DEGREE PROJECT IN COMPUTER SCIENCE AND ENGINEERING,
SECOND CYCLE, 30 CREDITS
STOCKHOLM, SWEDEN 2016

Eliminating the latency using different Kalman filters

for a virtual reality based teleoperation system

XUXIAO MA

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF COMPUTER SCIENCE AND COMMUNICATION

Eliminating the latency using different Kalman filters for a virtual reality based teleoperation system

Eliminera latensen med olika Kalman filter för en virtuell verklighet baserad teleoperation systemet

XuXiao Ma

DA221X Master Thesis in Media Technology 30 ECTS
DEGREE PROJECT AT CSC, KTH
Degree Project in: Media Tehchology
KTH E-mail: xuxiao@kth.se
Supervisor: Haibo Li
Examiner: Anders Hedman
Project Provider: Haibo Li

ABSTRACT

Latency has always been one of the essential problems within Virtual Reality (VR) domain since VR is inherently an interactive paradigm which performs the real-time estimation of human motions. From the user's point of view, the latency extremely reduces the presence experience of VR systems, especially when user won't able to perform interactions accurately. To compensate the excessive latency, different prediction methods on human motion were studied in recent years. Among them, Kalman Filter was the most popular choice. However, the effectiveness of using Kalman Filter to eliminate the latency for VR systems is not always satisfactory in practice since the accuracy of the estimation of the users' motion depends on several factors: the linearity of the motion, the prediction time, the computational time, and the algorithm's limitation.

Therefore, this thesis presents a VR-based haptic teleoperation system to study how to effectively eliminate the latency effectively using Kalman Filter. For investigating the performances of different prediction methods for VR systems with several factors considered, two types of Kalman Filter: Linear Kalman Filter (LKF) and Unscented Kalman Filter (UKF) have been used to predict the haptic motion dataset, under different amount of simulated latencies.

The result shows, both LKF and UKF provide a good performance at compensating the latency. For 200ms latency, both filters satisfactorily eliminate the latency and improve the interaction effectiveness. The comparative study shows, LKF provides better performance since the linear rotational motion dataset captured by haptic device was used; both filters show a reduced performance when the prediction time is increased. Besides, UKF requires more computational time than LKF.

ABSTRAKT

Latens har alltid varit en av de viktigaste problemen inom Virtual Reality (VR) domän eftersom VR är till sin natur en interaktiv paradigm som utför realtid uppskattning av mänskliga rörelser. Ur användarens synvinkel, latensen extremt minskar förekomsten erfarenhet av VR-system, i synnerhet när användaren kommer inte kunna utföra interaktioner noggrant. För att kompensera den överdrivna latens, var olika förutsägelsemetoder på mänsklig rörelse studerades under de senaste åren. Bland dem, Kalman Filter var det mest populära valet. Emellertid är effekten av att använda Kalman filter för att eliminera latens för VR-system inte alltid tillfredsställande i praktiken, eftersom noggrannheten hos uppskattningen av användarnas rörelser beror på flera faktorer: linearitet rörelse, förutsägelsen tid, beräkningstid och algoritmen är begränsningen.

Därför presenterar denna avhandling en VR-baserade haptiska teleoperation för att studera hur man effektivt eliminera latens effektivt med Kalman Filter. För att undersöka prestanda olika prognosmetoder för VR-system med flera faktorer som beaktas, två typer av Kalman Filter: Linear Kalman Filter (LKF) och Oparfymerad Kalman Filter (UKF) har använts för att förutsäga den haptiska rörelse dataset, under olika mängd simulerad latenser.

Resultatet visar, både LKF och UKF ge ett bra resultat vid kompensera latens. För 200 ms latency, båda filtren på ett tillfredsställande sätt eliminera latens och förbättra samspelet effektivitet. Den jämförande studien visar, LKF ger bättre prestanda eftersom den linjära roterande rörelse dataset fångas av haptiska enheten användes; båda filtren visar en reducerad prestanda när förutsägelse tiden ökar. Dessutom kräver UKF mer beräkningstid än LKF.

Keywords

Kalman Filter Algorithm, Teleoperation, Haptic, Comparative study

Acknowledge

Special thanks to Haibo Li and Anders Hedman for supervising and supporting the thesis work; Dr.Shafiq ur Rehman for his help and guidance; Magnus Bergvalls Stiftelse for project grant.

I would also like to thank PhD.Muhammad Sikandar Lal Kha for the guidance and discussion. My labmates, Jerry Fan, and Haky Rufianto for the nice team work of the implementation.

Table of contents

List of figures

List of tables

1 Introduction

In recent years, the development of the VR field is maturing and it has been used for many different domains such as education, medicine training, entertainment, and architectural design. By simulating a virtual environment for users, VR allows them to interact with the virtual objects with different sensory controls such as head motion, body motion, and haptic. The created environment can be either real (captured by cameras) or imagined (rendered by computers), which means VR also covers the concept of presence, which provides the immersive experience and makes users feel they are present in the computer generated environment. According to "Research on Presence in Virtual Reality: A Survey" [1], Presence is one of the essential concepts in VR, and the interactivity of VR environments is the most important cause of the presence. Particularly, the speed of the responses of the environment shows a clear contribution to presence up to a point.

In this case, it is usually not easy to deliver good presence experience to create a truly believable world in VR systems due to one of the essential shortcomings: latency. Undoubtedly, the latency extremely affects the user's experience, especially for the interactions. Imagine if the users' eyes receive markedly delayed frames from the display equipment such as VR glasses or head-mounted display, their perception of all the virtual objects will not be experienced in "real time". In other words, all the objects in the video are not in the positions they are supposed to be. In this case, it is hard to make users feel being present in the virtual environment since they are not able to interact with the virtual objects accurately.

According to "Entertainment Computing - ICEC 2015", [2] for general users, the latency of 50ms feels responsive but the delay is still noticeable for VR systems. To make the virtual world nearly indistinguishable from the reality, the acceptable latency is under 20ms. With the rapid growth of the VR technologies, people have been searching for different approaches to reduce the latency. The straightforward ways are for example improving the VR hardware tracking sensors to reduce the computational time, and improving the software of rendering graphics to reduce the display processing time. However as long as the physical limitations exist, [3] the problem cannot be solved fundamentally.

To overcome the physical limitation, the feasible way is compensating the latency. Specifically, the users' motions will be predicted, and then the VR frames or graphics will be generated according to the predicted data, therefore compensate the latency. According to "*HISTORY: The Use of the Kalman Filter for Human Motion Tracking in Virtual Reality*", [4] the most popular method for tracking and predicting the human motion within VR domain was the filter-based prediction algorithm, namely Kalman Filter. As an optimal estimator, Kalman Filter provides an efficient computational means to recursively estimate the state and error covariance of a process and it has been widely used for different areas such as the navigation and control of the vehicles, the track and guidance of the robotics, and the prediction of interactive computer graphics.

However, the effectiveness of using Kalman Filter to predict the human motion is not always satisfactory in practice. Many factors need be considered in order to have a good estimation result such as the linearity of the motions, the prediction time of the motions for different latencies (i.e. how far the motions need to be predicted), and the computational time.

Therefore, for investigating the performances of different prediction algorithms when using them to eliminate the VR latency, this thesis chose LKF and UKF to predict the user's haptic motions. Both algorithms use the same dataset captured from a VR-based haptic teleoperation system to keep the linearity of the motion constant. Different amount of latencies have been simulated for the system to explore how prediction time affect the estimation result. An experiment has been done to examine how latency causes the problems and affects the

effectiveness of users' interactions. A comparison result for both filters has been presented along with the result of how different factors affect the performances.

This thesis mainly focuses on the design and implementation of VR systems and two of the Kalman Filters: LKF and UKF. A literature study has been shown in Chapter 2. The theories and methods used for implementing the system have been described in Chapter 3. The implementation and comparison result has been shown in Chapter 4. The analysis of the performances has been described in Chapter 5. Then, the conclusions have been summarized in Chapter 6.

