Kalman Filtering and Prediction for Hand Tracking

An Advanced DSP Project

For

Dr. R.D. Dony / ENGG*6560

Ben Miners

April 20, 2001

Abstract

A need exists in today's society for intuitive hand gesture interaction between human and machine. Although a non-invasive technique is desirable, using a magnetic system for hand tracking avoids current issues with non-invasive techniques and allows one to concentrate on other aspects of gesture interaction. Unfortunately, magnetic tracking systems are highly susceptible to electromagnetic interference and signal noise. Filtering this sensory data will improve the quality of the input for interpretation, along with providing a measure of error for subsequent processes in hand gesture interpretation. This project evaluates the suitability of the extended Kalman filter and unscented Kalman filter for use in filtering and prediction of the hand movement as measured through a magnetic tracking system.

A computer simulation was used to evaluate the performance of the extended and unscented Kalman filters for use in hand tracking. This simulation used a simplified model of the hand dynamics since an accurate model for human body dynamics is a large problem on its own, and a non-linear observation model was used in each of the filters. Process and measurement noise was introduced and the performance was analyzed using the average mean squared error. The unscented Kalman filter resulted in a lower average mean squared error with a lower estimated variance on average. The improvements observed through use of the unscented Kalman filter were more significant in highly non-linear or discontinuous scenarios.

The simplicity of implementing a complex process or measurement model is a definite advantage of the UKF over the EKF, however, this simplicity comes at the expense of CPU time. The Matlab implementation of the UKF required almost twice as much CPU time as the equivalent EKF implementation in this project. Future research includes evaluation of the algorithms using real-world test data and investigation of alternative process models that more accurately model hand movement. Alternative techniques that are potentially useful and should be investigated further include the covariance intersection technique and interacting multiple models Kalman filter.

Table of Contents

1 Introduction

The dependence of humans on machines for assistance with a diverse range of everyday tasks is steadily increasing while interaction is often restricted to providing tedious and strict sets of instructions through a traditional keyboard and mouse and receiving feedback visually. These interfaces, although state of the art decades ago, leave considerable room for improvement in today's society. A need exists for intuitive hand gesture interaction between human and machine in which the machine does not only recognize gestures, but also interprets their meaning. The ability to interpret gestures will improve the ease and efficiency of society's interaction with machines, allowing the description of relatively complex tasks to machines without the traditional set of detailed instructions.

Hands are an important component of communication, and have historically been an area of interest as illustrated in Leonardo da Vinci's "Study of Arms and Hands" (*Figure 1*). There is a recent growth in interest and research in many aspects of hand gesture recognition, however, significant progress is still required before gesture recognition reaches the level at which it can form part of a primary interface between humans and machines in today's society. Machines must be able to interpret the meaning of continuous phrases of gestures before natural, intuitive communication can occur.

Gesture recognition alone will provide machines with the ability to identify simple commands, or the raw content of communication from humans. To interact seamlessly with humans in today's society, a machine must not only identify what commands were issued through gestures, but it must also identify the context and meaning of this information, as is accomplished in human face-to-face communication. Recognizing a gesture in isolation can facilitate simple commands, but in a natural HMI, individual gestures may have many different meanings depending on their context. Once machines interpret the meaning of continuous phrases of gestures, the communication possibilities between humans and machines expand dramatically from simple commands to a language in which the expression of complex concepts and instructions is possible.

The quality of the information used as input into the interpretation process will affect the ability of machines to interpret meaning from gesture. The sensory data acquired from input hand configuration, position and orientation information is commonly passed through a filter to reduce noise and compensate for the limited accuracy of the sensors. This project evaluates the performance characteristics of the extended Kalman filter and the unscented Kalman filter in the task of filtering hand position for the purpose of hand gesture interpretation.

In addition to reducing noise in the hand position information, the Kalman filtering techniques can be used for prediction. Prediction is useful both for compensation of obstructed measurements, and to meet real-time constraints imposed on an interactive gesture interpretation system.

Input information to the alternate forms of the Kalman filter is generated using a simulation of a 6DOF magnetic tracking system. This commonly used, and relatively inexpensive tracking technology determines hand position and orientation using a fixed electromagnetic transmitter and a mobile electromagnetic receiver on the hand. The receiver measures the proximity and angle to the transmitter, requiring a non-linear co-ordinate transformation from the measurements to the system model, where hand position relative to the body is used.

A gesture interpreting HMI provides the ability for natural interaction both in noisy environments where speech may not be feasible, and to complement alternate modalities of communication. Additional applications for a gesture interpreting system include a primary interface to service robots where household noise can obscure speech, and as a complementary interface in tele-assistance where considerable information can be communicated in a relatively short duration using a gesture language in combination with other modalities.

Figure 1 – "Study of Arms and Hands", Leonardo da Vinci (c. 1474)

2 System Overview

Before hand gestures can be interpreted, the basic hand configuration and orientation information must be acquired. A diverse range of approaches exist to accomplish this task, including statistical appearance based techniques [Freman and Roth, 1995], [Darrell and Pentland, 1995], feature-based techniques using elastic graph matching [Triesch and Malsburg, 1996], principle component analysis [Moghaddam and Pentland, 1995] [Hamdan et al., 1999] [Iwai et al., 1999], 3D model based [Ahmad, 1995] [Rehag and Kanade, 1993], and physical glove-based techniques [Lu et al., 1997].

Ideally, acquisition of hand configuration and orientation data should be non-invasive and transparent to the user. Unfortunately, existing non-invasive methods to acquire hand information are still in their infancy, susceptible to minor environmental changes, and may not provide adequate precision to distinguish between similar hand gestures. From a quality of information standpoint, a good data acquisition technique is use of a glove with joint angle sensors in combination with a six degree-of-freedom (6DOF) tracking device.

Although use of a glove with joint angle sensors and a 6DOF tracking device are not transparent to the user, the results of the interpretation can be applied to model-based vision recognition systems as their performance improves. Data from the glove with joint angle sensors can provide hand configuration, but does not provide orientation or hand motion information. The 6DOF tracking device is required to obtain hand position and orientation information required for gesture interpretation. Please refer to *Figure 2* for an illustration of hand configuration, orientation and position information.

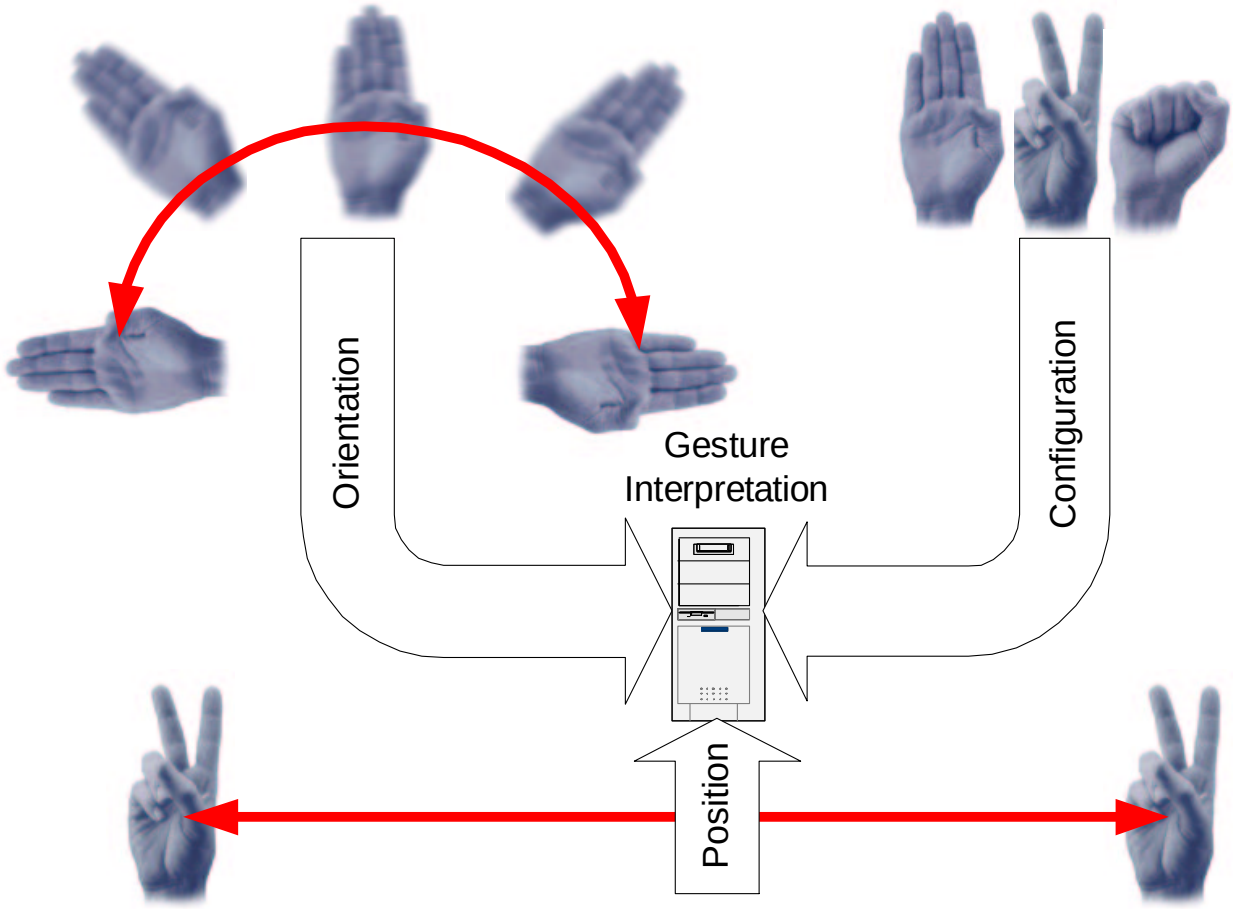

Figure 2 – Hand Position, Orientation, and Configuration

2.1 Selected Tracking Technologies

Some commercially available magnetic and optical tracking sensors are briefly summarized in this section. Magnetic tracking sensors are the most common in body tracking research, and optical tracking sensors are generally used in higher-end applications including movies and videogames. Other available technologies include ultrasound, inertial, and mechanical sensors. Magnetic tracking systems were selected for use in this project due to their relatively low cost and reasonable precision.

It should be noted that the specifications listed are manufacturer specifications, and are more than likely best-case results under ideal conditions.

2.1.1 Magnetic

Although magnetic tracking systems are less expensive than optical systems by at least an order of magnitude, they achieve a sampling rate approximately an order of magnitude less.

One of the major problems in using magnetic tracking systems is their susceptibility to electromagnetic interference. In any metallic objects are nearby, including within the floors or ceiling, the electromagnetic fields generated by the transmitter will be distorted and corrupt the position and orientation information in the receiver.

Another significant disadvantage of magnetic tracking systems is the fact that the sensors are on the object being tracked, either requiring cables or RF communication to transfer data from the object being tracked to where it is being processed.

Two different magnetic tracking systems are currently used, AC and DC. The AC technique used by Polhemus generates continuously rotating magnetic fields, and produces low latency measurements, but is more susceptible to interference from metallic objects than the DC technique. The DC technique generates a pulse along the transmitting coil, and the sensor measures the field a short time later to reduce the effects of Eddy fields around nearby metallic objects. This technique also requires measurement of the ambient magnetic fields before each position/orientation measurement, resulting in potentially longer latency than AC systems.

2.1.1.1 Ascension Flock of Birds

Technology: DC pulsed electromagnetic (with a fluxgate magnetometer)

Latency: 22 ms

Update Frequency: 144 Hz

Accuracy: 1.8mm RMS, 0.5 degrees RMS

Resolution: 0.5mm @ 30.5 cm, 0.1 degree @ 30.5 cm

2.1.1.2 Polhemus FasTrack

Technology: Continuous AC electromagnetic

Latency: 4 ms

Update Frequency: 120 Hz

Accuracy: 0.03” RMS positional, 0.15 degrees RMS (at 30”)

Resolution: 0.0002 inches of error per inch of transmitter & receiver separation, 0.025 degrees orientation

2.1.1.3 Polhemus IsoTrak II

Technology: Continuous AC electromagnetic

Latency: 20 ms

Update Frequency: 60 Hz

Accuracy: 0.1” RMS positional, 0.75 degrees RMS (at 30”)

Resolution: 0.0015 inch per inch of transmitter/receiver separation, 0.1 degrees orientation

2.1.1.4 Northern Digital Aurora

Technology: DC electromagnetic

Accuracy: 1mm RMS, 0.5 degrees RMS orientation

Update Frequency: 20 – 60 Hz

2.1.2 Optical

The high cost of optical tracking systems is related to the high sampling rate, passive sensors, and high computational complexity to calculate orientation information. In magnetic trackers, the receiver is on the object being tracked and directly measures distance and orientation, whereas in optical trackers, the receiver is external to the object being tracked and generally uses stereo-vision techniques to calculate distance. Orientation is calculated using the positions of multiple passive or active sensors.

Similar to the problems with metallic objects corrupting the signals in magnetic tracking systems, opaque objects will occlude the passive sensors, corrupting the signals of an optical tracking system. Reflective objects (jewlery, wet surfaces) and changes in lighting conditions, backgrounds, and clothing can also corrupt the tracking signal.

2.1.2.1 Northern Digital Optotrak

Uses active infrared markers on object being tracked

Accuracy: 0.1 mm for x & y co-ordinates @ 2.5 m from sensor,

0.15mm for z co-ordinate @ 2.5 m from sensor

3D resolution: 0.01 mm @ 2.5 m

2.1.2.2 Ascension LaserBird

Accuracy: 1mm RMS @ 1m, 1 degree

Resolution: 0.1mm, 0.1 degree

3 Magnetic Tracking Systems

The focus of this project is on the filtering and prediction of hand position information from a 6DOF tracking system. Magnetic tracking systems were selected for use in hand gesture interpretation due to their relatively low cost, reasonable precision, and no requirement for an optical line-of-sight. Current research in other fields requiring tracking systems also considers magnetic tracking systems to avoid the line-of-sight issues [Cleary et al., 2001]. Unfortunately, magnetic tracking systems are highly susceptible to electromagnetic interference and signal noise.

The 6DOF tracking system will be attached to the forearm at a small distance from the hand to minimize the obstruction of hand motion, while still measuring approximate hand position. Please refer to *Figure 3* for an illustration of the components in a magnetic hand tracking system.

