Video See-Through Augmented Reality Application on a Mobile Computing Platform Using Position Based Visual POSE Estimation

by

Daniel Fischer

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2013

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

A technique for real time object tracking in a mobile computing environment and its application to video see-through Augmented Reality (AR) has been designed, verified through simulation, and implemented and validated on a mobile computing device. Using position based visual position and orientation (POSE) methods and the Extended Kalman Filter (EKF), it is shown how this technique lends itself to be flexible to tracking multiple objects and multiple object models using a single monocular camera on different mobile computing devices. Using the monocular camera of the mobile computing device, feature points of the object(s) are located through image processing on the display. The relative position and orientation between the device and the object(s) is determined recursively by an EKF process. Once the relative position and orientation is determined for each object, three dimensional AR image(s) are rendered onto the display as if the device is looking at the virtual object(s) in the real world. This application and the framework presented could be used in the future to overlay additional informational onto displays in mobile computing devices. Example applications include robotic aided surgery where animations could be overlaid to assist the surgeon, in training applications that could aid in operation of equipment or in search and rescue operations where critical information such as floor plans and directions could be virtually placed onto the display.

Current approaches in the field of real time object tracking are discussed along with the methods used for video see-through AR applications on mobile computing devices. The mathematical framework for the real time object tracking and video see-through AR rendering is discussed in detail along with some consideration to extension to the handling of multiple AR objects. A physical implementation for a mobile computing device is proposed detailing the algorithmic approach along with design decisions.

The real time object tracking and video see-through AR system proposed is verified through simulation and details around the accuracy, robustness, constraints, and an extension to multiple object tracking are presented. The system is then validated using a ground truth measurement system and the accuracy, robustness, and its limitations are reviewed. A detailed validation analysis is also presented showing the feasibility of extending this approach to multiple objects. Finally conclusions from this research are presented based on the findings of this work and further areas of study are proposed.

Acknowledgements

I would like to thank all of the people who helped to make this work possible. Specifically I would like to extend great thanks to my supervisors, Professor William Wilson and Professor David Wang for their excellent advice, insight, and wisdom. Additionally I would like to thank members of my research group, specifically, Michael Tribou, Yi-Shiuan Chen, and Adam Gomes.

Additionally, I would like to thank BlackBerry and NSERC for providing funding and equipment, which made this work possible.

Dedication

To Karen, Madison, and Ethan.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

AR	Augmented Reality
PBVS	Position Based Visual Servo
IBVS	Image Based Visual Servo
POSE	Position and Orientation
EKF	Extended Kalman Filter
PBVP	Position Based Visual POSE
2D	Two Dimensions
3D	Three Dimensions
PDA	Personal Device Assistant
PC	Personal Computer
CPU	Central Processing Unit
WLAN	Wireless Local Area Network
CAD	Computer Aided Design
SIFT	Scale Invariant Feature Transform
SURF	Speeded Up Recognition of Features
LDB	Local Difference Binary
SLAM	Simultaneous Location and Mapping
PTAM	Parallel Tracking and Mapping
fps	frames per second
CCD	Charged Coupled Device
VCM	Voice Coil Motor
LCD	Liquid Crystal Display
CV	Constant Velocity
CA	Constant Acceleration
GTMS	Ground Truth Measurement System

Chapter 1 INTRODUCTION

1.1 Video See-Through Augmented Reality and Position Based Visual POSE Estimation for a Mobile Computing Platform

Augmented Reality (AR) has its roots in the mid twentieth century through the work of pioneers such as Ivan Sutherland [1], who invented the head-mounted display. Military flight simulators, film, video games, and various other items all collectively advanced the field of AR further but the term AR was not coined until the early 1990s by Caudell [2]. The first major paper in the field of AR was published in 1992 by Feiner et al [3] and since then, a proliferation of AR applications into various applications including mobile computing has occurred.

With the recent increases in processing power, AR applications are increasingly being developed for mobile computing applications. One such application, called video see-through AR, is to superimpose virtual objects on top of the real time viewfinder image captured by the device's camera. The AR image rendered would be placed in a known position and orientation (POSE) relative to some target object captured by the device's camera. This particular video see-through AR application would therefore require the ability to track the target object by determining its POSE relative to the target object and rendering the virtual object onto the display. One approach for target object tracking would be to use Position Based Visual POSE Estimation techniques on a mobile computing platform.

Visual Servoing techniques came about in the early 1980s [4] and use visual feedback from imaging sensors as the primary control mechanism. There are two major classes of visual servoing controllers as first defined by Weiss [5]: Image Based Visual Servoing (IBVS) and Position Based Visual Servoing (PBVS). IBVS control computes control inputs based on features in images and steers the system to a location in the 2D image. In PBVS control, feature points are extracted from the 2D image and the control inputs are based on the 3D geometries of the target object and the camera model. For object tracking and POSE estimation applications, both classes of visual servoing are able to handle the case of full 3D motion [6] and each control approach has both advantages and disadvantages. Chaumette and Hutchinson provide a more thorough treatment of visual servoing techniques in the tutorial papers [7] and [8]. In both approaches, however, a priori information is required to complete the relative POSE estimation calculations.

To implement a video see-through AR application on a mobile computing device, a framework needs to be developed that does the following four major steps. The first step is to initialize the application and set up the mobile computing device's camera to take a stream of images. In the second step, image processing techniques needs to be executed for each image frame to determine the required feature points used in the object tracking algorithm. There are various image processing techniques that can be used to extract the feature point locations as detailed by Forsyth and Ponce [9]. Once the feature point locations are known, the third step can be run, which is to complete the relative POSE estimation. For this work, an Extended Kalman Filter [6] [10] was used to determine the relative POSE. Finally the fourth step can be executed, which comprises of rendering the virtual object onto the device display to give the appearance of video see through AR. A high level tutorial for implementation of video see-through AR applications on Mobile Phones is presented by Wagner et al in [11] and [12] and by Oui et al in [13] detailing considerations such as architecture, image processing, POSE estimation, and AR rendering.

