Multi-touch displays: design, applications and performance evaluation

L.Y.L. Muller

Grid Computing
Section Computational Science
Universiteit van Amsterdam

July 2, 2008

Introduction

What is multi-touch?

- Interactive graphical device.
- Combines camera and tactile technologies.
- Allows multi-user and multi-touch input.

Why use multi-touch?

- Direct on-screen manipulation.
- Gesture based interaction.
- Multi-user task solving.
- Allows "natural interaction" with a computer.

Overview of this work

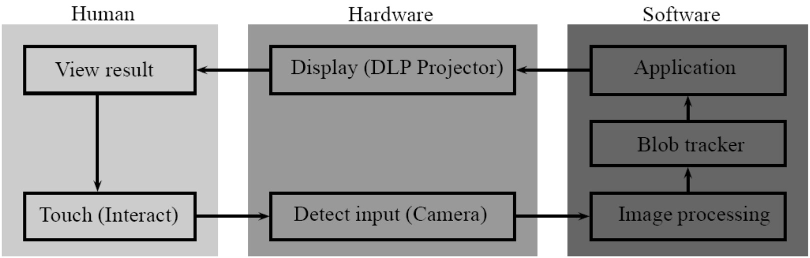
- The design and construction of a camera based multi-touch device.
- The design and implementation of a gestural interaction library.
- Implementation of test case applications.
- Performance evaluation of multi-touch enabled tasks.

Constructing a multi-touch device

Requirements

- NEMO Science Center
  - NEMO encourages the audience to participate in experiments, the system should be attractive.
  - The system needs to be suitable for an audience from 7 up to 70 years.
  - The system should encourage users to playing together.
  - The system needs to be a standalone device.
  - The hardware needs to be ‘child proof’.
- Universiteit van Amsterdam
  - Construct a reliable multi-touch panel.
  - Allow multiple users to collaboratively solve tasks.

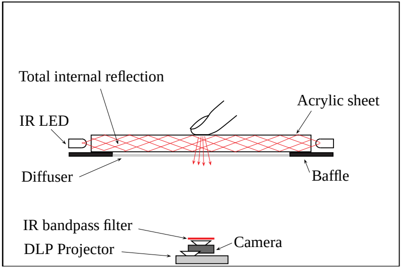
Camera based multi-touch device pipeline

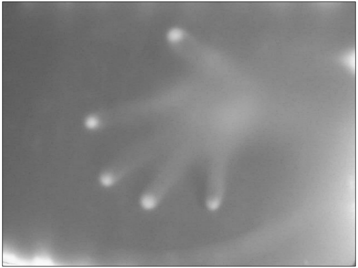


Camera based multi-touch techniques

- Frustrated Total Internal Reflection (FTIR)
- Diffused Illumination (DI)
  - Rear-side Illumination (RI)
  - Front-side Illumination (FI)

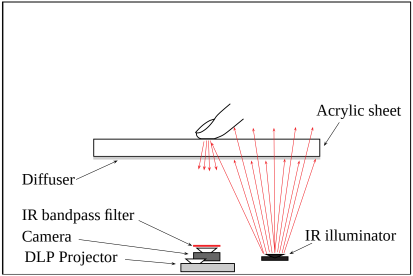
Frustrated Total Internal Reflection (FTIR)

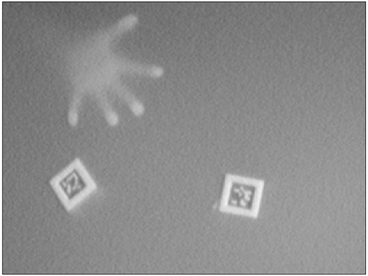




- Presented by Jeff Han (NYU) in 2005.
- Based on research from the early 80s.
- High contrast.
- Reliable finger tracking.