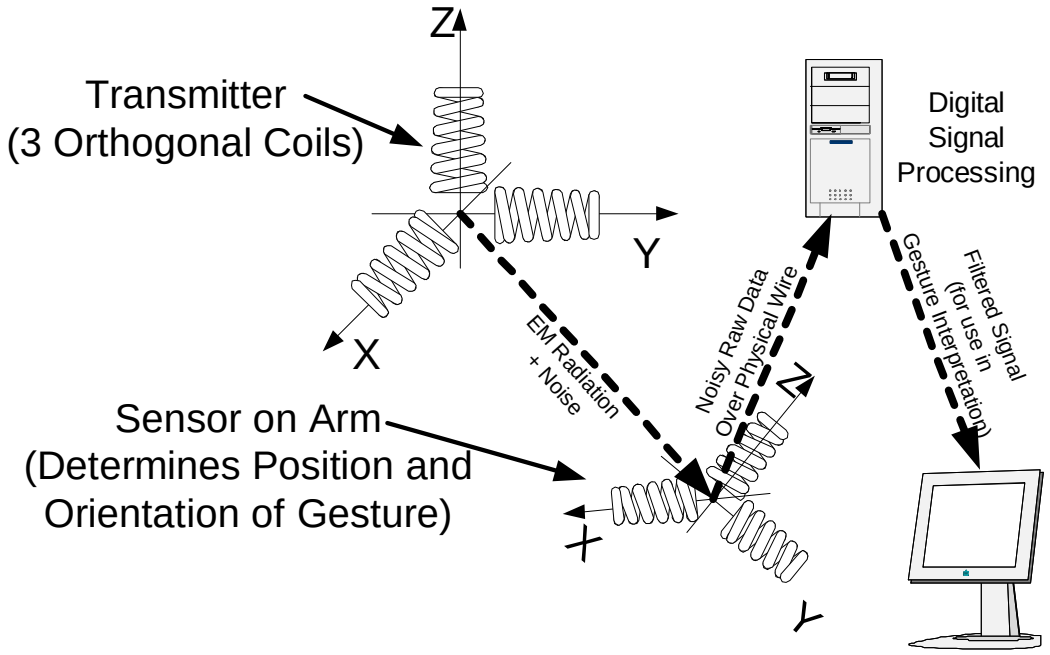

Figure 3 – Magnetic Hand Tracking System Components

The signals measured by the 6DOF tracking system are the distance and angles from the transmitter, and the orientation of the receiver with respect to the transmitter. These measurements are determined using three orthogonal coils in the transmitter to produce orthogonal fields that are measured by three orthogonal coils in the receiver. As the distance between the transmitter and receiver increases, the field strength decreases and less current is induced in the receiver. Similarly, as the orientation of the receiving coils rotate out of alignment from the transmitting coils, the current induced in the receiving coils is reduced. This section describes why Cartesian co-ordinates are not representative of the actual measurement, and why spherical co-ordinates are used in the measurement model for the Kalman filters.

The magnetic field intensity of a coil can be described using the following equation based on Ampere's Law and illustrated in *Figure 4*: [Smith and Dorf, 1992]

$$B = \frac{\mu N i}{2L}\left(\cos(\alpha) + \cos(\beta)\right)$$

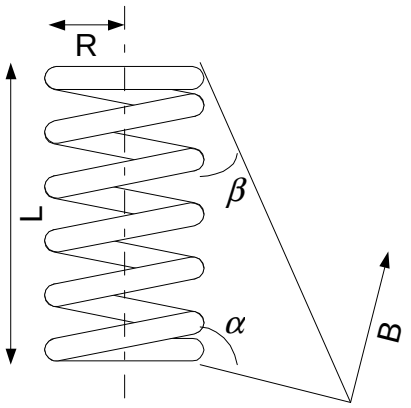

Figure 4 – Field Intensity of a Coil

Since the emitting coils are very small relative to the distance between the sensor on the arm and the emitting coils, this equation can be approximated using: [Baillot and Rolland, 1996]

$$B \approx \frac{\mu N i R^2}{2r^3}$$

With perpendicular and parallel components to the line between the sensor and emitter as follows an illustrated in *Figure 5* (based on [Baillot and Rolland, 1996]):

$$B_r = \frac{\mu N i R^2 \cos(\theta)}{2r^3}$$

$$B_\theta = \frac{\mu N i R^2 \sin(\theta)}{2r^3}$$

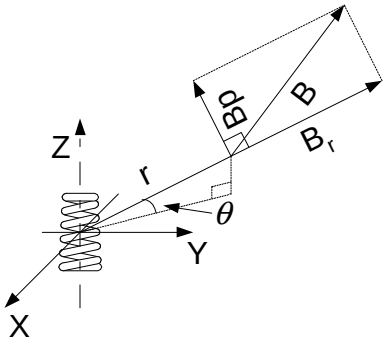

Figure 5 – Orthogonal Components of Magnetic Field

Where,

$i = Ie^{-j\omega t}$ is the current in the coil at time t, with frequency ϖ, maximum current amplitude I,

r is the distance between the sensor and transmitter,

θ is the angle between the axis of the transmitting coil and the line between the transmitter and sensor,

N is the number of coils in the transmitting coil,

R is the radius of the transmitting coil,

L is the length of the transmitting coil, and

μ is the permeability of the air between the transmitter and sensor

The relationship between magnetic field intensity (B) and the measured current (i) at the sensor can be approximated using the field intensity at the center of a very short coil: [Smith and Dorf, 1992]

$$i \approx \frac{4BR_{sensor}}{N\mu_{sensor}}$$

Where,

R_{sensor} is the radius of the sensor coil, and

μ_{sensor} is permeability of the material used in the core of the sensor coil

As described in [Baillot and Rolland, 1996], the position and orientation of the sensor relative to the transmitter can be determined using measurements of current along each of the three orthogonal sensor coils for three different transmission fields, resulting in a system of nine equations that must be solved.

In most practical implementations of magnetic tracking systems, fluxgate magnetometers are used rather than bare coils. In fluxgate magnetometers, two windings are wrapped around a core. One winding is used to produce a magnetic field that saturates the core. The second winding measures the change in magnetic field after the first winding stops producing a magnetic field. Rather than using current to measure the magnetic field intensity, the

rate of change of the field is used, but result in determining the same measurements - distance and angles from the transmitter to receiver.

4 Filtering and Prediction Techniques

As previously identified in literature, magnetic tracking systems are highly susceptible to both orientation and position noise, and the noise increases with the distance between the receiver and transmitter. Minimizing the noise in the tracking signal will improve the quality of hand gesture interpretation.

Wiener filters, although optimal in the least mean-squared error sense, require a priori knowledge of the statistical properties of both the signal and noise (auto and cross-correlation) and require the entire set of data. Traditional Kalman filters also require statistical properties of the signal, but are recursive in nature, reducing the computational complexity and storage requirements since all previous information does not need to be re-processed with the introduction of new observations.

Traditional Kalman filters require an accurate linear model of the system dynamics, along with an accurate linear model of the observation process to be optimal in a least mean-squared error sense. In scenarios where the system dynamics and / or observation models are non-linear, such as the conversion from spherical to Cartesian co-ordinates in a hand tracking system, the standard Kalman filter is no longer optimal.

A common approach to apply the recursive least squares Kalman filter to non-linear models is through the use of the extended Kalman filter. The extended Kalman filter (EKF) linearizes the model(s) to provide sub-optimal output using the standard Kalman filter equations. This technique is considered an ad-hoc filter since a proof cannot be derived in general to guarantee optimality in terms of the minimum mean squared error. If the linearization is not a good local approximation of the model, instability may result. Another drawback to the EKF is the computational complexity of calculating the required partial derivatives with each new sample. Several alternatives to the EKF have been proposed in recent research, including the unscented Kalman filter (UKF) introduced by Julier and Uhlmann [Julier et. al, 1995], and covariance intersection techniques developed by the Covariance Intersection Working Group [CI, 1997].

The unscented Kalman filter proposed by Julier and Uhlmann avoids the use of Jacobian matrices, simplifying the implementation of the filter and potentially reducing its complexity. In addition, the authors claim that the unscented Kalman filter provides a more accurate estimation result, reducing the potential instability issues that can arise with the EKF when the linearization is not a good approximation.

The covariance intersection technique provides a method to combine estimates without knowledge about their correlation. This technique guarantees that the covariance estimate will not be underestimated, resulting in a non-divergent filter even when the observations are not statistically independent. When the observations are statistically independent, the covariance intersection technique will result in less accurate results than the EKF. For this project, it will be assumed that a single hand position sensor is used, and observations are statistically independent, eliminating any advantage the covariance intersection technique may provide over the UKF and EKF.

Kalman prediction techniques can be used to reduce the lag in magnetic tracking systems, but the lag is replaced with potentially erratic positioning since a model for general hand motion will never be perfect.

A computer simulation was used to evaluate the performance of the extended and unscented Kalman filters for use in hand tracking. This simulation used a simplified model of the hand dynamics since an accurate model for human body dynamics is a large problem on its own, and a non-linear observation model was used in each of the filters. Process and measurement noise was introduced and an appropriate performance measure was used to evaluate the two techniques.

5 Linear Discrete Kalman Filter

The Kalman filter is an optimal recursive least squares estimator under the following conditions:

- Hand dynamics can be accurately modeled with a linear model
- A linear, accurate observation model exists
- Noise affecting the hand dynamics and sensor data has known covariance and is a Gaussian process with zero mean

The basic principle of the Kalman filter is to estimate successive states of the system based on the current estimated state of the system and (noisy) observations. The current estimated state of the system and the

observation are combined using the correlation matrices of the error in the current estimate and the error in the (noisy) observation. This technique is computationally efficient since the entire sequence of previous observations does not require reprocessing with the introduction of new observations. The information in the current state and its error correlation matrix summarizes the previous observations.

Although the linear discrete Kalman filter cannot be used directly with non-linear measurement or process models, it is summarized here for comparison with the EKF and UKF in subsequent sections. The notation from [Haykin, 1996] is used, which is summarized in section 14 (page 28).

The Kalman gain (a measure of the relative confidence between our past estimates and the new observation) is calculated,

$$\mathbf{G}(n) = \mathbf{K}(n, n-1)\mathbf{C}^H(n)\left(\mathbf{C}(n)\mathbf{K}(n, n-1)\mathbf{C}^H(n) + \mathbf{Q}_2(n)\right)^{-1}$$

The innovations (error in our estimate) are calculated with,

$$\boldsymbol{\alpha}(n) = \mathbf{y}(n) - \mathbf{C}(n)\hat{\mathbf{x}}(n \mid Y_{n-1}))$$

The estimate is updated with the measurements using the innovations weighted by the Kalman gain,

$$\hat{\mathbf{x}}(n \mid Y_n) = \hat{\mathbf{x}}(n \mid Y_{n-1}) + \mathbf{G}(n)\boldsymbol{\alpha}(n)$$

The correlation of the error in this estimate is calculated,

$$\mathbf{K}(n) = \left(\mathbf{I} - \mathbf{G}(n)\mathbf{C}(n)\right)\mathbf{K}(n, n-1)$$

The estimate for the next iteration based on current observations is predicted,

$$\hat{\mathbf{x}}(n+1 \mid Y_n) = \mathbf{F}(n)\mathbf{x}(n \mid Y_n)$$

And the correlation of the error in the predicted state is calculated,

$$\mathbf{K}(n+1, n) = \mathbf{F}(n+1, n)\mathbf{K}(n)\mathbf{F}^H(n+1, n) + \mathbf{Q}_1(n)$$

These are summarized in *Figure 6* (based on [Haykin, 1996] and [Brown, 1997])

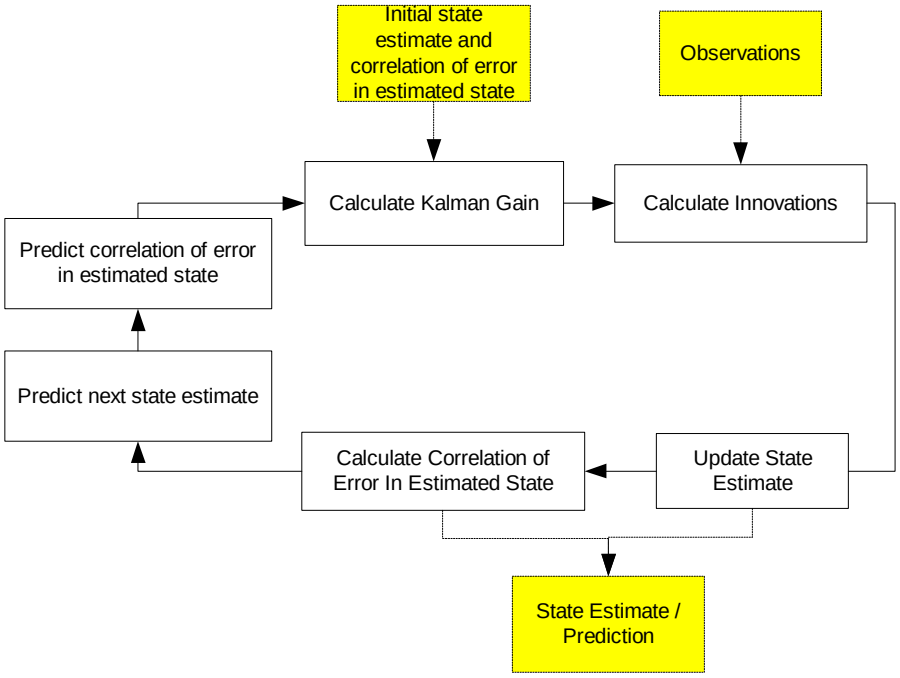

Figure 6 – Kalman Filter Summary

The ability to determine both an estimate of the system state and a measure of its error is potentially very useful in gesture interpretation. Without the correlation matrix of the estimate error, an assumption would have to be made about every estimate of hand position and without additional information, the accuracy of estimates would

be indistinguishable. With the correlation matrix, a high error in a hand position estimate can be considered in subsequent stages of gesture interpretation, as summarized in *Figure 7*.

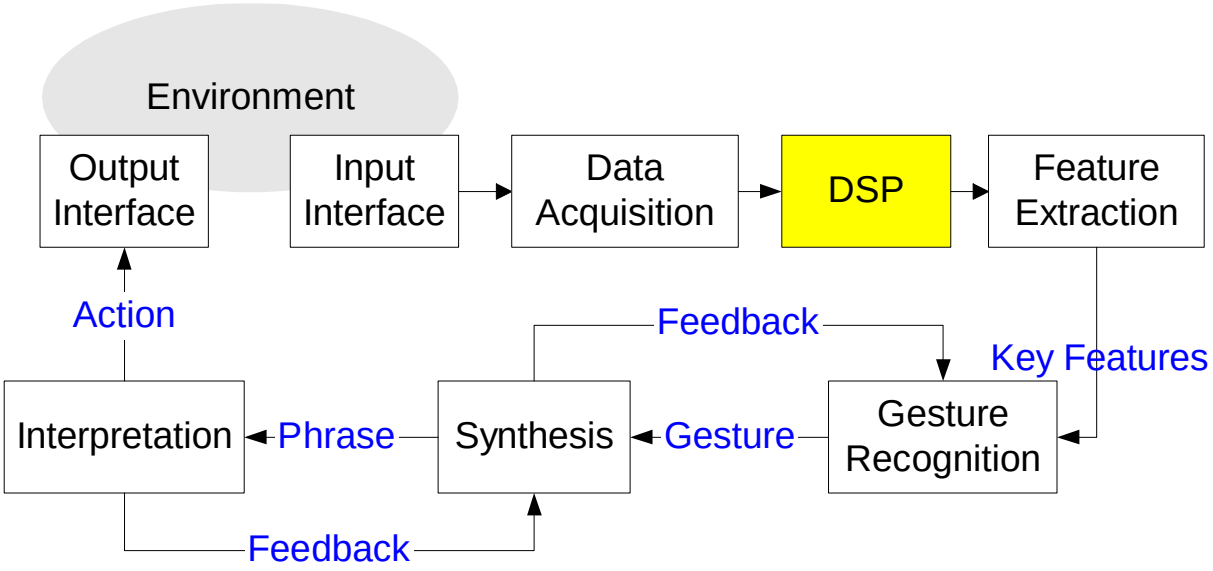

Figure 7 – Gesture Interpretation Overview

Before Kalman filters can be applied to gesture interpretation, a model of the system dynamics, a model of the observation process, and the statistical properties of both the system and observation noise are required.

6 Process Model

In any Kalman-based filter, both a model of the process and a model to convert from a system state to measurement variables are required. In hand gesture interpretation, the process model is a model of the dynamics of the hand.

