FILTERING FOR KINECT POINTING

GROUP MEMBERS

Yu Zhang
014345531

Lei Wang
014343711

Ayesha Ahmad
014109366

Arvi Nikunlassi
013473938

DEMO REPOSITORIES	
Filter Visualization	https://github.com/ayef/kinectProject
Filter Game Application	https://github.com/bluntos/hitMouseForKinectProject

Contents

1 Abstract

The advance of interface technologies nowadays has produced more and more possibilities in interacting with computers. Microsoft Kinect is a motion sensing input device that allows user to communicate with computer using whole body gestures. In this paper arm pointing (considering the hand and shoulder joints) using Kinect and its filtering are examined. Kinect filters are algorithms that allow filtering of the raw body skeleton data to produce more accurate interaction. In this paper various filtering techniques applied on the hand and shoulder joint data are examined and analyzed especially in the context of pointing applications. Different filters produce different results with respect to accuracy, latency and smoothing effect. The filtering techniques can also be combined to produce different results. For demonstration purposes an application built using Unity and Kinect is demonstrated. The application's main function will be demonstrating different types of filtering for pointing with the Kinect. During the demonstration filters can be changed to highlight their differences. The application demonstrates visually the performance and behavior of various filters.

2 Introduction

2.1 The Need for Filtering

In this research the focus is on filtering for Kinect pointing. Raw data from the skeletal tracking (ST) system of the Kinect is noisy and cannot be used directly in applications such as gesture detection and navigation of interfaces. Especially in the case of using Kinect data for pointing, user experience can be improved by incorporating smoothing or filtering of noise. Using the skeletal tracking joint data that Kinect provides in its raw form will result in jittery and inaccurate pointing. The user cannot see a stable point on screen even while holding their arm still. Jitter is especially problematic when a joint is occluded during pointing. Consequently some filtering is required for the joint positions before they can be used in pointing applications.

Figure 1. Hand joint positions from Kinect scattered around the true position over time
Source: http://msdn.microsoft.com/en-us/library/jj131429.aspx

2.2 Filtering and Pointing Principles

The accuracy of filtering can never be perfect. The acquired coordinate points always float around the theoretical perfect result. This inaccuracy of filtering is called noise and it is always appearing in filters [Azi00]. The most usual form of noise is the so called white noise that is constantly part of the filtering. The white noise is usually very light and not disturbing form of the noise and its origin is the basic inaccuracy of gesture recognition. The spike noise is very observable big deviation from the perfect recognition. These spikes are produced

when the trackable part of the body is in a situation where the recognition is unclear. Figure 2 presents the typical noise situation. Looking at the red line it is easy to distinguish the overall white noise and the small deviations from the big and high noise spikes. Usually the duration of these spikes is very low and that makes them look like some kind of a bug in software.

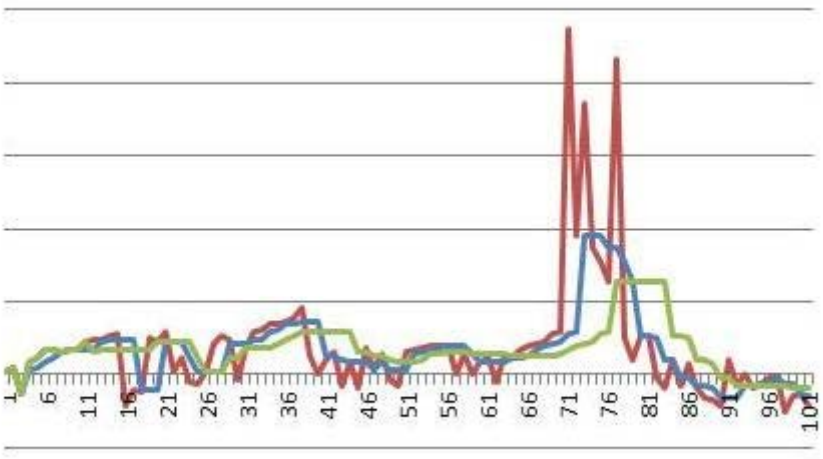

Figure 2. White noise and spike noise [Azi00]

Latency is a term that describes the amount of time required from the initiation of a process to the delivery of results. In the Kinect the latency can be analyzed as the time that is required to recognize the gestures, process and filter the data and output the results on the screen. The more the input gestures are filtered and processed the more time it will consume. In this way there's often a trade-off between filtering smoothness and low latency. Different filters produce different latencies. One way to reduce latency is forecasting the movements. However this approach is usable only in situations where the movements are easily forecast. In some situations it is better to have good smoothing over latency and vice versa. Filters can also be applied on different levels. Sometimes it is useful to have a distinct filter on every joint and sometimes a common filter for all is the best solution. Because of that it is important to choose the most suitable approach to every situation.