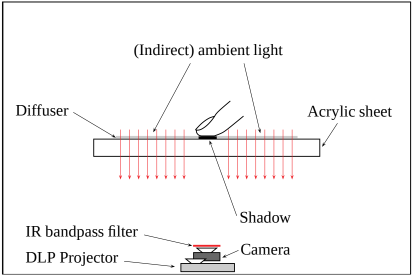
Rear-side Illumination (RI)

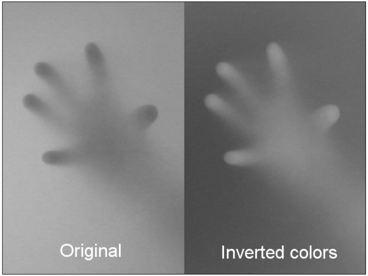




- Based on the technique used in the HoloWall (1997) and MS Surface (2007).
- Compared to FTIR it is easier to construct.
- Allows object tracking with fiducial markers.
- Reliable finger tracking.

Front-side Illumination (FI)



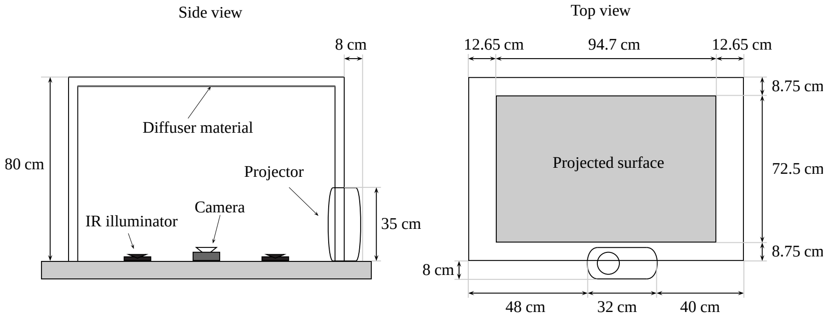


- Requires (stable) ambient light.
- Cheap and easy to construct.
- Less reliable than FTIR or RI.
- Does not allow object tracking.

Comparisons table

Comparison overview			
Item	**FTIR**	**RI**	**FI**
Component costs	High	Medium	Low
Construction complexity	High	Medium	Low
Closed box required	No	Yes	No
Blob contrast	Strong	Average	Average
Software tracking complexity	Low	Medium	Medium
Reliable finger tracking	High	High	Low
Allows object tracking (pencil)	Yes	No	No
Allows object tracking with fiducials	No	Yes	No
Influence of ambient light	Low	High	High

Hardware description



Hardware photos

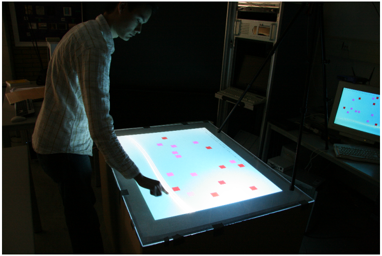


Figure: Using the multi-touch table.

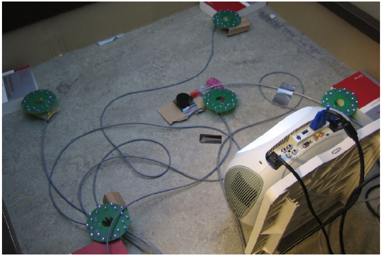


Figure: Inside the multi-touch table.

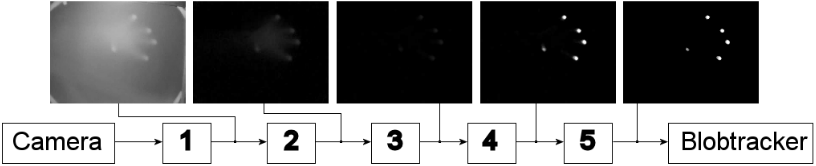
Multi-touch detection and processing

Multi-touch framework

Touchlib

- Open source multi-touch framework
- Cross platform (Windows, Linux, Mac OS X)
- Provides:
  - Video image processing (using Intel OpenCV, computer vision library).
  - Blob detection and tracking.

Video image processing



1. Capture filter
2. Background filter
3. Highpass filter
4. Scaler filter
5. Rectify filter

(Optional) Lens correction

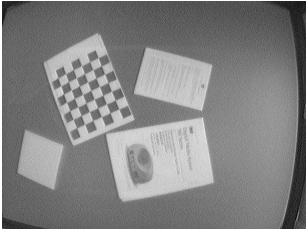


Figure: Distorted image.

