Linear Gesture Recognition and Large Screen Simulation of Gesture Select

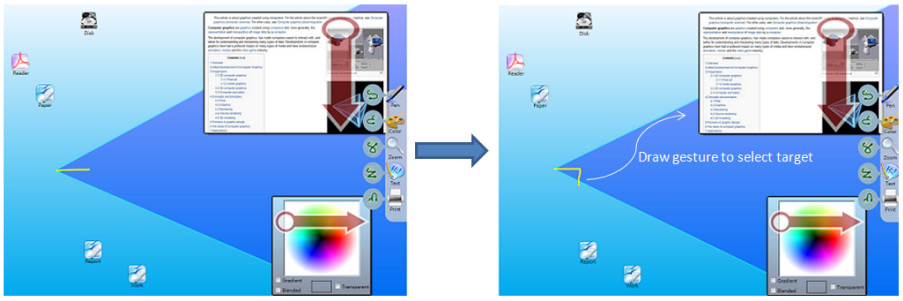
Hsu-Sheng Ko

Jan 8, 2010

1 Abstract

Gesture Select is a new technique for selecting remote targets on large screen.

Instead of walking to distant target and selecting it, users draw an initial stroke with a direction toward the target. This would create a region of interest in which every target would be assigned a linear gesture. Users continue on drawing the correspondent gesture of the target to have it selected.



Draw initial stroke Continue drawing gesture

To implement Gesture Select, there are two primary technical problems :

1. Linear gesture recognition :

Gesture Select needs a robust linear gesture recognizer that supports sufficient gestures. These gestures should be simple enough so that the user response time – the time spent on recognizing and completing the gesture – could be lower than 2.5 seconds. This recognizer should also be stable: the error rate must be in a reasonable range.

2. Large screen simulation :

Since there's no large screen device available on campus, we need to simulate a large screen by connecting two projectors and one smart board. A communication software runs between these devices to synchronize the input from users.

2 Linear Gesture Recognition

Algorithm Design:

A linear gesture is composed of a series of input points recorded by input device. Since parts of this algorithm relies on vector-based computation, input points will be translated to a series of vectors. The final output is a series of vectors with different directions. We call these vectors with different directions 'segment' in the following article.(Fig. 1)

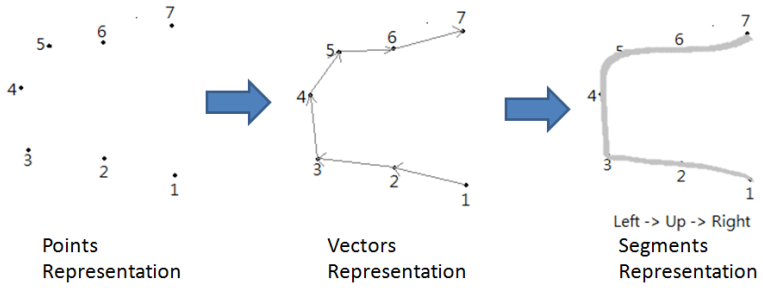


Figure 1

This gesture recognition algorithm contains 6 phases :

1. Resample and vector translation

Devices tend to record some extremely close points which can be considered as noise (This is probably caused by slight moves done by users unconsciously). In order to get rid of this noise, we resample the inputs points to gain a new series of points in which distance between two points is larger than a specific threshold. This new series of samples is then translated to vectors.

2. Curve Detection

A curve is detected by comparing angles between vectors within a specific distance. If the angle is larger than a specific value, there's a curve and these two vectors possibly belong to different segment.(Fig. 2) Note that the specific

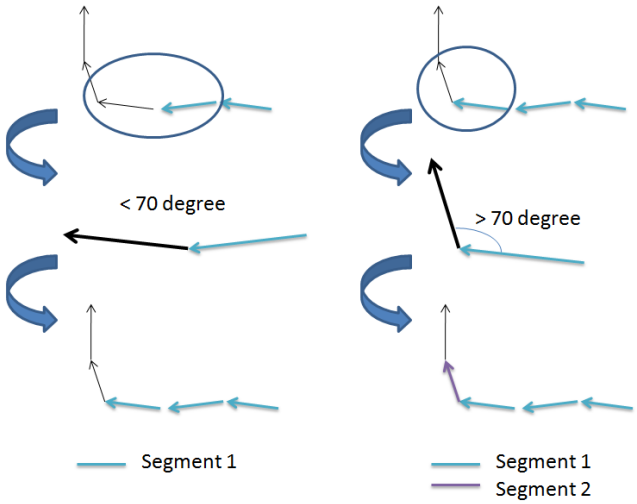


Figure 2

distance is defined as a relative value to the length of whole gesture. This is because different user tends to draw the same gesture in different size.

3. Erroneous Input Gesture Correction

Users may make mistakes shown in Fig. 3. To gain better user response time and lower error rate, the recognizer must be able to tolerate these errors. Again, the threshold is a relative value to the length of whole gesture. Those segments with shorter length would be removed.