One big part of body gestures is the trade-off between accuracy and speed in human motor movements [PTI13]. If the speed is increased on movements the accuracy and quality of the

movements is decreased. The same applies on the gesture recognition device. While very fast movements may be hard to recognize accurately also very standstill gestures may be hard to keep stable. The object that is going to be pointed and its context is also a big and important factor [MSB91].

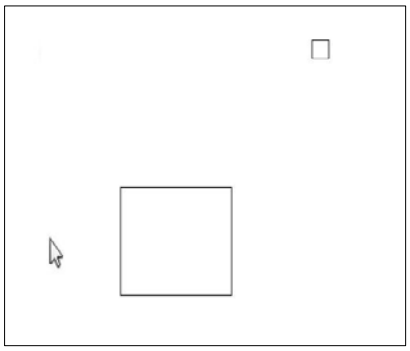

Figure 3.Fitts' law

Fitts' law is a model that predicts how much time it takes to point the target object. The time is determined by two factors, the size of the object and its distance from the pointer. The filtering algorithm has to be adjusted correctly to match different situations. Figure 3 is displaying a typical situation involving Fitts' law. It can be clearly observed that the bigger square is much easier and faster to reach and click than the smaller one

2.3 The Goals of this Project

The main goal of this project is to compare the different filters and their combination for Kinect pointing. For this purpose two demonstration applications are created. The first application concentrates on filtering on a game where user has to hit appearing targets. The type of filter can be changed to observe the different effects they provide. The second application is a technical demonstrative application. With this application the user can choose different filters and try to move around a circle. The filters produce line where the way of the movement can be examined.

The applications of this project demonstrate the smoothing capabilities of different filters. The user can easily found out that more smoothing filters produce more latency. When the

filters can be seen in real use it is easy to determine which filters fit best to some applications. When the filtering data is drawn on the screen it is easier to visualize than without drawing. That is why the approach in the applications is more visual than relying on data or numbers.

3 Background: Pointing, Filters and Natural User Interfaces

Arm pointing is a way to interact with the computer with only a user's hands. Besides the normal input techniques like the mouse and keyboard, gestures are starting to appear nowadays as a way to interact with computer. The most commercially successful systems may be the touchpad gestures, but gestures recognized by mere camera are now becoming more and more common.

In [Farzin12] researchers studied a camera based on the input system for the computers to perform typical desktop tasks by gestures. For example, when the touchpad is scrolled down with two finger gesture, it could be more intuitive way to use the computer when you could input the gestures on midair. The researchers built their system on the base of the Kinect 3D and the software kits used by OpenNI and NITE. [Farzin12] The research group has compared the usual input methods with these newly developed which has a comprehensive user experiment. They have found out that the arm gestures are superior when used with large displays. The test persons felt that the gestures were more natural and pleasant in that situation. [Farzin12] However, using gestures cause more fatigue to the users than using the mouse, because the users have to keep their hands on the air and keep pointing using their hands on the screen.

From [Yama04], we know that some limitations on the arm pointing technology and the result data from sensor will be influenced by changes in lighting and user's clothing. The research group from University of Tsukuba proposed a new camera system by using four stereo cameras placed in four corners of the room. These cameras looked down at an oblique angle to capture human entire bodies and faces simultaneously. This system did not depend on user's body position, direction and lighting environment. It could independently recognize arm-pointing gestures of several users. [Yama04]

There is still much space for arm pointing technology to develop. Especially how to identify gesture of multiple people simultaneously in a complex scenario and recognize both arms independently [Chien07]. In addition, the comfortable of arm pointing is another important research topic. Unlike ordinary input method (like mouse or touch screen), human cannot maintain the same posture for a long time.

Furthermore, three gestural pointing techniques and two clicking techniques for large displays are introduced and evaluated in [Vogel05]. The need for pointing and clicking from a distance was justified, for example, when the user was situated at a distance from the display, and was performing tasks such as sorting or navigating a large graphic. Important design characteristics of such techniques were identified, namely accuracy (reliable selection of small targets), acquisition speed (acquisition and release of the pointing device when in-use and not-in-use), pointing & selection speed (the pointing device is able to move to any location in the display without much effort), ease-of-use and smooth transition between up-close interaction to interaction at a distance.