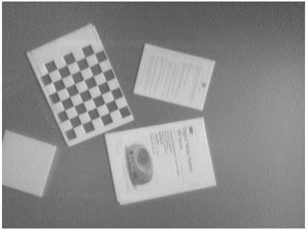


Figure: Undistorted image.

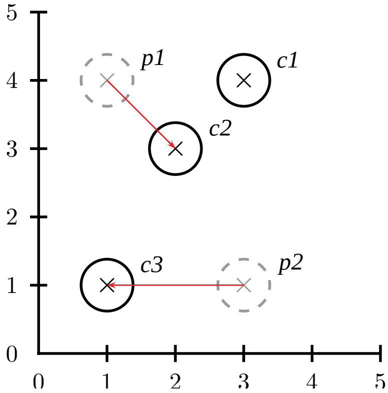
- Our system uses a wide-angle micro lens which suffers from radial distortion (barrel distortion).
- Correcting algorithm uses OpenCV functions to correct the camera image.

Blob detection and tracking I

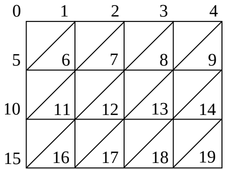
- Finding contours using *cvFindContours*.
- Position of the blobs are stored for each frame.

Blob detection and tracking II

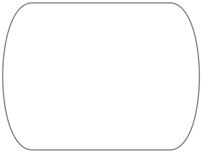
- Blobtracker generates a transition matrix.
- New blobs contain a distance value of zero.
- The blobtracker tries to find a state with the lowest distance value.



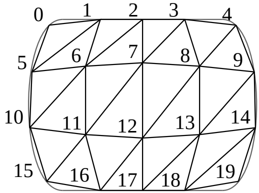
Position correction I



(a) Calibration grid



(b) Camera frame



Position correction II

(a) Distorted (b) Corrected

- Barycentric coordinates named α, β and γ.
- Represents normalized values of the areas of the subtriangles.
- $\alpha + \beta + \gamma = 1$.

Generating events

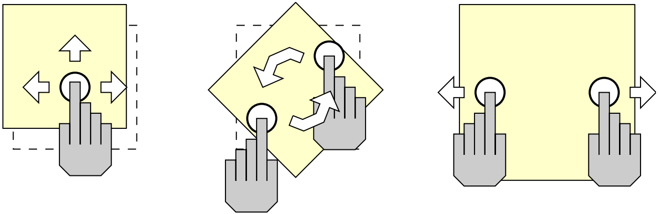
Three possible events:

- Touch down
- Touch update
- Touch up

Gesture classification

- Direct gestures
- Symbolic gestures

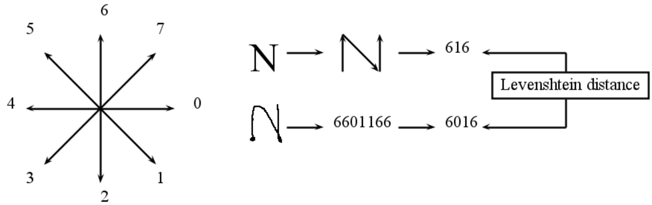
Direct gestures



(a) Translate (b) Rotate (c) Scale

- Direct gestures are implemented in the application.

Gesturelib I



- Each gesture is defined by an 8-direction sequence.
- The gestures are stored in a database.
- A Levenshtein distance is calculated between the captured sequence and the entries in the database.
- Based on the Levenshtein cost a candidate is selected.