Although human hand movement cannot be reliably predicted, a simplified hand model can be used to achieve sub-optimal filtering and prediction. As specified in the project proposal, a simplified hand model will be used where the hand is represented as a point in space with constant velocity. Any acceleration of the hand is modelled as white noise. This technique is commonly used to track objects that are difficult to model [Kohler, 1997].

The relationship between the process and measurements is illustrated in *Figure 8*, where the state is the position and velocity of the hand in Cartesian co-ordinates, and the measurement using a magnetic tracking device is in spherical co-ordinates.

It is assumed that the process noise and the measurement noise are independent, since measurement noise is based on magnetic interference and process noise is based on hand acceleration.

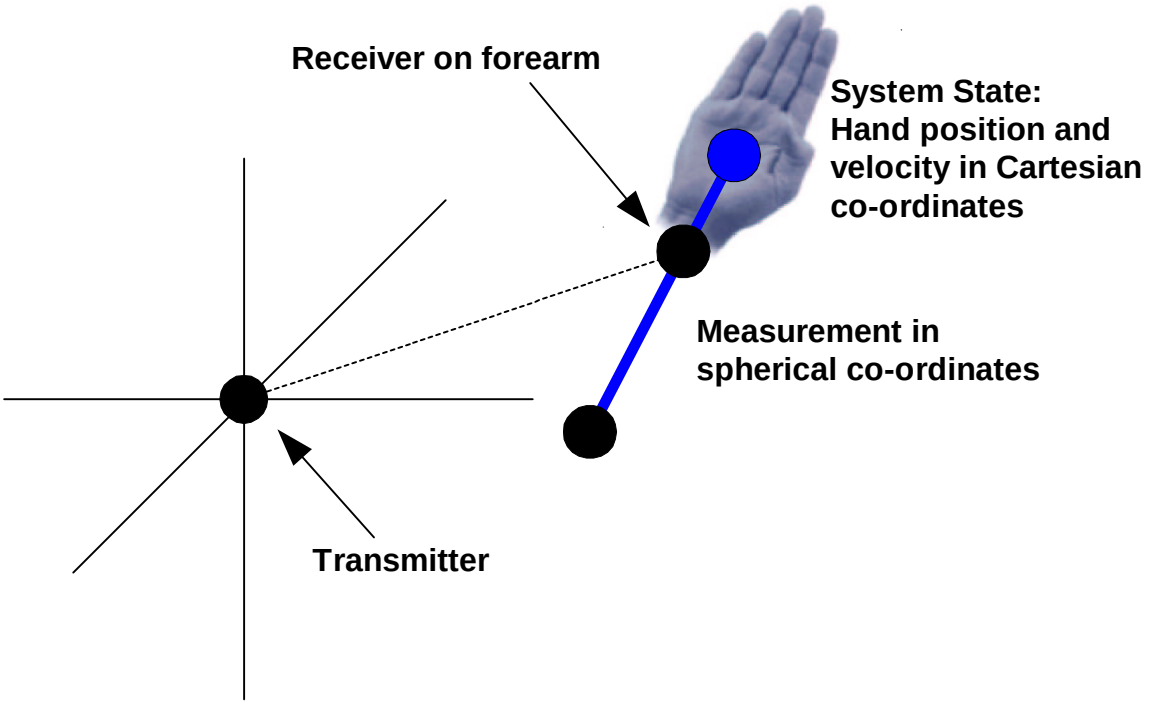

Figure 8 – System and Measurement Model Relationship

6.1 Process Model Equations

Using the simplification that the dynamics of the hand follows a linear motion pattern, the system model is as follows (adopting the variable naming conventions used in [Haykin, 1996]):

$$\mathbf{x}(n+1) = \mathbf{F}(n+1, n)\mathbf{x}(n) + \mathbf{v}_1(n)$$

Where **x** is a state vector that describes the state of the hand, **F** is the (MxM) state transition matrix, defining the relationship between **x**(n) and **x**(n+1) or how the state of the hand changes between time n and time n+1. Since the contribution of muscles in the arm and decisions on how the hand will move cannot be reliably modeled, it is included in the process noise vector, $\mathbf{v}_1$. The Kalman filter derivation is based on the process noise ($\mathbf{v}_1$) being a random stationary process with normal distribution and a mean of zero. Neglecting the muscles in the arm and the decisions of the person introduces inaccuracies into the model due to the simplification.

Since only the magnetic tracking receiver measurements are available, estimates of the process state must be used. **x**(n) and **x**(n+1) are hidden variables to the system. The resulting equation using the estimates of the process state is:

$$\hat{\mathbf{x}}(n+1) = \mathbf{F}(n+1, n)\hat{\mathbf{x}}(n) + \mathbf{v}_1(n)$$

Since it was assumed the hand follows a linear motion model, Newton's equations of motion can be used to model the motion:

$$x = x_0 + vt + \frac{1}{2}at^2$$

Assuming a constant velocity results in a = 0 and any acceleration that does occur will be considered process noise ($\mathbf{v}_1$). Using Newton's equations of motion in the state of the and results in the following state matrix:

$$\mathbf{x}(n) = \begin{bmatrix} \mathbf{P}(n) \\ \mathbf{V}(n) \end{bmatrix}$$

$$\mathbf{x}(n) = \begin{bmatrix} p_x(n) \\ p_y(n) \\ p_z(n) \\ v_x(n) \\ v_y(n) \\ v_z(n) \end{bmatrix}$$

Where **P**(n) is a 3 element vector of the position at time n in Cartesian co-ordinates, and **V**(n) is a 3 element vector of the velocity at time n. The state transition matrix is then:

$$\mathbf{F}(n+1,n) = \begin{bmatrix} \mathbf{I} & \Delta t\mathbf{I} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}$$

$$\mathbf{F}(n+1,n) = \begin{bmatrix} 1 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Where Δt is the time interval between time n and time n+1.

This state transition matrix allows the use of dead reckoning to estimate the position of the hand until new measurements are available.

7 Measurement Model

If the measurement errors in each component of the hand movement were uncorrelated, tracking of the hand can be simplified by using independent Kalman filters for each component. Unfortunately, determining position in a magnetic tracking system is based on spherical co-ordinates based on the distance from the transmitter, azimuth and elevation. Since the independence assumption cannot be made, a single Kalman filter must be used to include the correlations between different components of the sensor measurements. Otherwise, three separate (and simpler) Kalman filters could be used to reduce computational complexity.

7.1 Measurement Model Equations

The coordinates used by the magnetic tracking system must be converted to a common coordinate system in order to be used with other sensors, or to provide relevant input into a hand gesture recognition system. A co-ordinate transformation between Cartesian co-ordinates and spherical co-ordinates is required, as illustrated in *Figure 9*.

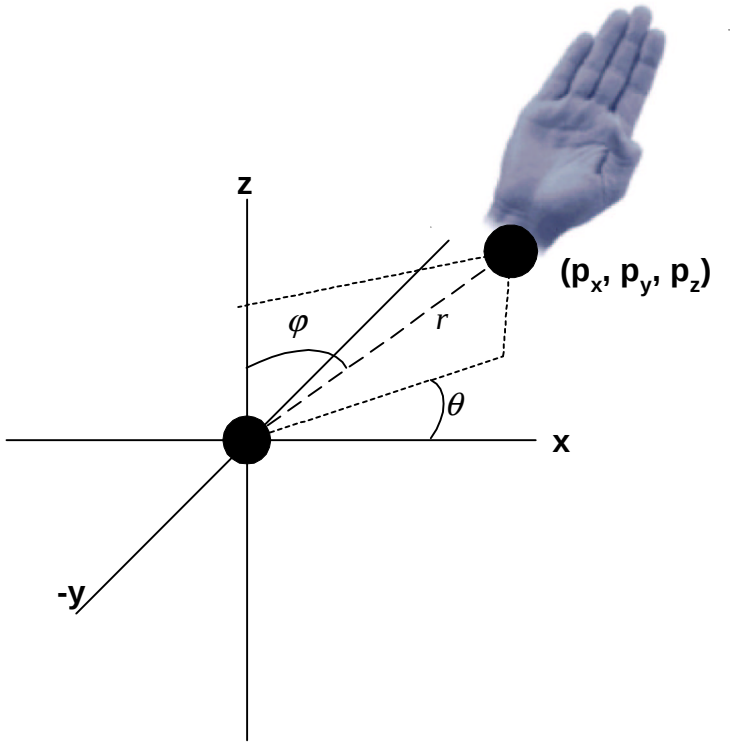

Figure 9 – Measurement Co-ordinate Transformation

The measurements made are the distance between the transmitter and receiver (r), azimuth (θ) and elevation (φ), resulting in the following observation vector:

$$\mathbf{y}(n) = \begin{bmatrix} r \\ \theta \\ \varphi \end{bmatrix}$$

The relationship between the position of the hand in Cartesian co-ordinates at time n to the observation in spherical co-ordinates is non-linear and can be described with the following three equations:

$$r = \sqrt{p_x^{\,2} + p_y^{\,2} + p_z^{\,2}}$$

$$\theta = \arctan\left(\frac{p_y}{p_x}\right)$$

$$\varphi = \arccos\left(\frac{p_z}{\sqrt{p_x^{\,2} + p_y^{\,2} + p_z^{\,2}}}\right)$$

And the reverse transformation can be performed with the following equations:

$$p_x = r\cos\theta\sin\varphi$$

$$p_y = r\sin\theta\cos\varphi$$

$$p_z = r\cos\varphi$$

Unfortunately, these equations cannot easily be converted into the measurement matrix as is required for the discrete linear Kalman filter. It is this coordinate transformation that introduces non-linearities into the hand model, violating the linearity constraint, and requiring alternative Kalman filters such as the EKF and UKF.

8 Extended Kalman Filter

Since the relationship between the measurements of the hand position are nonlinearly related to the state of the system, this violates the linear assumption of the standard Kalman filter. The extended Kalman filter (EKF) is an ad hoc technique to provide to use the standard Kalman filter on non-linear process or measurement models resulting in sub-optimal estimates. The measurement model and process model are linearized about the mean and covariance (the current operating point) at each iteration, and the standard Kalman filter is applied to the linearized models.

In the linear discrete Kalman filter, the state of the system can be updated with a straightforward matrix multiplication (**F**(n+1,n)). Similarly, converting from the state to measurement space is accomplished with another matrix multiplication (**C**(n)). Both of these matrices are approximated in the extended Kalman filter using a first-order Taylor expansion. To accomplish this, the Jacobian matrix of both the process model and the measurement model need to be calculated. Since the process model is already linear, the calculation is trivial,

$$\mathbf{F}(n,\mathbf{x}) = \begin{bmatrix} p_x + v_x\Delta t \\ p_y + v_y\Delta t \\ p_z + v_z\Delta t \\ v_x \\ v_y \\ v_z \end{bmatrix}$$

$$\frac{\partial \mathbf{F}(n,\mathbf{x})}{\partial \mathbf{x}} = \begin{bmatrix} 1 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

But, the Jacobian matrix of the nonlinear measurement model is nontrivial,

$$\mathbf{C}(n,\mathbf{x}) = \begin{bmatrix} \sqrt{p_x^{\,2} + p_y^{\,2} + p_z^{\,2}} \\ \arctan\left(\frac{p_y}{p_x}\right) \\ \arccos\left(\frac{p_z}{\sqrt{p_x^{\,2} + p_y^{\,2} + p_z^{\,2}}}\right) \end{bmatrix}$$

$$\frac{\partial \mathbf{C}(n,\mathbf{x})}{\partial \mathbf{x}} = \begin{bmatrix} \frac{p_x}{r} & \frac{p_y}{r} & \frac{p_z}{r} \\ \frac{-p_y}{-p_z{}^2 + r^2} & \frac{p_x}{-p_z{}^2 + r^2} & 0 \\ \frac{p_z p_x}{r^2\sqrt{-p_z{}^2 + r^2}} & \frac{p_z p_y}{r^2\sqrt{-p_z{}^2 + r^2}} & \frac{-\sqrt{-p_z{}^2 + r^2}}{r^2} \end{bmatrix}$$

These linearized matrices are incorporated into the extended Kalman filter using the following equations [Haykin, 1996],

The Kalman gain is now,

$$\mathbf{G}_f(n) = \mathbf{K}(n, n-1)\mathbf{C}^H(n)\left(\mathbf{C}(n)\mathbf{K}(n, n-1)\mathbf{C}^H(n) + \mathbf{Q}_2(n)\right)^{-1}$$

The innovations are calculated with,

$$\boldsymbol{\alpha}(n) = \mathbf{y}(n) - \mathbf{C}(n, \hat{\mathbf{x}}(n \mid Y_{n-1}))$$

The estimate is updated with the measurements using,

$$\hat{\mathbf{x}}(n \mid Y_n) = \hat{\mathbf{x}}(n \mid Y_{n-1}) + \mathbf{G}_f(n)\boldsymbol{\alpha}(n)$$

The error covariance for this estimate is calculated,

$$\mathbf{K}(n) = \left(\mathbf{I} - \mathbf{G}_f(n)\mathbf{C}(n)\right)\mathbf{K}(n, n-1)$$

The estimate for the next iteration based on current observations is predicted,

$$\hat{\mathbf{x}}(n+1 \mid Y_n) = \mathbf{F}(n, \mathbf{x}(n \mid Y_n))$$

And the error covariance for the next iteration is predicted,

$$\mathbf{K}(n+1, n) = \mathbf{F}(n+1, n)\mathbf{K}(n)\mathbf{F}^H(n+1, n) + \mathbf{Q}_1(n)$$

Where **F**(n+1,n) and **C**(n) are the Jacobian matrices evaluated at the current state estimate. Please refer to *Figure 6* for an illustration of how these equations interact in the Kalman filter.

The fundamental flaw in the extended Kalman filter is the fact that the probability distributions are no longer Gaussian after the non-linear transformation [Welch and Bishop, 2001], violating the assumption in the standard Kalman filter.

9 Unscented Kalman Filter

Julier, Uhlmann, and Durrant-Whyte identified the following three significant disadvantages in using the EKF [Julier et. al., 1995]:

- Linearization can produce highly unstable filter performance if the discrete time intervals are not sufficiently small
- Derivation of the Jacobian matrices are nontrivial
- Sufficiently small time intervals usually implies high computational overhead

The unscented Kalman filter (UKF) is based on the relatively low complexity in approximating a known statistical distribution rather than approximating a non-linear function. When reviewing literature on this technique, very few papers exist, and those that do exist include terse explanations or only describe portions of the UKF. To provide a relevant comparison between the algorithm for the UKF and the EKF, the algorithm for the UKF was interpreted from several variations mentioned in [Julier et al., 1995], [Janet, 1998], [Julier, 1999], and [Wan and van der Merwe, 2000] and presented here. Direct comparisons between the UKF and EKF equations are made in this section, and clarifications or explanations included where appropriate.

The extended Kalman filter calculates the Kalman gain and the correlation of the error in the state estimate based on a linearization of the measurement (C(n))and process (K(n,n-1)) models about the estimated state, using:

$\mathbf{G}_f(n) = \mathbf{K}(n, n-1)\mathbf{C}^H(n)\left(\mathbf{C}(n)\mathbf{K}(n, n-1)\mathbf{C}^H(n) + \mathbf{Q}_2(n)\right)^{-1}$ to calculate the Kalman gain, and

$\mathbf{K}(n) = \left(\mathbf{I} - \mathbf{G}_f(n)\mathbf{C}(n)\right)\mathbf{K}(n, n-1)$ to calculate the correlation of the error in the state estimate. At the same time, the actual (non-linear) process and measurement models are used to determine the innovations and state estimates. In contrast, the UKF calculates the correlation of the error in the state estimate, innovations, and state estimates together without passing values through the linearized process or measurement models. This significant difference can be observed in *Figure 10* and *Figure 11*.