The three pointing techniques discussed were absolute position ray casting, relative pointing, and a hybrid technique combining ray-casting and relative pointing. After experimentally evaluating these techniques, the paper found that absolute pointing was fast but has a high error rate. Relative and hybrid techniques were similar in terms of selection time and error rate. It also had a much lower error rate than absolute pointing, which suggested that these two techniques were equally suitable for selection of small targets. [Vogel05]

Then, a few other interesting issues discussed in the paper like use of visual and/or auditory feedback to compensate for lack of physical touch during clicking and visual feedback to user when gesture about to become ambiguous.[Vogel05]

Another increasingly important point is to develop adaptive input techniques with more and more large displays adopted by individuals. Adaptive input techniques can be used to interact with displays from several meters away. A Comparison of Ray Pointing Techniques for Very

Large Displays first described four different fundamental ray variants (laser pointing, arrow pointing, image-plane pointing and fixed origin pointing). The article [Jota20] concentrated on two factors which influence ray pointing performance: Control Type and Parallax. Then, it used three experiments (horizontal targeting, vertical targeting and tracing) to examine pointing performance of the four different ray pointing variants. These experiments results revealed that arrow pointing is a good choice when interaction required both targeting and tracing. It also had a close performance with laser pointing in selection tasks and better in steering jobs. When only considering tracing task, two parallax-free techniques (image-plane and arrow) performed best than the others. At last, it found evidence that control type affects targeting and parallax affects tracing. [Jota20] Moreover, this paper gives us strong evidence to support the use of angular indexes of difficulty for targeting and tracing tasks with any of the absolute ray pointing techniques.

Filtering of the human gesture input data is an area that has recently increased a lot of in numbers of publications [YOU12, ZHO13]. This is because the consumer reachable systems like Kinect have made these ways of interaction more popular [JUN13]. Filtering in all its forms has been researched in different fields long before Kinect and other devices. The re-search on gesture filtering is therefore based on the applying the filters to this gesture based way of input. There has also been much work done with reference to filtering of the depth image provided by the Kinect sensor [MNG13, CAM13, MAT11] the filtered depth image can then be used as input in many applications. The noise characteristics of the Kinect sensor have been studied in [NGU12]. In this project we focus on filtering of the joint data provided by Kinect in pointing applications.

4 Project Design

4.1 Pointing and Filtering Methods

Relative pointing has been used and smoothing has been performed on both the hand and shoulder joints. The confidence value of the joints has been taken into account, if the confidence falls below a certain threshold the application does not compute a smoothed output. Different filters can be used for filtering the hand and shoulder joints, especially as the underlying data model for the two joints is so different from one another. The hand joint shows much greater movement and velocity than the shoulder joint which is relatively stable. It is better to use less computationally intensive filters such as median for processing the shoulder joint which will effectively remove spiky noise from this joint. In addition to the joint filtering jitter removal has also been performed. If the change between the computed vector between hand and shoulder exceeds a certain threshold the jitter removal filter is applied. A simple functional diagram for the working of the filtering algorithm is shown in Figure 4.