This thesis will examine and present a video see-through AR application on a mobile computing device using a Position Based Visual POSE Estimation approach that is drawn from the PBVS theory. This work will also detail the extendibility and scalability of this position based visual POSE estimation approach to tracking multiple objects simultaneously and also different mobile computing devices for an AR application.

1.2 Organization

The thesis is organized in the following manner. The basic problem of video see-through AR applications on mobile computing device using the relative position and orientation (POSE) of an object with known geometry is discussed in Chapter 2 along with a review of the current work in this field. Chapter 3 provides a further examination of the general relative POSE estimation problem by providing the general mathematical framework and provides for a solution based on the Extended Kalman Filter (EKF). Chapter 4 discusses the design of an implementation for a mobile computing device based on the approach outlined in Chapter 3.

The design proposed in Chapter 4 is analyzed in a simulation environment in Chapter 5. Here a set of known conditions are used to analyze the performance of the overall system. The system proposed was implemented and validated in Chapter 6 using an optical ground truth measurement system. A set of

physical motions are tested/measured and a comparison of the results from the device to the ground truth measurement system is provided. The technical feasibility of the system is discussed along with a comparison of the simulation results. Finally conclusions and recommendations for future areas of research and improvements are detailed in Chapter 7.

Chapter 2 BACKGROUND AND RELATED WORK

The use of video see-through AR real-time tracking applications on mobile computing platforms is a relatively recent phenomenon with initial work based on tablet PCs, notebooks or even customized hardware [1]. In 1999, the ARToolkit library was presented [14] and it allowed for developers to produce computationally inexpensive optical tracking applications, which is important for the mobile computing environment. Initial mobile real-time tracking AR applications were based on a thin client approach due to limitations in computing power of the mobile computing device. The AR-PDA project [15] is an example of this approach, where the majority of computations were completed on a PC server and the pertinent information sent to the Compaq iPAQ 3630 Pocket PC PDA via a WLAN communications link.

The first self-contained AR System for a mobile phone was presented in 2004 by Möhring et al [16], which was based on a marker system. This system was limited by a slow microprocessor, memory, simple graphics, a low frame rate, and poor accuracy. Around the same time, Wagner et al presented a self-contained system on a PDA [17], which ported the ARToolkit and also critical OpenGL items needed for the AR rendering. In Wagner's initial application, the camera performance was the bottleneck with a camera image video stream of 7-8 frames per second. In both of the aforementioned cases, markers or fiducials, which are added to the scene or image space, were used to aid in the object feature tracking.

Many mobile computing AR applications today rely on marker or fiducial based tracking since these systems can be designed in a manner to optimize detection and tracking [18], which is important given the requirement for processing efficiency. Marker based approaches are used to facilitate both the feature extraction from the images and provide for reliable measurements that can be used in the POSE estimation. There are numerous applications of marker based real-time object tracking AR applications for mobile devices, such as [19] and [20] and the general theory behind marker based tracking is presented in the survey by Lepetit and Fua [21]. However the drawbacks of the marker based tracking approach, as noted by Schall et al [18], is that these markers obscure parts of the image space and also require the scene to be engineered by the addition of markers in order to run the object feature tracking application. Some work has been done to lessen the impact of markers, such as Wagner et al [22], where care is taken to minimize the marker design.

Given the drawbacks to markers, more research is being conducted in the area of markerless tracking, which rely on edges and feature points of the object. One of the primary benefits of a markerless approach is that the natural images of the scenes can be used without the addition of markers or modification to the

environment in question. As far as can be determined, the first markerless real-time tracking video see-through AR application done for a mobile phone was published by Wagner et al in 2007 [23]. Markerless tracking relies on features naturally present in the images and often 3D CAD models of objects are used. There are various approaches and methods to markerless tracking such as edge based, optical flow based, template matching, natural features. The various approaches to complete markerless tracking are outlined in the survey by Lepetit and Fua [21].

One of the prominent and most successful approaches in video see-through AR applications in mobile computing is using a modified SIFT/Ferns approach detailed by Wagner [23]. The Scale Invariant Feature Transform (SIFT), developed by Lowe [24], is based on an approach of extracting features from images and matching against an image feature database. Its computational cost is minimized by using a cascaded set of filtering steps (keypoint localization, feature description, and feature matching) with the most computationally expensive operation done only once during the first pass. The SIFT approach will generate a large number of features, which densely cover the image. For tracking purposes, the SIFT features are individually matched in the current image by comparing against the prior image. From this, a descriptor based on gradients that are weighted from their distance to a patch is generated. Ferns, which classifies features for tracking [25], uses statistical learning techniques to model the possible appearance of an image under various conditions to detect and match features. In this approach, close to perfect object/feature recognition is not required. Wagner [23] runs both a modified SIFT and Ferns algorithms concurrently but optimizes the algorithm by sharing the first (keypoint detection) and last (POSE estimation) steps along with further algorithmic adjustments that due to considerations for mobile phone computational power. The POSE estimation is completed based on the feature point information derived from both the SIFT and Ferns steps and is based on a Gauss-Newton iteration scheme. Wagner has also demonstrated successful real time tracking of multiple objects simultaneously [26] using this SIFT/Ferns approach. Others such as Skrypnyk and Lowe [27] use the SIFT features for recognition, tracking, and AR object placement.

