Human interfaces - Finger tracking applications

L. Vlaming

university of groningen

Department of Computer Science
University of Groningen

July 4, 2008

Contents

Acknowledgements

I would like to thank the following people, without them the project would have been a lot more difficult:

- Tobias Isenberg
  For guiding the project, and creating the gloves.
- Jasper Smit
  For writing the library
- Wouter Lueks
  For creating the final version of the LED array
- Dincla Plomp
  For providing the retro-reflective sheet
- Gieneke Hummel
  For testing the setup each time, without losing patience.
- Arnette Vogelaar
  For creating the first version of the fingertip retro-reflective sheet items.

Chapter 1

Introduction

Interaction for humans with machines is different than interaction with other humans. In time, a diverse group of devices has been created with different ways to interact with them. For interaction with most man made devices, the designer tries to create something that works as intuitively as possible. Until now, interacting with a computer has been done with some standard equipment, a mouse and a keyboard. This project aims to provide a different way to interact with (part) of a computer.

The focus for this project, is to try to create a new way for interacting with the computer using cheap hardware. Here the idea from a project called "Tracking your fingers with the Wiimote" from Lee [19] is used to give the computer the ability to "see" the fingers of a user. The innovation of this project is in the part where the interaction with the computer is created.

The reason to try to create this, is to make a more intuitive way of interacting with the computer available. Another reason is that to create the interaction, some new algorithms need to be thought of.

This interaction is realized through passive motion tracking, using a (in comparison to other hardware) very cheap Wii Remote, infrared LEDs, a power supply, and some reflective clothing normally used in traffic.

Chapter 2

Related work

The ideas used for this project originate from motion tracking and motion gestures. For this project motion capture is used to capture the movement and the position of our fingers, to interact with the computer. This has the advantage that no physical touching is needed in order to register where a finger is.

I will talk about most of the techniques briefly in the next sections.

2.1 Motion capture

Motion tracking started as an analysis tool for biomechanics, expanded to sports and training, and was recently adopted for video games and computer animations in films.

Motion tracking is today mainly used for creating animated versions of an actor. This has the advantage that the visual appearance of the actor in the final product can be changed quite easily while giving it a very realistic and believable way of moving. For example in the “the matrix” movie, the punch the head actor gives the bad guy [12] was completely computer animated. Also the movies “Happy feet”,“The Polar Express” and ”Beowulf” relied heavily on motion capture [26, 22].

There are also applications in the medical world, virtual reality, biomechanics, sports and training. For virtual reality motion tracking is most times used for CAD applications, but is used mostly for its abilities to show objects in 3D. In biomechanics, sports and training, real-time data can be used to diagnose problems, for example how to increase performance, or the best way to start in a skate contest [20].

To collect the information about position and movement of objects, several techniques were developed over the years:

- Optical systems
- Inertial systems
- Mechanical systems
- Magnetic systems

In general, when doing a motion tracking session, the position of certain points in an area are sampled very frequently each second. These points are then stored somewhere, could be transformed, and then mapped to a digital version of the subject. Today, most effort put in these systems is used to be able to do real-time processing of the captured data.

Most systems used today are optical systems, sometimes with addition of inertial, mechanical or magnetic systems.

2.1.1 Optical systems

Optical motion tracking systems use cameras from different angles to capture points placed on an object, and to triangulate the 3D position of the points. Several cameras are used to provide overlapping projections, and to be able to track a point even when occluded for a certain camera view. This technique provides 3 dimensions per marker, other dimensions like rotation can be inferred from the relative position of two or three markers. Using optical systems, there are passive and active ways to identify the points on a performer.

Passive markers

With passive markers, the performer can wear a suit containing a lot of retro reflective markers(see Fig 2.1), so when a lightsource shining in the view direction of the camera, the light is reflected back into the camera. This is called a passive optical system, because no direct light is sent from the performer, light is only reflected back. This way very simple reflecting material can be used to triangulate the points. Most times with this technique, infrared light is used because it does not interfere with regular light sources.

The advantages with this system (over active marker systems and magnetic systems) is that the performer does not need to wear wires or other electronic fragile equipment. Instead a lot of very cheap and easy to replace markers are put on the performer. This system can capture a lot of markers at very high framerates, and the performer is able to do all the movements he does normally without constraints.

One problem is that all markers appear identical. This also gives problems when a marker disappears (is occluded by the performers body for example) from the camera view. When a new marker appears, it is not known for sure which marker it is.

One way to partially solve this problem is to have cameras from many angles, trying to always capture the markers when the performer moves. Typical professional systems range from $50,000 to $100,000. Companies that sell these kind of systems are Vicon [5], OptiTrack [4] and MCTCameras [2].

Active markers

Active marker systems using active lightsouces attached to the performer (see Fig 2.2, illuminating one light source or multiple distinguishable lightsources at a time. Because active lightsources are used, bigger recording scenes can be used. Because active lightsources are used, each performer needs a power source, limiting the performer in the freedom of movement. Active marker systems are more used in real-time systems because less computation is needed to triangulate each marker. There is a trade-off between the number of distinguishable lightsources flashing at the same time and the final framerate.

Markerless

New computer vision techniques have lead to the development of markerless motion capturing. These systems do not require the subjects to wear special suits. Instead, computer algorithms are used to identify the human bodies, and break these into certain parts which are tracked. Sometimes special colored clothing is required to identify certain parts of the body. One system was recently featured during Intel CEO Paul Otellini's keynote address at the 2008 Consumer Electronics Show in Las Vegas (see Fig 2.3). During the demonstration, a singer performed live while tracking data generated in realtime by the markerless system was instantaneously rendered into a garage scene. These systems work well with large motions, but have problems with small motions and parts, like fingers.

Tracking faces

Most motion capture systems have problems capturing the natural way in which faces move. This is because, for example, the movement of the eyes is not captured well. The company Com [3] has created a new way to capture