4. Recognition

Based on the segments given by the previous phase, segment direction could be simply defined by computing the angle between segment and the x-axis and comparing this angle with angle ranges of different directions. The comparison logic can be easily modified to support four-direction recognition or eight-direction recognition.

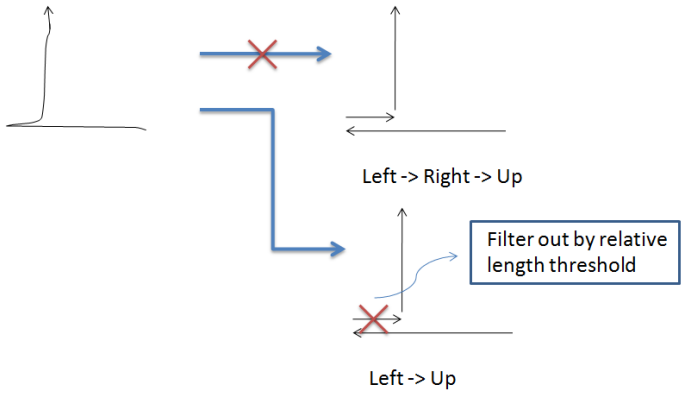


Figure. 3

5. Integration

In the Curvature Detection phase, for better sensitivity, the detection distance is set to a large value, and the threshold angle is set to a small value. This would cause the problem shown in Fig. 4. Since the continuous segment has the same direction, it should be considered as only one segment.

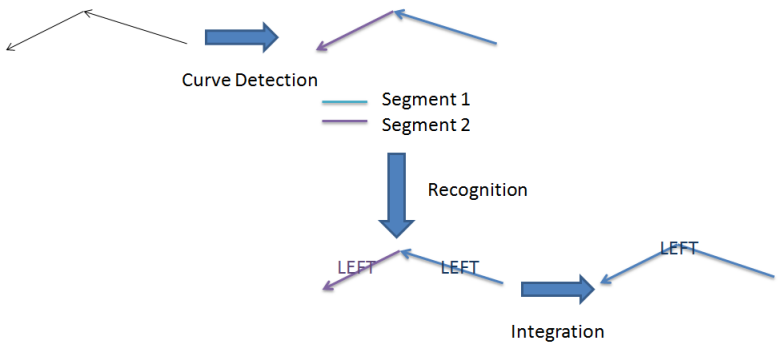


Figure. 4

6. Exception Handling

This phase deals with some erroneous cases which can't be detected in the previous phases.

If the user draws a very smooth curve as Fig. 5, the recognizer may fail to

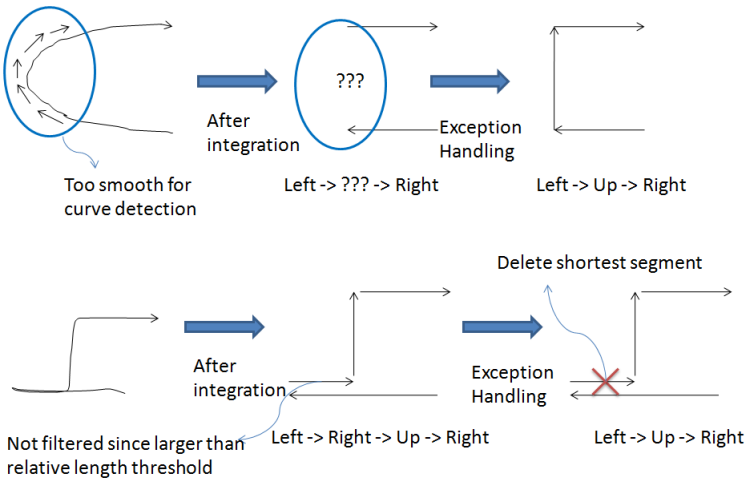
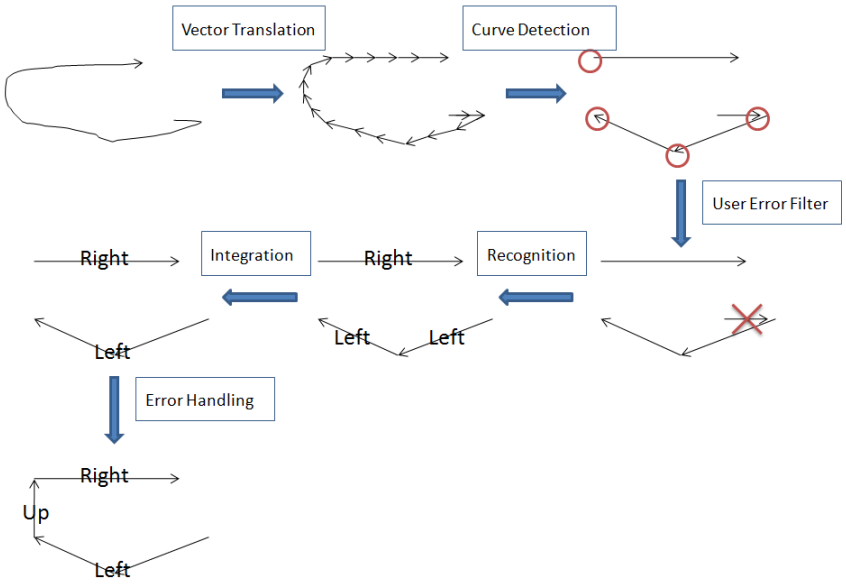


