Shung Huang
Carnegie Mellon University
linshung.huang@sv.cmu.edu

Feng-Tso Sun
Carnegie Mellon University
lucas.sun@sv.cmu.edu

Pei Zhang
Carnegie Mellon University
pei.zhang@sv.cmu.edu

1 Introduction

Touchscreens enable users to interact directly and intuitively with computers by simply touching the display area without requiring any intermediate devices. There are various touchscreen technologies that generally utilize resistive or capacitive panels. Typical touchscreens are constrained by the fixed size and high cost panels. Many research efforts have been made towards achieving multitouch functionality using vision-based systems. However, existing approaches have limitations such as relying on pre-defined gestures [5], requiring users to wear a glove with a custom pattern [4], or using infrared light pens [2].

We present TouchAble, a camera-based multitouch system using wireless camera sensor nodes to enable multitouch capability on any two-dimensional display surface (e.g. a projection screen or laptop monitor). The TouchAble system provides the following contributions:

- Enables multitouch gesture interaction directly on any two-dimensional display surface of arbitrary size using minimal hardware.
- Automatically calibrates arbitrarily-placed wireless camera sensors.
- Adapts to user-defined gestures through a gesture learning phase that requires only one snapshot per camera.

To develop a practical system that is easy to setup and achieves reasonable accuracy, we faced new challenges and interesting research questions as follows. How can the system learn to detect the user's gestures under different camera settings? Can we improve the accuracy of gesture detection with image processing and classification techniques?

2 System Architecture

The hardware of the TouchAble system is consisted of multiple Sun SPOT sensor nodes equipped with C328 camera modules, and one Sun SPOT base station connected to the laptop. Figure 3 shows the overview of our system. The Sun SPOT sensor nodes and base station communicate over IEEE 802.15.4 compliant 2.4 GHz radio antennas.

In our system, the sensor nodes continuously retrieve compressed images from the camera via UART, and stream them over the wireless sensor network to the base station, where the base station writes the received files to the laptop filesystem through USB. The compressed images are processed on the laptop to detect user interactions and update the display accordingly. When the TouchAble system initializes, it first performs automatic camera calibration. Then, it enters a short gesture learning phase which will adapt to user-defined gestures. Our prototype system is developed with the Sun SPOT Java development kit [3] and the OpenCV C++ library [1].

Our system achieves over 92% classification accuracy on a 46" flat screen TV when the camera resolution setting is higher than 320x240. Higher image resolutions provide better accuracy, as shown in Figure 4. However, the latency (time between snapshot and display update) increases when the resolution is higher, as shown in Figure 5. The latency increases sharply when the resolution changes from 320x240

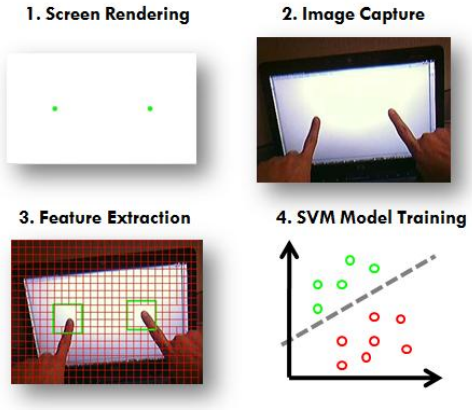

Figure 1. Steps of Gesture Learning Phase

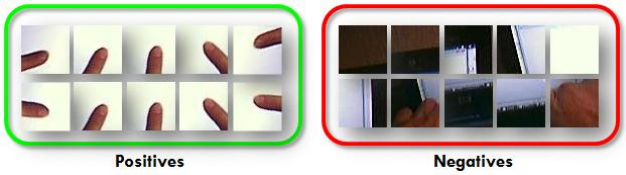

Figure 2. The sub-images centered on the two colored dots are cropped as positive training data; the other sub-images are used as negative training data. Positive sub-images are rotated to expand the scarce training data set.