Gesturelib II

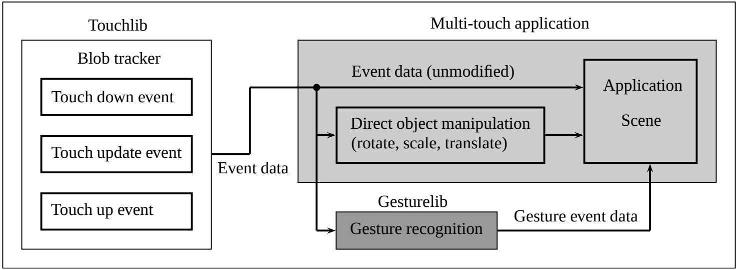
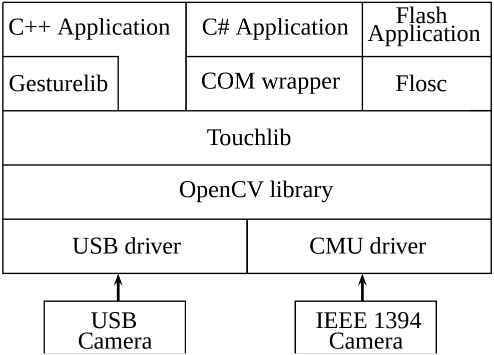


Figure: Schematic view of the events pipeline using Touchlib and Gesturelib in an application.

Multi-touch applications

Multi-touch application design



Real-time Fluid Simulation

Multi-user environment providing direct scene manipulation.

RTFD_short.wmv

Play Stop

Multi-touch Media Application

Multi-user photo and video viewer.

MMA_short.wmv

Play Stop

NASA World Wind

Gesture based interaction.

NASA_short.wmv

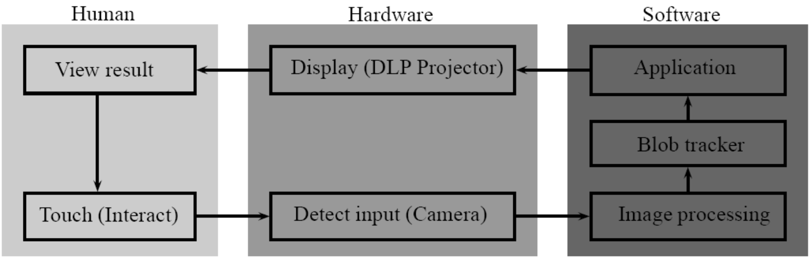
Play Stop

Performance measurements

Hypotheses

1. The performance of a task on a multi-touch device depends on the performance of the used hardware.
2. Some tasks can be performed faster on a multi-touch device than a mouse.
3. Collaboration on a multi-touch device increases task performance.

System latency



1. Camera
2. Touchlib
3. Application
4. Digital projector

System latency - Touchlib

Filter type	No active blobs		Five active blobs	
CMU capture	3.351 ms	10.49%	3.048 ms	9.63%
Background removal	0.569 ms	1.78%	0.565 ms	1.78%
Simple highpass	4.751 ms	14.87%	4.788 ms	15.12%
Scaler	2.767 ms	8.66%	2.806 ms	8.86%
Barrel distortion correction	19.962 ms	62.49%	19.913 ms	62.90%
Rectify	0.544 ms	1.70%	0.538 ms	1.70%
Total filter time	**31.944 ms**	**100%**	**31.658 ms**	**100%**
Finding blobs	1.276 ms	99.61%	1.491 ms	94.97%
Tracking blobs	0.004 ms	0.31%	0.061 ms	3.89%
Dispatching events	0.001 ms	0.08%	0.018 ms	1.15%
Total tracker time	**1.280 ms**	**100%**	**1.570 ms**	**100%**
Total Touchlib time	**33.225 ms**		**33.228 ms**	

Table: Touchlib image processing and blob tracker latency results.

note: Results exclude the touchlib ‘bug’ causing the image processing pipeline to stall for 32 ms.

System latency - Digital projector

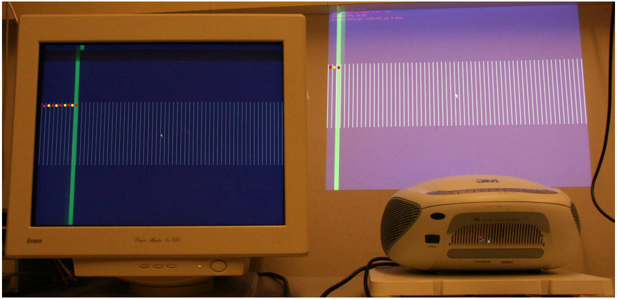


Figure: Comparing the latency of the digital projector with a CRT monitor using the latency tool.