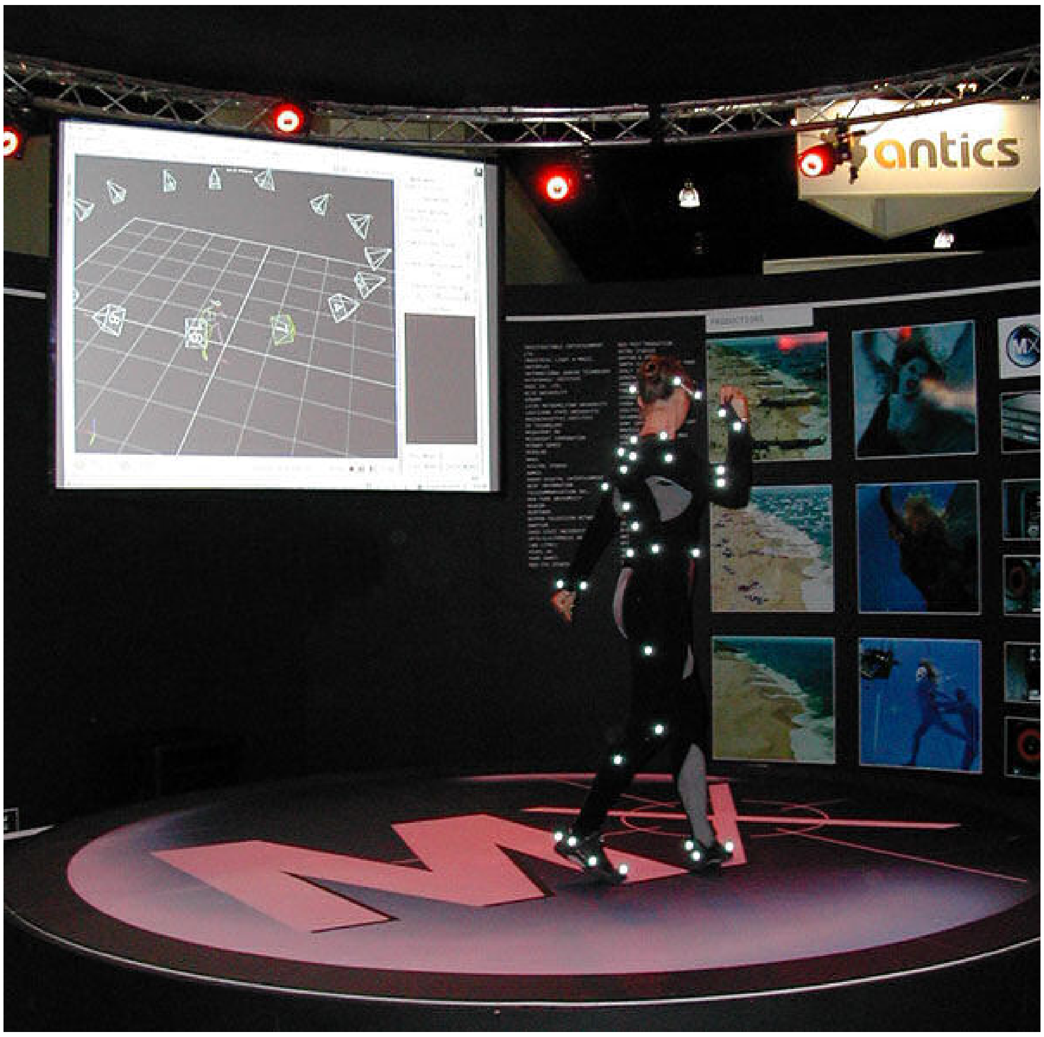


Fig. 2.1: Passive motion capture system, with one actor wearing a suit with passive markers. View from one of the cameras.

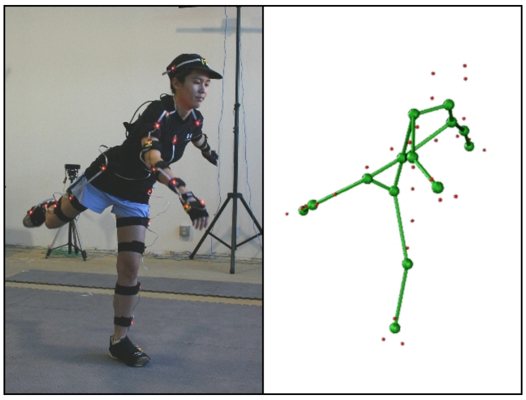
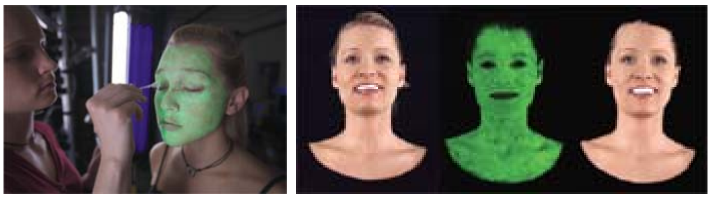


Fig. 2.2: Active motion capture system, with one actor wearing a suit with active markers. View from one of the cameras.

Fig. 2.3: Demonstration of Organic Motion's markerless motion capture featuring Steve Harwell of the band Smash Mouth during Intel keynote at CES 2008.

face movement [23], which they featured at SIGGRAPH 2006. Special phosphor makeup is applied to the performer, and with a 1.3 megapixel camera the movement of the face is followed and evaluated. This is done with a flashlight, flashing between 90 and 120 times per second. When there is no flash, the phosphorescent makeup glows, and digital cameras can capture the performer. This results in a high-quality 3D model (see Fig 2.4a and 2.4b).



(a) Applying the phosphorescent makeup. (b) Before the makup, the seen picture, and the reconstructed image.

Fig. 2.4: Phosphorescent makup.

Another company [3] doing face motion capturing [21] uses the physics behind facial movement to reconstruct a model. They try to give the computer an understanding of what is happening. The system works with one camera, but can use more cameras to provide more flexibility. The process uses the principle that one can contract a face muscle, which affects the eyes and mouth. They fit data of 28 groups of contractions into a model. This way they can map the facial movement on very different looking characters.

Lasers

One very specialized way to track an object is to track an object with a laser [13]. The approach used in the uses a laser, a laser diode and a few steering mirrors to track the object. This is another active way of tracking, but it does not use a camera. One problem here is that the current maximum distance of the tracking object may only be around 160cm, and that it does not use cheap hardware (a laser is needed) (see Fig 2.5).

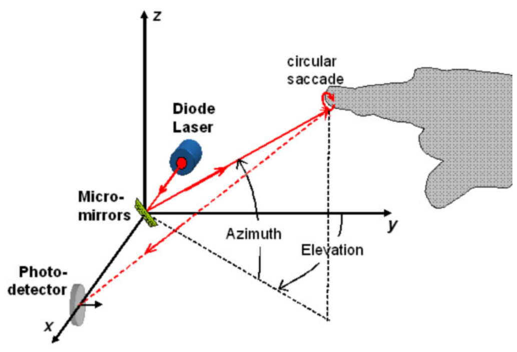


Fig. 2.5: The experimental setup the researchers use.

2.1.2 Intertial systems

Inertial motion capture systems [24] are based on inertial sensors. This type of systems can capture the full six degrees of freedom body motion in real-time, resulting in better models. The data is most times transmitted to a computer wirelessly. With these systems, no cameras, markers or emitters are needed to capture the motion. Another benefit is that there is no need for computation to get the correct data, no difficult camera setups are needed. Also large capture areas are possible, and there is a lot of freedom to move to any studio you want.

2.1.3 Mechanical systems

Mechanically motion capture systems are based on an exo-skeleton (see Fig 2.6), which captures the motion of each joint of the body. The exo-skeleton is a skeletal-like structure, which measures the position of each joint relative to another. These systems are real-time, relatively low-cost, and are wireless. Also there is no limit to the number of performers using an exo-skeleton, because there are no problems with occlusion.

Fig. 2.6: A setup where the position of the head is tracked.

2.1.4 Magnetic systems

Magnetic motion capture systems [24] use relative magnetic flux of three orthogonal coils on both the transmitter and each receiver (see Fig 2.7). The system can calculate position and orientation with this information, together with range and orientation. The markers do not have problem with nonmetallic object occlusion, but have problems with magnetic and electrical interference.

2.2 Multi touch interfaces

There are several multi-touch interfaces. Some of them are Microsoft Surface [7], the iPhone [10], CUBIT [17], “the Digital Desk” [27], DiamondTouch [6] and FTIR [14]. These multi-touch interfaces use different techniques to register multi-touch.