2 Related Researches

This chapter provides a literature study mainly about the early researches of applying prediction algorithms on human motion and also mentions the early comparative studies of analyzing the performances of prediction algorithms. The contributions of this thesis are also mentioned at the end of this chapter.

2.1 LKF Applied On Human Motion

LKF, as the most basic prediction algorithm, has been widely used on simple human motion tracking and predicting. However in VR domain, it has been abandoned for a long time since most of the human motions for VR systems are non-linear such as the head motion, hand motion, and body motion. Many recent related studies about applying LKF on Human motion were using the Kinect, a set of motion sensing input devices produced by Microsoft. For example, "Trajectory tracking of joint based on Kinect" [5] uses LKF to improve the precision of the tracking function of the Kinect camera. Specifically, Kinect extract the coordinate data from the users' skeleton motions, and the extracted data will be processed with LKF and send to a dual-axis motion control subsystem to control a turntable mechanical. "*Low-Latency Filtering of Kinect Skeleton Data for Video Game Control*" [6] presents a comparative study of four different filter-based approaches to reduce the latency of a simple video game, Pong. The game was also controlled by the skeleton data captured by Kinect sensors, and then different prediction methods: Holt double exponential smoothing filter, Arithmetic Average Filter, Linear Kalman Filter (with constant acceleration model), and Linear Kalman Filter (with Wiener Process Acceleration Model) were used to smooth the joint data and mitigate the latency. For both theses, the filters they used were limited to fit only the linear models. However, they didn't explore the performances of using non-linear prediction filters on the same data.

2.2 UKF Applied on Human Motion for VR

Compare with LFK, EKF and UKF have received more attention in VR domain. "*A Comparison of Unscented and Extended Kalman Filtering for Estimating Quaternion Motion*" [7] provides an evaluation to compare the performance of EKF and UKF for improving human head and hand tracking. Specifically, the human head and hand orientation motion signals are tracked by VR applications and represented with quaternion, and then EKF and UKF were used to improve the tracking process. The result shows that the additional computational overhead of the UKF and quasi-linear nature of the quaternion dynamics make the EFK becomes a better choice in VR applications. However, they didn't explore another critical factor in prediction algorithm determination: the prediction time, which is an important uncertainty and needs to be adapted according to different network situations.

2.3 Early Comparative Studies

There were also many early studies exist for investigating the performances of different Kalman Filters. For example, "*A Comparitive Study Of Kalman Filter, Extended Kalman Filter And Unscented Kalman Filter For Harmonic Analysis Of The Non-Stationary Signals*" [8] presents a comparative result of three Kalman Filters for the tracking of harmonic components of a dynamic signal in communication system. However, their evaluation was very specific for the signal domain, which is quite different from the human motion in VR domain.

For VR systems, "A Testbed for Studying and Choosing Predictive Tracking Algorithms in Virtual Environments" [9] provides a testbed for comparing the performances of different predictive tracking algorithms when used them to reduce the dynamic tracking error and masking latency in VR environment. They used a prediction algorithm library which contains a variety of different predictors such as simple extrapolation routines, integerized predictors,

filter-based approaches, and multiple model adaptive estimation. For user motion data repository, they used both head and hand motion data. Their testing application provides a number of useful features, by setting special parameters such as sampling rate, prediction time, noise variance, and algorithmic parameters, the predictor's performance can be represented by the commonly used error metrics. However, their main focus was the implementation of the testbed application; therefore the dataset they used was pre-collected, not captured from the real implemented VR system.

2.4 Contributions

This thesis contributes a design, implementation and comparative study of using both LKF and UKF to predict the linear haptic rotational motion and eliminate the latency. A VR-based haptic teleoperation system has been implemented for the experiment; an evaluation has been made with same dataset under different simulated latencies to examine how latencies reduce the user experience and how different factors affect different Kalman filters' performances.

3 Theory and Method

This chapter presents the theories and methods used for implement the prototype system, including the description of the system model, the human motion dataset, two Kalman Filter algorithms: Linear Kalman Filter and Unscented Kalman Filter, and the methods of image distortion process.

3.1 Introduction of System Model

This thesis provides a fixed camera teleoperation system based on VR, which purposes to apply several Kalman Filters for eliminating the latency.

Specifically, the users are able to remote control a 3D graphic robotic arm using a haptic device "Phantom Omni", and also perceive the real-time surrounding environment by a simple Head-Mounted Display (HMD), Google Cardboard. Figure 1 shows the basic flow chart of the system

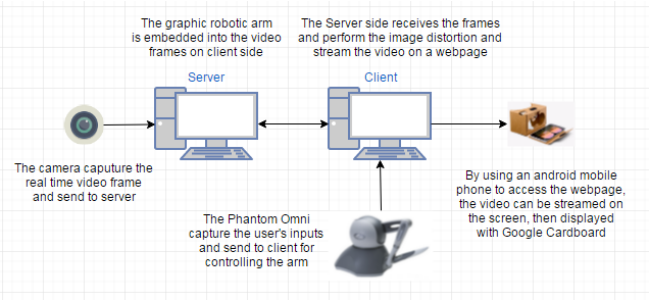

Figure 1. The flow chart of the system

The system is based on Client-server model. The client is connected with a camera used to capture the real-time environment surrounding the imaginary robotic arm, using Open Source Computer Vision (OpenCV). The graphic arm is generated by Open Graphics Library (OpenGL), an application programming interface (API) for rendering 2D and 3D vector graphics. Then the graphic arm is embedded into the video frames and encoded by H.265 (High Efficiency Video Coding, HEVC), using FFmpeg, a software provides libraries and programs for handling multimedia data. The communication between client and server is based on User Datagram Protocol (UDP); Server receives and decodes the frames, and also sends the filtered user input data back to client. The user input data is captured by Phantom Omni, using OpenHaptic Toolkits, which includes the Haptic Device API (HDAPI), the Haptic Library API (HLAPI), and also the PHANTOM Device Drivers (PDD). The received frames are adapted to VR frames by using Radial distortion and Stereoscopy, and stream to a webpage, using Hypertext Transfer Protocol (HTTP) and Motion JPEG (MJPEG). Then the processed frames are displayed by the Google Cardboard.

3.2 Haptic dataset

In order to control the graphic robotic arm, users need to use the haptic device Phantom Omni. Phantom Omni is a 6 degree-of-freedom (DOF) haptic device which can easily get the positions, angles, and force feedbacks from users with the joints and stylus. The communication interface is IEEE-1394 Fire Wire port and it supports C++ by using OpenHaptic Toolkits. Figure 2 shows the basic design of the Phantom Omni.

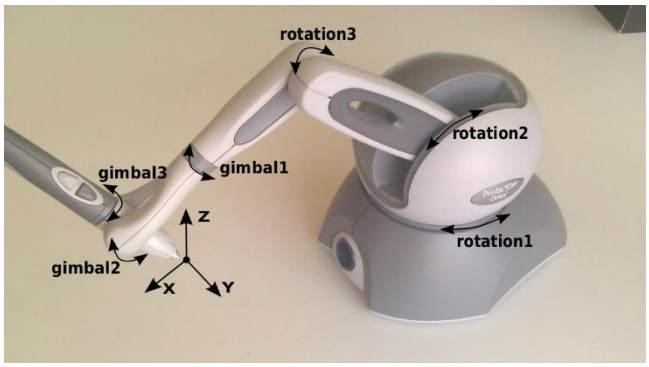

Figure 2. The overview design of Phantom Omni

For the prototype, the coordinates of the stylus(x, y, and z), and three joints angles (rotation1, rotation2, and rotation3) were used to control the graphic robotic arm, which means the user's haptic motion can be represented by the joints' linear rotational motions. The two buttons on the stylus were used to control the "fingers" of the arm for grabbing and releasing functions. In order to determine whether the "fingers" reach the objects, a vibration feedback was added, users can feel the vibration when the "fingers" touch the objects. The graphic arm model is designed by Giorgi Pataraia [10] as Figure 3 shows:

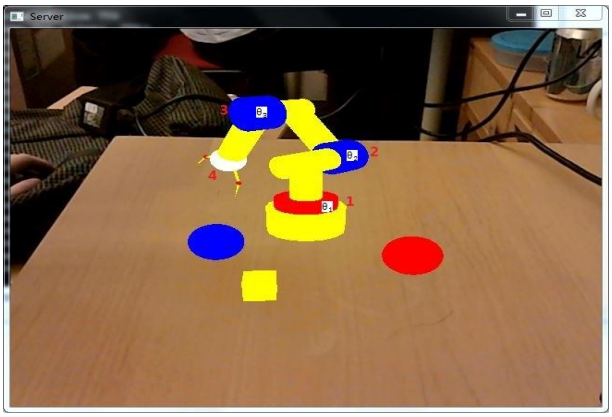

Figure 3. The pre-designed robotic model

The graphic model above represents a 3 DOF robotic arm, which has three turnable joints (1, 2, and 3) corresponding to the three joints of the Phantom Omni respectively.

