The Implementation of a Glove-Based User Interface

Chris Carey

June 3, 2010

Abstract

Multi-touch interfaces have a strong advantage over the standard mouse interface because their multiple points of input can simplify the execution of certain complex commands. However, the requirement that the user physically touch the screen remains a disadvantage. A glove-based interface can provide the utility of a multi-touch interface without the proximity restriction. This project resulted in the implementation of a glove-based user interface using a system of battery-powered infrared LED gloves and a webcam modified to receive infrared light. LED detection and gesture recognition software was written to allow the user to execute several cursor-based commands in a custom photo manipulation application using the glove interface. The completion of various tasks performed in this photo application with single-handed glove commands, double-handed glove commands, and standard mouse commands was evaluated. The single-handed glove commands which matched the complexity of the mouse commands required more time for executeion. The double-handed glove commands matched and exceeded the speed at which the corresponding mouse commands were excuted. This indicates that command simplifcation is necessary in order to render the glove interface more efficient than the mouse interface. Developing gestural commands that take advantage of the glove interface's multiple points of input is the key, and in various applications would provide for more efficient and natural interfaction between the human and computer.

1 Introduction

1.1 Purpose

The purpose of this project is to explore the glove-based user interface, an emerging interface allowing for gestural commands that are not restricted to a 2D surface. The commands in this project are performed in a 2D plane in 3D space and provide the gestural utility of multi-touch, but without the restriction of physically touching a surface.

1.2 Scope

The goal of implementing a glove-based user interface is to determine where its advantages and disadvantages lie. A focus on task completion is necessary to evaluate the effectiveness of such an interface [1], and it must be allow for the evaluation of gestures relevant to controlling various applications such as software for geo-spatial imaging, 3D modeling, information visualization, and presentations.

1.3 Background

As the tools and technologies for building alternative user interfaces have become more readily available, alternatives to button and mouse interfaces have emerged. Multi-touch interfaces have been implemented as early as the mid 1980s [2], and have grown in usage both independently and commercially over the past few years as a result of improved accessibility to the required technology [3]. Recent advances in infrared-based multi-touch technologies have been moving towards reducing the need for the user to physically touch the screen, by instead allowing them

to hover over the screen [4]. A glove-based user interface, based on nearly identical IR LED technology, would eliminate this restriction all together. And though the idea of a glove-based user interface dates back to the beginnings of virtual reality [5], with proven applications in IR LED sensing now available in todays popular consumer electronics provide the glove-based interface with the potential for wider usage.

2 Implementation

2.1 Hardware Implementation

This research project was written in Java using the Java Media Framework and was tested on an HP Pavilion Laptop with a 2.1 GHz processor and 2.75 usable RAM running 32-bit Windows 7 Professional.

A modified Logitech USB webcam is used to provide a live 320 x 240 resolution video feed of infrared light. Its internal IR-blocking filter was removed and a visible-light blocking filter was externally installed, in order to ensure that only IR light is received by the webcam. The brightness and contrast of the video feed was set to ideal values using the webcams driver software.

The user controls the interface with two wireless gloves. Each glove contains three 950nm IR LEDs located on tips of the thumb, pointer finger, and middle finger, and are powered by three 1.5V AAA batteries. The circuits have an on/off switch and are sewn into the fabric of the glove at various points.

Fig. 1 - IR LED Gloves

2.2 Language and Structure

This research project is written in Java using the Java Media Framework in an effort to make the software more accessible and more efficient. A modular architectural framework is utilized in order to add recognizable gestures more easily.

3 Procedure

3.1 LED Detection

Each captured video frame is evaluated through a process of binary rasterization each pixel is marked as either above or below a certain brightness value. The optimal threshold value is automatically determined by creating a histogram of the pixel brightness values and choosing a value past the brightness of the peak histogram frequency level [6]. Fig. 2 shows a histogram in which the optimal threshold value is represented by the vertical orange line.

Using brightness as a representation of mass, each blob of bright pixels that meets a predetermined size and aspect ratio requirement is evaluated by splitting the blob with the diagonals of its bounding-box and determining if the pixels are equally distributed in each quadrant, indicating a circular shape. If the pixels are unequally distributed in a manner that indicates two overlapping LEDs, the blob is divided along the appropriate midpoint division line and the centers of mass of the two halves are calculated.

If it is determined that this unequally distributed blob represents more than two overlapped LEDs, the Circle Hough Transform algorithm is run on the blob to locate the centers of the multiple circles within the blob.