Other descriptor based approaches like the Speeded-Up Robust Features (SURF) algorithm [28] have been used in mobile AR Applications. Descriptor based approaches like SURF can be generally broken down into three steps. First a correspondence between two images in time is found and is done by creating a map of interest points such as corners. SURF optimizes this *detection* step by using a Hessian matrix and relies on integral images to reduce computational time. Next a feature descriptor vector that describes the neighbourhood for each interest point is created in the *description* step. In the SURF method [28], the descriptor is constructed by a distribution of Haar-wavelet responses within the interest point

neighbourhood. Finally the *matching* step is conducted by comparing the descriptor vectors between different images. SURF further optimizes this step by using a 64 dimension descriptor vector instead of 128 dimensions used in most SIFT applications.

The SURF algorithm, while potentially more efficient than SIFT, was first implemented by Tackacs et al [29] for use in mobile AR application. However the focus of Tackacs' work was on detection quality on not real time POSE estimation. Ta et al [30] present a modified algorithm called SURFTrac, which is more efficient but still does not achieve real time tracking. Another descriptor approach is the Local Difference Binary (LDB) [31] introduced by Yang and Cheng, which was found to be capable of achieving real time tracking capabilities for a Mobile AR application. A good survey of descriptor based approaches is given by Mikolajczyk and Schmid [32].

An alternate approach by Klein and Murray was to implement a keyframe-based Simultaneous Location and Mapping (SLAM) system on a mobile phone [33]. SLAM, as detailed by Durrant-Whyte and Bailey [34], is a technique used by devices, robots, and autonomous vehicles in either a known or unknown environment and the SLAM approach updates or creates a map of the environment while simultaneously tracking the object's location. In the modified keyframe-based SLAM approach developed by Klein and Murray, which is denoted as Parallel Tracking and Mapping (PTAM), the mobile phone's position is tracked and mapped in an unknown environment and AR images are then superimposed. However the performance of this system was constrained by processing power and camera hardware capabilities and thus did not achieve real-time tracking capability. Klein and Murray's work was extended further by Verbelen et al [35] by implementing a collaborative and distributed PTAM, where multiple mobile devices share or rather divide up components to optimize performance.

For the markerless methods listed above, the common theme is that large numbers of feature points are tracked for descriptors or maps, resulting in a large number of required computations for each frame. These approaches tend to be computationally expensive and may not be able to achieve real-time tracking performance when considering the various other services and applications typically running on a mobile phone such as the cellular radio, messaging services, etc. An alternate approach would be to use a minimum set of features based on CAD models of known objects. In such an approach, the image processing would identify only objects of interest in the field of view based on the CAD object model library. The initialization or detection phase may possibly be more computationally expensive to identify all objects of interest against the CAD object libraries models. However, the tracking phase would be much more computationally efficient, as only a few features would be tracked, i.e. tens versus hundreds to thousands

in the approaches detailed above, allowing us to achieve real-time tracking performance when considering the various other features already running on the mobile phone environment.

The methods of position based visual POSE estimation could therefore be applied to a video see-through mobile computing AR application by creating a CAD object model library for all objects of interest, implementation of the position based visual POSE estimation method based on Wilson's approach in [6] to implement the real time tracking or relative POSE estimation, and rendering of AR onto the display. At present, there are no known published works that utilize the position based visual POSE estimation method for a video see-through mobile AR Application.

While other approaches such as Wagner [26] have demonstrated real-time tracking performance, applying position based visual POSE estimation methods to a video see-through mobile AR application will allow for improved computational efficiency by reducing the number of computations required for each time step. The reduced computational efficiency in the application will allow for multiple objects to be tracked simultaneously and also minimize performance impact to the mobile computing device in general.

Therefore the primary contribution of this work is to propose a video see-through AR Application on a mobile computing device using the position based visual POSE estimation methods. The EKF will be used to complete real time markerless tracking based on known target object models and a virtual object will be rendered on the display. A framework for extension to multiple object tracking is provided along with the framework for implementing on various mobile computing devices. For the purposes of this work, a BlackBerry Dev Alpha A device was used for the implementation.

Chapter 3 MODELING AND THEORY

Implementing a video see-through AR application on a mobile computing device requires the development of a real time object tracking framework to first determine the relative POSE between the device and the target object and then render a virtual object on the device's display based on the relative position and orientation. Hence, it is assumed that there is a known target object with features that are well defined and a relative 3D motion occurring between the mobile computing device and the target object. As discussed above in Chapter 2, there are various approaches that can be used for real-time object tracking. The methodologies of position based visual POSE estimation lends itself very well to this particular mobile AR application, as it operates solely based on the relative position and orientation between the mobile device and the target object.

As most mobile computing devices contain a camera sensor, the camera sensor is the most appropriate and logical choice to determine the relative position and orientation between the device and the target object. The mobile computing device's camera is used to capture a stream of real-time images and for each time step the device will locate the target object's feature points using image processing techniques. Once the feature points of the target object have been located, the real time object algorithms based on position based visual POSE estimation techniques can be executed providing us with the relative position and orientation between the mobile computing device and the target object.

To render the virtual object, it is assumed that the virtual object is fixed in its position with respect to the target object. Once the relative POSE between the mobile computing device has been calculated and the target object, it then becomes a simple matter of calculating the virtual object's position and rendering it on the display.

In this chapter the model of the system under scope is presented, detailing the mathematical formulation. A method is proposed to solve this POSE estimation problem using recursive estimation techniques specifically through the EKF. Finally, a discussion is presented on extending this method to rendering multiple virtual objects based on multiple target objects.