3.3 Kalman Filter Algorithms

Kalman Filter algorithm (KFA), named after Rudolf E. Kálmán [11] by 1960 is the most popular optimal estimator algorithm today. Theoretically, Kalman Filter is based on Bayesian model and it is similar to a hidden Markov model except the state space of the latent variables is continuous and all latent and observed variables have a Gaussian distribution. The Kalman Filter algorithms basically have two processes: prediction and correction. In prediction process, the estimates of the current state variables will be produced, along with the uncertainties which refer to the process noises. Then the estimates will be updated using a weighted average in the correction process after the new measurement data (including the errors) is observed. Here it also shows the great success of this algorithm in two aspects. Firstly, this algorithm has small computational requirement; Secondly, it is recursive so it can be used for real time processes.

To predict user motions using Kalman Filters, the prediction process needs to be repeated several times due to the lack of measurement data. The repeat time is according to the prediction time, which is the value corresponding to the latency of the system.

There are many variants of the standard LKF for different system models such as the EKF and the UKF. Both of them are two nonlinear version of the LKF, which purpose to be used for non-linear system models. In this chapter, both LKF and UKF have been presented in detail along with the parameters used for the prototype system.

3.3.1 Linear Kalman Filter Algorithm

LKF is the standard algorithm compare with other Kalman extensions. Basically, the State Space Model of this dynamical system contains two equations: state equation and measurement equation.

The state equation describes how the unobserved state evolves at a time *t* from a prior state at time *t*-1 according to

$$\mathrm{x_t} = F_t \cdot x_{t-1} + B_t \cdot u_t + W_t \quad W_t = N(0, Q_t) \tag{1}$$

In equation (1), $\mathrm{x_t}$ is the state vector containing the interest for the system at time t; u_t is the control vector containing all the control inputs, F_t is the state transition matrix which applied to the prior state x_{t-1}, B_t is the control input matrix which applied to the control vector u_t , W_t is the process noise for the state parameters, which assumed to be a normal distribution zero mean Gaussian white noise with covariance given by the covariance matrix Q_t.

The measurement equation describes how the observed variables depend on the unobserved state of the model, according to

$$\mathrm{z_t} = H_t \cdot x_t + V_t \qquad V_t = N(0, R_t) \tag{2}$$

In equation (2), $\mathrm{z_t}$ is the measurement vector; H_t is the transformation matrix which maps the state parameters into the measurement space, V_t is the measurement noise which also assumed to be a zero mean Gaussian white noise with covariance given by covariance matrix R_t.

As a recursive estimator, Linear Kalman filter has two distinct phases: predict and update. In order to produce the estimate for current state, the estimated state from the previous time t-1 and the current observed measurement state are needed.

Firstly, the predicted state estimate and predicted estimate covariance are calculated according to

$$\hat{\mathrm{x}}_{\mathrm{t|t-1}} = F_t \hat{x}_{t-1|t-1} + B_t \cdot u_t \tag{3}$$

$$\mathrm{P_{t|t-1}} = F_t P_{t-1|t-1} {F_t}^T + Q_t \tag{4}$$

In equation (3), $\hat{\mathrm{x}}_{\mathrm{t|t-1}}$ represents the predicted estimate of state vector x at time t given measurements up to t-1, it is also called priori state estimate since the measurement information from current time t is not included. In equation (4), $\mathrm{P_{t|t-1}}$ represents the predicted estimate covariance, it is used to measure the estimated accuracy of the state estimate. Then, the update equations are given by

$$\mathrm{K_t} = P_{t|t-1} \cdot {H_t}^T \cdot (H_t \cdot P_{t|t-1} \cdot {H_t}^T + R_t)^{-1} \tag{5}$$

$$\hat{\mathrm{x}}_{\mathrm{t|t}} = \hat{x}_{t|t-1} + K_t \cdot \left(z_t - H_t \cdot \hat{x}_{t|t-1}\right) \tag{6}$$

$$P_{t|t} = P_{t|t-1} - K_t \cdot H_t \cdot P_{t|t-1} \tag{7}$$

From equation (6) and (7), it is obviously to see that the posteriori state estimate $\hat{\mathrm{x}}_{\mathrm{t|t}}$ and posteriori estimate covariance $P_{t|t}$ are updated by $\mathrm{K_t}$, the Optimal Kalman gain represents a weighting matrix used to calculate how much the state estimate needs to be changed according to the measurement.

For the prototype system, the LKF described above has been implemented for estimating three joints angle $\theta_x, \theta_y, \theta_z$, and their velocities V_x, V_y, V_z. The state vector x of the dynamic system is then described as $(\theta_1, \theta_2, \theta_3, V_1, V_2, V_3)^T$, and the measurement vector z is described as$({\theta_1}', {\theta_2}', {\theta_3}')^T$, which are the angle outputs of the Phantom Omni sensor. To simplify the implementation, and also based on the real situation. The rotational motion of the joints when users controlling the Phantom Omni is assumed to be uniform, therefore the

velocities are considered to be constant, which means the accelerations for three joints have been set to 0. According to the second order equations of motion, the state evolution function of the rotational motion can be expressed as

$$x_t = \begin{bmatrix} \theta_{1,t-1} + V_{1,t-1} \cdot dt + \frac{A \cdot dt^2}{2} \\ \theta_{1,t-1} + V_{2,t-1} \cdot dt + \frac{A \cdot dt^2}{2} \\ \theta_{3,t-1} + V_{3,t-1} \cdot dt + \frac{A \cdot \Delta t^2}{2} \\ V_{1,t-1} + dt \cdot A \\ V_{2,t-1} + dt \cdot A \\ V_{3,t-1} + dt \cdot A \end{bmatrix}$$

Therefore, the parameters for LKF have been set as follow:

State transition matrix F_t:

$$\begin{matrix} 1 & 0 & 0 & dt & 0 & 0 \\ 0 & 1 & 0 & 0 & dt & 0 \\ 0 & 0 & 1 & 0 & 0 & dt \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{matrix}$$

For the state process noise W_t, we experimentally found that $Q_t = \text{diag}[1,1,1,0.01,0.01,0.01]^T$ provides a good model, which means for angles, a standard deviation of 1° is considered as noise, and for velocities, a standard deviation of $0.01^\circ/dt$ is expected.

Measurement transformation matrix H_t:

$$\begin{matrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{matrix}$$

For the measurement noise V_t, according to the phantom sensor, we found that $V_t = \text{diag}[1,1,1]^T$ gives the best result, which means 1° change is allowed for each angle as noise.