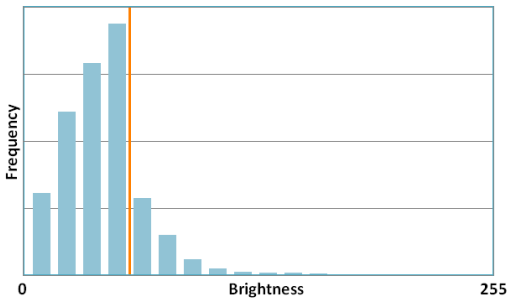


Fig. 2 - Histogram of Pixel Brightness Values with Optimal Threshold

3.2 LED Tracking

All LEDs detected in the video frame are matched to corresponding LED objects, and receive a group (left or right) and an ID (pointer, clicker, or auxiliary) classification to uniquely identify them. Tracked LED objects are created or deleted each time step to match the number of detected LEDs. The user is required to display either three or six LEDs on-screen for initial LED classification. If a new LED later appears on-screen, it is matched with the group it is closest to without exceeding the group limit, and receives the appropriate remaining ID.

Groups and IDs are nested within brackets so that when two LEDs combine to form a single LED, the group and ID of both previously separated LEDs are preserved and can be re-distributed to them when the combined LED separates. For example, consider the pinch gesture:

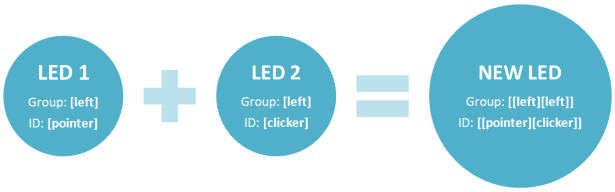


Fig. 3 - Nested LED Classifications

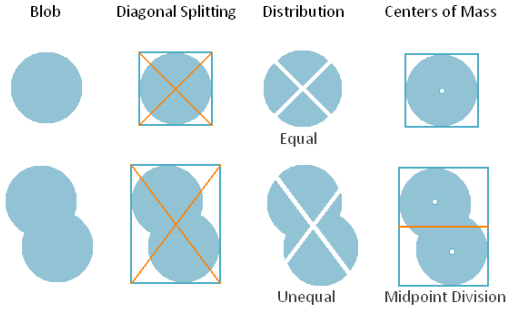


Fig. 4 - Diagonal Splitting of Two Overlapped LEDs

The depth of nesting in the new LED allows it to be identified as actually representing two combined LEDs, whose group and ID information has been preserved. This allows for the management of multiple LED overlaps without loss of data.

However, occasionally LED combinations, separations, entrances, and exits occur more quickly than what the application can handle, leading to the misclassification of these LEDs. Therefore, the LED groups and IDs are overridden at each time step depending on the number of LEDs being used on-screen, which is found not by counting the LEDs on-screen, but by finding the total sum of their nesting depths. From this, the application forces the LEDs to receive groups and IDs based on their positions on-screen. Though this requires the user to perform gestural commands within these parameters, it ensures proper LED tracking.

Fig. 5 - Gestural Command Parameters

3.3 Photo Manipulation Application

A simple photo manipulation application was created for testing and demonstration purposes, in which the user can drag, rescale, and rotate photos using both the mouse and glove interfaces. The Java-standard AffineTransform class is used to perform geometric transformations on a photo and convert on-screen coordinates to points in the photo's transformed coordinate space. The viewpoint perspective at which the user views the photos can also be controlled by panning or zooming. The AffineTransform class also handles transformations between the window coordinate space and the applications transformed coordinate space.

3.4 Gesture Recognition and Command Execution

3.4.1 Photo Manipulation Commands

Cursor Control The cursors are positioned proportionally to the centers of the LEDs in the video frame. A *two-finger pinch* is executed by bringing two LEDs in close proximity of each other. A *three-finger grab* is executed by bringing three LEDs close together.

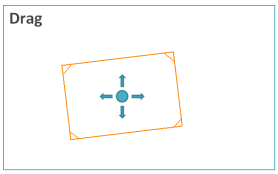


Photo Drag A photo is dragged by dragging with a two-finger pinch anywhere on the image except the marked corners.

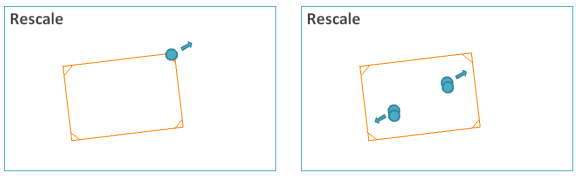


Photo Rescale A photo is rescaled with a double two-finger pinch gesture by dragging two cursors anywhere on the photo and moving them together or apart. A photo is also rescaled by dragging any of the photo's marked corners towards or away from the center to decrease or increase the photo's size respectively.