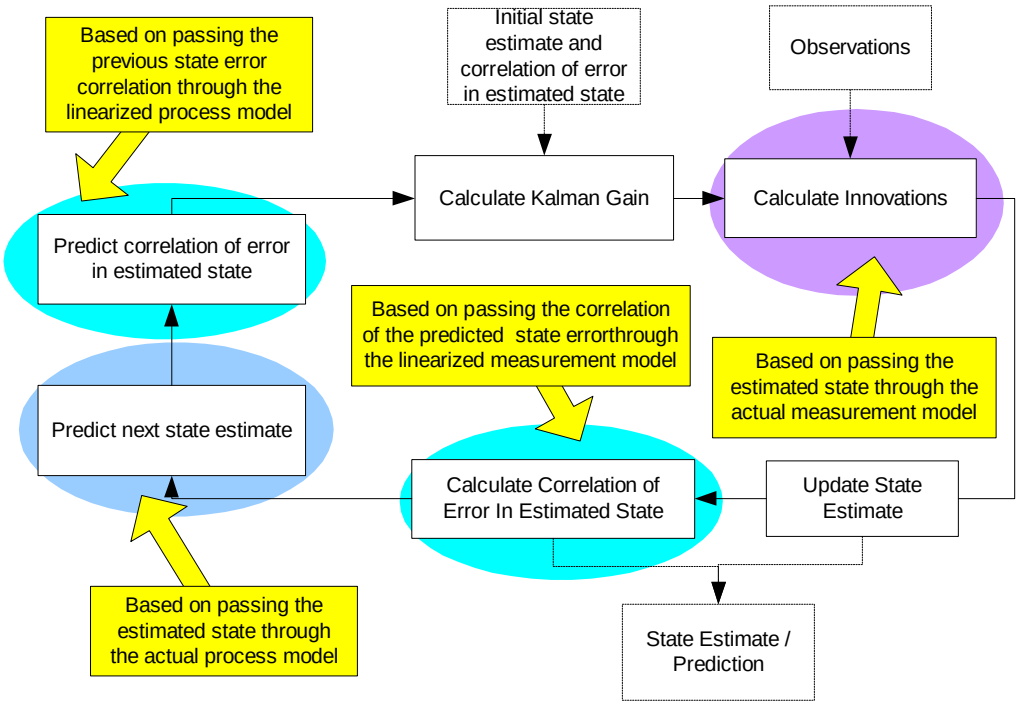

Figure 10 – Linearized and actual models used in EKF with estimated means and correlations

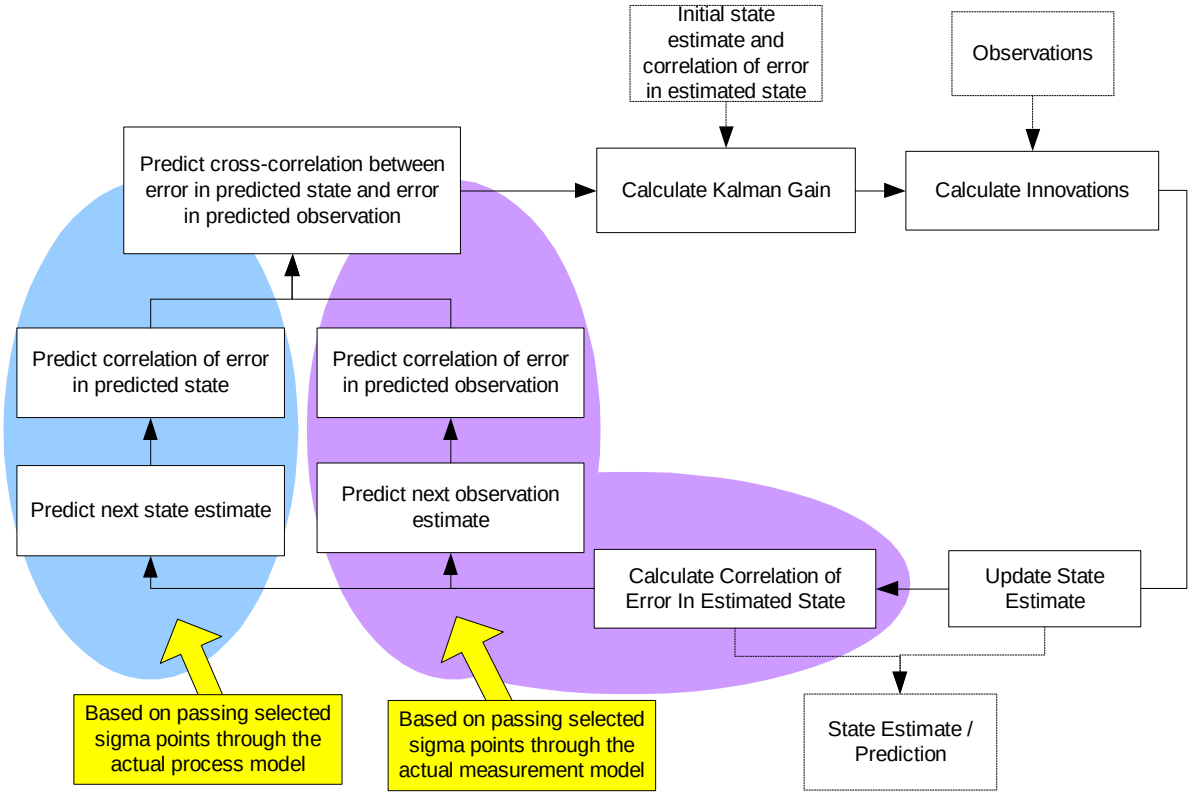

Figure 11 – Actual models used in UKF with sigma vectors

9.1 UKF and EKF algorithm comparison

The basis of the UKF is the unscented transform, where a distribution is approximated using a set of vectors, and these vectors are passed through the (non-linear) function to determine the probability distribution of the output from the function rather than linearizing the function itself. The unscented Kalman filter is summarized and compared against the EKF in the following set of equations, using the notation in [Haykin, 1996] for consistency:

9.1.1 Kalman Gain

In the UKF, a series of sigma vectors, or vectors selected to be representative of the probability distribution are used to determine the cross-correlation between the error in the estimated state and error in the estimated observations, along with the correlation matrix of the error in the estimated observation. Since these two matrices are explained later, they will be simplified here by using $\mathbf{P}(n, n-1)$ to represent the cross correlation between the error in the estimated state and error in the estimated observations at time n, and $\mathbf{R}(n, n-1)$ representing the correlation matrix of the error in the estimated observation. This is

$$\mathbf{G}_f(n) = \mathbf{P}(n, n-1)\left(\mathbf{R}(n, n-1)\right)^{-1}$$

9.1.2 Innovations

In the EKF, the innovations are calculated with,

$$\boldsymbol{\alpha}(n) = \mathbf{y}(n) - \mathbf{C}(n, \hat{\mathbf{x}}(n \mid Y_{n-1})) = \mathbf{y}(n) - \hat{\mathbf{z}}(n, n-1)$$

Whereas in the UKF, the innovations are calculated using the set of selected sigma vectors rather than the single state estimate,

$$\boldsymbol{\alpha}(n) = \mathbf{y}(n) - \sum_{i=0}^{2(2M+N)} W_i \mathbf{C}_u(n, \boldsymbol{\chi}_i(n, n-1), \boldsymbol{\chi}_i^{v_2}(n-1, n-1)) = \mathbf{y}(n) - \hat{\mathbf{z}}(n, n-1)$$

Where the weighted average of 2(2M+N)+1 sigma vectors is used to determine the mean observation. This will be described in more detail following the EKF/UKF comparison.

9.1.3 Update State Estimate

The estimate for both the EKF and the UKF is updated with the measurements (included in the innovations) using,

$$\hat{\mathbf{x}}(n \mid Y_n) = \hat{\mathbf{x}}(n \mid Y_{n-1}) + \mathbf{G}_f(n)\boldsymbol{\alpha}(n)$$

9.1.4 Correlation of Error in State Estimated

The correlation of the error in the current state estimate is calculated in EKF by passing the correlation of the predicted state estimate through the linearized measurement model,

$$\mathbf{K}(n) = \left(\mathbf{I} - \mathbf{G}_f(n)\mathbf{C}(n)\right)\mathbf{K}(n, n-1) = \mathbf{K}(n, n-1) - \mathbf{G}_f(n)\mathbf{C}(n)\mathbf{K}(n, n-1)$$

Whereas in the UKF, the correlation of the error in the current estimate takes into account the correlation of the error in the estimated observation,

$$\mathbf{K}(n) = \mathbf{K}(n, n-1) - \mathbf{G}_f(n)\mathbf{R}(n, n-1)\left(\mathbf{G}_f(n)\right)^{-1}$$

9.1.5 Predict Next State Estimate

The estimate for the next iteration based on current observations is predicted,

$$\hat{\mathbf{x}}(n+1 \mid Y_n) = \mathbf{F}(n, \mathbf{x}(n \mid Y_n))$$

Whereas, using the UKF technique, sigma vectors are passed through the process model to determine estimates of the sigma vectors for the next iteration,

$$\boldsymbol{\chi}_i(n+1, n) = \mathbf{F}_u(n, \boldsymbol{\chi}_i(n, n), \boldsymbol{\chi}_i^{v_1}(n, n))$$

It should be noted that the sigma vectors passed are based on an augmented system that includes the process variables, process noise terms, and observation noise terms and are passed into a slightly different process model that includes the relationship between noise and the system state. This is the similar to the technique used when the process and measurement noise is correlated [Brown and Hwang, 1997], however, the augmented system state is used only in this step rather than maintaining the process noise as part of the process state variables. Representing measurement, process, and state information this manner allows the modelling of correlations between them, although we assume they are uncorrelated in this project.

The mean of the estimates of the sigma vectors for the next iteration is then calculated using specified weights (please refer to section 9.2 for details) for each of the estimated sigma vectors that were passed through the process model,

$$\hat{\mathbf{x}}(n+1 \mid Y_n) = \sum_{i=0}^{2(2M+N)} W_i \boldsymbol{\chi}_i(n+1, n)$$

9.1.6 Predict Correlation of Error in Next State Estimate

The correlation of the error in the estimate for the next iteration is predicted using the linearized process model with the EKF,

$$\mathbf{K}(n+1, n) = \mathbf{F}(n+1, n)\mathbf{K}(n)\mathbf{F}^H(n+1, n) + \mathbf{Q}_1(n)$$

Whereas the correlation of the error in the estimate calculated using the estimated sigma vectors with the UKF,

$$\mathbf{K}(n+1,n) = \sum_{i=0}^{2(2M+N)} W_i \left(\boldsymbol{\chi}_i(n+1,n) - \hat{\mathbf{x}}(n+1 \mid Y_n)\right)\left(\boldsymbol{\chi}_i(n+1,n) - \hat{\mathbf{x}}(n+1 \mid Y_n)\right)^H$$

The process noise correlation matrix $\mathbf{Q}_1(n)$ does not appear in this equation since it was incorporated into an augmented system when estimating the sigma vectors using $\mathbf{F}_u(n, \boldsymbol{\chi}_i(n,n), \boldsymbol{\chi}_i^{v_1}(n,n))$.

9.1.7 Predict Next Observation Estimate + Correlation Matrix

The two new correlation matrices for the error in the introduced to simplify the Kalman gain equation are now calculated in the UKF. The correlation matrix of the error in the estimated observation is calculated using the predicted observation sigma vectors and estimated observation mean,

$$\mathbf{R}(n+1,n) = \sum_{i=0}^{2(2M+N)} W_i \left(\mathbf{Z}_i(n,n-1) - \hat{\mathbf{z}}(n+1,n)\right)\left(\mathbf{Z}_i(n,n-1) - \hat{\mathbf{z}}(n+1,n)\right)^H$$

Where $\mathbf{Z}_i$ is the result of passing the i[th] sigma point through the augmented observation model as follows,

$$\mathbf{Z}_i(n+1,n) = \mathbf{C}_u(n, \boldsymbol{\chi}_i(n+1,n), \boldsymbol{\chi}_i^{v_2}(n,n))$$

And the estimated observation was calculated from a weighted combination of the sigma vectors similar to how the estimated state was calculated,

$$\hat{\mathbf{z}}(n+1,n) = \sum_{i=0}^{2(2M+N)} W_i \mathbf{Z}_i(n+1,n)$$

The observation noise correlation matrix $\mathbf{Q}_2(n)$ does not appear in this equation, although it did appear in the EKF equation to calculate the Kalman gain. This correlation matrix was incorporated into an augmented system when estimating the sigma vectors using $\mathbf{C}_u(n, \boldsymbol{\chi}_i(n+1,n), \boldsymbol{\chi}_i^{v_2}(n,n))$.

The cross-correlation matrix between the error in the estimated state and error in the estimated observations is then calculated as follows,

$$\mathbf{P}(n+1,n) = \sum_{i=0}^{2(2M+N)} W_i \left(\boldsymbol{\chi}_i(n+1,n) - \hat{\mathbf{x}}(n+1,n)\right)\left(\mathbf{Z}_i(n+1,n) - \hat{\mathbf{z}}(n+1,n)\right)^H$$

It can be seen that the differences between the UKF and the EKF are relatively minor, and are only due to using sigma vectors to determine the statistical properties after the process or observation models are applied, as illustrated in *Figure 10* and *Figure 11*.

9.2 Unscented Transform

Determining the sigma vectors is accomplished using the unscented transform [Julier et al., 1995] as follows:

$$\boldsymbol{\chi}_0^a(n,n) = \hat{\mathbf{x}}^a(n \mid Y_n)$$

$$\boldsymbol{\chi}_i^a(n,n) = \hat{\mathbf{x}}^a(n \mid Y_n) + \left(\sqrt{(2M+N+\lambda)\mathbf{K}^a(n)}\right)_i \text{ for 1<=}i\text{<=2M+N}$$

$$\boldsymbol{\chi}_i^a(n,n) = \hat{\mathbf{x}}^a - \left(\sqrt{(2M+N+\lambda)\mathbf{K}^a(n)}\right)_{i-n^a} \text{ for (2M+N) +1<=}i\text{<= 2(2M+N)}$$

Where,

$$\hat{\mathbf{x}}^a(n \mid Y_n) = \begin{bmatrix} \hat{\mathbf{x}}(n \mid Y_n) \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}$$ is the augmented state vector of length 2M+N, including the process and observation noise terms, and $\mathbf{K}^a$ is the correlation matrix for the augmented state vector,

$$\mathbf{K}^a(n) = \begin{bmatrix} \mathbf{K}(n) & 0 & 0 \\ 0 & \mathbf{Q}_1(n) & 0 \\ 0 & 0 & \mathbf{Q}_2(n) \end{bmatrix}$$

and $\left(\sqrt{(2M+N+\lambda)\mathbf{K}^a(n)}\right)_i$ is the i[th] column of the matrix square root. λ is an ad-hoc scaling parameter, $\lambda = \beta^2(M+\kappa) - M$, where β determines how far the sigma vectors are from the mean, κ is an ad-hoc scaling parameter, and M is the length of the normal state vector (not augmented with process or observation noise).