The initial state $\hat{x}_{0|0}$ is: $(0,15,-21,0,0,0)^T$

The initial covariance $P_{0|0}$ is an eye matrix since the initial position is known:

$$\begin{matrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{matrix}$$

3.3.2 Unscented Kalman Filter Algorithm

The Unscented Kalman Filter, proposed by Julier and Uhlman [12] is an alternative to the EKF. Different from the LKF which uses Gaussian random variable (GRV) to approximate the state distribution, UKF uses a deterministic sampling approach to represent the state distribution, where a minimal set of carefully chosen sample points are used to capture the true mean and covariance of the GRV. Compare with EKF, UKF elimates the need of derivation and evaluation of Jacobian matrices preserves the normal distributions throughout the nonlinear transformations and partially incorporates contributions of higher order information into the estimates, therefore achieves 3rd order accuracy for any arbitrary non-linear systems. The basic derivation can be summarized as follow:

Considering a random state vector x, a N dimensional vector, propagated through a nonlinear function $y = g(x)$. Assume that x has mean $\bar{x}$ and covariance P_x. To calculate the statistics of y, a matrix X can be formed which contains 2N+1 sigma points X_i with corresponding weights W_i, according to

$$X_0 = \bar{x} \tag{8}$$

$$X_i = \bar{x} + (\sqrt{(N+\lambda)\,P_x})_i, \qquad i = 1, \dots, N \tag{8}$$

$$X_i = \bar{x} - (\sqrt{(N+\lambda)\,P_x})_{i-N}, \qquad i = N+1, \dots, 2N \tag{8}$$

$$W_0^{\,m} = \frac{\lambda}{N+\lambda} \tag{9}$$

$$W_0^{\,c} = \frac{\lambda}{N+\lambda} + (1 - \alpha^2 + \beta) \tag{9}$$

$$W_i^{\,m} = W_i^{\,c} = \frac{\lambda}{2(N+\lambda)} \qquad i = 1, \dots, 2N \tag{9}$$

In above equations, $\lambda = \alpha^2(N+\kappa) - N$ is a scaling parameter, where the α and κ controls the spread of the sigma points around $\bar{x}$, α is usually a small positive value set by 10^{-3}, [13] and κ provide an extra degree of freedom to adjust the higher order moments of the approximation to reduce the overall prediction errors, β is related to the distribution of x and it is usually set by 2 for Gaussian distributions. The expression $(\sqrt{(N+\lambda)\,P_x})_i$ means the *i*th row of the matrix square root of $(N+\lambda)\,P_x$.

Then, these sigma vectors are propagated according to the non-linear function $y = g(x)$, expressed as

$$y_i = g(X_i) \qquad i = 1, \dots, 2N \tag{10}$$

The mean and the covariance for y are approximated using a weighted sample mean and covariance of the sigma points, expressed as

$$\bar{y} = \sum_{i=0}^{2N} W_i^{(m)} y_i \tag{11}$$

$$P_y \approx \sum_{i=0}^{2N} W_i^{(c)} \cdot (y_i - \bar{y}) \cdot (y_i - \bar{y})^T \tag{11}$$

For non-linear dynamical systems, the State Space Model is given as follow:

$$x_t = f(x_{t-1}, u_t) + W_t \qquad W_t = N(0, Q_t) \tag{12}$$

$$z_t = h(x_t) + V_t \qquad V_t = N(0, R_t) \tag{13}$$

In equation (12) and (13), function f and h are both differentiable functions to describe a non-linear system, W_t and V_t are the noises of state and measurement process and both of them are assumed to be zero mean multivariate Gaussian noises with covariance Q_t and R_t.

With respect the same State Space Model (12) and (13), the UKF can be summarized up according to above equations as follow:

Predict:

Firstly, augment the estimated state and covariance to include the mean and covariance of the process noise, expressed as

$$\mathrm{x}^{\mathrm{a}}_{\mathrm{t-1|t-1}} = \left[\hat{x}^{T}_{t-1|t-1} \quad E(W^{T}_{t})\right]^{T} \tag{14}$$

$$\mathrm{P}^{\mathrm{a}}_{\mathrm{t-1|t-1}} = \begin{bmatrix} P_{t-1|t-1} & 0 \\ 0 & Q_t \end{bmatrix} \tag{14}$$

Then, use the augmented state and covariance to derive a set of 2N + 1 sigma points, where N is the dimension of the augmented state. According to equation (8), expressed as

$$\mathrm{x}^{0}_{\mathrm{t-1|t-1}} = x^{a}_{t-1|t-1} \tag{15}$$

$$\mathrm{x}^{\mathrm{i}}_{\mathrm{t-1|t-1}} = x^{a}_{t-1|t-1} + (\sqrt{(N+\lambda)P^{a}_{t-1|t-1}})_{i}, \quad i = 1, \dots, N \tag{15}$$

$$\mathrm{x}^{\mathrm{i}}_{\mathrm{t-1|t-1}} = x^{a}_{t-1|t-1} - (\sqrt{(N+\lambda)P^{a}_{t-1|t-1}})_{i-N}, i = N+1, \dots, 2N \tag{15}$$

Propagate the sigma points through the non-linear transition function f, according to equation (10), expressed as

$$\mathrm{x}^{\mathrm{i}}_{\mathrm{t|t-1}} = f\left(\mathrm{x}^{\mathrm{i}}_{\mathrm{t-1|t-1}}\right), \quad i = 1, \dots, 2N \tag{16}$$

The predicted state and predicted state covariance are then produced by the weighted sigma points, according to equation (11), expressed as

$$\hat{\mathrm{x}}_{\mathrm{t|t-1}} = \sum_{\mathrm{i=0}}^{\mathrm{2N}} \mathrm{W_i}^{(\mathrm{m})} \cdot \mathrm{x}^{\mathrm{i}}_{\mathrm{t|t-1}} \tag{17}$$

$$\mathrm{P}_{\mathrm{t|t-1}} = \sum_{\mathrm{i=0}}^{\mathrm{2N}} \mathrm{W_i}^{(\mathrm{c})} \cdot \left(\mathrm{x}^{\mathrm{i}}_{\mathrm{t|t-1}} - \hat{\mathrm{x}}_{\mathrm{t|t-1}}\right) \cdot \left(\mathrm{x}^{\mathrm{i}}_{\mathrm{t|t-1}} - \hat{\mathrm{x}}_{\mathrm{t|t-1}}\right)^{\mathrm{T}} \tag{17}$$

In above equation, $\mathrm{W_i}^{(\mathrm{m})}$ and $\mathrm{W_i}^{(\mathrm{c})}$ are calculated according to equation (9).

Update:

The predicted state and covariance are augmented again with the mean and covariance of the measurement noise, expressed as

$$\mathrm{x}^{\mathrm{a}}_{\mathrm{t|t-1}} = \left[\hat{x}^{T}_{t|t-1} \quad E(V^{T}_{t})\right]^{T} \tag{18}$$

$$\mathrm{P}^{\mathrm{a}}_{\mathrm{t|t-1}} = \begin{bmatrix} P_{t|t-1} & 0 \\ 0 & R_t \end{bmatrix} \tag{18}$$

Same as the predict process, a set of 2N + 1 sigma points is derived from the augmented state and covariance, expressed as

$$\mathrm{x}^{0}_{\mathrm{t|t-1}} = x^{a}_{t|t-1} \tag{19}$$

$$\mathrm{x}^{\mathrm{i}}_{\mathrm{t|t-1}} = x^{a}_{t|t-1} + (\sqrt{(N+\lambda)P^{a}_{t|t-1}})_{i}, \quad i = 1, \dots, N \tag{19}$$

$$\mathrm{x}^{\mathrm{i}}_{\mathrm{t|t-1}} = x^{a}_{t|t-1} - (\sqrt{(N+\lambda)P^{a}_{t|t-1}})_{i-N}, i = N+1, \dots, 2N \tag{19}$$

Then, the sigma points are propagated through the non-linear transition function h, expressed as

$$\gamma^{\mathrm{i}}_{\mathrm{t}} = h\left(\mathrm{x}^{\mathrm{i}}_{\mathrm{t|t-1}}\right), \quad i = 1, \dots, 2N \tag{20}$$

The predicted measurement (the prediction of the current measurement, given previous observed measurement) and predicted measurement covariance are also produced by the weighted sigma points, according to equation (11), expressed as

$$\hat{z}_t = \sum_{i=0}^{2N} W_i^{(m)} \cdot \gamma_t^i \tag{21}$$

$$P_{z_t z_t} = \sum_{i=0}^{2N} W_i^{(c)} \cdot \left(\gamma_t^i - \hat{z}_t\right) \cdot \left(\gamma_t^i - \hat{z}_t\right)^T \tag{21}$$