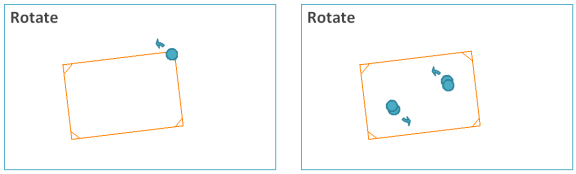
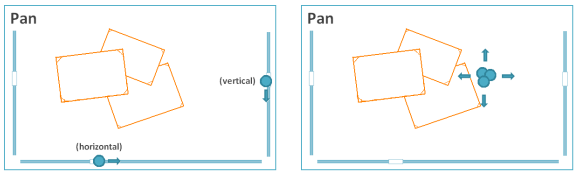
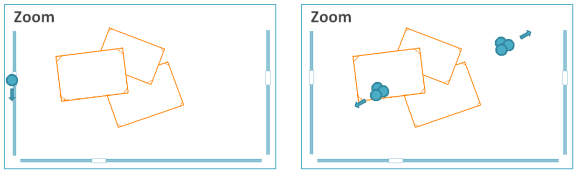


Photo Rotate A photo is rotated with a double two-finger gesture by dragging two cursors anywhere on the photo and rotating them around their midpoint. A photo is also rotated by dragging any of the photo's corners and rotating it around the photo's center point.

3.4.2 Application Viewpoint Commands



Viewpoint Pan The application viewpoint is panned horizontally and/or vertically by dragging anywhere on the screen with a three-finger grab. The viewpoint can also be panned using the on-screen scrollbars.



Viewpoint Zoom The zoom level of the application viewpoint is controlled with a double three-finger

grab gesture by dragging two cursors anywhere in the application window and moving them together or apart. The zoom level is also controlled by the left on-screen scrollbar.

4 Experiment

The glove interface was tested through the performance of a series of tasks. These tasks fall into two categories: (1) photo tasks, and (2) viewpoint tasks. In a photo task, a photo is placed on-screen with an initial position and orientation. A faded-out copy of that photo is placed on-screen in a different position and orientation. To complete the task, the user must drag, rescale, and/or rotate the photo so that it matches its faded-out copy within a certain degree of accuracy (Fig. 6). In a viewpoint task, the application scrollbars are set to predefined initial positions, and green rectangles appear on-screen along the scrollbars. To complete the task, the user must pan and/or zoom the applications viewpoint so that the scrollbar sliders fit inside the green rectangles along their scrollbar tracks (Fig. 7).

Fig. 6 - Photo Task Completion

Fig. 7 - Viewpoint Task Completion

During the completion of a task, the following data is collected and exported to a CSV (Comma-Separated Value) each time step:

- cursor x- and y-coordinates
- photo x- and y-coordinates
- photo angle and scale values
- application viewpoint pan and zoom values

The tasks were performed by the projects developer using both the mouse and glove interfaces. An effort was made to perform the tasks as naturally as possible erratic or unrepresentative task performances were not used for analysis. Each task was performed eleven times in succession with each interface. To eliminate outliers, only the middle five trials based upon total completion time were chosen for analysis. The tasks were performed in a seated position using the users dominant hand when applicable.

The following **photo task** was performed using the mouse interface, and the glove interface with one hand:

- Task #1: Drag a photo 500 pixels from right to left

The following **photo tasks** were performed using the mouse interface, the glove interface with one hand, and the glove interface with two hands:

- Task #2: Rescale a photo from 50% to 150% of its original size
- Task #3: Rotate a photo 3.0 radians clockwise

The following **viewpoint tasks** were performed using the mouse interface, and the glove interface with one hand:

- Task #4: Pan left 1000 pixels
- Task #5: Pan right 1000 pixels
- Task #6: Pan up 600 pixels
- Task #7: Pan down 600 pixels
- Task #8: Pan up 600 pixels and left 1000 pixels
- Task #9: Pan down 600 pixels and right 1000 pixels

The following viewpoint task was performed using the mouse interface, and the glove interface with two hands:

- Task #10: Zoom in from 75% zoom to 150% zoom

To begin the task, the user must click on the task start button at the bottom of the application. The **command acquisition time** (or command activation time) is defined as the time between the beginning of the task and the first use of the command. The **command manipulation time** (or command execution time) is defined as the time between the first use of the command and the termination of the command. The photo drag, rescale, and rotate tasks and the viewpoint zoom task were only performed in one direction because both the command acquisition and manipulation methods are the same regardless of direction, and therefore would achieve similar results.