The corresponding weights used in the unscented transform are:

$W_0 = \frac{\lambda}{2M+N+\lambda}$ when calculating the mean,

$W_0 = \frac{\lambda}{2M+N+\lambda} + (1-\beta^2)$ when calculating correlations (since the mean should not be scaled using α).

$W_i = \frac{1}{2(2M+N+\lambda)}$ for 1 <= *i* <=2(2M+N)

Although the UKF introduces two ad-hoc parameters for tuning, it eliminates the calculation of the Jacobian matrices as is required in the EKF since the measurement and process models are not linearized. The elimination of the Jacobian matrices allows discontinuous functions to be used in the measurement and process models without any special considerations.

10 Results

To evaluate the performance of the EKF and UKF for tracking the hand using the simplified hand model, several test cases were developed. Performance of the filters was evaluated using the Polhemus FasTrack specifications for measurement errors, and the measurement noise introduced into the system matched the given measurement error statistical properties. The scaling parameter β was set to 1 (unscaled), and the other tuning parameter, κ was set to the recommended value in [Julier, 1999], M – 3. The initial position of the hand was at 762mm (30") from the transmitter, at $\varphi = \pi/4$ and $\theta = \pi/4$.

The effects of noise introduced as normally distributed noise to velocity, acceleration, and the effects of a discontinuous model on the EKF and UKF for hand tracking are summarized in this section.

10.1 Velocity Noise

Linear motion is introduced in this test case, where the x, y, and z velocity of the hand follow a Gaussian white noise process. Both the EKF and UKF are initialized with the same initial parameters and observe the same sequence of observations. Typical results are illustrated in *Figure 12*, *Figure 13*, and *Figure 14*. In *Figure 12*, the solid black line represents the actual position, the dashed red line the EKF estimate, and the dashed blue line the UKF estimate. In *Figure 13*, the black "+" markers indicate actual hand position, the green circles are the measurements, the red line the EKF estimate, and the blue line the UKF estimate. In *Figure 14*, the solid blue line represents the mean squared error in the UKF estimate, the solid red line the mean squared error in the EKF estimate, the dashed blue line the estimated standard deviation of the error in the estimate for the UKF, and the dashed red line for the EKF. If the distinction between the lines is not clear, please refer to http://wolfman.eos.uoguelph.ca/~bminers/track for a colour PDF version of this document.

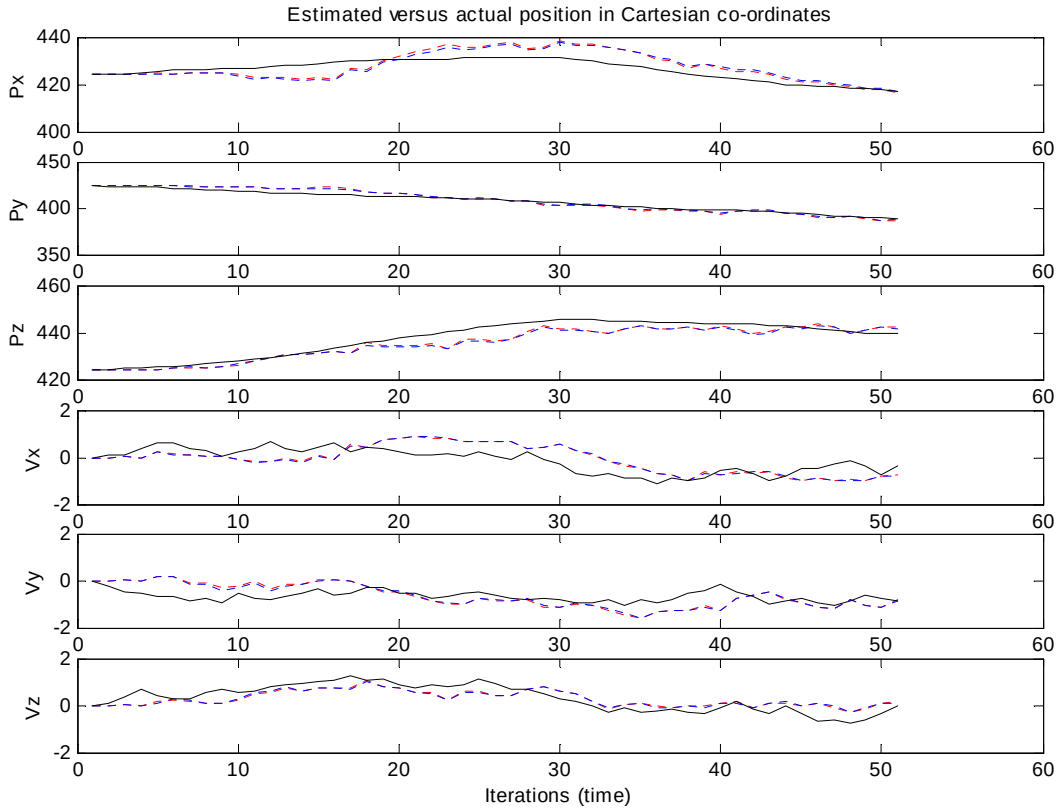

Figure 12 – Velocity Noise – Estimated versus Actual position and velocity in Cartesian co-ordinates

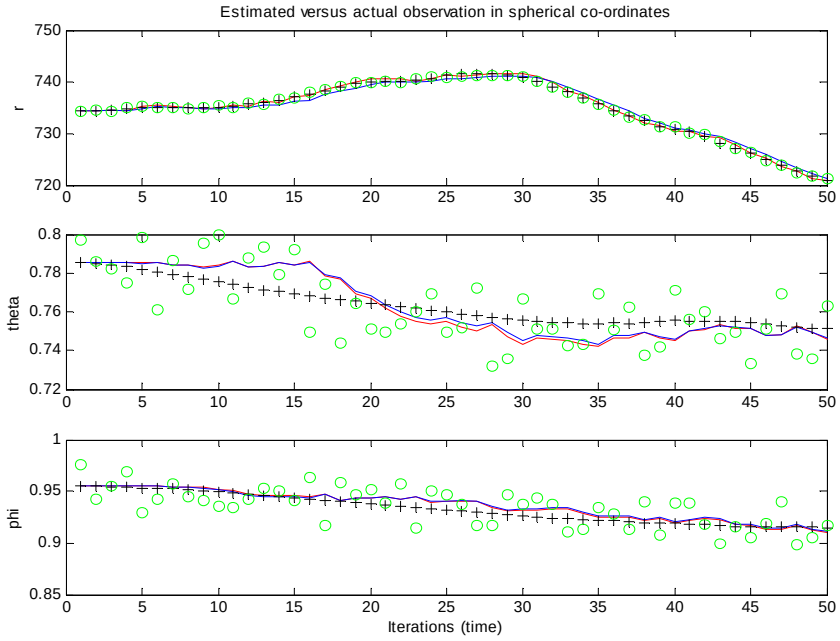

Figure 13 – Velocity Noise – Estimated versus actual observations in spherical co-ordinates

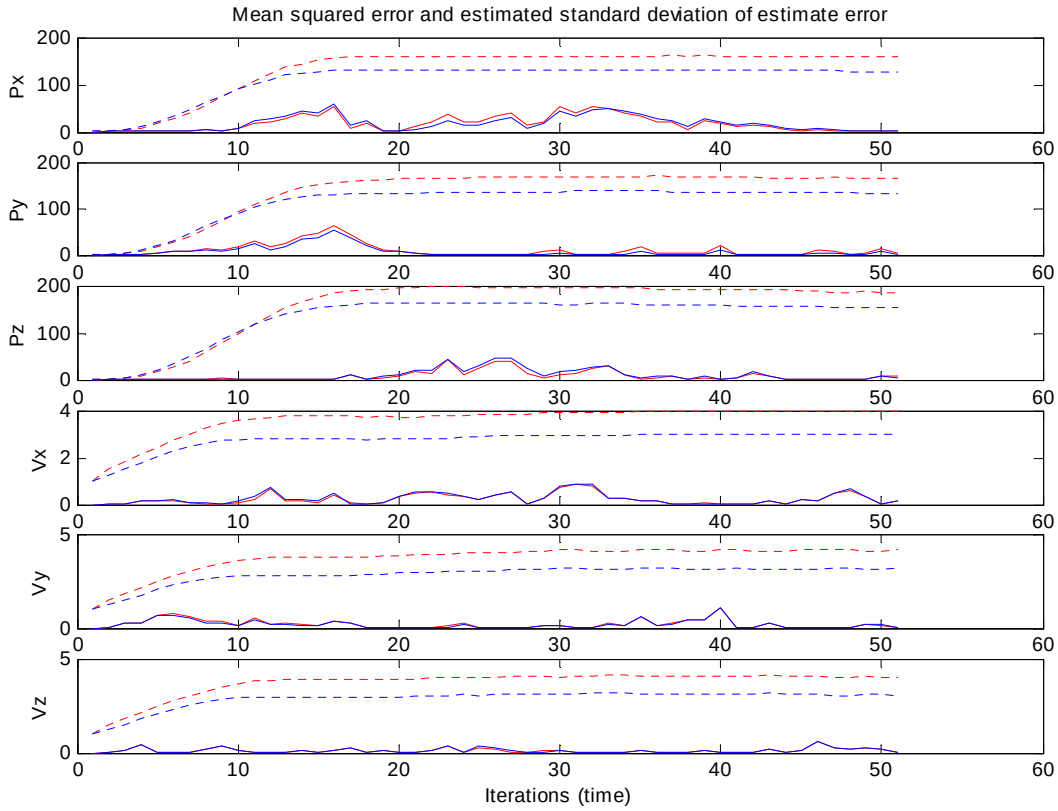

Figure 14 – Velocity Noise – MSE and estimated σ of estimate error for linear perturbation

It can be observed that the performance of the UKF is very similar to the performance of the EKF in this scenario in terms of average mean squared error. On average after 10 runs, the UKF resulted in a decrease of approximately 6% in mean squared error, and a lower standard deviation of the error in the estimate was consistently predicted by the UKF.

10.2 Acceleration Noise

Since the velocity of the hand does not realistically move with sudden changes in velocity following a Gaussian distribution, random noise was introduced as acceleration rather than velocity. Typical results from this are illustrated in *Figure 15*, *Figure 16*, and *Figure 17*.

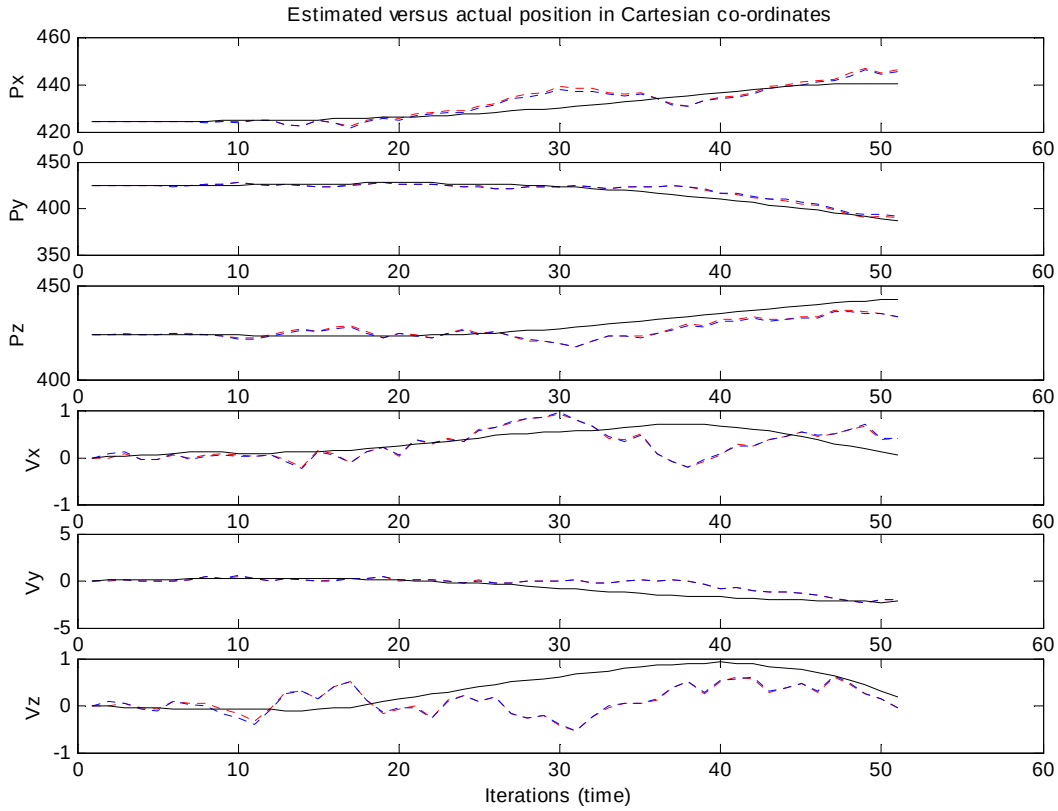

Figure 15 – Acceleration Noise – Estimated versus Actual position and velocity in Cartesian co-ordinates

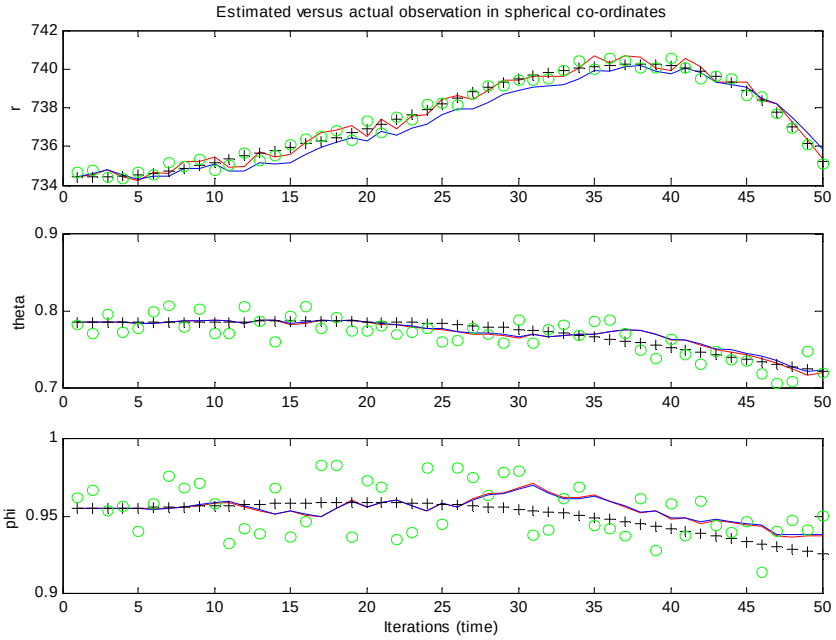

Figure 16 – Acceleration Noise – Estimated versus actual observations in spherical co-ordinates