CUBIT, Microsoft Surface and “the Digital Desk” all use cameras to capture the positions of the fingers, although they use different techniques to make the fingertips visible to the cameras.

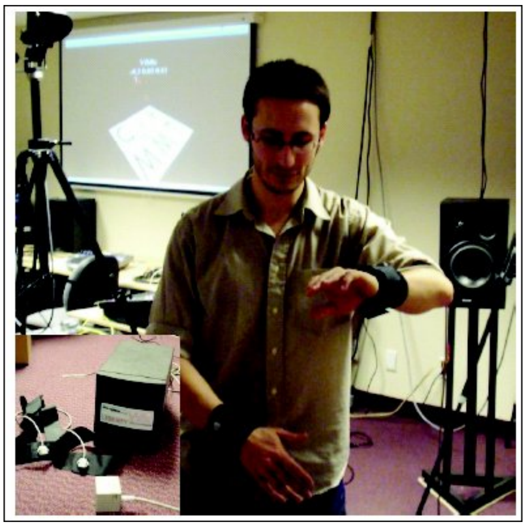


Fig. 2.7: A small setup where the arms are tracked.

Also more and more demonstrations are created using Johnny Lee's technique using a stable Wii Remote on top of the screen, and capturing the fingertips positions using infrared.

These multi-touch interfaces use different techniques to actually let a user interact with the interface. There are the screen type surfaces, which require users to actually touch the interface. When touching the interface, a point is registered, and the interface reacts.

Another way toget user input is by registering points in space with a camera. Because the points are then seen most of the time, there needs to be a special motion to register a touch. This can for example be a pinch action or a tap action.

Chapter 3

Multi touch interaction in free air

The idea for this project came from the Minority Report movie, combined with the idea from Lee [19]. The idea is to use a Wii Remote to track four points (the maximum a Wii Remote can track), pair these points so the program knows which points are from which hand, and then create multi touch interaction with this technique.

3.1 Concept

The Wii Remote can track four points. This is used to track four fingers in a 2D virtual plane. To create multi touch interaction, the fingers are “paired”, and with this and pinching motions, it is possible to interact with objects. This specific application uses this interaction methods to create a program with which a person can do a presentation.

3.1.1 Inspiration

My inspiration for my demo application was the movie “Minority Report”. In this movie the main actor uses gloves to interact with a multi touch screen. He uses several gestures to interact with movies and pictures. He uses the way his hands are oriented (towards the screen or turned) to grab something. When his hands are parallel to the screen, a grab action is initiated. Also here tracking of a total of six fingers is done, and the tracking is in 3D. This allows gestures like zooming to be oriented in depth. If we were to really implement the system shown in Minority Report, there would be some difficulties:

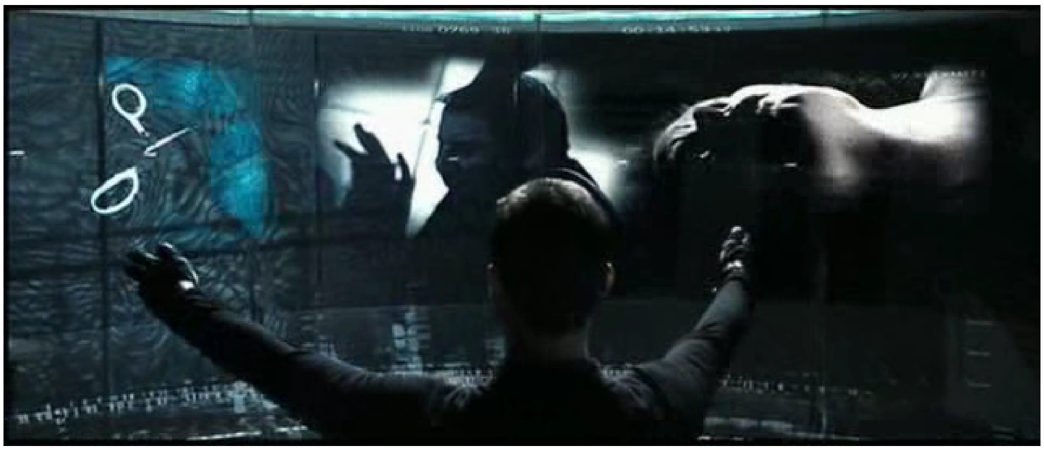


Fig. 3.1: The strange gesture the actor does.

- The audio mastering
  The audio mastering is done very carefully in the movie. Only one stream is heard at a time, but the others can be recognized too. For example you can hear the scream in another video stream. Also the talking you can hear clear, but if you were to play three videos at the same time yourself, you would probably end up not being able to distinguish them (when playing all of them at the same volume).

- Occlusion
  In the movie a light is placed on each finger, to suggest that these are tracked. If so, there would be a big problem actually tracking these lights, since occlusion would happen with most of the gestures shown in the movie. Also, with only these lights, it is a problem to stably track the orientation of the hand.

- Gestures
  There is one particular strange gesture (see Fig 3.1), which would be very difficult to implement. In the very beginning of the movie, the main actor grabs the video stream, and splits it into three streams, each with one subject: the man, the wife and the murder object. At the moment this would not be possible to implement.

Of course there are more references to multi touch interfaces working in free air. For example this is also used in the movie “Iron Man” and the video clip “Laurent Wolf - No Stress”. In the Iron Man movie both direct interaction with the hands is done, but also interaction with a pointing device is done. In the clip “Laurent Wolf - No Stress” the interaction is shown, but it is very difficult to see what actually is done to create the interaction.

Also some other projects are implementing multi touch in free air. Some examples are Project Maestro [9], the iPoint Presenter [11] and Cam-Trax [1].

3.1.2 Grabbing an object

Because a Wii Remote is used (which limits the number of tracked points to four points), and we want to track four fingers at a time, it is not possible to see depth. Grabbing an object by for example using the orientation of your hand is impossible this way. We want to track four points at a time, because we want to interact with two hands, so of each hand two fingers can be tracked. The fingers that are tracked are the thumb and forefinger.

Using this approach, basically three types of grabbing motions can be used to grab an object. You could "tap" an object to grab it (see Fig 3.2), and tap again to release. Another motion is to pinch your two fingers of one hand together (see Fig 3.3b) to grab the object and make your fingers distinct from each other (see Fig 3.3a) to release. The last option, is when a point appears, a "grab" action is done, and when it is removed, the object is released again.

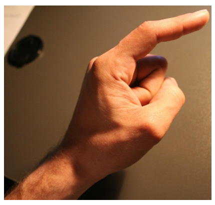
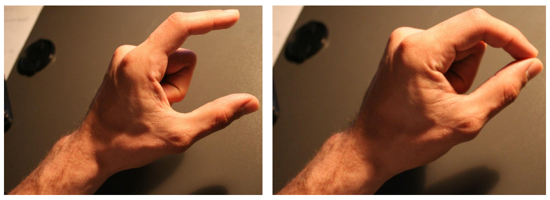


Fig. 3.2: Tapping motion.



(a) No pinching. (b) Pinched, and grabbed an object.

Fig. 3.3: Pinching motion.