- Comparing the position of the green bar.
- Each section is equal to 16 ms.
- Six frames delay (100 ms).

System latency - Total system latency

Used projector	**3M DMS 700**	**3M DMS 700** (improved touchlib)	**Sharp PG-A10X** (improved touchlib)
FireWire Camera	30 ms	30 ms	30 ms
Touchlib	33 ms	33 ms	33 ms
Touchlib 'bug'	32 ms	0 ms	0 ms
Digital projector	100 ms	100 ms	0 ms
Total latency	195 ms	163 ms	65 ms

Table: Comparing the total system latency of different hardware and software combinations. Latency time is measured in milliseconds.

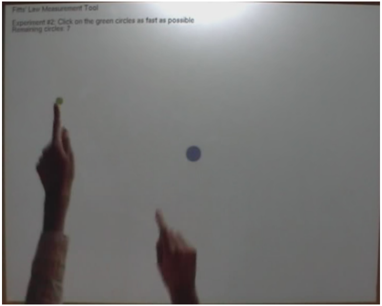
note: The camera latency values are provided by manufacturer specifications.

Experiment design

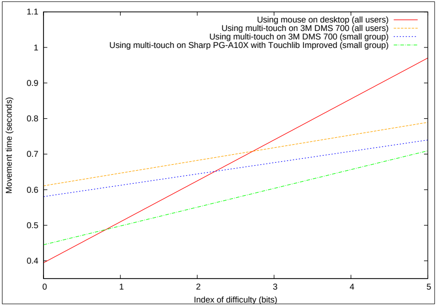
- 22 test persons (4 female and 18 male).
- Used input devices:
  - A Logitech standard mouse with three buttons.
  - A multi-touch table using RI (3M / Sharp).
- Two experiments:
  - A Fitts' Law model.
  - Gesture based manipulation.

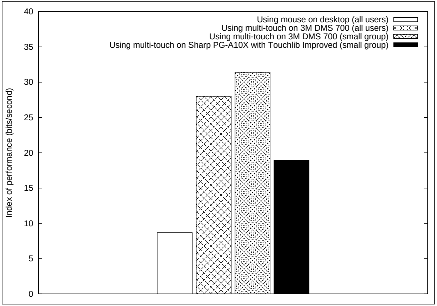
Experiment 1: Fitts' Law

- Prediction model of movement time.
- Difficulty depends on width and distance.
- 4 different widths.
- 5 different distances.
- 20 objects per test (40 objects in total).



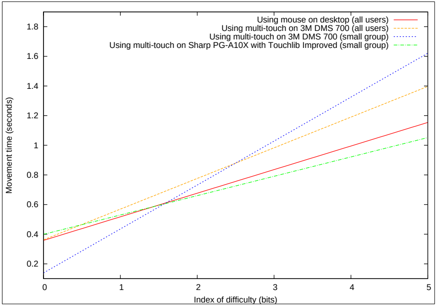
Results - Fitts' Law 1D test

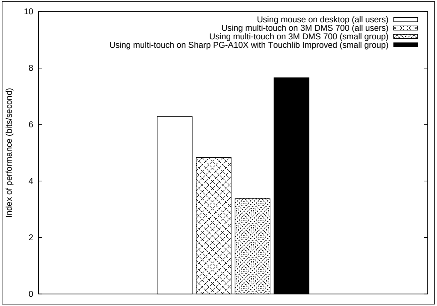




- Index of difficulty is calculated from the target distance and width.
- Index of performance is calculated from the model fit slope results.
- The multi-touch device outperforms the mouse device.