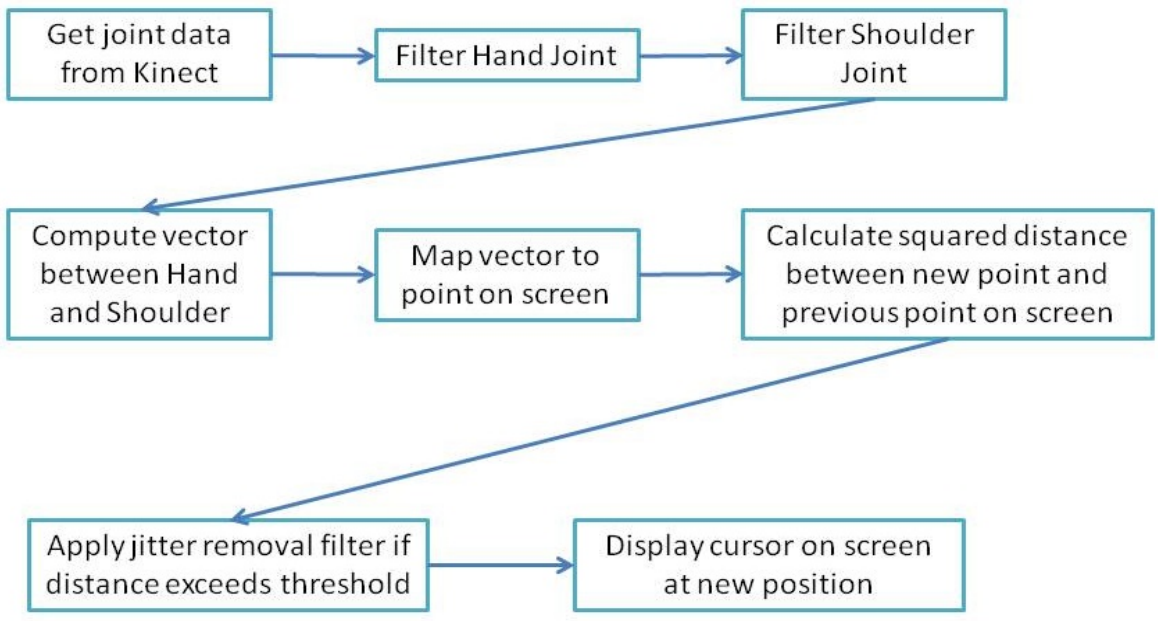

Figure 4: Simplified filtering algorithm

4.2 Filters Implementation

Kinect filtering is mainly based on averaging the results and smoothing the movements. The raw movement by itself is very fuzzy and imprecise and has a lot of spikes in movements. The main task of the filtering is to smooth that data. The filter responds to the input data outputting data that is more flat than the input. ARMA -filter is in some sense the base of the filtering [Azi00]. It contains just the basic filtering operations. Its main function is to take all the previous actions and their filtering and to filter the future actions based on this data.

The simple averaging filter takes a chosen number of recent inputs and does the filtering based on that. The double moving average filter is useful when the movement has a trend that it is following. The exponential smoothing filter is based on the exponential diminishing weights of the past inputs. The double exponential smoothing filter is similar to the double moving average filter. It doubles the exponential smoothing. The adaptive double exponential smoothing filter is an extension of the double exponential smoothing filter. The Taylor series filter is based on the mathematical Taylor series. The median filter is based on the median value of some number of previous inputs. Jitter removal filter is used as its name suggests to remove jitter. The main function of the filter is to limit the maximum changes of movement. In this way spikes due to noise can be dampened.

The filters are based on different equations to calculate the most correct results. By using different amounts of historical joint data and weights, different results are achieved.

The filters used are primarily special cases of moving average filters. All of the filters use previous inputs to calculate the smoothed current output. One of the filters - the Taylor Series filter - has been used to forecast data. Combinations of filters have also been studied for example using different types of filtering for the hand and shoulder joint as each has a very distinct data model. The filters have been compared on the basis of pointing accuracy and latency introduced by filter application. The filters implemented in this project are given in Table 1.

Name	Formula	Description
Moving Average	$\hat{X}_n = \sum_{i=0}^{N} a_i X_{n-i}$	The weighted sum of previous 'n' inputs, weights sum to 1
Simple Average	$\hat{X}_n = \frac{1}{N+1}\sum_{i=0}^{N} X_{n-i}$	Average of previous 'n' inputs
Double Moving Averaging	$MA_n^{(1)} = \frac{1}{N+1}\sum_{i=0}^{N} X_{n-i}$ $MA_n^{(2)} = \frac{1}{N+1}\sum_{i=0}^{N} MA_n^{(1)}$	Moving average of the moving average
Exponential Smoothing	$\hat{X}_n = \alpha \sum_{i=0}^{n} (1-\alpha)^i X_{n-i}$	Weighted average of past 'n' inputs with weight reducing exponentially
Double Exponential Smoothing	Trend: $b_n = \gamma(\hat{X}_n - \hat{X}_{n-1}) + (1-\gamma)b_{n-1}$ Filter output: $\hat{X}_n = \alpha X_n + (1-\alpha)(\hat{X}_{n-1} + b_{n-1})$	Smoothes smoothed output by second application of exponential filter
Adaptive Double Exponential Smoothing	$\alpha_n = \begin{cases} \alpha_{low} & v_n \le v_{low} \\ \alpha_{high} + \frac{v_n - v_{high}}{v_{low} - v_{high}}(\alpha_{low} - \alpha_{high}), & v_{low} \le v_n \le v_{high} \\ \alpha_{high} & v_n \ge v_{high} \end{cases}$	Adapts the weights α and γ according to υ – the joint velocity
Taylor Series	$f^{(3)}(n) = f^{(2)}(n) - f^{(2)}(n-1) = X_n - 3X_{n-1} + 3X_{n-2} - X_{n-3}$ $X_{n+1\|n} = \frac{8}{3}X_n - \frac{5}{2}X_{n-1} + X_{n-2} - \frac{1}{6}X_{n-3}$	Taylor series output using three previous inputs. Second equation shows how the forecasted value is computed
Median	Median of previous 'n' inputs, useful in removing spike noise	
Jitter Removal	$\hat{X}_n = \begin{cases} X_n, & if \left\|X_n - \hat{X}_{n-1}\right\| < threshold \\ \alpha X_n + (1-\alpha)\hat{X}_{n-1}, & otherwise \end{cases}$	Dampens spikes by limiting amount of change allowed between inputs

Table 1.A list of implemented filters, with their formulae. Filter implementations are all available in the Filter.cs file in both demo applications

There is also another filter we need to mention, which is Kalman Filter. Currently the Kalman filter has been considered as the most optimal approach to provide best estimate in linear systems with Gaussian white noise [X2RDF]. It infers parameters from uncertain observations, and uses recursive method to process new measurements when they arrive. In addition, it is very convenient for real time processing and easy to apply in general scenarios. But in this report, we do not plan to work more detail on this filter.