3.1 Problem

3.1.1 Coordinate Frame Definition

To implement a real time 3D object tracking method using position based visual POSE estimation, the system needs to be defined and described mathematically. Given a mobile computing device with a camera, a coordinate frame is defined as fixed to the camera frame and is denoted as C. Similarly for a known target object, a coordinate frame can be defined for the object frame and is denoted as O. The relative POSE between the camera frame C and the object frame O can be defined through the six element POSE vector W as follows:

$$W = [X, Y, Z, \emptyset, \theta, \psi]^T \tag{3.1}$$

where X, Y, Z give the Cartesian coordinates of the object frame O origin expressed in the camera frame C and $\emptyset, \theta, \psi$ represent the Euler angles for roll, pitch, and yaw with respect to the camera frame's Z_C, Y_C, and X_C axes respectively. A right handed coordinate system was selected based on the work of Spong and Vidaysagar [36] and a visual representation of the system is presented in Figure 3-1 below and the roll, pitch, and yaw angles are graphically depicted in Figure 3-2 below.

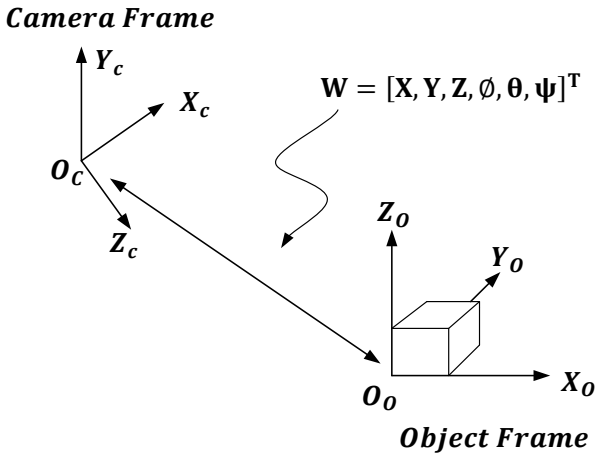

Figure 3-1: Geometric representation of system model detailing both the object frame and the camera frame along with the relative POSE vector W based on a right handed coordinate system.

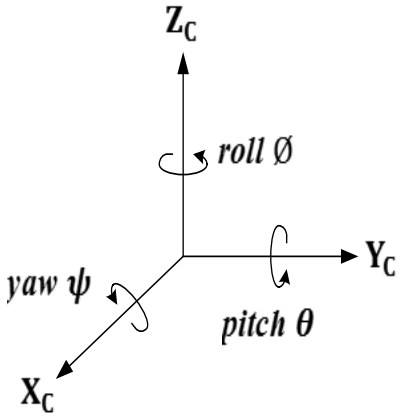

Figure 3-2: Roll, pitch, and yaw angles defined with respect to the device camera frame C with the coordinate system based on the right handed coordinate system.

The relative position of the object frame O with respect to the camera frame C is defined as the vector $T = [X, Y, Z]^T$. The relative orientation of the object frame O with respect to the camera frame C is defined through the roll $\emptyset$, pitch θ, and yaw ψ angles. The relationship can be described through a product of three rotation matrices about each of the three coordinate axes, respectively, as follows and is explained further in Appendix A.

$$R_O^C = R_{zC}(\emptyset) R_{yC}(\theta) R_{xC}(\psi) \tag{3.2}$$

Thus, the rotation matrix R_O^C is defined as,

$$R_O^C = \begin{bmatrix} C_\theta C_\psi & S_\emptyset S_\theta C_\psi - C_\emptyset S_\psi & C_\emptyset S_\theta C_\psi + S_\emptyset S_\psi \\ C_\theta S_\psi & S_\emptyset S_\theta S_\psi + C_\emptyset C_\psi & C_\emptyset S_\theta S_\psi - S_\emptyset C_\psi \\ -S_\theta & S_\emptyset C_\theta & C_\emptyset C_\theta \end{bmatrix}, where\ C_\theta = cos\,\theta, S_\theta = sin\,\theta\,, etc \tag{3.3}$$

To solve the relative POSE estimation problem, it is required that the pre-defined number of feature points of the target object be located in the mobile computing device's camera image plane I through image

processing techniques. This feature point data in the camera image plane I is then used to compute the relative POSE vector W based on the position based visual POSE estimation techniques. First, however the transformation of a feature point in the object frame O to the camera frame C needs to be considered before the transformation from the camera frame C to the camera image plane I can be dealt with. For a point j in the object frame, its position in the object frame P_j^O can be defined as follows:

$$P_j^O = \begin{bmatrix} x_j^O \\ y_j^O \\ z_j^O \end{bmatrix} \tag{3.4}$$

where x_j^O, y_j^O, and z_j^O are expressed with respect to the origin of the object frame O. The one-to-one transformation of a point j in object frame O to a point j in the camera frame C can be defined through a translation followed by a rotation as follows:

$$P_j^C = T + R_O^C(\emptyset, \theta, \psi)P_j^O \tag{3.5}$$

where P_j^C represents the Cartesian coordinates of the j^{th} feature point location in the camera frame C. Expanding Equation 3.5, the j^{th} feature point location in the camera frame can be written as follows:

$$x_j^C = X + C_\emptyset C_\theta x_j^O + \left(C_\emptyset S_\theta S_\psi - S_\emptyset C_\psi\right)y_j^O + (C_\emptyset S_\theta C_\psi + S_\emptyset S_\psi)z_j^O \tag{3.6}$$

$$y_j^C = Y + S_\emptyset C_\theta x_j^O + \left(S_\emptyset S_\theta S_\psi + C_\emptyset C_\psi\right)y_j^O + (C_\emptyset S_\theta S_\psi - S_\emptyset C_\psi)z_j^O \tag{3.7}$$

$$z_j^C = Z - S_\theta x_j^O + C_\emptyset S_\psi y_j^O + C_\theta C_\psi z_j^O \tag{3.8}$$