The state-measurement cross-covariance matrix can be calculated by

$$P_{x_t z_t} = \sum_{i=0}^{2N} W_i^{(c)} \cdot \left(x_{t|t-1}^i - \hat{x}_{t|t-1}\right) \cdot \left(\gamma_t^i - \hat{z}_t\right)^T \tag{22}$$

Then, the Kalman gain is calculated by

$$K_t = P_{x_t z_t} \cdot P_{z_t z_t}^{-1} \tag{23}$$

The estimate state vector and the state covariance are updated by Kalman gain, expressed as

$$\hat{x}_{t|t} = \hat{x}_{t|t-1} + K_t \cdot [z_t - \hat{z}_t\,] \tag{24}$$

$$P_{t|t} = P_{t|t-1} - K_t \cdot P_{z_t z_t} \cdot K_t^T \tag{25}$$

For the prototype system, we kept the linearity of the motion same, therefore the state vector x and state equation for UKF remains the same as LKF, which is $(\theta_1, \theta_2, \theta_3, V_1, V_2, V_3)^T$. The state evolution function can be expressed as

$$f(x, \Delta t) = \begin{bmatrix} \theta_{1,t-1} + V_{1,t-1} \cdot dt + \dfrac{A \cdot dt^2}{2} \\ \theta_{1,t-1} + V_{2,t-1} \cdot dt + \dfrac{A \cdot dt^2}{2} \\ \theta_{3,t-1} + V_{3,t-1} \cdot dt + \dfrac{A \cdot \Delta t^2}{2} \\ V_{1,t-1} + dt \cdot A \\ V_{2,t-1} + dt \cdot A \\ V_{3,t-1} + dt \cdot A \end{bmatrix} \tag{26}$$

For measurement vector z, we used a different model according to "the Kinematics of Phantom Omni" [14]. Therefore instead of using three angles, we used coordinates obtained by the Phantom Omni sensor, describe as $(x, y, z)^T$. The measurement evolution function is then

$$h(x, \Delta t) = \begin{bmatrix} -sin\theta_1(L_2 sin\theta_3 + L_1 cos\theta_2) \\ -L_2 cos\theta_3 + L_1 sin\theta_2 + L_3 \\ L_2 cos\theta_1 sin\theta_3 + L_1 cos\theta_1 cos\theta_2 - L_4 \end{bmatrix} \tag{27}$$

Where $L_1 = L_2 = 133.35\text{mm}$, $L_3 = 23.35\text{mm}$ and $L_4 = 168.35\text{mm}$ represents the length as Figure 4 shows

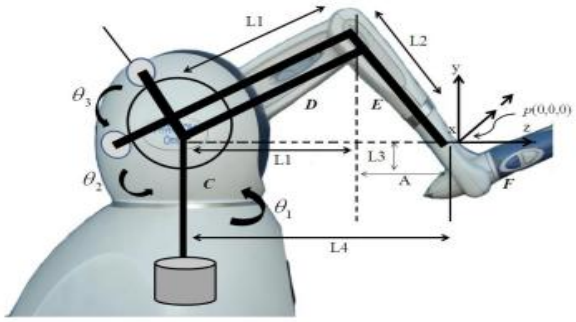

Figure 4. Initial condition of Phantom OMNI [14]

The noise model, initial state, and initial covariance also remain the same as LKF.

3.4 Data points smoothing

The limitation of the prediction algorithm is one of the factors that affect the prediction performance. Kalman Filter algorithms also have one critical limitation: the algorithms (both LKF and UKF) contain the statistical noise of state process and measurement process, making the estimation values floating around the true value. In order to overcome this limitation, the data points need to be smoothed. In this thesis, Savitzky–Golay filter has been applied for the estimation values. The equation of Savitzky–Golay filter can be expressed as

$$D_j = \frac{1}{35}\left(-3 \cdot d_{j-2} + 12 \cdot d_{j-1} + 17 \cdot d_j + 12 \cdot d_{j+1} - 3 \cdot d_{j+2}\right)$$

Where point d_j will be updated by D_j, for 5-point quadratic polynomial, 5 points are used as reference points, therefore 2 additional values need to be predicted.

Figure 5 shows how Savitzky-Golay filter smooth a set of points in curve without greatly distorting the data.

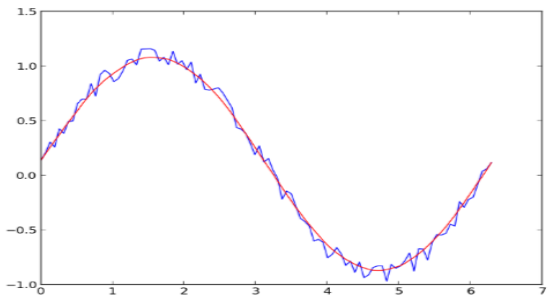

Figure 5. Savitzky-Golay smoothing

3.5 Binocular disparity and Stereoscopy

Since our VR teleportation system is based on video captured by webcam, the captured video frames (See Figure 1) have to be adapted and displayed on the Head-Mounted Display (HMD). Basically, there are two types of HMD, monocular HMD, and binocular HMD. For the prototype, we used Google Cardboard, which belongs to the binocular HMD. Therefore, a technique called stereoscopy has been used for creating two images for left and right eyes based on binocular disparity. Binocular disparity [15] refers to the differences produced when two eyes look at an object from slightly different angles, which results the eyes' horizontal separation, also called parallax. Human's brain uses the binocular disparity to extract depth information. With stereoscopy images, the visual system fuses two images into a single perception and converts the disparity between the two images into the

perception of depth. Figure 6 shows the basic principle of how human's eyes extract the depth information from 2D images

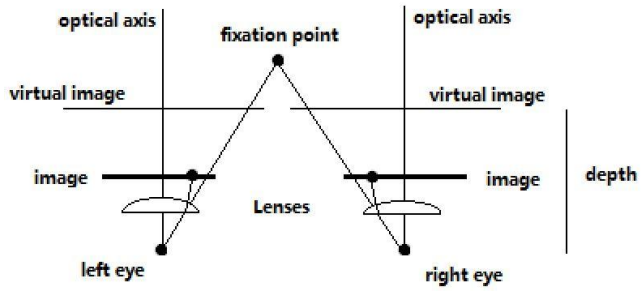

Figure 6. The optical model for both eyes

To simulate the 3D vision with 2D images, for the left eye, the image needs to be shifted to right, and for the right eye, the image needs to be shifted to left. The amount of shift pixel depends on the Interpupillary Distance (IPD) which represents the distance between the centers of the pupils of the two eyes, and also the distance between lenses and eyes. For Google Cardboard used in this prototype, we found that a good shift radio is 1/16 of the width of the image. Specifically, for 640*480 real time frames captured by build-in webcam, we firstly create two duplicated images, and then cut 1/16 of the image from right for the left eye image, and cut 1/16 of the image from left for the right eye image, Figure 7 shows the result of the frames after applying Stereoscopy.

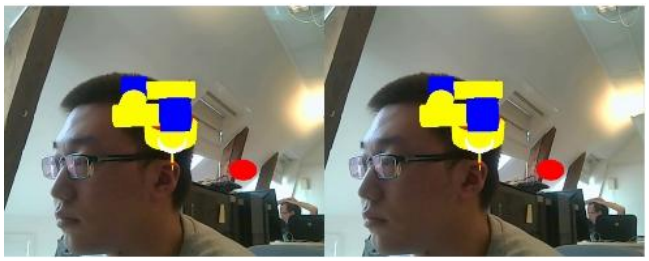

Figure 7. The stereoscopy of the captured frames

3.6 Radial (Optical) distortion

Before display the stereoscopy frames on HMD, there is also another important process needs to be done here, the Radial distortion.