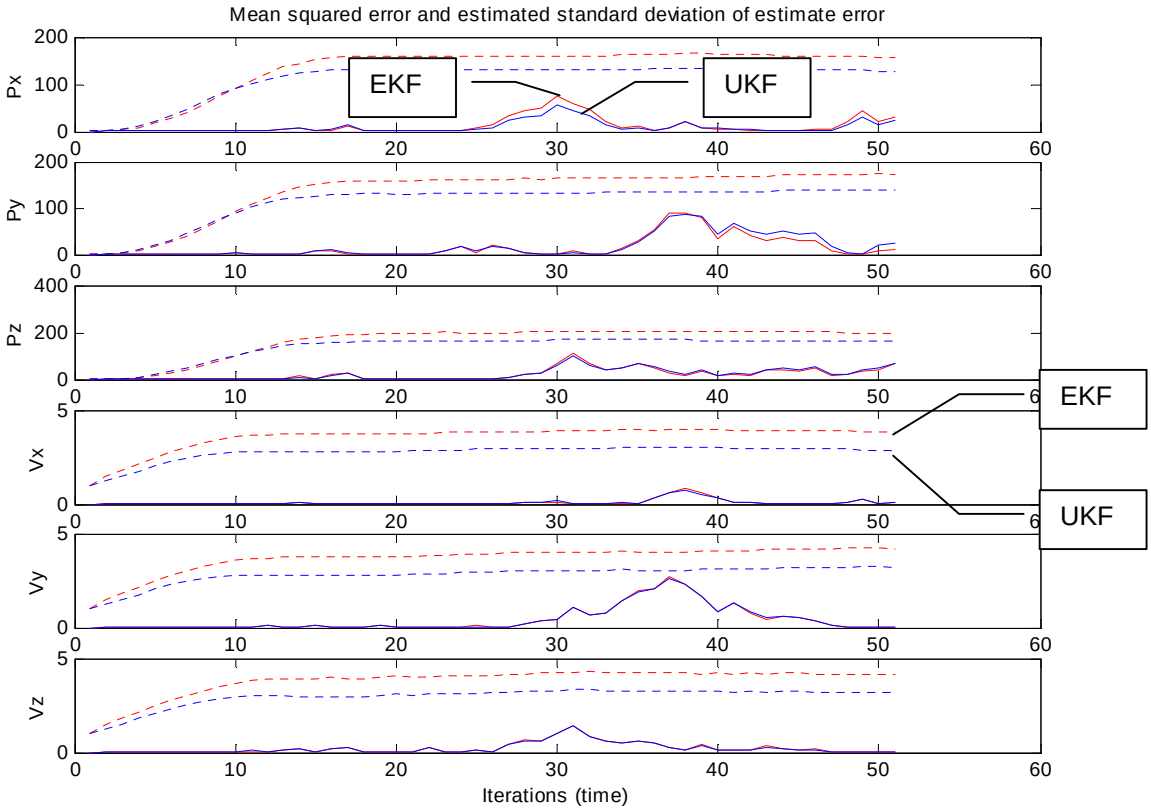

Figure 17 – Acceleration Noise – MSE and estimated σ of estimate error for linear perturbation

It can be observed that the performance of the UKF is very similar to the performance of the EKF, similar to when noise was introduced into the acceleration. On average after 10 runs, the UKF resulted in a decrease of 0.3% in mean squared error, and a lower standard deviation of the error in the estimate was consistently predicted by the UKF.

10.3 Discontinuous Process Model

One of the weaknesses of the EKF is in the Jacobian matrices that must be calculated. If the process or measurement models are discontinuous, the Jacobian matrix must be calculated for either one side of the discontinuity or the other. With the UKF, the sigma vectors are passed through the model, allowing the possibility for some vectors to be evaluated on one side of the discontinuity and some vectors to be evaluated on the other side.

A "bounding box" was implemented in the process function to restrict the hand movement within specified Cartesian co-ordinates, and acceleration noise was introduced into the process model. Typical results appear in *Figure 18*, *Figure 19*, and *Figure 20*.

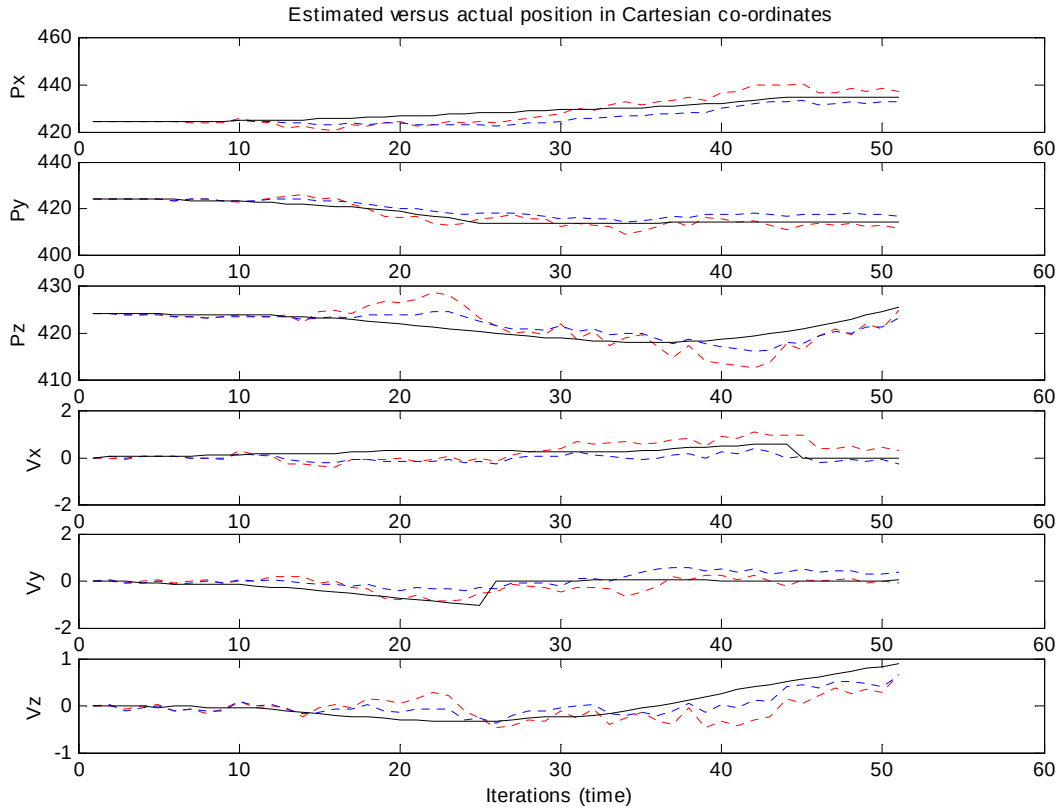

Figure 18 – Discontinuity – Estimated versus Actual position and velocity in Cartesian co-ordinates

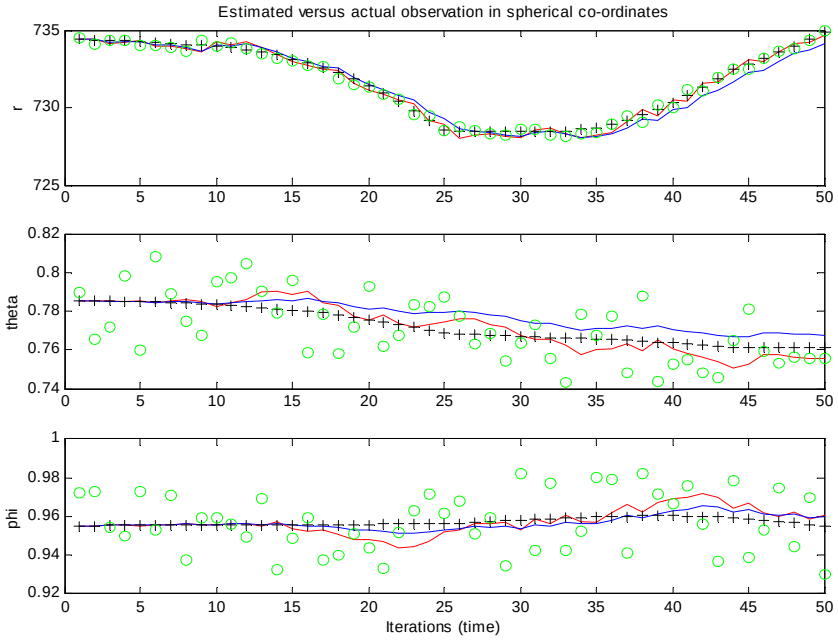

Figure 19 – Discontinuity – Estimated versus actual observations in spherical co-ordinates

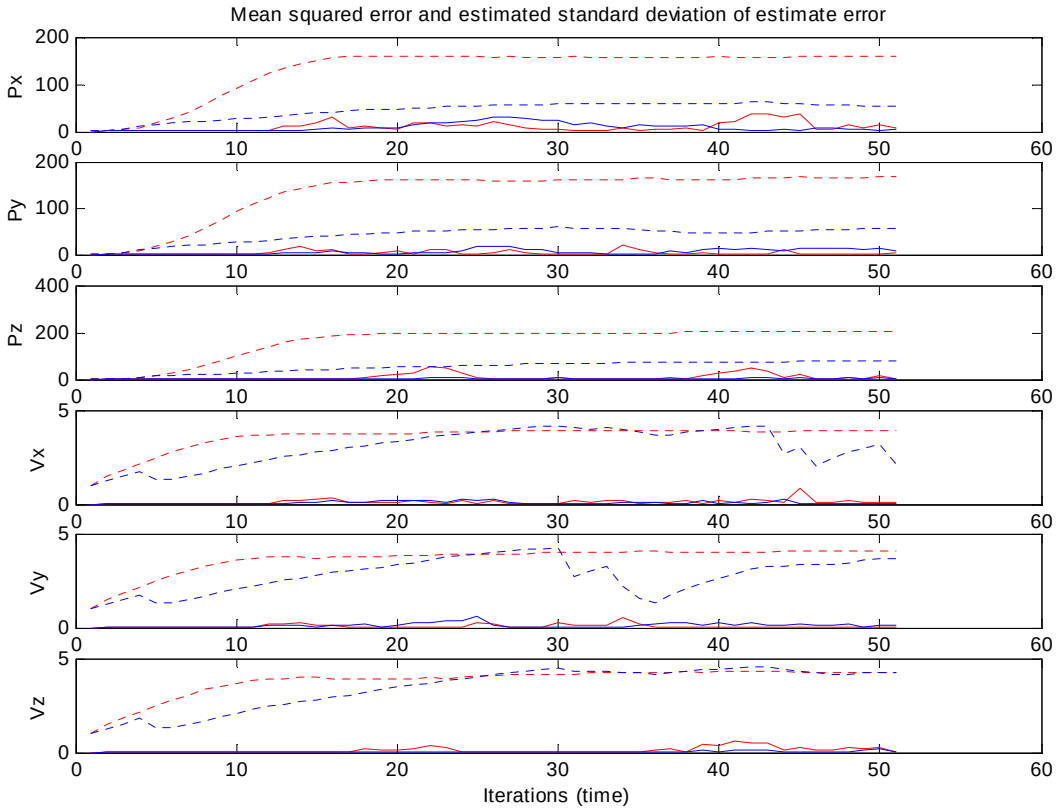

Figure 20 – Discontinuity – MSE and estimated σ of estimate error for linear perturbation

It can be seen from these results that the UKF performs better than the EKF in terms of mean squared error when a discontinuous process model is used. On average after 10 runs, the UKF resulted in a decrease of 28% in mean squared error over the EKF using this discontinuous process model.

11 Conclusion

The evaluation of the performance of the extended Kalman filter and the unscented Kalman filter in the tasks of filtering and prediction of hand position is an important step toward hand gesture interpretation. Using either the EKF or the UKF for non-linear filtering and prediction of hand position will aid in gesture interpretation. Filtering the sensory data will improve the quality of the input for interpretation, along with providing a measure of error for subsequent processes in hand gesture interpretation.

The unscented Kalman filter, as proposed in [Julier et. al., 1995] was shown to be suitable for use in tracking the hand using a magnetic 6DOF tracker for the purpose of gesture interpretation. The claims in [Julier and Uhlmann, 1997] and [Wan and van der Merwe, 2000] that the UKF is consistently better than the EKF are verified in this specific application, however, the improvement that the UKF provides is considerably more significant with a highly nonlinear or discontinuous model. As expected, the difference in terms of mean squared error between the UKF and EKF is negligible on models that are approximated well using linearization.

The simplicity of implementing a complex process or measurement model is a definite advantage of the UKF over the EKF, however, this simplicity comes at the expense of CPU time. The Matlab implementation of the UKF required almost twice as much CPU time as the equivalent EKF. It has been noted in literature that the UKF and EKF have the same order of complexity [Wan and van der Merwe, 2000], so the observed difference in computation time may be due to implementation differences.

12 Future Research

Janet briefly discusses the use of the UKF in his thesis [Janet, 1998], emphasizes its accuracy, and recommends future work in the area. The applicability of this technique to non-linear systems appears very promising, although the computational cost was higher than the EKF in manner it was implemented for this project. A complexity analysis should be performed on the two algorithms using models appropriate for hand tracking. For a more accurate CPU time analysis, both algorithms should be implemented using a lower-level language with less unknown overhead.

Although the recommended values from literature used in this project appeared to work well, a sensitivity analysis of the UKF to the various ad-hoc tuning parameters should be performed with respect to the process and measurement models used to track the hand.

The range of movements applied in the simulations avoided angular discontinuities, but this may not be possible in a practical environment. Since quaternions can represent angles without the discontinuities found in spherical co-ordinates, their applicability to hand tracking using a magnetic tracker should be investigated.

Future research into developing a more accurate process model for the hand will improve the accuracy of the Kalman filter, as will a more accurate measurement model. It has been shown that the noise is proportional to the fourth power of the transmitter-receiver separation [Nixon et al., 1998]. In addition, [Livingston and State, 1997] concluded that orientation error is a function of both the receiver position and its orientation. Incorporating these correlations into the model should improve its accuracy. Use of feedback from the gesture recognition to assist the tracking process may also improve the accuracy.

Use of real-world test cases rather than simulated scenarios will provide a more accurate and practical comparison between the UKF and EKF for use in hand gesture interpretation.

Other alternative techniques that may be useful include use of covariance intersection since the actual correlation matrices may not be accurately known, and use of the interacting multiple model (IMM) technique, where several simultaneous Kalman filters are used. Each Kalman filter in the IMM technique uses a different process model describing certain dynamics of hand movement. The state estimate from the Kalman filter with the minimum estimation error is used. Although the computational cost is increased with the use of multiple simultaneous Kalman filters, it may improve the accuracy if several different simplified process models together can describe the dynamics of most complex hand movements. Other techniques such as the decentralized Kalman filters and use of Kalman filters for fusion may not be directly applicable to magnetic tracking for hand gesture interpretation, but may be useful in multimodal interfaces.

13 References

[Julier et. al., 1995] S. J. Julier, J.K Uhlmann, H.F. Durrant-Whyte, “A New Approach for Filtering Nonlinear Systems”, American Automatic Control Conference, 1995.

[Julier and Uhlmann, 1996] S.J. Julier and J.K. Uhlmann, “A General Method for Approximating Nonlinear Transformation of Probability Distributions”, Technical report, RRG, Oxford, 1996. (http://www.robots.ox.ac.uk/ ~siju/work/publications/letter size/Unscented.zip)

[Julier and Uhlmann, 1997] S.J. Julier and J. K. Uhlmann, “A New Extension of the Kalman Filter to Nonlinear Systems” , The Proceedings of AeroSence: The 11th International Symposium on Aerospace/Defense Sensing, Simulation and Controls, SPIE, 1997.

[Julier, 1999] S. Julier, “The Scaled Unscented Transformation”, Draft, 1999 (http://citeseer.nj.nec.com/julier99scaled.html).