The transformation from the object frame to the camera frame can also be expressed using a homogenous transformation matrix as follows:

$$\begin{bmatrix} x_j^C \\ y_j^C \\ z_j^C \\ 1 \end{bmatrix} = T_O^C \begin{bmatrix} x_j^O \\ y_j^O \\ z_j^O \\ 1 \end{bmatrix} \tag{3.9}$$

where

$$T_O^C = \begin{bmatrix} R_O^C & \begin{matrix} X \\ Y \\ Z \end{matrix} \\ 0_{1x3} & 1 \end{bmatrix} \tag{3.10}$$

3.1.2 Monocular Camera Model

In order to solve the POSE estimation problem for each time step, the feature points need to be located in the camera image plane I based on a transformation of these same feature points in the camera frame C. In order to determine the relationship between the camera frame C and the camera image plane I, an appropriate model of the monocular camera sensor is required. For the purposes of this thesis, a pinhole camera measurement model was selected, as it typically provides for an acceptable approximation [9]. The pinhole model, which approximates the monocular camera sensor, maps 3D points in the camera frame C to 2D points in the camera image plane I as shown in Figure 3-3 below. The camera image plane is at a directed distance of $-f$ in the Z_C axis direction from the optical center of the camera frame O_C. It is noted that images in the camera image plane are inverted from the camera frame.

The camera frame's origin O_C represents the camera pinhole and is also by definition the center of projection. The camera image plane's origin O_I or image center can be defined by a line that is both perpendicular to the camera image plane I and a line that also passes through the pinhole, i.e. is assumed to lie along the camera frame's Z_C axis.

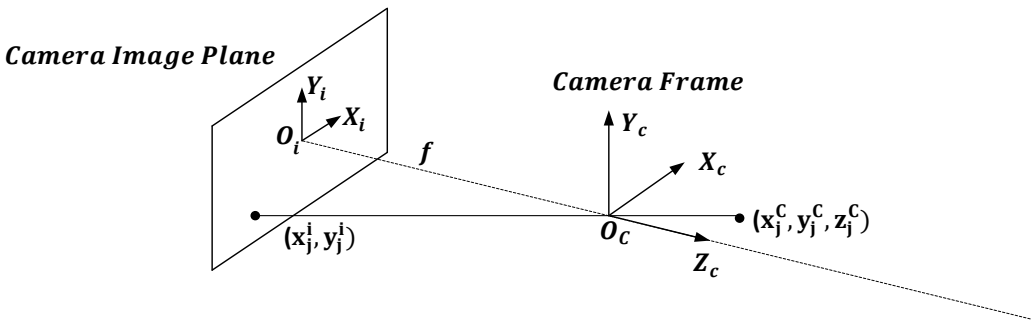

Figure 3-3: The pinhole camera model is used to represent transformation of feature points in camera frame to the camera image plane.

Considering the three-dimensional point P_j^C with coordinates (x_j^C, y_j^C, z_j^C) in C, the two-dimensional projection onto the camera image plane I can be defined as P_j^i having coordinates (x_j^i, y_j^i) in I. The point P_j^i represents the intersection on the camera image plane I of the line from the point P_j^C through the center of projection $\mathrm{O_C}$. The projection of the point j onto the camera image plane from the camera frame is defined as:

$$x_j^i = \frac{-fx_j^c}{P_X z_j^c} \tag{3.11}$$

$$y_j^i = \frac{-fy_j^c}{P_Y z_j^c} \tag{3.12}$$

where f is the focal length in meters of the monocular camera sensor and P_X and P_Y are the inter-pixel spacing of the camera along the X^i and Y^i axes and are in m/pixel. The three intrinsic camera parameters f, P_X, and P_Y can be obtained either through the manufacturer's data sheet or through camera calibration tests.

3.1.3 Mathematical Model

In summary, there are the following steps in the system. First the mobile computing device will capture a stream of images using the camera sensor. For each time step, feature points of a known target object are located in the camera image plane. Additionally for each time step, the six element relative POSE vector W needs to be determined in order to properly render the virtual object. The transformation of the j^{th} feature point from the object frame O to the camera image plane I can therefore be expressed by combining Equations 3.6, 3.7, 3.8, 3.11, and 3.12 to give us the following two expressions as follows,

$$x_j^i = \frac{-f(X+C_\emptyset C_\theta x_j^O + (C_\emptyset S_\theta S_\psi - S_\emptyset C_\psi) y_j^O + (C_\emptyset S_\theta C_\psi + S_\emptyset S_\psi) z_j^O)}{P_X(Z - S_\theta x_j^O + C_\emptyset S_\psi y_j^O + C_\theta C_\psi z_j^O)} \tag{3.13}$$

$$y_j^i = \frac{-f(Y+S_\emptyset C_\theta x_j^O + (S_\emptyset S_\theta S_\psi + C_\emptyset C_\psi) y_j^O + (C_\emptyset S_\theta S_\psi - S_\emptyset C_\psi) z_j^O)}{P_Y(Z - S_\theta x_j^O + C_\emptyset S_\psi y_j^O + C_\theta C_\psi z_j^O)} \tag{3.14}$$

where x_j^i and y_j^i represent the two Cartesian coordinates of the feature points in the camera image plane. Figure 3-4 illustrates the mapping of the j^{th} feature point from the object through the camera frame to the camera image plane.

In examining the above equations, it is clear that the six unknown elements of the relative POSE vector W are contained in Equations 3.13 and 3.14. In order to solve for W, a minimum of six independent equations are required. This implies that a minimum of three target object feature points are needed. However through Euclidean axioms [37], it is known that any two points in a vector space will be collinear. Based on the relative position and orientation between the device and the target object, it is assumed that there will be cases when these two collinear points lie directly on the principal ray of the camera and will be mapped to the same position on the camera image plane, i.e. the second feature point is directly behind the first. In these instances, the solution will be non-unique. Therefore to obtain a unique solution to this problem, a minimum of four object feature points with the constraint that no three feature points are collinear are required.