As Jeff Han said in his presentation at TED [15], to get the feeling that someone is actually interacting with something, the person needs to feel pressure on their fingers. Because with this setup, we do not actually have an object that we are working with, or a screen to press on, the pinching action is preferred. The pinching action implicitly gives the idea that someone is grabbing something, because pressure is felt. Although this is pressure from just putting the fingers together, it feels more intuitive than the tapping action.

Also the tapping action is not very accurate. Because the user is doing a very fast motion, it is not known where the tapping action initiated from very accurately. This would create the need to keep some sort of history to make the tracking more accurate and it would need to make users move quite slowly when tapping. It will be a problem to make the distinction between tapping and moving fast, when moving in a non-linear line.

One problem with the pinching technique is the orientation of the hands. Because the Wii Remote can only see four points in a plane, when doing the grabbing action, this needs to do this in a virtual plane. If the finger pairs of the hands are not oriented in parallel to the view of the Wii Remote, problems arise. For example is it not possible to grab an object as if it was floating in the air parallel to the view plane of the Wii Remote (having, from the view point of the Wii Remote, one finger behind the other). This is because then you would have your fingers perpendicular to the plane while grabbing, and the Wii Remote would register only one point, and would not see the actual grabbing motion.

Because the pinching action feels more natural, this is the motion that was chosen for the application. One problem that remains in a 2D view plane, is the forced orientation of the hands.

3.1.3 Tracking of the points

Because of the way the library I used works (see Chap. 4) to get the points from the Wii Remote, combined with the not perfect tracking the Wii Remote provides itself, it is not that obvious which point is which. For example the library gives me identification numbers per point, but they are switched sometimes. The Wii Remote itself does not keep identification numbers per point itself. The library generates these identification points, and just gives the visible points numbers in the order they are retrieved from the Wii Remote. The identification numbers switch when someone for example puts (for both hands) the two fingers together, and then at almost the same time shows all four fingers again. There are several ways to keep track of relatively stable identification number per point; the algorithm I used in my final version of the demo is quite standard, but not the most intuitive.

The algorithm always matches the closest previous points to the new points. Then the identification numbers are copied over from the old points to the new points (since the identification numbers from the library switch actually).

When a point is removed or added, a slightly different approach is used. When a point is added, all old points are matched to a new point, and the identification numbers are copied over again to the new points. The new points that have not been handled yet, will be assigned a new identification number, and will be placed at the end of the array.

Using this approach, when a point is added, it is easy to pick the new added point, because it is at the end of the array. This way when doing the pairing, we do not have to search for the added point again.

When a point is removed, all new points are matched to an old point. Then also the identification numbers are copied.

This algorithm does not do motion estimation, so when two points are occluded, and shown again, the identification numbers can be switched. This is however repaired most of the times by the pairing algorithm, as explained in the next section.

3.1.4 Pairing of the points

There are several possible algorithms to pair the points. I tried two algorithms, an instant algorithm and an incremental version. The final version of my program uses the incremental version, because it is more stable, and has less problems generating the actual interaction events.

The instant algorithm

This algorithm uses the total distance the two pairs of points generate. When four points are available, for each combination of pairs the distance is calculated. The total minimum distance that a combination of pairs generate, is the correct pair combination. One solution to make this algorithm more stable, is to favor the y-direction over the x-direction. This is because when doing the interaction, the hands will most likely be oriented vertical (see Fig 3.4a and 3.4b), with the two points of one pair most times beneath one another. The factor used to favor the y-direction is multiplying the x-distance by a factor 1.8. To keep the identification of the pairs stable, an algorithm that looks which pair is which can be used to keep the identification of the pairs more stable.

Using this approach gives a problem when less than four points are available. Using history, it is possible to keep track of which point belonged to which pair, but this is not very practical. When only keeping track of the

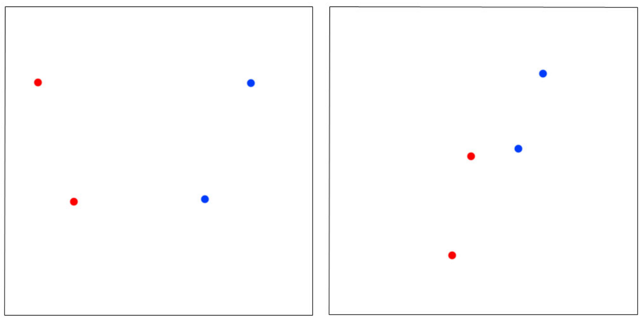

(a) Two hands distant from each other.
(b) Hands close to each other.

Fig. 3.4: Pairing of points.

pairing of the previous frame, certain things can be matched, but soon a lot of states will be created in the algorithm, just to keep track of which point belonged to which pair. This becomes a bigger problem when points disappear. For example, when only two points are left, is it better to assign the two left points each to one “pair”, or to put both in one pair. It would be better to keep the two points in one pair, if for example one hand moved out of sight of the camera. On the other hand, when both pairs of fingers are put together at the same time (which happens quite often), each pair should only have one finger, and two grab events should be registered. Using this instant algorithm, with checking later, gives a problem deciding when to put two points in one pair, and when to put them in distinct pairs.

The best version that worked quite well, just put each point in a pair when there were one or two points available. When three or four points were available, pairing was done as described above. With three points the only difference is that one pair has no distance to add. Also some very basic use of history was put in, to make the pairs not switch too often.

The incremental algorithm

This algorithm uses the the information of the pairing of points done before as input to decide what to do with the current input. When points are added, different actions are taken depending on the number of points available. When points are deleted, each point that does not exist anymore is deleted from the pair.

To keep track of the points of each pair,a structure was used that can keep track of identification numbers of the points in each pair, the count of active points in the pair, and the last known distance that was between the points, when two points were still available.

For the adding of the points, the following logic is used:

- When the first point is added, it is added to the first pair.

- When adding the second point, first the pair that has no active points is looked up. This is necessary since points can disappear too, and for example when previously one point was removed from the first pair, the new point should be added to the first pair instead of the second pair. So the new point is added to the pair that has no active points, and the active number of points in that pair is set to one.

- When adding the third point, some more logic is needed. Since here an actual pair is formed, reordering of the pairs is allowed. If, for example, first you make your thumb and forefinger of your left hand visible, these fingers would each be assigned a pair. When then the thumb of the right hand would be added, reordering of the pairs is necessary, since otherwise a pair would be formed between one finger of the left hand and one of the right hand (see Fig **??**).

  To decide which two points need to form the pair, just calculate the distance between each pair that can be created with three points. The pair that would has the smallest distance between its points is the pair that is chosen. Also favor y-direction over the x-direction here by a factor 1.8, to make the algorithm more stable, and make the chances of choosing the right points for a pair better.

  Problems arise when for example the first hand is occluded by the second, and is then shown again. If one point appears between the two points of the second hand, the new point from the first hand would form a pair with one of the points of the second hand, which would not be the result we want.

  One way to solve this problem would be to introduce an extra check which would check if one of the pairs still had two points. When this is the case, just add the new point to the other pair. Unfortunately it seems this does not work, since the input from the Wii Remote seems too unstable in the cases this logic is needed.

  At last, when the point is added, and the pairs are reordered, the distance of the pair that has two points needs to be updated.

- When adding the fourth point, just add the new point to the pair that only has one point, and update the distance and point count of that pair.