[Wan and van der Merwe, 2000] E. Wan and R. van der Merwe, “The Unscented Kalman Filter for Nonlinear Estimation”, In Proceedings of the IEEE Symposium 2000 on Adaptive Systems for Signal Processing, Communications, and Control, Alberta, 2000. (http://www.ece.ogi.edu/~ericwan/pubs.html)

[CI, 1997] Covariance Intersection Working Group, “A Culminating Advance in the Theory and Practice of Data Fusion, Filtering, and Decentralized Estimation”, 1997.

[Welch and Bishop, 2001] Welch, G. and G. Bishop, “An Introduction to the Kalman Filter”, TR 95-041 (revised 2001), University of North Carolina, 2001.

[Janet, 1998] J.A. Janet, “Pattern Analysis, Tracking and Control for Autonomous Vehicles Using Neural Networks”, Ph.D. Thesis, Electrical and Computer Engineering, North Carolina State University, 1998.

[Livingston and State, 1997] M. Livingston and A. State. "Magnetic tracker calibration for improved augmented reality registration", Presence: Teleoperators and Virtual Environments, 6(5):532-546, 1997.

[Nixon et al., 1998] M. Nixon, B. McCallum, W. Fright, and N. Price, "The effects of metals and interfering fields on electromagnetic trackers", Presence: Teleoperators and Virtual Environments, 7(2):204-218, 1998.

[Ikits et al., 2001] M. Ikits, J.D. Brederson, C. Hansen, and J. Hollerbach, "An Improved Calibration Framework for Electromagnetic Tracking Devices", in Proceedings of the IEEE Virtual Reality Conference 2001, pp. 63-70, 2001. (http://www.sci.utah.edu/research/vr/Publications.html)

[Cleary et al., 2001] K. Cleary, M. Clifford, M. Freedman, J. Zeng, S. Muna, V. Watson, F. Henderson, "Technology Improvements for Image-guided and Minimally Invasive Spine Procedures", Submitted to IEEE Transactions on Information Technology in Biomedicine, January 2001.

[Baillot and Rolland, 1996] Y. Baillot and J. Rolland, "Fundamental Principles of Tracking Technology for Virtual Environments", TR96-004, University of Central Florida, 1996.

[Smith and Dorf, 1992] R. Smith and R. Dorf, "Cicuits, Devices and Systems", John Wiley & Sons, Toronto, 1992.

[Kohler, 1997] M. Kohler, "System Architecture and Techniques for Gesture Recognition in Unconstrained Environments", Proceedings of the International Conference on Virtual Systems and Multimedia, 1997.

[Ahmad, 1995] S. Ahmad, "A Usable Real-Time 3D Hand Tracker", Proceedings of the 28th IEEE Asilomar Conference on Signals, Systems and Computers, pp. 1257-1261, 1995.

[Rehag and Kanade, 1993] J. Rehag, T. Kanade, "DigitEyes: Vision-Based Human Hand Tracking", Technical Report CMU-CS-93-220, Carnegie Mellon University, 1993.

[Fong et al., 2000] T. Fong, F. Conti, S. Grange, and C. Baur, "Novel Intefaces for Remote Driving: Gesture, Haptic and PDA", SPIE Telemanipulator and Telepresence Technologies VII, Boston, MA, November 2000.

[Haykin, 1996] S. Haykin, "Adaptive Filter Theory", Prentice Hall, 3rd edition, 1996.

[Brown and Hwang, 1997] R. Brown and P. Hwang, "Introduction to Random Signals and Applied Kalman Filtering", John Wiley & Sons, 3rd edition, 1997.

[Freeman and Roth, 1995] Freeman, William T. and Michal Roth, "Orientation Histograms for Hand Gesture Recognition", IEEE International Workshop on Automatic Face and Gesture Recognition, Zurich, June 1995.

[Darrell and Pentland, 1995] Darrell, Trevor J. and Alex P. Pentland, "Task-specific Gesture Analysis in Real-Time using Interpolated Views", MIT Media Lab TR-364, 1995. (Portions appeared in "Space-Time gestures", Proceedings, IEEE Computer Vision and Pattern Recognition, pp. 335-340, 1993)

[Triesch and Malsburg, 1996] Triesch, Jochen and Christoph von der Malsburg, "Robust Classification of Hand Postures against Complex Backgrounds", IEEE Proceedings of the Seconds International Conference on Automatic Face and Gesture Recognition, 1996.

[Moghaddam and Pentland, 1995] Moghaddam, Baback and Alex Pentland, "Probabilistic Visual Learning for Object Detection", 5th International Conference on Computer Vision, 1995.

[Hamdan et al., 1999] Hamdan, Raouf, Fabrice Heitz, and Laurent Throaval, "Gesture Localization and Recognition using Probabilistic Visual Learning", Proceedings, Computer Vision and Pattern Recognition, vol 2, pp.98-103, 1999.

[Iwai et al., 1999] Iwai, Yoshio, Hiroaki Shimizu, Masahiko Yachida, "Real-Time Context-Based Gesture Recognition Using HMM and Automaton", International Workshop on Recognition, Analaysis, and Tracking of Faces and Gestures in Real-Time Systems, 1999.

[Lu et al., 1997] Lu, Shan, Seiji Igi, Hideaki Matsuo and Yuji Nagashima, "Towards a Dialogue System Based on Recognition and Synthesis of Japanese Sign Language", International Gesture Workshop Proceedings, pp. 259-271, 1997.

14 Appendix A – Nomenclature

The nomenclature followed in [Haykin, 1996] is used wherever possible for consistency. All variables used are described in *Table 1*. Note – this table is an augmented version of table 7.1 in [Haykin, 1996] to include all additional variables used in this project.

Table 1 – Nomenclature

Variable	Description	Dimension
$\mathbf{x}(n)$	State vector at time n	Mx1
$\mathbf{y}(n)$	Observation vector at time n	Nx1
$\mathbf{F}(n+1,n)$	Linearized state transition matrix from time n to n+1	MxM
$\mathbf{F}(n,\mathbf{x})$	(non-linear) process model function from time n to time n+1	Returns MxM
$\mathbf{F}(n,\mathbf{x},\mathbf{x}^{v})$	(non-linear) process model function including effects of process noise from time n to time n+1	Returns MxM
$\mathbf{C}(n)$	Linearized measurement matrix at time n	NxN
$\mathbf{C}(n,\mathbf{x})$	(non-linear) measurement model function at time n	Returns NxN
$\mathbf{C}(n,\mathbf{x},\mathbf{x}^{v})$	(non-linear) measurement model function at time n including effects of measurement noise	Returns NxN
$\mathbf{Q}_1(n)$	Correlation matrix of process noise at time n	MxM
$\mathbf{Q}_2(n)$	Correlation matrix of measurement noise at time n	NxN
$\hat{\mathbf{x}}(n \mid Y_n)$	Estimated state vector at time n given observations up to time n	Mx1
$\hat{\mathbf{x}}(n+1 \mid Y_n)$	Predicted estimate of the state vector at time n+1 given observations up to time n	Mx1
$\hat{\mathbf{z}}(n)$	Estimated observation at time n	Nx1
$\mathbf{P}(n,n-1)$	Cross-correlation between the error in the predicted state and error in the predicted observations at time n	MxN
W_i	Weight used in the unscented transform	Scalar
λ	Ad-hoc scaling parameter (a function of κ and β)	Scalar
κ	Ad-hoc scaling parameter for the UKF based on the size of the state vector	Scalar
β	Ad-hoc scaling parameter for the UKF based on the distance between the sigma vectors and the mean	Scalar
$\boldsymbol{\chi}_i(n+1,n)$	Predicted sigma vector *i* at time n+1	Mx1
$\boldsymbol{\chi}_i^a(n,n) = \begin{bmatrix} \boldsymbol{\chi}_i(n,n) \\ \boldsymbol{\chi}_i^{v_1}(n,n) \\ \boldsymbol{\chi}_i^{v_2}(n,n) \end{bmatrix}$	Augmented sigma vector at time n (represents the statistical properties of the state, process, and measurement noise)	(2M+N) x 1
$\mathbf{G}(n)$	Kalman gain at time n	MxN
$\boldsymbol{\alpha}(n)$	Innovations vector at time n	Nx1
$\mathbf{R}(n)$	Correlation matrix of the innovations vector at time n	NxN
$\mathbf{K}(n+1,n)$	Correlation matrix of the error in the predicted state at time n+1	MxM
$\mathbf{K}(n)$	Correlation matrix of the error in the estimated state at time n	MxM
$\mathbf{K}^a(n)$	Correlation matrix of the error in the augmented state at time n	(2M+N) x (2M+N)

15 Appendix B – Maple Code

The Jacobian matrices required for the linearization of the measurement model in the EKF were calculated in Maple using the following commands:

```
> assume(R,real);
> assume(x,real);
> assume(y,real);
> assume(z,real);
> assume(R >= 0);

> r(x,y,z) := sqrt(x^2 + y^2 + z^2);
```

$$\mathrm{r}(x\sim, y\sim, z\sim) := \sqrt{x\sim^2 + y\sim^2 + z\sim^2}$$

```
> theta(x,y,z) := arctan( y / x);
```

$$\theta(x\sim, y\sim, z\sim) := \arctan\left(\frac{y\sim}{x\sim}\right)$$

```
> phi(x,y,z) := arccos( z / sqrt(x^2 + y^2 + z^2));
```

$$\phi(x\sim, y\sim, z\sim) := \arccos\left(\frac{z\sim}{\sqrt{x\sim^2 + y\sim^2 + z\sim^2}}\right)$$

```
> Diff(r,y) = diff(r(x,y,z),y);
> Diff(r,z) = diff(r(x,y,z),z);
> Diff(r,x) = diff(r(x,y,z),x);
```

$$\frac{\partial}{\partial y\sim} r = \frac{y\sim}{\sqrt{x\sim^2 + y\sim^2 + z\sim^2}}$$

$$\frac{\partial}{\partial z\sim} r = \frac{z\sim}{\sqrt{x\sim^2 + y\sim^2 + z\sim^2}}$$

$$\frac{\partial}{\partial x\sim} r = \frac{x\sim}{\sqrt{x\sim^2 + y\sim^2 + z\sim^2}}$$

```
> Diff(theta,x) = algsubs(x^2+y^2+z^2 = R^2,simplify(diff(theta(x,y,z),x)));
> Diff(theta,y) = algsubs(x^2+y^2+z^2 = R^2,simplify(diff(theta(x,y,z),y)));
> Diff(theta,z) = algsubs(x^2+y^2+z^2 = R^2,simplify(diff(theta(x,y,z),z)));
```

$$\frac{\partial}{\partial x\sim} \theta = -\frac{y\sim}{-z\sim^2 + R\sim^2}$$

$$\frac{\partial}{\partial y\sim} \theta = \frac{x\sim}{-z\sim^2 + R\sim^2}$$

$$\frac{\partial}{\partial z\sim} \theta = 0$$

```
> Diff(phi,x) = algsubs(x^2+y^2+z^2 = R^2,simplify(diff(phi(x,y,z),x)));
> Diff(phi,y) = algsubs(x^2+y^2+z^2 = R^2,simplify(diff(phi(x,y,z),y)));
```

```
> Diff(phi,z) = algsubs(x^2+y^2+z^2 = R^2,simplify(diff(phi(x,y,z),z)));
```

$$\frac{\partial}{\partial x\sim}\phi = \frac{z\sim\, x\sim}{\sqrt{-z\sim^2 + R\sim^2}\; R\sim^2}$$

$$\frac{\partial}{\partial y\sim}\phi = \frac{z\sim\, y\sim}{\sqrt{-z\sim^2 + R\sim^2}\; R\sim^2}$$

$$\frac{\partial}{\partial z\sim}\phi = -\frac{\sqrt{-z\sim^2 + R\sim^2}}{R\sim^2}$$

16 Appendix C – Matlab Code

16.1 Kalman_extended.m

```
% kalman_extended
%
% The Extended Kalman Filter
%
% Input Parameters:
% z = Current observation
% F = process model function name, accepting a
%     state vector (Mx1), a process noise vector (Mx1), and
%     a time interval and returns the resulting state vector
%     as predicted by the process model after the given time interval
%
% C = measurement model function name, accepting a state vector (Mx1),
%     a measurement noise vector (Nx1), and returns the resulting
%     measurement for the given state vector
%
% Q1 = process noise correlation matrix
% Q2 = measurement noise correlation matrix
% K_predicted = Correlation matrix of the error in the predicted
%               state estimate for the current time step
% x_hat_previous = Previous estimate of the system state
%
% timestep = elapsed time between iterations

function [x_hat, K_predicted_next] = ...
    kalman_extended(z, F, C,
        Q1, Q2, K_predicted, x_hat_previous, timestep)

  m = length(x_hat_previous); % Length of state vector

  % Linearize the measurement model
  C_linearized = eval([C '_jacobian( x_hat_previous ) ']);


  % Kalman Gain
  Gf = K_predicted * C_linearized' * inv( C_linearized * K_predicted * C_linearized' + Q2
);


  % Innovations
  alpha = z - eval([ C ' ( x_hat_previous, [] ) ']);

  % Estimation
  x_hat_corrected = x_hat_previous + Gf * alpha;
  x_hat = eval([ F ' ( x_hat_corrected, [], timestep ) ']);
```

```
  % Linearize the system model
  x_hat_corrected;
  F_linearized = eval([ F '_jacobian( x_hat_corrected, timestep ) ']);

  % Error Correlation
  K_filtered = (eye(m) - Gf * C_linearized) * K_predicted;
  K_predicted_next = F_linearized * K_filtered * F_linearized' + Q1;

  % Just prediction:
  % x_hat = eval([ F_function ' ( x_hat_corrected ) ']);

% Linearization of the process model
% (yes, this one is already linear...)
function F_jacobian = hand_process_model_jacobian( x_state, dt )
  F_jacobian = [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 ];


% Linearization of the measurement model
function C_jacobian = hand_measurement_model_jacobian( x_state )
%  Jacobian: dh/dx = [ dh1/dx1 dh1/dx2 dh1/d3 . . . ; dh2/dx1 dh2/dx2 . . . ];

  x = x_state(1);
  y = x_state(2);
  z = x_state(3);
  r = sqrt(x^2 + y^2 + z^2);
  if (r == 0)
      C_jacobian = [ 0 0 0 ; 0 0 0 ; 0 0 0];
  else
      % Partial derivative of r with respect to x, y, and z
      drdx = x / r;
      drdy = y / r;
      drdz = z / r;
      % Partial derivative of theta with respect to x, y, and z
      if ((r == 0) & (z == 0))
          dthetadx = 0;
          dthetady = 0;
      else
          dthetadx = y / (z^2-r^2);
          dthetady = x / (-z^2 + r^2);
      end
      dthetadz = 0;
      % Partial derivative of phi with respect to x, y, and z
      if (z == 0)
          dphidx = 0;
          dphidy = 0;
      else
          dphidx = z * x / ( sqrt(- z^2 + r^2 ) * r^2 );
          dphidy = z * y / ( sqrt(- z^2 + r^2 ) * r^2 );
      end
      dphidz = -sqrt( - z^2 + r^2 ) / r^2;

      C_jacobian = [ drdx     drdy     drdz     0 0 0;
                     dthetadx dthetady dthetadz 0 0 0;
                     dphidx   dphidy   dphidz   0 0 0 ];
  end
```