Radial distortion refers to an optical aberration that deforms and bends physically straight lines and makes them appear curvy in images. Generally, Radial distortions are caused by the optical design of lenses and there are three known types of optical distortion: Barrel distortion, Pincushion distortion, and moustache distortion.

Depending on which type of the lens are used, the VR frames need to be adapted for correcting the lens error [16], so that the displayed frames are not deformed in users' eyes. For instance, wide angle lenses cause the barrel distortion, therefore the opposite of barrel distortion, Pincushion distortion is needed to be used to adapt the frames. Conversely, simulating barrel distortion effect on frames corrects the Pincushion distortion cause by telephoto lenses.

According to Brown–Conrady distortion model, also known as decentering distortion, these radial distortions can be corrected by applying suitable algorithmic transformations to the frames. For the prototype, we used a pair of

biconvex lens to assemble with the simplest VR device, Google Cardboard. Therefore, the barrel distortion needs to be simulated for the frames, with the equation of decentering distortion

$$x_d = x_u \cdot (1 + K \cdot r^2 + K \cdot r^4 + \cdots)$$

$$y_d = y_u \cdot (1 + K \cdot r^2 + K \cdot r^4 + \cdots)$$

In above equation, x_d and y_d are the distorted image points and x_u and y_u are undistorted image points, K is the radial distortion coefficient which controls the amount of distortion, $r = \sqrt{(x_u - x_c)^2 + (y_u - y_c)^2}$ is the radial value, where x_c and y_c are the center points of the image.

In practical, the radial distortion equation can be simplified with only the first two terms of the infinite series, expressed as

$$x_d = x_u \cdot (1 + K \cdot r^2 + K \cdot r^4)$$

$$y_d = y_u \cdot (1 + K \cdot r^2 + K \cdot r^4)$$

Figure 8 shows the changes after applying barrel distortion.

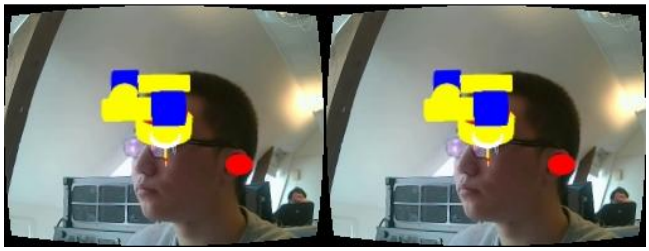

Figure 8. Barrel distortion effect

The final result after applying both Stereoscopy and barrel distortion has been shown as following Figure 9.

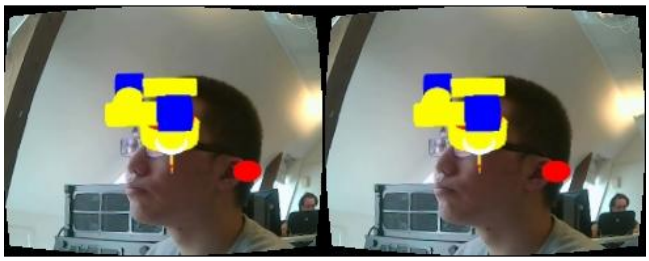

Figure 9. The frames after applying stereoscopy and barrel distortion

4 Implementation and Experiment Result

This chapter has been divided into two parts. The first part describes the experiment for examine how latency affect the user's interactions from effectiveness aspect, the second part describes the comparative study for LKF and UKF to examine how different factors affect the prediction performance.

4.1 Interaction Effectiveness

In order to examine how latency affects the interactions in VR systems, a compare experiment has been done. Firstly, the user controls the robotic arm in real network situation to verify the fundamental latency between server and client. The result shows that the fundamental latency when connect the server and client in real network situation is around 200ms, containing the rendering time of the graphic arm, the computational time of the filtering, the transmission time between server and client, and client to the webpage, the encoding/decoding time of video frames, and the image processing time of adapting video frames for VR.

Then, additional latencies (0ms, 200ms, and 600ms) have been simulated for different amount of latencies (200ms, 400ms, and 800ms). After that, smoothed LKF and UKF have been applied to compensate the latency with different prediction time corresponding to the latencies. In video technology, 24p (24 frames per second) is the commonly used standard for video format. Therefore the prediction times for different simulated latencies are shown in Table 1.

	Latency 200ms	Latency 400ms	Latency 800ms
Prediction time	5+2(frames)	10+2(frames)	20+2(frames)

Table 1. Prediction time for different latencies

To examine the effectiveness of the interactions, user performs the same actions in different settings mentioned above:

1. Move the graphic arm from the initial position to the object.
2. Grab the object and put it down to a certain fixed position.
3. Move the graphic arm back to the initial position.
4. Try to keep the rotational motion velocity constant for every time.

The spending time for above interactions is around 2850ms when user controls the graphic arm locally (without latency).

Table 2 shows the average spending time (10 times for each setting, in order to reduce overall spending time error), and the deviation time (compare with standard spending time) of the user's interactions.

	Real network	Smoothed LKF	Smoothed UKF
200ms Latency	3108ms dt:258ms	2873ms dt:23ms	2893ms dt:43ms
400ms Latency	3351ms dt:501ms	2985ms dt:135ms	3021ms dt:171ms
800ms Latency	3876ms dt:1026ms	3207ms dt:357ms	3483ms dt:633ms

Table 2. Average spending time of the interactions for different settings

The result shows that the latencies slow down the users' actions. The mismatch of the movement of the graphic arm makes user hard to perform the interaction effectively, thus more time is required to perform the same actions. By comparing the latencies with the deviation time, it is clearly to see that for all the cases, the deviation time is greater than the latency, no matter how much the latency is, and along with the latency increased, the deviation time also increased, which shows the more latency, the worse condition for user to perform the interactions.

With Smoothed LKF and UKF applied, the result becomes much better. For 200ms latency with LKF applied, the spending time is close to the standard spending, and the deviation time is 23ms, close to the ideal latency for VR systems. However, for 400ms and 800ms, the deviation times are increased, which means the performance of the filters is reduced. But still, the latency is eliminated to a certain extent. Besides, the smoothed LKF provides better performance compare with smoothed UKF. The comparative study of these two filters has been shown in Chapter 4.2.

Figure 10 shows the view from user's perspective of using the system with 200ms latency and without prediction filters. The left figure shows the movement of robotic arm in real-time, which is simulated locally on client side (Moving from left to right); the right figure shows the received frames on client side, which represents the delayed robotic arm. For analyze purpose, Stereoscopy and barrel distortion are not applied.

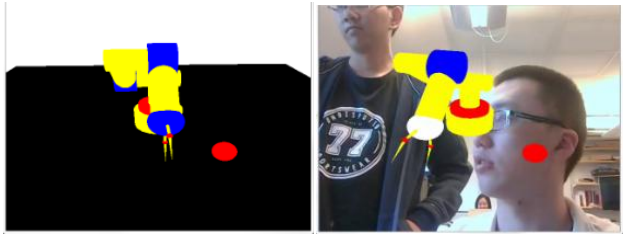

Figure 10. Real time movement Vs. Delayed movement

From the visual point of view, above figure also shows that latency extremely affects the accuracy and effectiveness of the interactions. Users have to wait the delayed robotic arm to catch up their real time motion before they can perform the next action.

Figure 11 and Figure 12 respectively shows the view from user's perspective of using the system with smoothed LKF and UKF applied for compensating the 200ms latency, compare with the real time movement.