When removing a point, the easiest way to determine which points are gone, is to just loop through the pairs, and try to find the identification numbers of the points that should be in the pair. When a point is gone, remove it from the pairs. One practical implementation issue to make the implementation of the adding of points easier is to make the last point of the array of points of a pair always empty, when one of the points from a pair is removed. This saves logic when adding the third and fourth point to the pairs.

To create the interaction events, when a third or fourth point appears, and triggers the creation of a finger pair, the "pair lost" event is fired. This is because this means that for that specific pair, two fingers are shown (again), and so the "grab" is released.

Similarly, when a point is removed, and the number of active fingers in the pair was one, the "pair lost" event is fired.

The other event "pair created" is fired, when the number of active fingers in a pair was two, and the distance between the two points was less than 0.3. The distance checking is important, because otherwise when a hand is moved outside the view of the Wii Remote (and no grabbing should be done), a grab action would be registered too, which is unwanted.

There is a way to fix the pair mixing, which even with this approach still happens sometimes. The pair mixing happens sometimes, when one hand moves behind the other, and then appears again in the neighborhood of the hand. When the pairs are mixed, the points of one pair would most probably move in opposite directions. Normally two points of one hand would move most certainly in almost the same directions. So an extra check to fix eventual mix ups of the pairs would be to check for the opposite movement of the points of one pair. If such movement would appear, then the points of of the pairs would need to switch in such a way that they would move in the same direction again.

3.1.5 Interaction

Now that we have interaction events, we can create the interaction.

The way to interact with the windows in my demonstration program is interaction with one pair (just translation or RNT), and interaction with two pairs, resulting in scaling, stretching and translation all together.

The translation is done when the object is grabbed in the middle of the object, while RNT is done when the object is grabbed on one of the corners of the object. This is because when doing RNT when the object is grabbed in the middle, the resulting movement becomes non-intuitive, and numerical instability also starts to play a role.

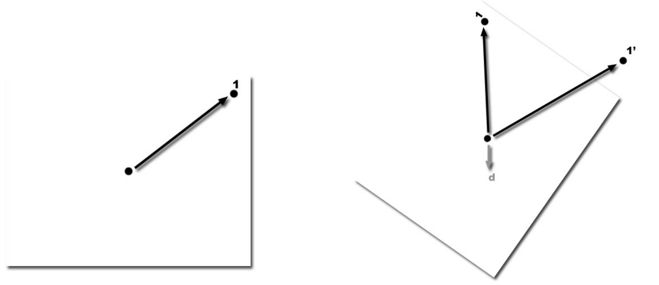
Translation

As mentioned before, translation happens when the object is grabbed in the middle. The percentage of the object chosen as “the middle” depends of the application. For my demonstration about 25% of the length is used, where the length is the distance from the middle to one of the corners.

RNT

When the object is grabbed at the corner, RNT [18] is used (see Fig 3.5a and 3.5b). This algorithm uses a size vector s, which stays the same during the movement of the object. The vector s is initiated with the distance between the grabbing point and the center of the object. Also the object is rotated with the change in angle between the start grabbing point $P1$ and the new point $P2$ later in the movement. Because this translation reacts like how an object in the physical world would react, this is a good algorithm to use for interaction of an object with one pair.

For a more thorough explanation of the algorithm, please have a look at the article from Kruger et al. [18].



(a) The object is grabbed. (b) The objects is translated and rotated to point 1 from the original point 1’.

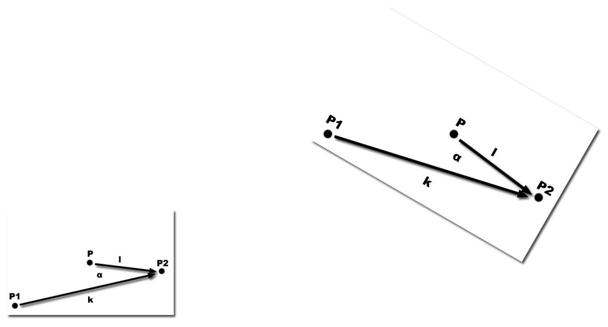
Fig. 3.5: RNT.

Two pair interaction

With this translation, the result expected is that the grabbing points stay on the same position on the image, although the grabbing points themselves move (see Fig 3.6a and 3.6b). This means that rotation, translation and scaling will be done. What we want to achieve, is that the angle between

the vector $P_1 - P_2$ and the vector $P - P_2$ stays the same (with $P_n = P^1$ or $P_n = P^2$). The new scale is then calculated with $s = \frac{||P_1^1 - P_2^1||}{||P_1^2 - P_2^2||}$.
The rotation is calculated with a function in Ogre, *getRotationTo*, which requires two normalized vectors, and outputs a quaternion describing the angle the objects turns. This is multiplied by the orientation the transformation started with.
At last, the translation of the center point P is done with the following formula: $P^2 = P_2^2 + r + (||P^1 - P_2^1|| * s * |P_1^2 - P_2^2|)$
Where r is the rotation between the vectors $|(P_1^1 - P_2^1)|$ and $|P^1 - P_2^1|$. And Where s is the scale factor calculated earlier.

For a more thorough explanation of the algorithm, please have a look at the article from Hancock et al. [16].



(a) The object is grabbed with points p1 and p2.

(b) The objects is translated, rotated and scaled.

Fig. 3.6: Two pair interaction.

3.2 Realization

For the realization of the presentation program, the render engine used is Ogre [8]. This is combined with the use of a few pieces of code from their wiki, providing video playing support on a texture in the render engine. Also included is the library from Smit [25], to get the input from the Wii Remote.

3.2.1 Displaying the pairs

Because users want to know where their fingers are in the view of the interaction system, it is handy to display where their fingers are. There are two basic ways to show these points. The position of the hand could be displayed (because the finger pairs are recognized this is possible), or just each finger can be shown independently. The difference here is how the user wants to interact with the objects. For the presentation program all four points are shown, and the fingerpoints of one pair a shown in a separate color. This way each point is shown, but the user knows which points are recognized as a pair. The reason to do this, is because this feels more natural for most people we tested with, because they think of grabbing objects with their fingers instead of a whole hand. Also now feedback is given back when grabbing an object, because the user sees clearly when they grab something, they can see their fingers move to each other. One last important reason to show all points, is because sometimes the pairing still breaks, and the user needs to move both hands out of sight, and make them visible again to get the correct pairing. (Although already a possible solution to the pair breaking problem is given, at the time of writing this is not tested yet).

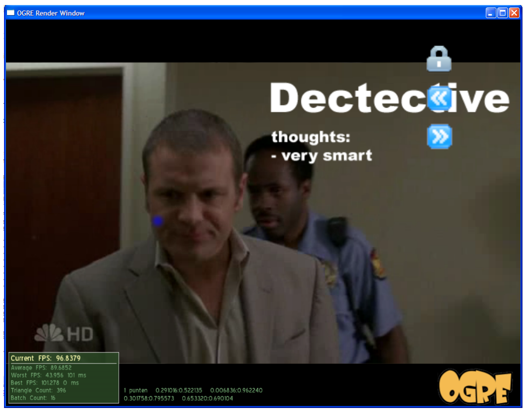
3.2.2 Interface elements

Because the inspiration of the project was the movie “Minority Report”, some interface elements were created like the ones in “Minority Report”. Also because the program is aimed to be a presentation program, three types of windows were created: pictures, movies, picture frames. A picture element show just a single picture. The movie element displays a movie, which basically plays the movie. The picture frame is created specifically for the presentation, because it can contain more than one picture, and you can walk through the pictures. To make the window interaction a little bit more intuitive, a very basic window reordering algorithm was implemented, which puts the window in front that is grabbed. For the movies also a volume changer was implemented, which sets the volume of the movie that is interacted with to full ($0dB$), and of all the other movies to $-17dB$. This way the movie that is interacted with can be heard clearly, and the other movies can still be heard a bit. This requires though all audio of the movies to be normalized first.