Results - Fitts' Law 2D test

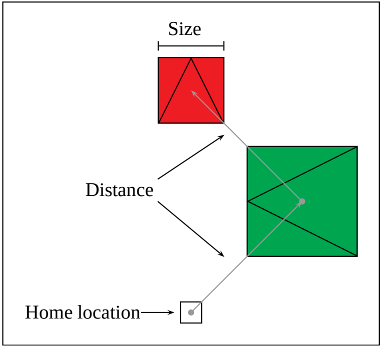




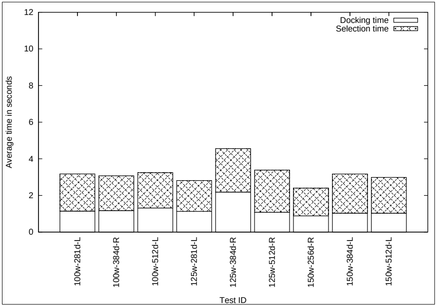
- Index of difficulty is calculated from the target distance and width.
- Index of performance is calculated from the model fit slope results.
- Only the multi-touch device using the Sharp projector outperforms the mouse device.

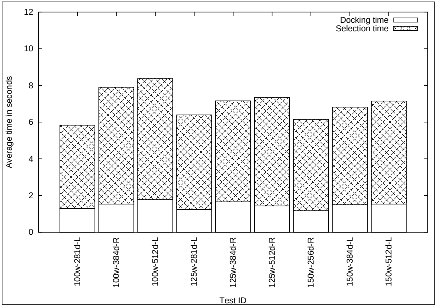
Experiment 2: Object manipulation

- Comparing mouse manipulation with gesture based manipulation.
- Using the direct gestures set (Rotate, Scale and Translate).
- 3 different sizes.
- 3 different scales.
- 3 different angles.
- 3 different distances.



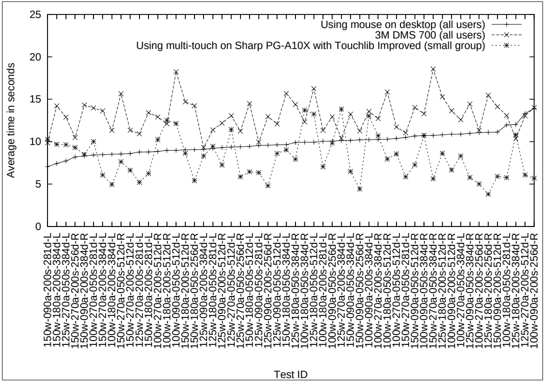
Results - Object manipulation I





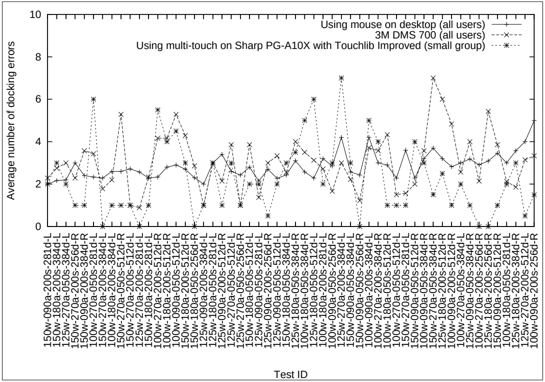
- Completion times are lower when using the mouse.
- Multi-touch requires more time for docking (precision)

Results - Object manipulation II



- Sorted by mouse task difficulty.
- Impact of the used projector and system latency.

Results - Object manipulation III



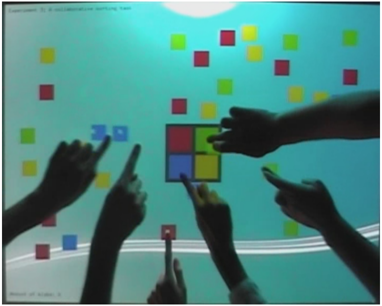
- Docking errors
- Sorted by mouse task difficulty.

Experiment design

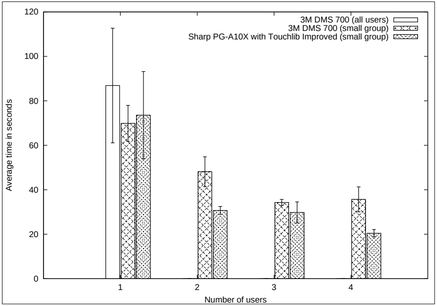
- 22 test persons for single user session.
- Multi-user sessions are performed by the 'small group' users.
- Two experiments:
  - Sorting task.
  - Point and selection task.