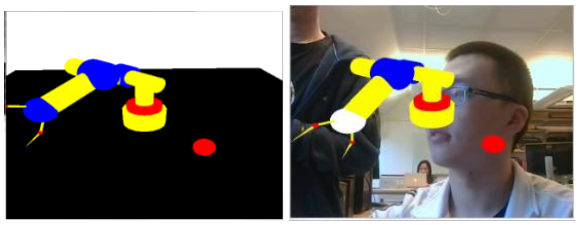

Figure 11. Real time movement Vs. LKF predicted movement under 200ms latency

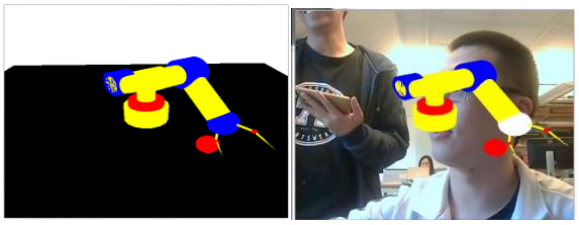

Figure 12. Real time movement Vs. UKF predicted movement under 200ms latency

From the visual point of view, above figures also show that with smoothed LKF and UKF applied, the robotic arm are close to the real time movement, which makes the users easier to perform the actions.

4.1 Performance Comparison

For the performance comparison of smoothed LKF and UKF, the true values of three angles in time domain have been observed under different simulated latencies. (200ms, 400ms, 800ms) In order to quantitatively analyze the filtering effect, Sum of squared errors of prediction (SSE) was used for the whole trace; the formula can be expressed as

$$\sum_{i=1}^{n}[(\theta_1-\widehat{\theta_1})^2+(\theta_2-\widehat{\theta_2})^2+(\theta_3-\widehat{\theta_3})^2]$$

In above equation, $\widehat{\theta_1},\widehat{\theta_2},\widehat{\theta_3}$ are the estimation values represent three angles respectively, $\theta_1,\theta_2,\theta_3$ are the corresponding true values, n is the number of frames.

Figure 13, 14, and 15 respectively shows the comparison result of three angles with smoothed LKF and UKF applied for 200ms latency. The true values were shifted according to the prediction time. (See Table 1)

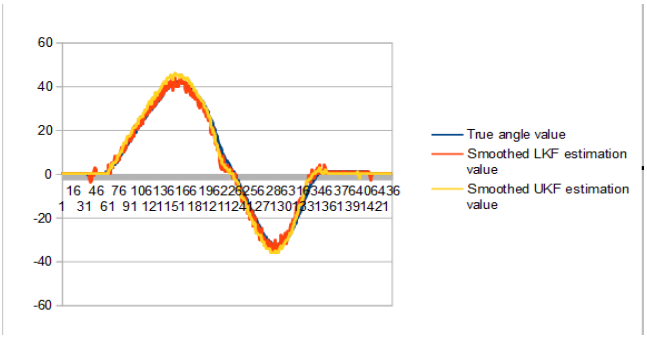

Figure 13. The true value, LKF estimation value, and UKF estimation value of angle θ_1 under 200ms latency.

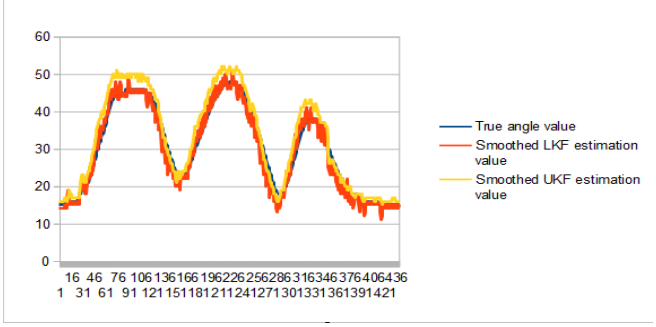

Figure 14. The true value, LKF estimation value, and UKF estimation value of angle θ_2 under 200ms latency.

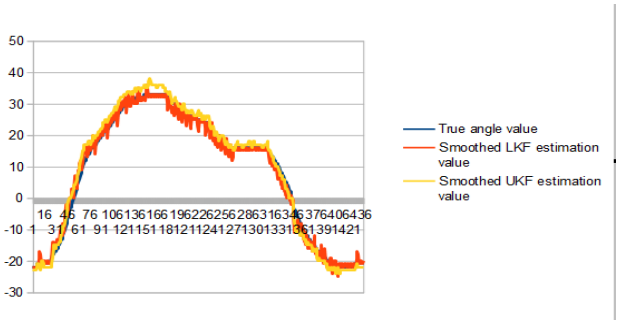

Figure 15. The true value, LKF estimation value, and UKF estimation value of angle θ_3 under 200ms latency.

Table 3 shows the SSE values of same amount of sample frames (340 frames) for different settings under different amount of latencies.

	Smoothed LKF	Smoothed UKF
200ms latency	5769(degree^2)	9035(degree^2)
400ms latency	17312(degree^2)	22634(degree^2)
800ms latency	49179(degree^2)	57729(degree^2)

Table 3. SSE values for smoothed LKF and UKF under different latencies

The table above shows that the prediction time extremely affects the performance of the prediction filters, both LKF and UFK give unacceptable SSE values when the latency increased. The predictions become worse due to the limitation of Kalman Filters. Theoretically, Kalman algorithm estimate the future state based on the previous measurement by updating the covariance and Kalman gain. If users suddenly change their motion (i.e. stop or change direction), the Kalman algorithms will still predict the future frames based on the old measurement and take few steps to adjust the great changes after the new measurement observed. Therefore, a larger prediction time brings worse estimation result. Figure 16, 17, 18 respectively shows the estimation result of smoothed LKF and smoothed UKF for angle θ_1 under different amount of latencies.

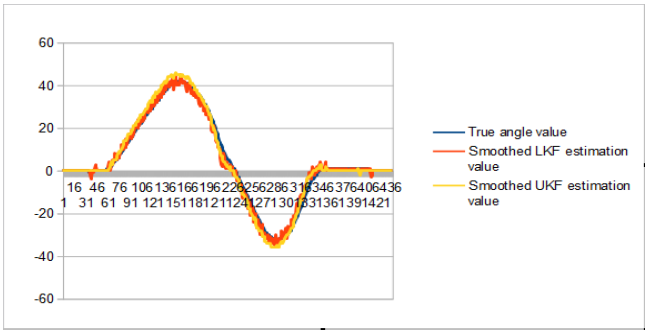

Figure 16. The true value, LKF estimation value, and UKF estimation value of angle θ_1 under 200ms latency.

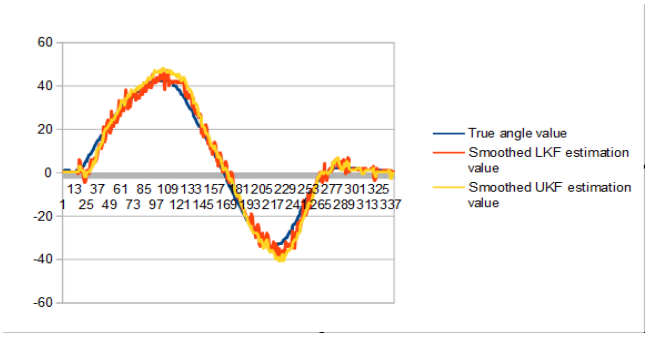

Figure 17. The true value, LKF estimation value, and UKF estimation value of angle θ_1 under 400ms latency.

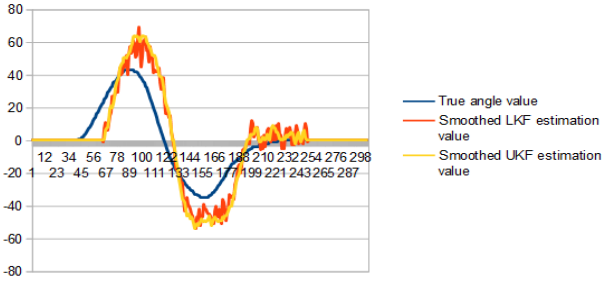

Figure 18. The true value, LKF estimation value, and UKF estimation value of angle θ_1 under 800ms latency.

Above figures show that with 200ms latency, both LKF and UKF provide a satisfactory prediction. With 400ms latency, the prediction is still acceptable but become worse, when the latency increases to 800ms, the prediction is unacceptable, which results the virtual object mismatch in VR frames, thus affect the interactions.

For the computational time, UKF requires a larger computational overhead, Table 4 shows the computational time of LKF and UKF when process the same sample frames (340 frames) for prediction time(different amount of latencies).