3.2.3 Interface widgets

To be able to present with the program also some widgets were implemented. The interactions that can be done with a normal window are just locking. Locking a window is handy when presenting (see Fig ??), because one can not accidentally move the window anymore. For the movie elements, also buttons for starting the movie, stopping the movie, and rewinding the movie were created (see Fig 3.8a and 3.8b). These are shown when appropriately, so either the start button is shown, or the stop and rewind button.

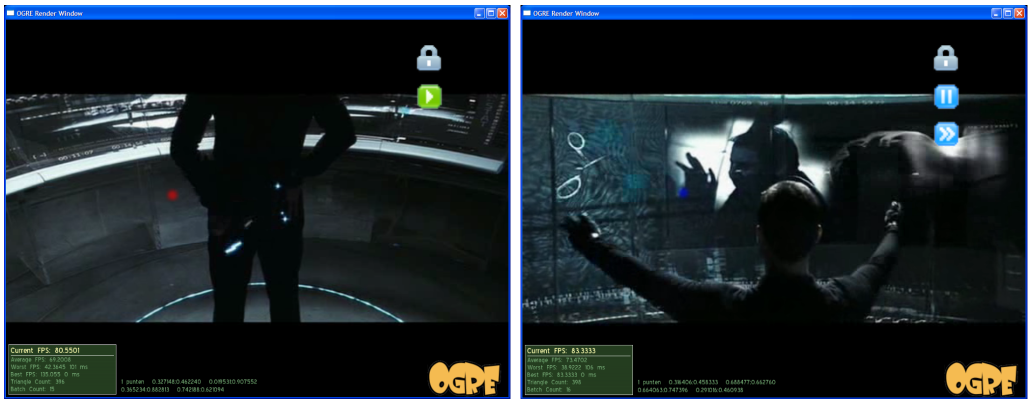
For the picture frame elements (see Fig 3.7), also buttons for going to the next and previous slide were created.



```
Current FPS: 96.8379
Average FPS: 89.6852
Worst FPS: 43.956 101 ms
Best FPS: 101.278 0 ms
Triangle Count: 396
Batch Count: 16
```

```
1 punten    0.291016:0.522135    0.006836:0.962240
0.301758:0.795573    0.653320:0.690104
```

Fig. 3.7: The pictureframe widgets.



```
Current FPS: 80.5501
Average FPS: 69.2008
Worst FPS: 42.3645 101 ms
Best FPS: 135.055 0 ms
Triangle Count: 396
Batch Count: 15
```

```
1 punten    0.327148:0.462240    0.01953t0.907552
0.365234:0.882813    0.742188:0.621094
```

```
Current FPS: 83.3333
Average FPS: 73.4702
Worst FPS: 38.9222 106 ms
Best FPS: 83.3333 0 ms
Triangle Count: 398
Batch Count: 16
```

```
1 punten    0.316406;0.458333    0.688477:0.662760
0.664063;0.747396    0.291016:0.460938
```

(a) The movie is paused. (b) The movie is playing.

Fig. 3.8: The movie widgets.

These interaction buttons are only shown when an object is actually grabbed, because otherwise they would occlude vision to other objects, and annoy people when for example presenting. The buttons are always shown on the right upper side of the display, to have a stable position of each button, and to not have to search for a button each time (when for example the buttons were tied to the window). It is also important to make the buttons big enough, and to keep enough space between them, because it is quite difficult when presenting to do very precise grabbing of an interaction button.

For presentation use, the minimum and maximum size of the objects is constrained. This is because if the object was to be sized really small, it is very difficult to actually make it bigger again, because the Wii Remote would just see one point if someone keeps their hands very close to each other. The maximum size constraint is because when presenting it is annoying when accidently objects are enlarged very fast. When then the user wants to get back a somewhat more normal size of the object, resizing several times can be quite annoying.

3.2.4 Configuration file

To allow reuse of the presentation program without recompiling and extensive knowledge of the program, a configuration file was created. This configuration file is specified in the INI-format[1]. It has only one section, named "Media", with settings for a picture, picture frame and a video. Each element can be specified more than once, and triggers the creation of an object. All the parameters of an element need to be specified. For each window type, the x and y can be specified, which are specified from 0 to 1. This sets the position of the middle of the object. Also the width of the object can be specified, which can be in the range from 0 to 1, where 1 an object the size of the screen, and 0 would be invisible. There are some size constraints though, no object (see "Interface widgets") smaller than a scale 0.3 is allowed. The last common parameters are the anti clockwise rotation, and whether or not the object is locked. The movie window type also has an extra parameter that specifies if the movie is playing initially or not.

3.2.5 Precision and reliability

The precision of the system is quite good, but there are some problems. Problems arise when the Wii Remote sees more than four points, and the point recognition is not stable anymore. Also it is very difficult to grab something small very accurate, because the user is working in free air.

[1]Accepted standard for configuration files, see, e. g., http://en.wikipedia.org/wiki/INI_file.

Some other people have used the system, with varying results. Most people need to learn to keep their hands parallel to the view plane of the Wii Remote. Most (technical) people have mastered this when working 15 minutes with the device. Some people tend to need more time to learn this though. Also a very small amount of people had problems making the link between the points shown on the screen, and where their fingers are. For example they did not know when their fingers left the view of the Wii Remote, and did not understand why they could not grab objects anymore.

When working with the setup for a longer time, some problems arise too. When presenting, the users arms get tired, and people tend to put down their arms, accidently moving objects in ways they did not want.

3.2.6 Possible uses

There are several uses for this system, most of them are in the sector where lots of data need to be viewed or organized.

Presentation system

The system should (with a few extensions) be quite usable as a presentation system. The things that are missing for the moment are seeking and manual volume settings for movies. Also missing are features like drawing on the screen.

Organizing data

Also, but you would need more extensions, this system can be used to organize big datasets. You would require something like a beamer then, to be able to display the data. Then with the use of this system, you would be able to work with big datasets. The big thing that needs to be improved here, is the interface. Currently only a really small set of interface elements and interactions are available.

3.2.7 Implementation problems

There were some problems with the implementation. To get Directshow working in combination with Ogre, some SDKs are needed. The DirectX SDK is needed, in combination with the Directshow part of the Platform SDK. However, most versions of the DirectX SDK do not include a proper version of some of the Directshow header files. The version recommended by Microsoft Support dates of November 2007.

Also problems arose when the presentation program was compiled under 64bit Windows. This could not be fixed.

To allow Directshow to actually play a variety of media files, ffdshow was used.