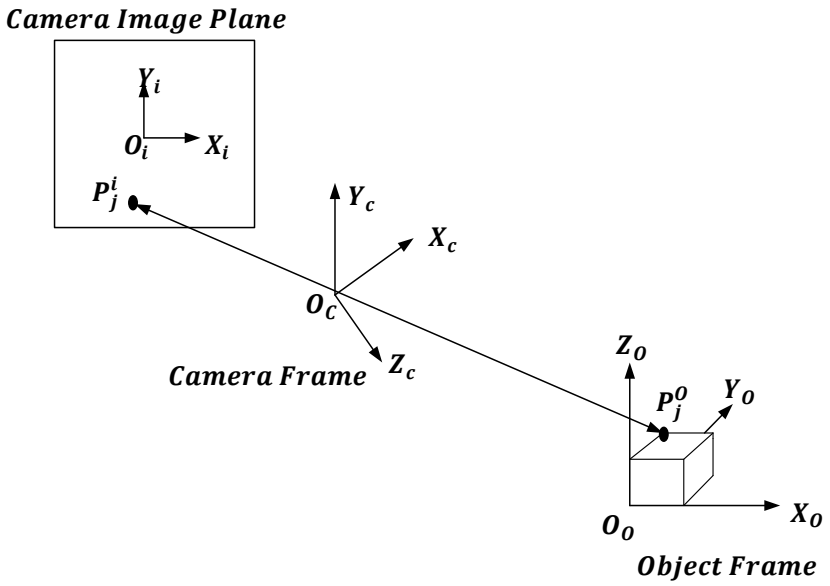

Figure 3-4: Graphical depiction of the mapping of the j^{th} feature point from the object frame, through the camera frame to the camera image plane.

Once the relative POSE vector W has been solved for each time step, rendering the video see-through augmented or virtual reality object in the camera frame becomes trivial. As was stated above, it is assumed that the virtual object is located in a fixed position relative to the target object and the virtual object feature points can be described in relation to the origin of the object frame. Figure 3-5 shows an example of a virtual object location fixed relative to the object frame.

The mathematical relationship which describes the transformation of the virtual object into the camera image plane can be easily computed using a forward calculation for each of the $M \geq 1$ virtual object feature points. A description is provided below detailing how the j^{th} point of the virtual object is transformed into the camera image plane as follows,

$$P_{AR\,j}^{O} = \begin{bmatrix} x_{AR\,j}^{O} \\ y_{AR\,j}^{O} \\ z_{AR\,j}^{O} \end{bmatrix} \tag{3.15}$$

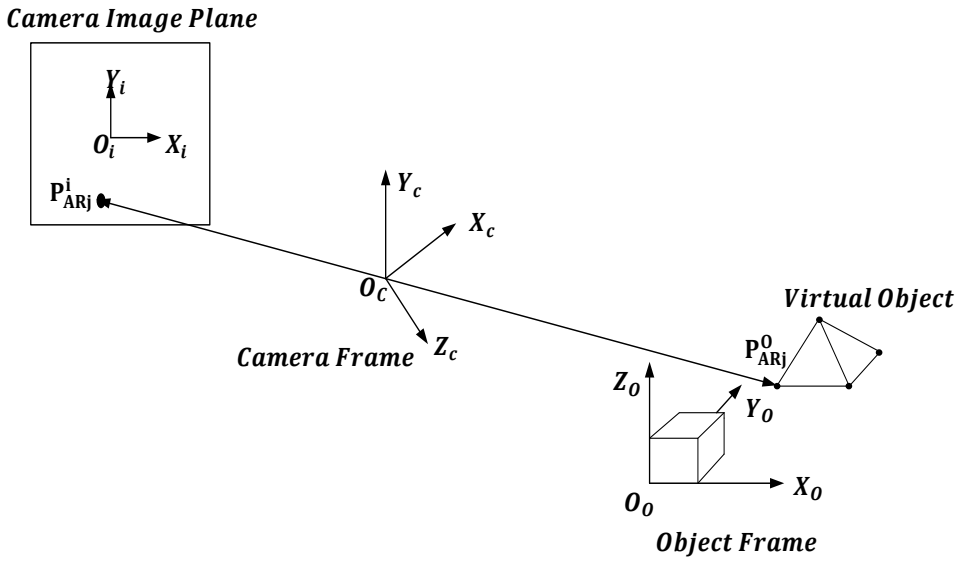

Figure 3-5: Depiction of the fixed distance between the virtual object and target object and transformation of virtual object feature points is into the camera image plane.

noting that $x^O_{AR\,j}$, $y^O_{AR\,j}$, and $z^O_{AR\,j}$ are the coordinates of the j^{th} point of the virtual object relative to the origin of the target object frame. The transformation of the j^{th} point of the virtual object to the camera frame can be described using the same relationship for transforming target object feature points into the camera frame and is shown below:

$$P^C_{AR\,j} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + R^C_O(\emptyset, \theta, \psi) P^O_{AR\,j} \tag{3.16}$$

where X, Y, Z, $\emptyset, \theta, \psi$ are the six relative POSE parameters to be determined. Once the coordinates of the j^{th} point of the virtual object are transformed in the camera frame, the transformation to the camera image plane is as follows:

$$x_{AR\,j}^{i} = \frac{-f x_{ARj}^{c}}{P_X z_{ARj}^{c}} \tag{3.17}$$

$$y_{AR\,j}^{i} = \frac{-f y_{ARj}^{c}}{P_Y x_{AR\,j}^{c}} \tag{3.18}$$

where $x_{AR\,j}^{i}$, $y_{AR\,j}^{i}$describe the virtual object's j[th] feature point locations in the camera image plane and f, P_X and, P_Y are the focal length and inter-pixel spacing parameters previously defined.