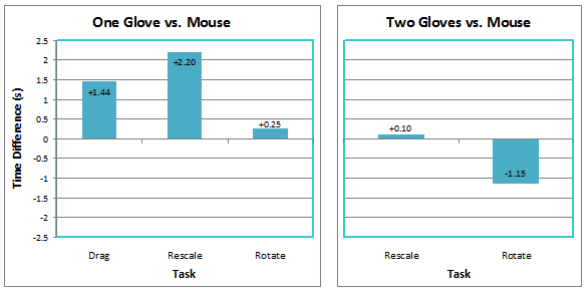
The viewpoint horizontal and vertical pan tasks were performed in two directions because the location command acquisition is different in each case. Using the mouse interface, the user must first move the mouse to the scrollbar slider, which is closer to the task start button in Task #5 (Pan right) and Task #6 (Pan up) than in Task #4 (Pan left) and Task #7 (Pan down). Differences in time between Task #4 and Task #5 and between Task #6 and Task #7 would uncover the relation between command acquisition and command manipulation times between the mouse and glove interfaces.

The viewpoint horizontal and vertical pan tasks were also combined not only to further investigate the difference in acquisition mentioned above, but also to demonstrate an instance in which the glove interface simplifies a multi-step task to a single gestural command. If the mouse interface has only one button available for dragging, the user must acquire and drag two scrollbars separately. The glove interface can perform the same task without having to acquire any target and without having to pan horizontally and vertically separately.

5 Results

5.1 Photo Tasks

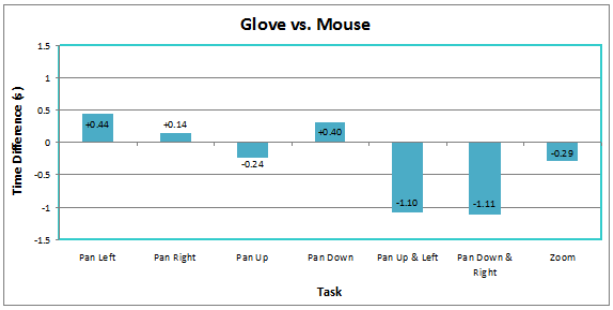
Photo Task Completion Time Differences



+ indicates glove perfomed slower than mouse

5.2 Viewpoint Tasks

Viewpoint Task Completion Time Differences



+ indicates glove perfomed slower than mouse

6 Analysis

6.1 Single-Handed Dragging

In single-handed cursor dragging gestures, the glove interface consistently spent more time acquiring and dragging the required points than the mouse interface.

6.2 Double-Handed Dragging

Double-handed gestures showed a significant time improvement over single-handed gestures, which can be attributed to the availability of two cursors to drag anywhere on the photo rather than at the

photos corners. This larger target area reduced acquisition time. Dual cursors also doubled the speed of the command execution.

The double-handed rotation gesture outperformed the mouse in both acquisition and manipulation time. This can be attributed to (1) the larger target area, and (2) a more natural up-down motion as opposed to an in-out motion.

Overcorrection and poor photo corner acquisition times with single-handed control have led to the conclusion that the glove interface cannot match the accuracy of the mouse interface, at least with its current sensitivity level. A wider camera lens and more operational space would be required to decrease the sensitivity and increase the accuracy.

6.3 Horizontal and Vertical Panning

Task #4 (Pan left) and Task #5 (Pan right) both required more time to complete with the glove interface than with the mouse interface. Command acquisition with the mouse interface was substantially quicker in Task #4 than in Task #5 because the scrollbar slider was much closer to the task start button.

Task #6 (Pan up) was the only task of the four individual panning tasks that the glove interface completed faster than the mouse interface; there was a significantly faster acquisition time (23.5%) and only a slightly slower manipulation time (8.0%). On the other hand, Task #7 (Pan down) had the second-largest positive time difference.

Because the task start buttons are located at the bottom of the screen, the user can instantly activate panning with the glove interface and pan up. To pan down however, the user must move the cursor to the upper half of the screen in order to give enough room to pan down. It is clear that in all directions, the actual execution of the pan command took more time with the glove than with the mouse. The difference is in the distance between the cursor starting point and the command activation point.

The time needed to activate a command is reduced whenever the amount of motion required to activate it is reduced. This occurred with the mouse interface in Task #4 (Pan left) when the task start button and the scrollbar slider were very close together, and with the glove interface in Task #6 (Pan up) when the three-finger grab could be executed instantly after clicking the task start button.