Figure. 5

recognize it in curve detection phase and only finds two vectors and a blank between them. For this case, we could assume there's something in-between, which is a 'UP' segment.

Consider also the case shown in the same figure. The recognizer might return a recognition result with 4 segments. Assume our recognizer only supports a set of 3-segments gestures, the shortest segment would be removed.



Flow Chart of Algorithm 1

The above figure shows all phases of this algorithm and an example of how a linear gesture is recognized.

Tuning Algorithm :

Before starting the tuning process, we need some real gesture data from users. We conducted a few user experiments (described in next section) and recorded the results. For those gestures failed to be recognized, we run a visual tool to help us find out the reason (Fig. 7). If it's a mistake made by the user, it can be ignored. If it's a problem that could be solved by modifying algorithm or adjusting parameters defined in algorithm, we do the correction. However, sometimes, these changes may solve the current case by produce more error cases. Therefore, after doing some corrections, a program would be executed to input all the gesture data to the recognition algorithm and output the average error rate and user response time.

By continuously doing corrections and comparing the error rate and user response time, finally, we will have a robust gesture recognizer.

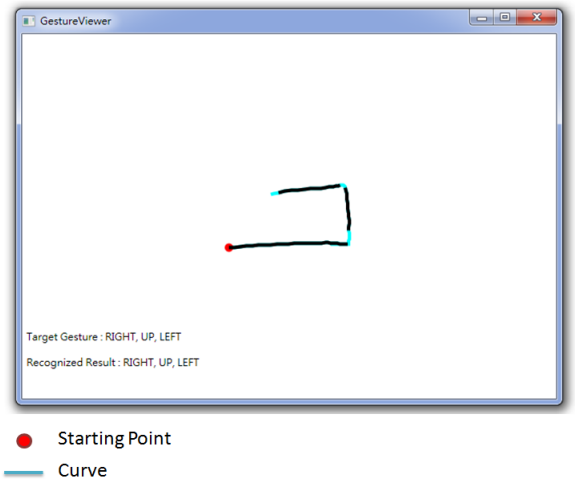


Figure 7

Experiment Design :

We conducted two experiments with different variables to inspect the performance of Gesture Select under various circumstance :

Experiment 1

This experiment has three variables : target distance (46", 68.4", 90"), target size (2" x 2", 4" x 4", 6" x 6"), and distractor count (16, 32, 64).

We have :

10 participants
x 4 blocks (1 training + 3 measured)
x 3 distances
x 3 widths
x 3 distractor counts
x 4 trials
= 4,320 trials completed

Experiment 2

This experiment has three variables: target distance (63.3", 78.2"), direction (E, NE, SE, W, SW, NW), and path distractor density (0%, 40%, 80%).

We have :

10 participants
x 4 blocks (1 training + 3 measured)
x 2 distances
x 6 directions

x 3 path distractor densities
= 1,440 trials completed

Result :

The result of above experiments shows that target distances, target size, direction and number of distractors has very small influence on the user response time and error rate of Gesture Select.

1. Eight-directional recognition :

The average user error rate of a Eight-directional recognizer is higher than 15%, which is not acceptable. We found that users can't distinguish between these eight directions accurately. The average user response time is 2.2 second, which is acceptable.

2. Four-directional recognition :

Four-directional recognizer performs much better than an eight-directional one. The average user error rate is 8%. Among these errors, only 35.3% are caused by failure of recognition. Others are wrong gestures done by users.

3 Large Screen Simulation

First, we tried to run MaxiVista - a screen sharing software to extend the screen from smart board to two projectors, but there's a severe lag problem.

We then came up with another solution. The idea is to have the large screen application running on all the computers connected to the output devices and synchronize the inputs. This would create several copies on the large screen application running on different computer. We only need to apply different translation on computers to show different part of the large screen application. To synchronize the inputs, we have a server program running on the computer connected to the smart board. This server would send all the inputs received from users to all the clients (two computers connected to the two projectors).

However, if the server sends all mouse events to two clients, there would be too many packets flown through the network and thus causes lag between server and clients. A tricky way to solve this is rather than sending all mouse events to clients, the server only send the recognized results to clients. Since server is the only computer connected to the smart board and could receive the gesture drawn by users, this would generate exact the same effect as sending all mouse

inputs to two clients. The following figure shows how the entire solution works.