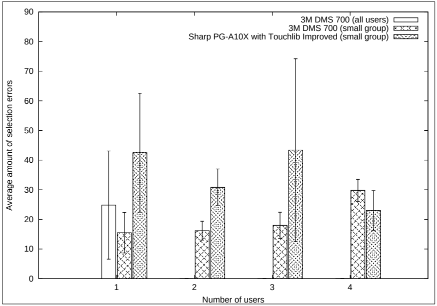
Experiment 3: Sorting task

- Sorting colored tiles.
- 40 objects.
- 4 colors.
- 4 containers.



Results - Sorting task I





- Task completion time.
- Impact of the used projector.

Results - Sorting task II

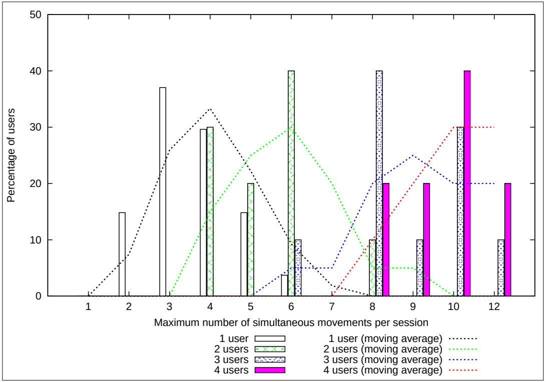
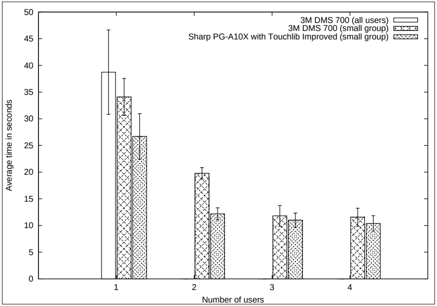


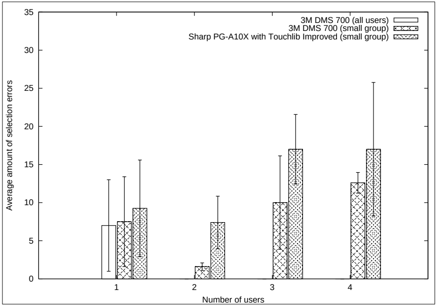
Figure: A comparison of parallel activity with different number of users.

Experiment 4: Point and selecting task

- Colors need to be selected in the right order.
- 50 objects.
- 5 different colors.

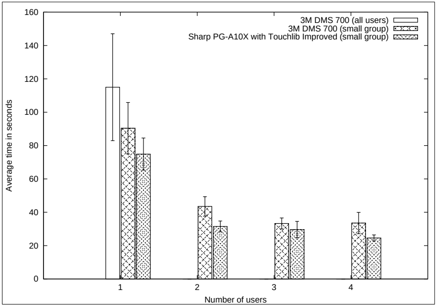
Results - Point and selecting task I

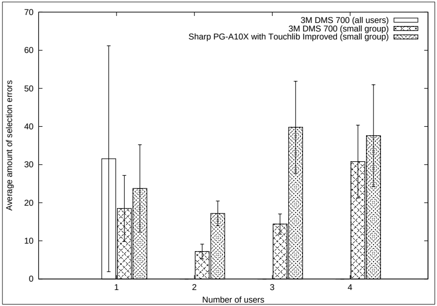




- Pointing task.

Results - Point and selecting task II





- Selection task.
- Surface friction.

Conclusion and Future work

Conclusion

- System performance depends on used hardware (precision, reliability, responsiveness).
- Demo applications show how new and existing applications can benefit from multi-touch.
- The multi-touch device should not be considered as a replacement for a mouse device (precision).
- Collaboration shows significant improvements when increasing the number of users.

Future work

- Optimizing multi-touch software:
  - Perform barrel distortion correction only on touched position.
  - Improving the blob tracker algorithm.
  - Using GPUCV (GPU port of OpenCV) to perform image processing.
- Optimizing used hardware:
  - Improving touch surface material (reducing friction).
  - Using a faster camera.

The End

multitouch-videomix1.wmv?