The glove interface has an extra advantage in that the three-finger grab and drag gesture can be performed in the same fluid motion. This is exemplified in Task #6 when the viewpoint manipulation time was also reduced, since the cursor was already in the correct position for command activation and execution.

6.4 Composition Panning

Task #8 (Pan up and left) and Task #9 (Pan down and right) took significantly less time to execute (21.8% and 17.0%, respectively) with the glove interface than with the mouse interface. This is simply because of how the pan command was defined. The glove interface can pan horizontally and vertically simultaneously. The mouse interface must pan horizontally and vertically separately. This results in a slower acquisition and manipulation time for the mouse interface.

6.5 Zoom

Task #10 (Zoom) took notably less time to activate and execute with the glove interface than with the mouse interface. The shorter command acquisition time can be attributed to the larger target area of performing two three-finger grabs anywhere on-screen with the glove interface, compared to maneuvering the mouse cursor to the left scrollbar. The zoom command took less time to perform as well since the simultaneous movement of both gloves moves the scrollbar twice as fast as the movement of the scrollbar slider with a single mouse cursor.

6.6 Overall Assessment

After analyzing experimental data and observations, the following general advantages and disadvantages of the glove interface have been identified:

6.6.1 Advantages

1. Performs wirelessly at a distance
2. Requires fewer on-screen controls
3. Allows for the combination and simplification of multi-step commands
4. Allows for more naturally-defined commands

6.6.2 Disadvantages

1. Less sensitivity and accuracy control
2. Longer physical command execution time when command complexity matches
3. Time delay between gesture performance and command execution
4. Arm fatigue

7 Conclusion

This project featured a glove-based user interface that when used effectively, could match and even exceed the efficiency of the standard mouse interface, despite its disadvantages. The hardware was created from consumer accessible webcam, IR LED, and IR filter technology. The software bypassed complex machine vision techniques in favor of proving the effectiveness of gestural commands. Single-handed gestures which merely matched the complexity of the mouse commands took longer to complete and were less efficient than the mouse commands. Slower acquisition times, command execution times, and general inaccuracy were obstacles to the glove interfaces success. However, double-handed gestures were able to match and even exceed the speed at which a task could be completed. Multi-step commands for the mouse interface were reduced to single-step gestures for the glove interface and resulted in a substantial decrease in execution times.

For the implementation of a glove-based user interface, all of these findings point to the necessity of a focus on task completion. The major difference between the mouse and glove interfaces is the number of input points. When defined tasks for the glove interface took advantage of its multiple points of input, they were completed faster and in some cases, with fewer actions. Another difference between the mouse and glove interface is how the command matches what occurs on-screen. When the actions required in a gestural command for the glove interface matched the actions occurring on-screen more closely than the command for the mouse interface, an improvement in command execution speed also ensued. This confirms the need for a glove-based user interface to have a foundation in task completion and simplification in order to provide for a more natural and effective human-computer experience.

References

[1] Molina, Jose P., et al. 'The Development of Glove-Based Interfaces with the TRES-D Methodology.' *Virtual Reality Software and Technology.*, pp. 216-219, 2006.

[2] Lee, SK, William Buxton, and K. C. Smith. 'A Multi-Touch Three Dimensional Touch-Sensitive Tablet.' *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.* San Francisco, pp. 21-25, 1985.

[3] Kim, Ji-Sun, et al. 'iPhone/iPod Touch as Input Devices for Navigation in Immersive Virtual Environments.' *Virtual Reality Conference, 2009. VR 2009. IEEE.*, Lafayette, pp. 261-262, 2009.

[4] Izadi, Shahram, et al. 'ThinSight: A Thin Form-Factor Interactive Surface Technology.' *Communications of the ACM 52.12*, pp. 90-98, 2009.

[5] Sturman, David J., and David Zeltzer. 'A Survey of Glove-based Input.' *Computer Graphics and Applications, IEEE.*, pp. 30-39, 1994.

[6] Baek, SeongHo, et al. 'IRED Gun: Infrared LED Tracking System for Game Interface.' *Lecture Notes in Computer Science 3768/?2005.*, pp 688-699, 2005.

[7] Wobbrock, Jacob O., Meredith Ringel Morris, and Andrew D. Wilson. 'User-Defined Gestures for Surface Computing.' *Proceedings of the 27th International Conference on Human Factors in Computing Systems.* Boston, pp. 1083-1092, 2009.