16.2 Kalman_unscented.m

```
% kalman_unscented
%
% Based on [Julier et al., 1995] (http://www.robots.ox.ac.uk/~siju/work/work.html)
%
% Input parameters:
% z = Current observation
% F = process model function name, accepting a
%     state vector (Mx1), a process noise vector (Mx1), and
%     a time interval and returns the resulting state vector
%     as predicted by the process model after the given time interval
%
% C = measurement model function name, accepting a state vector (Mx1),
%     a measurement noise vector (Nx1), and returns the resulting
%     measurement for the given state vector
%
% Q1 = process noise correlation matrix
% Q2 = measurement noise correlation matrix
% K_predicted = Correlation matrix of the error in the predicted
%               state estimate for the current time step
% x_hat_previous = Previous estimate of the system state
%
% timestep = elapsed time between iterations
%
function [x_hat, K_predicted_next] = ...
    kalman_unscented(z, F, C, Q1, Q2, K_predicted, x_hat_previous,timestep)

% Ad-hoc scaling parameter that can be used to adjust the
% distance between the mean and the sigma vectors
% beta = 1 to eliminate the effects of the scaling parameter
beta = 1;

% Ad-hoc scaling factor based on the dimensions in the state
% As recommended in [Julier, 1999], kappa = M-3 works well.
kappa = length(z) - 3;

M = length(x_hat_previous); % Size of state vector
N = length(z); % Size of measurement vector
if (sum(Q1) > 0)
    v1_size = M;    % Process noise exists
else
    v1_size = 0;    % No process noise
    Q1 = [];
end
if (sum(Q2) > 0)
    v2_size = N;    % Measurement noise exists
else
    v2_size = 0;    % No measurement noise
    Q2 = [];
end

% lambda is an ad-hoc scaling parameter based on
% beta, kappa, and the size of the state vector (M)
lambda = (beta^2)*(M + kappa) - M;

% Total number of sigma vectors
nsp = 2*(M + v1_size + v2_size)+1;

% Calculate sigma points and their weights

% The sigma points are formed by stacking up PEst and Q to give
% the augmented sigma point set.
```

```
if (v1_size + v2_size > 0)
    % Noise exists, so an augmented matrix that includes
    % the process noise, measurement noise, and state estimation
    % correlation information must be constructed.
    % It is assumed the process noise and measurement noise is uncorrelated,
    % as are the noises with the state estimation correlation.
   Q1_Q2 = [ Q1                       zeros(v1_size,v2_size); ....
             zeros(v2_size,v1_size)   Q2                    ];
   K_augmented = [ K_predicted                  zeros(M, v1_size + v2_size); ...
                   zeros(v1_size + v2_size, M)   Q1_Q2                      ];
else
    % No noise, so an augmented matrix does not need to be used.
   K_augmented = K_predicted;
end;

% Calculate the matrix square root
K_augmented_sqrtm = real( sqrtm( ...
    (M + v1_size + v2_size + lambda) * K_augmented) );

% Calculate the sigma vectors that will be input to the process and
% measurement models
Chi_augmented = [zeros(M + v1_size + v2_size, 1) ...
                -K_augmented_sqrtm K_augmented_sqrtm];

% Array of the weights for each sigma point
W = [lambda 0.5*ones(1, 2*(M + v1_size + v2_size))] / ...
    (M + v1_size + v2_size + lambda);

% Initialise intermediate values
xPred         = zeros( M, 1);
xPredSigmaPts = zeros( M, nsp);
zPred         = zeros( N, 1);
zPredSigmaPts = zeros( N, nsp);
KPred         = zeros( M, M);
alphaPred     = zeros( N, N);
PPred         = zeros( M, N);

% Pass the sigma points through the process and measurement models
% to determine the statistical properties of the outputs from
% these models
for count=1:nsp,
    % Pass the sigma vectors through the process model to get
    % estimates of where the sigma vectors will be based on the model
   xPredSigmaPts(:,count)=eval([F '(Chi_augmented(1:M, count)+x_hat_previous,
Chi_augmented(M+1:M+v1_size,count), timestep)']);
    % The mean value is the weighted sum of each sigma vector
    xPred=xPred+W(count)*xPredSigmaPts(:,count);

    % Pass the estimates of the sigma vectors based on the process model
    % through the measurement model
   zPredSigmaPts(:,count)=eval([C '(xPredSigmaPts(:, count),
Chi_augmented(M+v1_size+1:M+v1_size+v2_size,count))']);
    % The mean value is the weighted sum of each sigma vector
    zPred=zPred+W(count)*zPredSigmaPts(:,count);
end;

% The weights are adjusted here to adjust the distance of the sigma vectors
% from the mean
W(1) = W(1) + (1 - beta^2);

% Work out the correlations and the cross correlations
for count=1:nsp,
        % Distances beween the sigma vectors and the mean
```

```
        exSigmaPt=xPredSigmaPts(:,count) - xPred;
        ezSigmaPt=zPredSigmaPts(:,count) - zPred;

        % Prediction covariance
        KPred = KPred + W(count) * exSigmaPt * exSigmaPt';

        % Innovation covariance
        alphaPred = alphaPred + W(count) * ezSigmaPt * ezSigmaPt';

        % Cross-correlation between observations and states
        PPred = PPred + W(count) * exSigmaPt * ezSigmaPt';
end;

% Calculate Kalman gain
G = PPred * inv( alphaPred );

% Update the state estimate
x_hat = xPred + G * ( z - zPred );

% Update the correlation of error in the estimate
K_predicted_next = KPred - G * alphaPred * G';
```

16.3 Kalman_standard.m

```
% Kalman Filter
%
% Following with the conventions of [Haykin, 1996],
%
% y = (n x k) observations (n x 1) for each sample at time k
% F = (m x m) matrix of the system model
% C = (n x m) matrix of the measurement model
% Q1 = (m x m) system noise covariance
% Q2 = (n x n) measurement noise covariance
%
% x = (m x k) process state vector at time k
% P = (m x m x k) error covariance matrix at time k
%
function [x_hat, K_predicted_next] = ...
    kalman_standard(y, F, C, Q1, Q2, K_predicted, x_hat_previous)

  % Kalman Gain
  G = F * K_predicted * C' * inv( C * K_predicted * C' + Q2 );
  % Innovations
  alpha = y - C * x_hat_previous;
  % Estimation
  x_hat = F * x_hat_previous + G * alpha;
  % Error Correlation
  K_filtered = K_predicted - F * G * C * K_predicted;
  K_predicted_next = F * K_filtered * F' + Q1;

  % Just prediction:
  % x_hat(:,t+1) = F * x_hat_previous(:,t);
```

16.4 Hand_measurement_model.m

```
% hand_measurement_model
%
% Translate from state matrix to measurement matrix
%
% Input parameters:
% x_state           = state of system
% measurement_noise = noise introduced into the non-linear model (UKF)
%
% Returns a measurement corresponding to the given system state
```

```
%
function C = hand_measurement_model( x_state, measurement_noise )
  % Convert x,y,z to r, theta, phi
  x = x_state(1);
  y = x_state(2);
  z = x_state(3);

  % distance from transmitter to receiver on forearm
  r = sqrt(x^2 + y^2 + z^2);

  % A bit of a problem if the separation is not far enough,
  % but most specs state that there must be at least a
  % 30cm separation
  if (r == 0)
      disp('zero r');
      C = [ 0 ; 0 ; 0 ];
  else
      theta = atan( y / x);
      phi = acos( z / r );

      C = [r ; theta ; phi ];
  end

  % Incorporate measurement noise.  It's additive and uncorrelated.
  if (length(measurement_noise) > 0)
      C = C + measurement_noise;
  End
```

16.5 Hand_process_model.m

```
% Process model for hand tracking
%
% Input parameters:
% x_state = state of system
% system_noise = noise introduced into nonlinear model (UKF)
% time_step = length of interval between now and the returned value
%
% returns the state of the system at time now + time_step
%
function F = hand_process_model( x_state, system_noise, time_step )

  if (length(system_noise) > 0)
    x_state = x_state + system_noise;
  % Implement a "bounding box" in which the hand must remain
  %  for i = 1:3
  %      if (x_state(i) > 424 + 10) & (x_state(i+3) > 0)
  %          x_state(i+3) = 0;
  %      elseif (x_state(i) < 424 - 10) & (x_state(i+3) < 0)
  %          x_state(i+3) = 0;
  %      end
  %   end
  end

  px = x_state(1);
  py = x_state(2);
  pz = x_state(3);
  vx = x_state(4);
  vy = x_state(5);
  vz = x_state(6);

  F = [ px + time_step*vx ;
        py + time_step*vy;
        pz + time_step*vz;
        vx ;
```

```
        vy ;
        vz ];
```

16.6 Hand_tracking.m

```
% hand_tracking
%
% Run simulations to evaluate the EKF and UKF
%
%

function hand_tracking

% Iterations:
max_t = 50;

% System State
% -----------------

% Actual system state (x,y,z, vx, vy, vz)
x_actual = zeros(6,max_t+1);
% Hand at 735mm distance from transmitter, stationary
x_actual(:,1) = [424;424;424;0;0;0];

% Initial state
x_hat_e = zeros(6,max_t+1);
x_hat_e(:,1) = x_actual(:,1);
x_hat_u = x_hat_e;

% System noise covariance
Q1 = [ 0.025 0 0 0 0 0 ; ...
       0 0.025 0 0 0 0 ; ...
       0 0 0.025 0 0 0 ; ...
       0 0 0 0.5 0 0 ; ...
       0 0 0 0 0.5 0 ; ...
       0 0 0 0 0 0.5 ];

% Observations
% -----------------

% Actual Observations (r, theta, phi)
y_actual = zeros(3,max_t+1);    % Actual uncorrupted measurement
y = zeros(3,max_t+1);           % Corrupted with noise (as measured)
y(:,1) = hand_measurement_model(x_actual(:,1),[]);

y_hat_e = zeros(3,max_t+1);       % Measurement as predicted by Kalman filter
y_hat_u = zeros(3,max_t+1);       % Measurement as predicted by Kalman filter

% Measurement noise covariance:
% (based on the Polhemus FasTrack = 0.03" positional = 0.762mm, 0.15 degrees):
Q2 = [ 0.762    0         0       ;
       0    .15/180*pi    0       ;
       0        0       .15/180*pi ];

% Initial correlation of error in prediction
K_predicted_e = zeros(6,6,max_t+1);
K_predicted_e(:,:,1) = [ 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 ];
K_predicted_u = K_predicted_e;
```

```
% Continue from the end of a previous filter run
%load('filtered.mat','max_t','y_hat*','x_hat*','K_predicted*');
%K_predicted(:,:,1) = K_predicted(:,:,max_t);

% Generate a sequence of measurements and actual
% positions that can be used to empirically
% evaluate each filter type
a = zeros(3,1);
for t = 1:max_t
    y_actual(:,t) = hand_measurement_model(x_actual(:,t),[]);

    % Introduce measurement noise
    y(:,t) = hand_measurement_model(x_actual(:,t),sqrt(Q2) * (rand(3,1) - 0.5));

    % Use actual measurement (no noise)
    % y(:,t) = y_actual(:,t);

    % Introduce process noise

    % Based on actual covariance
    x_actual(:,t+1) = hand_process_model(x_actual(:,t),sqrt(Q1) * (rand(6,1) - 0.5), 1);

    % Based on movement of the hand, accelerating and decelerating
    % (Acceleration noise)
    %a = a + 0.05 * (rand(3,1) - 0.5);
    %noise = [0; 0; 0 ; a ];
    %x_actual(:,t+1) = hand_process_model(x_actual(:,t),noise, 1);

    % Use actual process model (no noise)
    %x_actual(:,t+1) = hand_process_model(x_actual(:,t),[],1);
end
save('test.mat','max_t','y','x_actual','y_actual');

%cov((y - y_actual)')

load('test.mat');

% Pass the simulated measurements through the filters
%
for t = 1:max_t

    % Standard Kalman Filter
    %[x_hat(t+1), K_filtered, K_predicted] = ...
    %    kalman_standard(y(t), F, C, Q1, Q2, K_predicted, x_hat(t) );

    % Extended Kalman Filter
    [x_hat_e(:,t+1), K_predicted_e(:,:,t+1)] = ...
        kalman_extended(y(:,t), 'hand_process_model', ...
                        'hand_measurement_model', Q1, Q2, ...
                        K_predicted_e(:,:,t), x_hat_e(:,t), 1 );

    % Unscented Kalman Filter
    [x_hat_u(:,t+1), K_predicted_u(:,:,t+1)] = ...
        kalman_unscented(y(:,t), 'hand_process_model', ...
            'hand_measurement_model', Q1, Q2, ...
            K_predicted_u(:,:,t), x_hat_u(:,t), 1 );

    % Calculate where the Kalman filter estimated its
    % measurement
    y_hat_e(:,t) = hand_measurement_model(x_hat_e(:,t),[]);
    y_hat_u(:,t) = hand_measurement_model(x_hat_u(:,t),[]);

end
```

```
save('filtered.mat','max_t','y_hat*','x_hat*','K_predicted*');

% Plot estimated verus actual position.
ylab = ['Px';'Py';'Pz';'Vx';'Vy';'Vz'];
for i = 1:6
    subplot(6,1,i);
    plot(1:max_t+1,x_hat_e(i,:),'r:',...
     1:max_t+1,x_hat_u(i,:),'b:',...
     1:max_t+1,x_actual(i,:),'k-');
    if i == 1
        title('Estimated versus actual position in Cartesian co-ordinates');
    end
    ylabel(ylab(i,:));
end
xlabel('Iterations (time)');
pause

% Plot estimated versus actual measurement
ylab = ['  r  ';'theta';' phi '];
for i = 1:3
    subplot(3,1,i);
    plot(1:max_t,y_hat_e(i,1:max_t),'r-',...
     1:max_t,y_hat_u(i,1:max_t),'b-',...
     1:max_t,y_actual(i,1:max_t),'k+',...
     1:max_t,y(i,1:max_t),'go');
    if i == 1
        title('Estimated versus actual observation in spherical co-ordinates');
    end
    ylabel(ylab(i,:));
end
xlabel('Iterations (time)');
pause

%subplot(4,2,[7 8]);
%ellipsoid(K_predicted(:,:,max_t));

mse_e = (x_hat_e-x_actual).*(x_hat_e-x_actual);
mse_u = (x_hat_u-x_actual).*(x_hat_u-x_actual);

ylab = ['Px';'Py';'Pz';'Vx';'Vy';'Vz'];
for i = 1:6
    subplot(6,1,i);
    stats = plotstuff(  mse_e(i,:), mse_u(i,:), ...
            K_predicted_e(i,i,:), K_predicted_u(i,i,:));
    if i == 1
        title('Mean squared error and estimated standard deviation of estimate error');
    end
    ylabel(ylab(i,:));
    disp(stats)
end
xlabel('Iterations (time)');


function stats = plotstuff(mse_e, mse_u, cov_e, cov_u)
len = length(mse_e);
% Plot MSE versus estimated covariance
plot(1:len,mse_e,'r-',...
     1:len,cov_e(:),'r:',...
     1:len,mse_u,'b-',...
     1:len,cov_u(:),'b:');

% Peak and average MSE
mse_peak_e = max(mse_e);
```

```
mse_avg_e = sum(mse_e) / length(mse_e);
mse_peak_u = max(mse_u);
mse_avg_u = sum(mse_u) / length(mse_e);

% Peak and average covariances
K_peak_e = max(cov_e);
K_avg_e = sum(cov_e) / length(cov_e);
K_peak_u = max(cov_u);
K_avg_u = sum(cov_u) / length(cov_u);

stats = [mse_peak_e, mse_avg_e, mse_peak_u, mse_avg_u, ...
        K_peak_e, K_avg_e, K_peak_u, K_avg_u];
```