	LKF	UKF
5+2(frames)	67ms	631ms
10+2(frames)	76ms	1022ms
20+2(frames)	153ms	1772ms

Table 4. Computation overhead for smoothed LKF and UKF under different latencies

The result above shows that a larger prediction time requires more computational overhead for both LKF and UKF. Compare with LKF, UKF requires 10 times more computation overhead, which becomes the additional latency when display the VR frames.

5 Discussion

The purpose of this thesis was the study of eliminating the latency using different Kalman filters for a VR-based teleoperation system. In order to effectively eliminate the latency, how different factors affect the filters' performances were studied by the implementation, experiment and evaluation.

The result shows, for the linear rotational motion dataset captured from Phantom Omni sensor, the smoothed LKF provides better prediction performance than smoothed UKF, which proves that the linearity of the human motion needs to be considered when choosing prediction algorithm to eliminate the latency for VR systems. The prediction time is also the important factor for prediction algorithms. By simulating different amount of latencies for the prototype system, the result shows that a larger prediction time normally returns worse estimation accuracy. For 200ms latency, LKF provides the best performance where the compensated latency satisfactory for users to perform the interactions. UKF provides slightly worse performance but the compensated latency is still unnoticeable. For 400ms latency, both filters have larger SSE values, which result the virtual objects' mismatch in VR frames. However, compare with the frames with 400ms latency under real network situation, the results of both filters are still acceptable. For 800ms, the predictions are unsatisfactory; the SSE values are extremely big for both filters, which extremely reduce the effectiveness of users' interactions. For computational time, smoothed LKF is faster than UKF since UKF uses a deterministic sampling approach with the sigma points. The prediction time also affect the computational time, where more computation overhead is needed for larger prediction time.

6 Conclusion and Future Work

In Conclusion, both smoothed LKF and UKF provides a satisfactory result for eliminating the latency of the prototype VR teleoperation system, where the effectiveness of the interaction is significantly increased. Compare the performance of both filters, LKF stands out since the human motion is haptic based which means linear rotational motion dataset was used for the prediction. The prediction time affect not only the accuracy of the prediction for both filters but also affect the computational time, where larger prediction time returns worse prediction accuracy and additional computational overhead.

For the future work of this thesis, different type of human motion dataset could be collected such as head motion, body motion, and hand motion, which are the non-linear motions in VR systems. Different prediction algorithm could be explored such as Extended Kalman Filter, Particle Filter, and Wiener filter. A computer graphic based VR system could be implemented, instead of the video based VR system. A real robotic arm could be used instead of graphic robotic arm, as well as the interactive objects.

7 Sustainability Considerations

This thesis provides a promising result with the implementation of a VR-based teleoperation system. The prototype application was aimed to benefit the telemedicine domain, where the idea of combining teleoperation and VR-based telecommunication could be used for developing telemedical haptic device to support in-home care. From sustainability aspect, with the help of telemedical device, the patients in isolated communities and remote regions are able to receive the health care from doctors or specialists without the need of travelling to visit them. Medical treatment such as palpation, medical massage, and even surgery are possible to be achieved from a distance, if the telemedical device is precision enough.

8 Ethical Considerations

For the experiment of this thesis, the autonomy of users has been ensured. Also, all the figures used in this thesis have obtained the full consent of the related people. All the experiment data shown in Chapter 4 is real and integrity, no falsification and deception. The discussion is also based on the experimental data, no conjecture and exaggeration. The prototype application for the experiment could slightly harm the users, since it is a VR-based teleoperation system. It may cause VR sickness if users use it for a long time. Therefore, the experiment of this thesis was running in short time period, and the users can withdraw from the experiment at anytime they want.

All the “original text” used in this thesis has the clearly references, the graphic robotic arm model of the prototype application has been used for study purpose, and it is designed by GameDevGP [10].

9 References

[1] Martijn J. Schuemie, Peter van der Straaten, Merel Krijn, and Charles A.P.G. van der Mast. *Research on Presence in Virtual Reality: A Survey.* CyberPsychology & Behavior., Vol.4. Pages 183-201. (Jul. 2004)

DOI: 10.1089/109493101300117884.

[2] Chorianopoulos, K., Divitini, M., Baalsrud Hauge, J., Jaccheri, L., and Malaka, R. *Entertainment Computing - ICEC 2015*. 14th International Conference, ICEC 2015, Trondheim, Norway, September 29 - October 2, 2015, Proceedings.

[3] Yulita P. 2008. *Reducing Latency When Using Virtual Reality for Teaching in Sport.* In 2008 International Symposium on Information Technology, Vol. 3, Pages 1-5.(Aug.2008) DOI: 10.1109/ITSIM.2008.4632076

[4] Gregory F. Welch. *HISTORY: The Use of the Kalman Filter for Human Motion Tracking in Virtual Reality*. Presence. Vol. 18, No. 1, Pages 72-91 (Feb.2009).

DOI= 10.1162/pres.18.1.72

[5] Cui, J. Fu, J. Tao, Z. Tong, L. Hu, G. Zhang, Y. Li, X. 2015. *Trajectory Tracking of Joint Based on Kinect.* In Proceedings – 2015 7th International Conference on Intelligent Human-Machine Systems and Cybernetics, IHMSC 2015, Vol. 1, 20, Pages 330-333. (Nov.2015)

DOI: 0.1109/IHMSC.2015.124

[6] Matthew Edwards. , Richard Green. *Low-Latency Filtering of Kinect Skeleton Data for Video Game Control.* Proceedings of the 29th International Conference on Image and Vision Computing New Zealand Pages 190-195. (2014). DOI= 10.1145/2683405.2683453.

[7] Joseph J. LaViola Jr. *A comparison of unscented and extended Kalman filtering for estimating quaternion motion.* American Control Conference, 2003. Proceedings of the 2003, Vol.3, Pages 2435-2440. (Jun.2003)

DOI= 10.1109/ACC.2003.1243440

[8] A.UmaMageswari, J.Joseph Ignatious, R.Vinodha. *A Comparitive Study Of Kalman Filter, Extended Kalman Filter And Unscented Kalman Filter For Harmonic Analysis Of The Non-Stationary Signals*. International Journal of Scientific & Engineering Research, Vol.3, Issue 7.(Jul.2012)

[9] Joseph J. LaViola Jr. *A Testbed for Studying and Choosing Predictive Tracking Algorithms in Virtual Environments.* Proceedings of the workshop on Virtual environments 2003. Pages 189-198. (2003) DOI= 10.1145/769953.769975

[10] *3D Robot Arm Simulation in OpenGL*. (Jan.2014)

< https://gamedevgp.wordpress.com>

[11] R. E. Kalman. *A New Approach to Linear Filtering and Prediction Problems.* Journal of Basic Engineering, VOL. 27, No.1 (Mar.1960). DOI= 10.1115/1.3662552.

[12] Simon J. Julier, Jeffrey K. Uhlrnann and Hugh F. Durrant-Whyte. *A new approach for filtering nonlinear systems.* American Control Conference, Proceedings of the 1995 Vol.3, Pages 1628 – 1632 (Jun.1995).

DOI= 10.1109/ACC.1995.529783

[13] Eric A. Wan and Rudolph van der Merwe. *The Unscented Kalman Filter for Nonlinear Estimation*. Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000. AS-SPCC. The IEEE 2000. Pages 153 – 158(Oct.2000). DOI= 10.1109/ASSPCC.2000.882463

[14] Alejandro J. 2009. *Phantom Omni Haptic Device: Kinematic and Manipulability.* In Electronics, Robotics and Automotive Mechanics Conference. Pages 193-198.(Sep.2009)

DOI: 10.11-9/CERMA.2009.55

[15] Joseph S. Lappin. What is binocular disparity? *Front Psychol. Vol.5. (Aug.2014).*DOI= 10.3389/fpsyg.2014.00870

[16] Wolfgang Hugemann. Correcting Lens Distortions in Digital Photographs.(Jan.2011)

<http://www.imagemagick.org/Usage/lens/>