3.2 Recursive Estimation

3.2.1 General Problem

The solution to the above nonlinear photogrammetric Equations 3.13 and 3.14 is both non-trivial and non-linear. Additionally the relative POSE vector *W* is required to be solved at each time step. It is therefore noted that the use of a recursive estimation technique is desired, as this approach lends itself to this nature of problem. Based on the position based visual POSE estimation research work already done in this area most notably by Wilson [6], the EKF was selected to solve the relative POSE estimation problem. For the system described above, it can be represented by the following nonlinear discrete time state-space system [38] as follows,

$$x_k = F(x_{k-1}) + \eta_k \tag{3.19}$$

$$y_k = G(x_k) + \nu_k \tag{3.20}$$

where $x_k \in \mathcal{R}^n$ is the system state vector at time step k, $\mathrm{y_k} \in \mathcal{R}^{\mathrm{m}}$ is the output measurement vector, $F(x)\colon \mathcal{R}^{\mathrm{n}} \to \mathcal{R}^{\mathrm{n}}$ is the process model, and $G(x)\colon \mathcal{R}^{\mathrm{n}} \to \mathcal{R}^{\mathrm{m}}$ is the output measurement model, η_k is the disturbance noise vector and of zero mean Gaussian noise with covariance Q_k and ν_k is the measurement noise vector also of zero mean Gaussian noise with covariance R_k. Each of the above quantities will be discussed in subsequent sections.

As per Wilson [6], the optimality for the application of the Kalman Filter method depends on the accuracy of the state dynamic and output models and also on the assumptions that the disturbance noise η_k and measurement noise ν_k have zero mean Gaussian noise. If the aforementioned conditions are not met, good approximations are needed to achieve a near optimal solution.

3.2.2 Assumptions

The following explicit assumptions are required to implement the video see-through AR application on the mobile computing device:

a. It is assumed that the target object is known and a rigid body with a defined number of feature points. The number of feature points N is known and their Cartesian coordinates in the object frame O are known.

b. The object feature points are distinguishable in the camera image and their perspective projection can be measured at each time step using image processing techniques.

c. The monocular camera of the device is considered to be fixed within the body of the device and has an associated coordinate camera frame C.

d. It is assumed that the device camera has the ability to focus on an object that is an arbitrarily large distance away in the X, Y, and Z directions.

e. It is assumed that the camera has the ability to form images on the image plane for relative pitch θ and yaw ψ angles that fall along the camera frame Y_C and X_C axes respectively.

f. It is assumed that the image captured on the camera image plane is directly correlated to the image rendered on the device's display.

g. It is assumed that the disturbance and measurement noise processes can be accurately represented by Gaussian distributed zero mean noise with known covariance. Additionally it is assumed that these noise processes are uncorrelated.

h. Both the target object and the device are assumed to be independent with respect to their motions and it is assumed that the relative 3D motion between the camera of the device and the target object is smooth and can be approximated by a constant velocity model.

i. The virtual object feature points $M \geq 1$ are known and these virtual object feature points position are fixed relative to the object frame O.

j. It is assumed that the sampling rate for the discrete time system is constant.

3.2.3 Relative Motion Dynamics

In the physical system, there is a mobile computing device and a target object. There is no constraint for either the device or the object to be fixed in its position and both can experience independent and non-correlated motions. As a result, the relative motion characteristics between the device and the target object are generally unknown. However, it is assumed that the relative motion is smooth. Therefore a generic constant velocity model was selected to describe the relative motion over one discrete time step [6], as this is consistent with the prior research in the field of PBVS and position based visual POSE estimation. The system state Equation 3.19 can be rewritten based on our assumption of the constant velocity model as follows,

$$x_k = x_{k-1} + \delta_k \dot{x}_{k-1} + \eta_k \tag{3.21}$$

where δ_k is defined as the change in time between time steps k and $k-1$. The variable $\dot{x}_{k-1}$ represents the velocity at this time step and η_k is a vector of zero mean Gaussian disturbance noise with covariance Q. Based on the assumption of a constant velocity process model, the system state vector x_k can be represented by the six POSE parameters and their respective velocities as follows

$$x_k = \left[X, Y, Z, \emptyset, \theta, \psi, \dot{X}, \dot{Y}, \dot{Z}, \dot{\emptyset}, \dot{\theta}, \dot{\psi}\right]^T \tag{3.22}$$

Equation 3.22 can be further simplified by noting that the first six elements of the system state vector x_k represents the position and the second six elements represent the first derivatives/velocities.

$$x_k = \begin{bmatrix} I_{6x6} & \delta_k I_{6x6} \\ 0_{6x6} & I_{6x6} \end{bmatrix} x_{k-1} + \eta_k \tag{3.23}$$

where I_{6x6} represents the identity matrix and 0_{6x6} represents a 6x6 matrix of zero entries. Equation 3.24 therefore allows us to define the state space system matrix A as follows and allows us to write the discrete time state equation in the traditional form as follows,

$$A = \begin{bmatrix} I_{6x6} & \delta_k I_{6x6} \\ 0_{6x6} & I_{6x6} \end{bmatrix} \tag{3.24}$$

$$x_k = Ax_{k-1} + \eta_k \tag{3.25}$$

Since the system state vector x_k will be updated for each time step, it is noted that the velocities will also be estimated for each time step. Additionally, the constant velocity model allows for errors in the dynamic model to be included through the disturbance noise η_k, which acts as inputs to the system state equations. Sampling rates will also affect the dynamic model accuracy for each time step. It follows that a smaller time step or higher frequency between samples will decrease the modeling error, as can be seen from the following definition of model error for each time step,

$$Dynamic\ Model\ Error \propto \delta_k * |\dot{x}_{actual} - \dot{x}_{model}| \tag{3.26}$$

Hence a higher sample rate is generally recommended as it will decrease the error. Furthermore it follows from Equation 3.26 that when the difference between the actual system's velocity and predicted or estimated velocity is large, higher modeling errors will be seen, which are encapsulated in the disturbance noise η_k as per the definition in Equation 3.21. Other relative motion dynamic models such as the constant

acceleration model could be selected. While this could potentially improve the model prediction, it also has the effect of increasing model complexity and therefore this approach was not pursued.