4.3 The Demo Applications

4.3.1 Filter Visualization Application

The main aim of the project is to demonstrate various filters we designed in a Kinect pointing application and compare them with respect to their accuracy, latency and lack of jitteriness. One way to compare these filters is to draw specific geometric graph like circles under different filters, then we can compare the outcome graph to evaluate performance of different filters.

The first demo allows comparison of smoothness and responsiveness between two or more filters. The user can select a number of filters or even no filtering and then draw on a screen to see the result. In this application filter latency cannot be measured but the user can visually ascertain the differences between filters in terms of accuracy and smoothness. A screenshot of the application is shown in Figure 5 and a block diagram outlining the flow of information is shown in Figure 6.

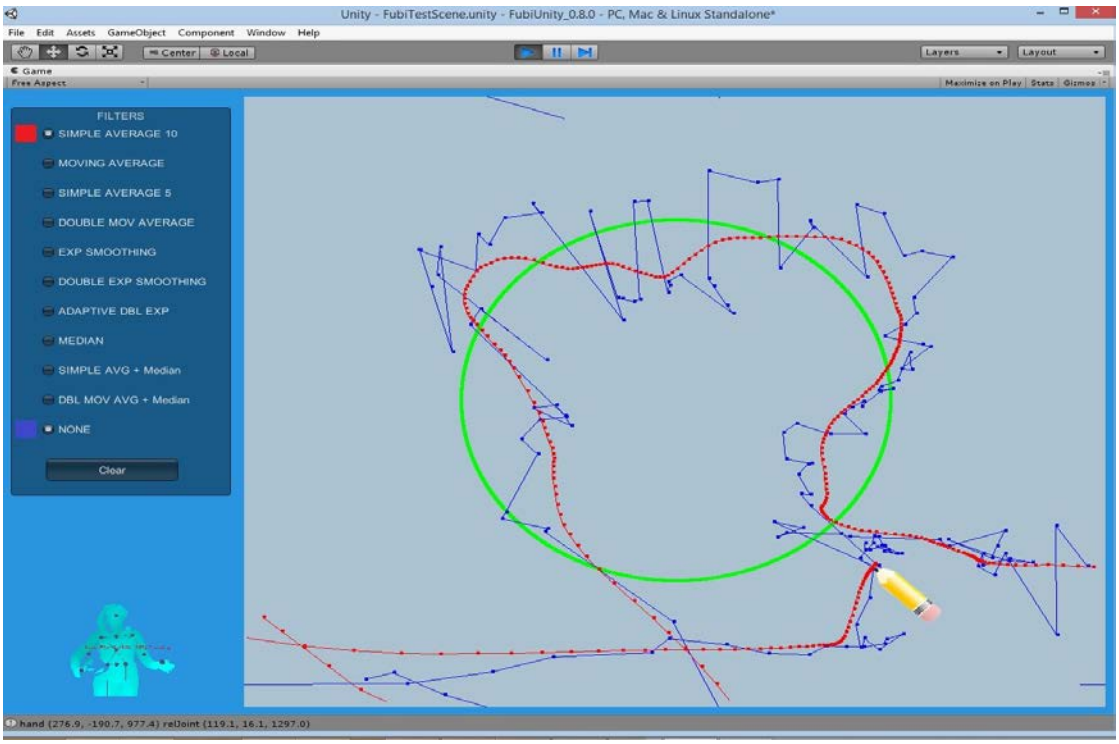

Figure 5. Screenshot of filter visualization application

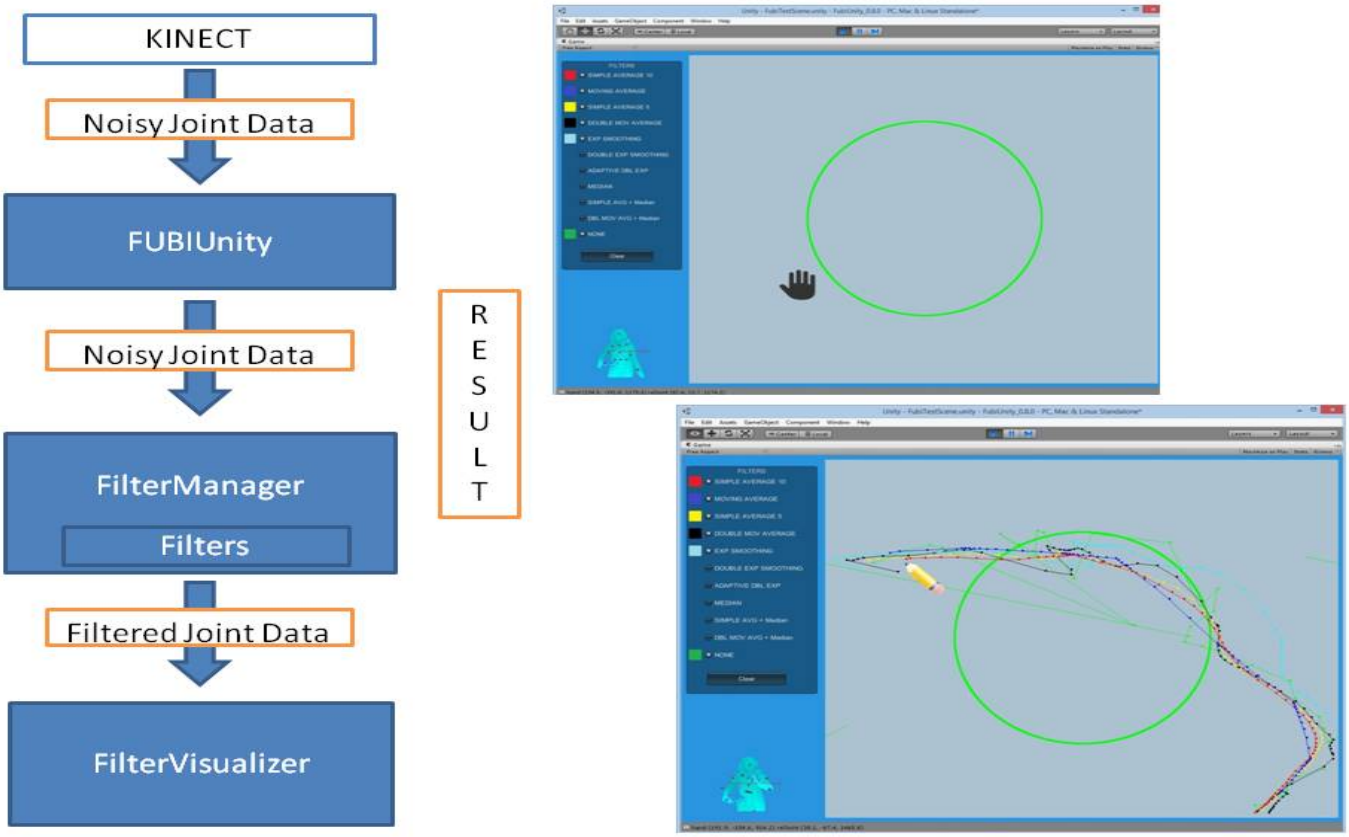

Figure 6.Functional block diagram of filter visualization application operation

4.3.2 HitMouse Game Application

Another way to compare these filters is from the viewpoint of the user, which is how well a user can accomplish a task using a certain filter combination. To demonstrate this from the user's perspective, a game called "Hit Mouse" has been developed. The higher score implies better performance of the selected filter. To develop this game, we also use the FubiUtility as base to do the basic data acquisition from Kinect.

The basic idea of this game is to use Kinect Pointing under certain kinds of filters to hit mouse as much as possible. Then compare the game scores under different filters to demonstrate the performance of specific filter. The following Figure 7 gives a whole view of this game.

Figure 7. Screenshot of "Hit Mouse" game

At the left side of Figure 7, there are 11 toggle boxes which give choice of different filters. Before click "Start Game" button, player need choose the specific filter he or she want to use. Once game starts, timer at the right top of game panel will count down. At the same time, mouse will randomly come out from one of the six holes on the game terrain at every 2 seconds. Players need to move cursor on to the mouse which has come out of the hole, and then mouse gets hit and disappears. Player can get 10 scores at each successful hit. If player

cannot hit the mouse which has come out of hole for more than 5 seconds, mouse will disappear. When the timer counts down to 0, game is over. Player can change the filter and then start game again.

After we choose several filters to complete game, we can click the “Show Summary” button to get the score comparison under different filters, which are considered as indicator of rough performance of corresponding filter.

The flow chart of whole game has given in the following Figure 8.