SenSys'10, November 3–5, 2010, Zurich, Switzerland.
ACM 978-1-4503-0344-6/10/11

Figure 3. System Overview

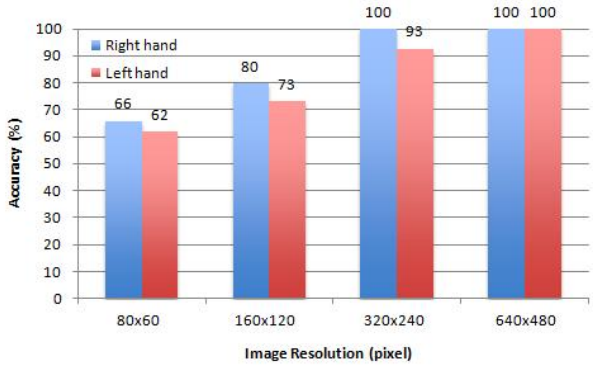

Figure 4. Accuracy vs Image Resolution

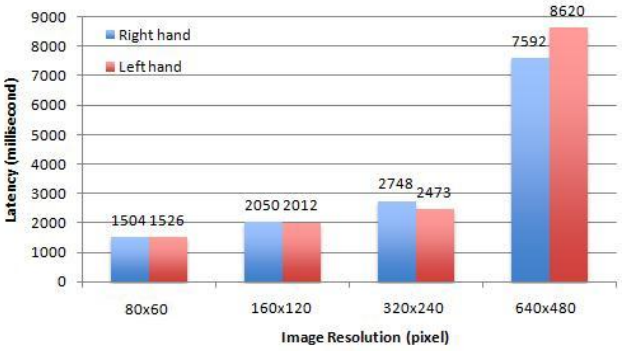

Figure 5. Latency vs Image Resolution

to 640x320, constrained by the network throughput bottleneck of approximately 30 kbps.

2.1 Automatic Camera Calibration

Since the camera sensor nodes are placed arbitrarily at different angles and distances in respect to the display area, the system needs calibration to correctly locate the display area from each camera perspective. We compute the projective mapping between the camera view coordinates and the display area coordinates. The calculation of this transformation matrix is implemented in OpenCV by rendering and detecting a chessboard pattern on the display area.

2.2 Adaptive Gesture Detection

To achieve adaptive gesture detection for different environments and users, we designed a gesture learning phase during system initialization to let users define their gestures under their camera and screen settings. For example, the user may chose to touch with fingers on a laptop monitor and with palms on a wide screen TV due to the different sizes of input surfaces. As shown in Figure 1, the gesture learning phase includes four steps:

1. Screen Rendering: Two colored dots are rendered at the center of the left half and right half of the display area.
2. Image Capture: The camera captures a snapshot of the user touching the colored dots on display area.
3. Feature Extraction: The captured image is cropped into sub-images and used as training data, as shown in Figure 2. To increase the accuracy of gesture classification using only one image for training, we expand the positive training data set by rotating the original image from -60° to 60° with 10° increments.
4. SVM Classification Model Training: We compute the histogram of oriented gradients (HOG) from the training data to build the support vector machine (SVM) model for gesture classification.

3 Demonstration

Our demonstration setup involves two individual display surfaces with their own arbitrarily-placed camera sensors. Our system performs the following steps:

- Automatic Camera Calibration: A chessboard is rendered on the display surface while our system computes the projective mapping for each camera perspective.
- Gesture Learning Phase: Users place self-defined gestures on two dots displayed on screen, while our system learns the user-defined gestures.
- Tic-tac-toe Game: Two users interact in the game on two separate display surfaces.

4 References

[1] OpenCV 2.1 C++ Reference. `http://opencv.willowgarage.com/documentation/cpp/`.

[2] J. Lee. Low-Cost Multi-point Interactive Whiteboards Using the Wiimote. `http://johnnylee.net/projects/wii`.

[3] Sun Microsystems, Inc. Sun SPOT Documentation. `http://www.sunspotworld.com/docs/`.

[4] R. Y. Wang and J. Popović. Real-time hand-tracking with a color glove. In *SIGGRAPH '09: ACM SIGGRAPH 2009 papers*, 2009.

[5] A. D. Wilson. Robust computer vision-based detection of pinching for one and two-handed gesture input. In *UIST '06: Proceedings of the 19th annual ACM symposium on User interface software and technology*, 2006.