3.3 Summary

A presentation program was realized through recognition of fingerpairs, allowing the user to grab and interact with objects. There are some limitations to the current technique though.

Chapter 4

Retrieving the points

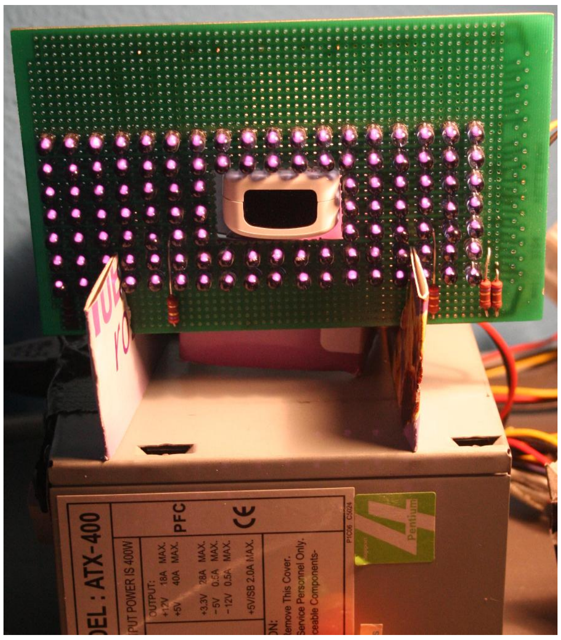
To retrieve the points, a library is used for getting points from the Wii Remote is the library written by Smit [25]. This library retrieves the points from the Wii Remote, and gives them identification numbers in the order it receives the numbers. Also the points are scaled between 0 and 1. For the precise working of the library, please have a look at the thesis.

It is very easy to use the library, since the setup to use the library (connecting to the Wii Remote, and setting the callback function) takes only about three statements.

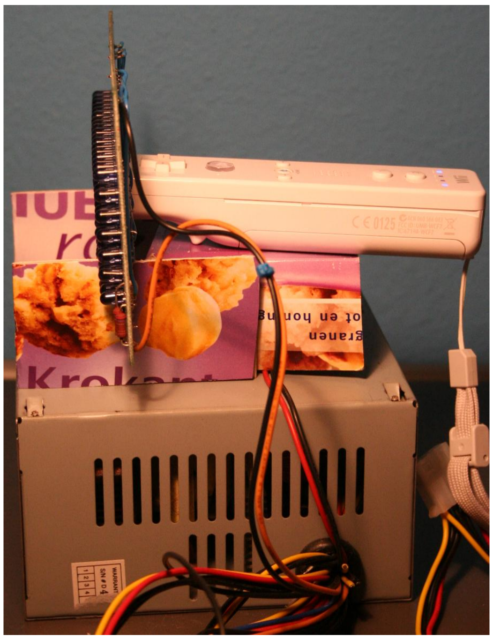
The original idea to keep the Wii Remote placed static somewhere around the screen, while the points are moving is from Lee [19]. He suggested using a IR LED setup around the Wii Remote, sending out light parallel to the Wii Remote. He used about a hundred IR LEDs in his setup, and said this was kind of overkill. He also suggested using retroreflective tape to put on the fingertips.

The first version of the LED array setup for the project was quite unstable, also because my soldering skills are not very good. So someone else was asked to create the device. He recreated most of the setup from Johnny Lee. However, he used a power supply to power the setup, and used about 140 LEDs on the setup (see Fig 4.1a and 4.1b). Using a power supply to power the setup is handier in the long run, because it saves the use of a lot of batteries. This setup would use about twelve batteries in one or two hours running.
As can be seen, the way the LED array is placed is not very stable, some paperboard is used to keep things in place. If a device would need to be created for real use, these things would need to be taken care of.

For reflecting the IR light, quite some materials were tried. Aluminium foil, parts from a cut CD, tape and mirrors were tried. All these objects do not reflect the light good enough, or do not reflect the light diffuse enough



(a) Front view.



(b) Side view.

Fig. 4.1: The IR LED array.

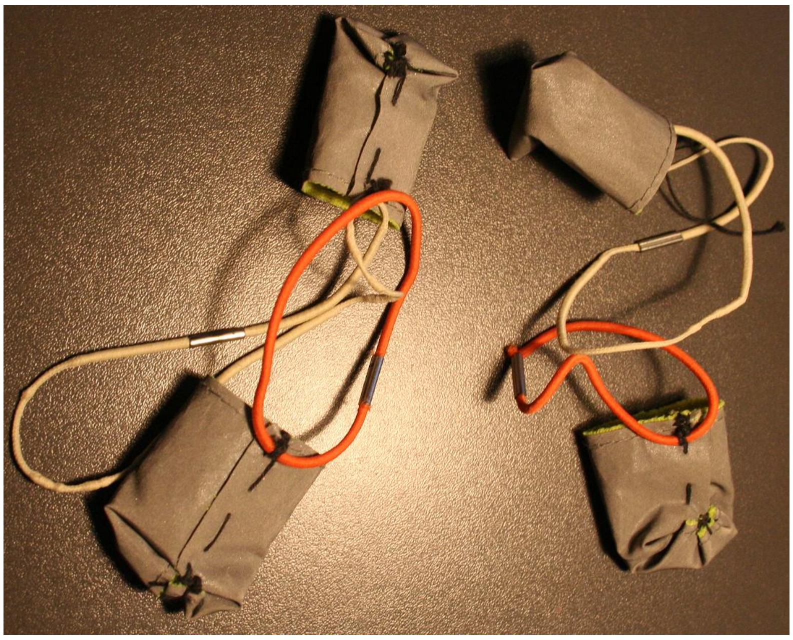
to get stable point recognition. Finally reflective sheet from some traffic clothing was used, which worked quite well. This is easy to get, and can be transformed in things like the versions on Fig 4.2a and 4.2b.

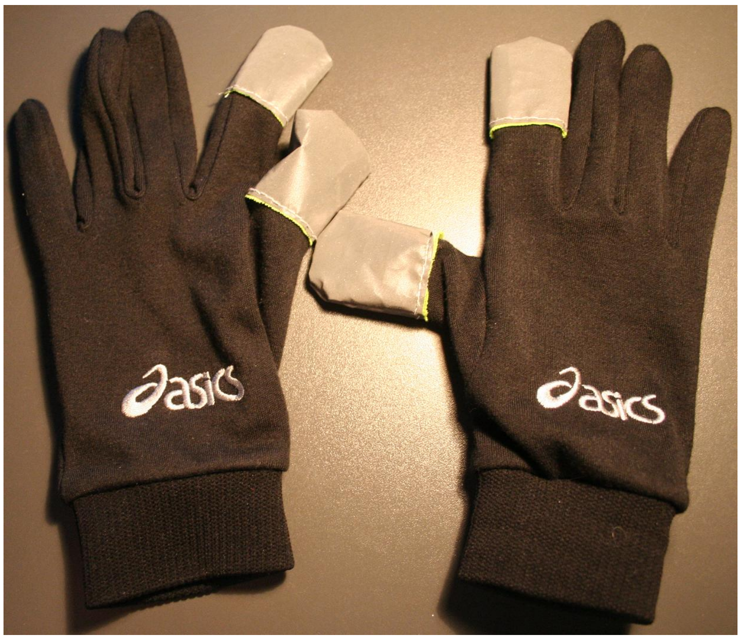
The fingertip version on Fig 4.2a was the first version, and had some problems. Because the stiches are on the outside, when you point these towards the Wii Remote, it has problems doing blob detection when you move away more than 1.5 meters. With the gloves version, this problem was almost gone, and in the same conditions (the same room), the blobdetection would run fine until 2.5 till 3 meters.