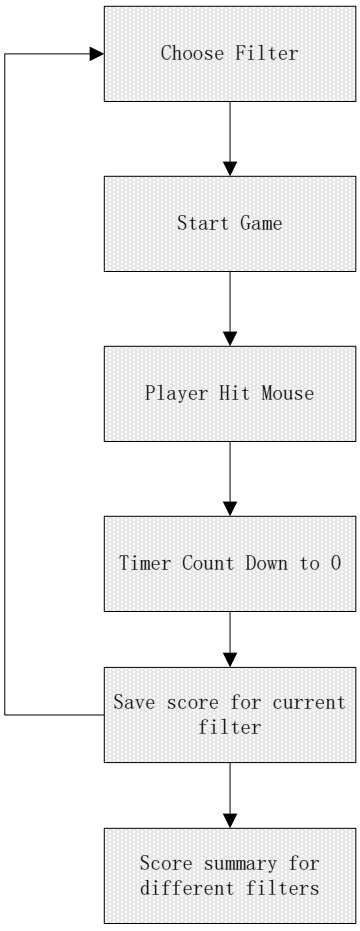

Figure 8.Flow chart of game

We have tested all these filters, and results have been given in the following Figure 9.

Figure 9. Screenshot of summary results of “Hit Mouse” game

4.3.3 Filter Visualization Application Important Classes and Functions

Name	Description
Class FilterManager	Manages all the added filters
AddFilter()	Adds a filter to FilterManager’s list
UpdateJointFilters()	Calls update for every added filter for joints
UpdateVectorFilters()	Calls update for every added filter for the new and old screen position
Class FilterVisualization	Manages drawing of the filtered points
DrawCircle()	Draws a circle on the screen the user can use as a guide
SetPixels()	Writes on the screen to show the filter output
Class Filter	Specifies a filter
Update()	Receives a joint position, applies the filter updates all of its internal lists like input history, output history etc

4.3.4 HitMouse Game Application Important Classes and Functions

Name	Description
Class hitMouseGame	Manages the logic of whole game
startGame()	Start Game
stopGame()	Stop Game
selectMouseToBeAvailable()	Randomly select one mouse among six to come out of the hole, then Mouse can be hit.
mouseToBeUnavaiable()	After successfully hit the mouse, disappear the hit mouse.
updateMenuPanel()	Update display of game menu panel
updateGamePanel()	Update display of game playing panel
Class FubiUnity	Manage acquisition, coordination of data from Kinect
showFilterMenu()	Show filter menu on the left side of game panel
getPixelPosition()	Get pixel position of cursor
Class GameStartStopTextShow	Show “Game Start” and “Game Over” GUI text on the game panel
Update()	Show “Game Start” and “Game Over” GUI text on the game panel

4.4 Further Work

There is still more work that can be done in this project for example implementation or refinement of filters to determine the optimal amount of history to keep for a filter and also to determine other parameters such as the best value of weights for different filters. It would also be interesting to compare the performance of the filters on some quantitative mathematical basis rather than just through visual comparison. Analysis of the type of noise and variation experienced by different skeletal joints provided by Kinect would also be useful in determining the most effective combination of joints and filtering to use for specific applications.

5 Discussion and Analysis of the Project

This project had a lot of new techniques, challenges and failures. The project started in a meeting on the class where groups were formed. This is the first crucial step for this kind of project. One observation that can be made about the group formation is that no consideration was given to the members' background and experience in programming. This can lead to several problems concerning the results of the project and grading. For example our group had two experienced programmers thanks to them the project could be carried out quite efficiently. If for example we had no experienced programmers, the work would be too hard to implement and there would be several problems. It would be more efficient if the same principles, like questionnaires about skill levels, would be used on this course similar to those in the 'Software Engineering Lab'.

Our group had some problems getting things running and the programming work started. In the start of the project one member began not to appear at the regular meetings and then abandoned the whole course. The programming platforms were quite unknown to the members of the group. Putting up a development environment took quite lot of time. It was also a hindrance that the computers at the university didn't contain software for the Kinect development. It is not realistic to assume that all the participants would have their own devices. A few weeks before the project deadline the underlying software being used for communicating with the Kinect device had to be changed causing a significant increase in work to learn and modify the new software (the software was changed from ‘Kinect Extras’ by RF Solutions to FUBIUnity) .

Also in the initial phases there were some problems concerning the topic of this project. At the initial phase the topic was comparing relative and absolute pointing. After discussion the topic was changed to the comparison of Kinect filtering. This kind of imprecision within the topics is also kind of a problem. When the task topics are too hard, unclear or require significant scientific research it is harder to implement some working software.

After all two working pieces of software were produced. These applications efficiently demonstrate the differences in filtering. These applications form a combination and support each other. In the game application the filters can be observed in a real action. In the technical application the smoothing procedure can be examined clearly. Even though there were problems getting the project started and one member dropped out it turned out the majority of the project goals were met. The created applications contain working code and they fit their purpose in demonstrating the filters. Every member participated well into the work and gave their best to finish the project.

6 Conclusion

In this project different filters on the Kinect device were compared. Two different applications were developed to demonstrate filters. The first application was a game where appearing objects had to be hit to gain more scores. The second application was a demonstration application where different Kinect filters and their traces can be seen.

The project's main point was to demonstrate the different outputs that the filters can provide. Using filters the raw data can be made smoother and the movements can be adjusted to fit the context. All the movement data can be filtered but in this work the arm movement was the main point of concern because hand movements are usually the main way to interact with Kinect. It became clear that smoothness comes at the cost of responsiveness and latency, and that the filter needs to be tailored to the type of application and the type of joint data being considered.

References

Azi00 Skeletal Joint Smoothing White Paper, Kinect for Windows 1.5, 1.6, 1.7, 1.8, by Mehran Azimi, Advanced Technology Group, http://msdn.microsoft.com/en-us/library/jj131429.aspx

MSB91 I. S. MacKenzie, A. Sellen, and W. Buxton, “A comparison of input devices in elemental pointing and dragging tasks,” in Proceedings of ACM CHI ‘91 Conference on Human Factors in Computing Systems, 1991, pp. 161–166

PTI13 A. Pino, E. Tzemis, N. Ioannou, and G. Kouroupetroglou, “Using Kinect for 2D and 3D pointing tasks: performance evaluation,” in Human-Computer Interaction. Interaction Modalities and Techniques, Springer, 2013, pp. 358–367

FARZIN12 Farzin, F., Reza, G. and Ali, A., Empirical study of a vision-based depth-sensitive human-computer interaction system, ACM, USA, 2012

YAMA04 Yamamoto, Y., Yoda, I. And Sakaue, K., Arm-pointing gesture interface using surrounded stereo cameras system, Proceedings of the 17th International Conference on Vol.4, No., PP.965,970, USA, 2004

CHIEN07 Ching-Yu Chien, Chung-Lin Huang and Chih-Ming Fu, A Vision-Based Real-Time Pointing Arm Gesture Tracking and Recognition System, 2007 IEEE International Conference on Vol.4, No., PP.983,986, China, 2007

JOTA20 Jota, R., Nacenta, M. A., Jorge, J. A., Carpendale, S. and Greenberg, S., A comparison of ray pointing techniques for very large displays, ACM, Canada,2010.

Vogel05 Vogel, D., and Balakrishnan, R., Distant freehand pointing and clicking on very large, high resolution displays. ACM, Toronto, 2005.

MNG13 Mengyao Zhao, Fuwen Tan, Chi-Wing Fu, Chi-Keung Tang, Jianfei Cai, Tat Jen Cham, High-quality Kinect depth filtering for real-time 3D telepresence, Pub-

lished in IEEE International Conference on Multimedia and Expo (ICME) 2013, pp 1 - 6, San Jose, CA

NGU12 Nguyen, C.V., Izadi, S., Lovell, D., Modeling Kinect Sensor Noise for Improved 3D Reconstruction and Tracking, Published in Second International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT) 2012, pp 524 - 530, Zurich

CAM13 Camplani, M., Mantecon, T., Salgado, L., Depth-Color Fusion Strategy for 3-D Scene Modeling With Kinect, Published in IEEE Transactions on Cybernetics (Volume:43 , Issue: 6) 2013, pp 1560 - 1571

MAT11 Matyunin, S., Vatolin, D., Berdnikov, Y., Smirnov, M., Temporal filtering for depth maps generated by Kinect depth camera, Published in 3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON), 2011, pp 1 - 4, Antalya

YOU12 Youwen Wang, Cheng Yang, Xiaoyu Wu, Shengmiao Xu, Hui Li, Kinect Based Dynamic Hand Gesture Recognition Algorithm Research, Published in 4th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC) 2012, Volume 1, pp 274 - 279, Nanchang, Jiangxi

JUN13 Jungong Han, Ling Shao, Dong Xu, Shotton, J., Enhanced Computer Vision With Microsoft Kinect Sensor: A Review, Published in IEEE Transactions on Cybernetics 2013, Volume 43, Issue 5, pp 1318 - 1334

ZHO13 Zhou Ren, Junsong Yuan, Jingjing Meng, Zhengyou Zhang, Robust Part-Based Hand Gesture Recognition Using Kinect Sensor, Published in IEEE Transactions on Multimedia 2013 Volume 15, Issue 5, pp 1110 - 1120

X2RDF Hervier, Thibault, Silvere Bonnabel, and François Goulette. "Accurate 3D maps from depth images and motion sensors via nonlinear Kalman filtering."Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on. IEEE, 2012.