When doing the presentation, some problems were found with the gloves. It is important to carefully align the Wii Remote and the IR LED array. Because the setup when presenting was not aligned that good (the IR LED array stood more horizontal then the Wii Remote), problems occured when interacting with the upper part of the screen. This problem did not occur earlier, because before the setup was only used with everything standing horizontally. The presentation was done standing, so the Wii Remote had to look up.

One way to fix this problem, is to attach the Wii Remote to the IR LED setup in a way they are always aligned correctly.

The precision of the Wii Remote is quite good, as long as enough light was reflected back to the Wii Remote, and the room did not have other very bright objects in its view. Problems occur when the objects get too small, or when you move away too far, and the amount of reflected light is too low. A way to fix this problem is to do active motion tracking, putting the IR LEDs in the gloves. The typical distance we had to keep to the Wii Remote was between 0.5 meter and 3 meters. In the user is standing to close to the Wii Remote, the points start flickering, and are not recognized stable anymore.


(a) First version.



(b) Second version.

Fig. 4.2: The fingertip version and gloves

Chapter 5

Summary and Conclusion

For this projection a finger tracking system was created using the Wii Remote and gloves fitted with retroreflective clothing. The system is very cheap. It only requires a Wii Remote, which costs only 40 euros, if a Wii Remote is not already owned. Besides the Wii Remote also some retroreflective clothing is needed, about 140 LEDs and preferrably a power supply. The total cost of the LEDs is about 17 euros, since each LED costs about 12 cents. The total cost is then about a hundred euros, taking into account a power supply is used. This is still very cheap, since most multi touch screens today cost a lot more.

In this project a multi touch interface in free air was created using cheap hardware, and a very basic interface for presentation use was created. The people that worked with it, once they got used to keeping their fingers in the view plane from the Wii Remote, were able to work quite well with the interface, and described it as working intuitive.

Chapter 6

Future Work

A few ideas that popped up during the creation of my demonstration program are listed below. Some of them require a lot of work, others just minor extensions of the demonstration program.

- Instead of having the Wii Remote oriented horizontal, orient it vertical, and put it under the desk, or above the desk. When put under the desk, a glass desk is needed. When the setup is oriented vertically, the advantages are that your hands can rest on the table most of the time. This saves a lot of strength for interacting with the machine.
- Use two Wii Remotes instead of one, to be able to get depth information too. The second Wii Remote could for example be positioned above the subject, so the second camera can supply the depth and x axis. This way an algorithm could be created that matches these points, and gives the finger points in 3D, without much occlusion.
- Using cameras instead of Wii Remotes for any of the ideas. This is more expensive, but also allows more fingers to be tracked at a time. Also the suggestion would be to not just do blob detection anymore, but use an algorithm that can extract more information from each blob by the form of the object. This could determine for example more reliable whether or not two finger points are pressed together.
- To identify each finger uniquely in an easy way, use LEDs width different infrared colors on the top of each finger. This is an active marker setup, but allows (with combination of cameras instead of Wii Remotes) you to easily track each finger.
- To identify each finger uniquely in a somewhat more sophisticated way, you could vary the sizes and forms of the retro reflective sheet on each

of the fingers. This requires a high resolution camera, but then each finger could be tracked more easily, with the use of pattern recognition. Using this technique (and for example putting some pattern on a finger instead of just one form), the rotation of a finger could be determined as well.

- Creating a driver for Linux (and possibly Windows) generating mouse (and keyboard) input from the input from the finger tracking device.

Bibliography

[1] Cam-trax. URL http://www.cam-trax.com.

[2] Mctcameras, . URL http://www.mctcameras.com/.

[3] Mova, . URL http://www.mova.com/.

[4] Optitrack, . URL http://www.naturalpoint.com/optitrack/.

[5] Vicon, . URL http://www.vicon.com/.

[6] Diamondtouch. URL http://www.merl.com/projects/DiamondTouch/.

[7] Microsoft surface. URL http://www.microsoft.com/surface/.

[8] Ogre rendering engine. URL http://www.ogre3d.org/.

[9] Cynergy labs: Project maestro. URL http://labs.cynergysystems.com/Flash.html.

[10] iphone. URL http://www.apple.com/.

[11] Fraunhofer hhi: ipoint presenter. URL http://www.hhi.fraunhofer.de/index.php?1401&L=1.

[12] George Borshukov. Making of the superpunch. Technical report, ESC entertainment, November 2004.

[13] Alvaro Cassinelli, Stephane Perrin, and Masatoshi Ishikawa. Smart laser scanner for human-computer interface, 2005. URL http://www.k2.t.u-tokyo.ac.jp/fusion/LaserActiveTracking/.

[14] Jeff Han. Multi-touch interaction research, 2006. URL http://www.cs.nyu.edu/ jhan/ftirtouch/.

[15] Jeff Han. Unveiling the genius of multi-touch interface design, 2006. URL http://www.ted.com/tedtalks/tedtalksplayer.cfm?key=j_han.

[16] Mark S. Hancock, Frederic D. Vernier, Daniel Wigdor, Sheelagh Carpendale, and Chia Shen. Rotation and translation mechanisms for tabletop interaction. In *Proceedings of IEEE International Workshop on Horizontal Interactive Human-Computer Systems (TableTop2006)*, pages 79–86. IEEE Computer Society, 2006.

[17] Stefan Hechenberger and Addie Wagenknecht. Cubit.

[18] Russell Kruger, Sheelagh Carpendale, Stacey D. Scott, and Anthony Tang. Fluid integration of rotation and translation. In *CHI '05: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 601–610, New York, NY, USA, 2005. ACM.

[19] Johnny Chung Lee. Johnny lee's personal webiste. URL `http://www.cs.cmu.edu/ johnny/projects/wii/`.

[20] Kadaba MP, Ramakrishnan HK, and Wootten ME. Measurement of lower extremity kinematics during level walking. *Journal of Orthopedic Research*, may 1990.

[21] Steve Perlman. Volumetric cinematography: The world no longer flat. Technical report, Mova, October 2006.

[22] Daniel Restuccio. Riding the polar express to a digital pipeline: Sony pictures imageworks' first full-length animated feature takes advantage of mocap and imax 3d. *Post LLC*, nov 2004. http://findarticles.com/p/articles/mi_m0HNN/is_11_19/ai_n8575087.

[23] Barbara Robertson. Big moves. *Computer Graphics World*, 29(11), November 2006.

[24] Daniel Roetenberg. Intertial and magnetic sensing of human motion, May 2006.

[25] Jasper Smit. Head tracking, July 2008.

[26] Julie E. Washington. 'beowulf' takes motion-capture filmmaking to new technological and artistic heights. *The plain dealer*, nov 2007. url:http://blog.cleveland.com/top_entertainment/2007/11/_imagine_youre_a_hollywood.

[27] Pierre Wellner. Digital desk, 1991.