3.2.4 Relative POSE Estimation

As noted above, a minimum of $N \geq 4$ object feature points are required to generate a unique solution assuming that no three object feature points are collinear. Furthermore, it has been shown that performance will not be improved significantly for more than six feature points [39] [40]. While additional feature points beyond six will result in an improvement to the accuracy solution of the system equations, a balance between computational complexity and accuracy is required. As the focus of this work is to implement a real-time system on a mobile computing device with various applications running, the required number of feature points for this system was mathematically set as follows,

$$4 \leq N \leq 6 \tag{3.27}$$

In this system, the output measurement vector y_k represents the feature point locations of the N known feature points on the camera image plane and can be represented as follows:

$$y_k = \begin{bmatrix} x^i_{1_k} & y^i_{1_k} & x^i_{2_k} & y^i_{2_k} & \dots & x^i_{N_k} & y^i_{N_k} \end{bmatrix}^T \tag{3.28}$$

Based on the mathematical model of this system and specifically Equations 3.11 and 3.12 that define the feature point locations in the camera image plane, an expression for the output model $G(x)$ can be written as follows:

$$G(x_k) = -f \begin{bmatrix} \frac{x^c_{1_k}}{P_X z^c_{1_k}} & \frac{y^c_{1_k}}{P_Y z^c_{1_k}} & \frac{x^c_{2_k}}{P_X z^c_{2_k}} & \frac{y^c_{2_k}}{P_Y z^c_{2_k}} & \dots & \frac{x^c_{N_k}}{P_X z^c_{N_k}} & \frac{y^c_{N_k}}{P_Y z^c_{N_k}} \end{bmatrix}^T \tag{3.29}$$

The above expression is noted to be non-linear in the six unknown relative POSE parameters $[X, Y, Z, \emptyset, \theta, \psi]^{\mathrm{T}}$ and needs to be solved at each time step. In order to solve the relative POSE estimation problem for each time step, the Extended Kalman Filter framework was selected [6] given its applicability and reliability to problems of this nature. For this system, the EKF will be used to estimate the states recursively at each time step and will be refined or steered to the actual value by comparing the locations of the feature point pairs in the camera image plane with an estimate of these feature points.

The EKF requires an initialization step, which is driven by the physical model of the system and discussed further in Chapter 3.2.6. Once the initialization has been completed, the EKF has two main elements that proceeds recursively for each time step in the system.

The first step is the *Prediction Step*, which performs an optimal estimate of the system state based on the process model of the system dynamics. Assuming that the current time step is k, the current state estimate $\hat{x}_{k,k-1}$ is calculated using the process model A and the estimate of the prior state $\hat{x}_{k-1,k-1}$ at the *k-1*th time step,

$$\hat{x}_{k,k-1} = A\hat{x}_{k-1,k-1} \tag{3.30}$$

The estimate covariance $P_{k,k-1}$ is determined through the process model A, the estimate covariance $P_{k-1,k-1}$ (meaning estimate covariance of the *k-1*th time step using measurements up to the *k-1*th time step), and the disturbance noise covariance Q_k,

$$P_{k,k-1} = AP_{k-1,k-1}A^T + Q_k \tag{3.31}$$

The next step is the *Estimate Update Step*, where measured feature point locations on the camera image plane are used to adjust the prediction of the state estimates. The output measurement model $\mathrm{G}(\mathrm{x_k})$ described in Equation 3.24 is linearized about the current state estimate $\hat{x}_{k,k-1}$ resulting in the measurement Jacobian C_k,

$$C_k = \left.\frac{\partial G(x)}{\partial X}\right|_{x=\hat{x}_{k,k-1}} \tag{3.32}$$

The mathematical derivation and resulting equations of the measurement Jacobian are detailed in the Appendix B. An estimate of the output $\hat{y}_k$, which in this case represents an estimate of the feature point locations on the camera image plane, is then calculated using the measurement Jacobian C_k and the current state estimate $\hat{x}_{k,k-1}$,

$$\hat{y}_k = C_k \hat{x}_{k,k-1} \tag{3.33}$$

The Kalman Gain K is then calculated using the estimate covariance $P_{k,k-1}$ calculated in equation 3.31, the measurement Jacobian C_k determined in equation 3.32, and the measurement noise covariance R_k,

$$K = P_{k,k-1}\, C_k^T \left(R_k + C_k P_{k,k-1} C_k^T\right)^{-1} \tag{3.34}$$

Finally, the state estimate $\hat{x}_{k,k}$ and estimate covariance $P_{k,k}$ are updated for the current time step using the Kalman Gain K and the *innovation*, which is defined as the difference between the measured output y_k and the estimated output $\hat{y}_k$, and is used to steer the control system to the actual values,

$$innovation = y_k - \hat{y}_k \tag{3.35}$$

$$\hat{x}_{k,k} = \hat{x}_{k,k-1} + K(y_k - \hat{y}_k) \tag{3.36}$$

$$P_{k,k} = P_{k,k-1} - K C_k P_{k,k-1} \tag{3.37}$$

3.2.5 System Model

Using the recursive state estimation technique of the EKF detailed above, this system can now be mathematically described as a linearized discrete time state space as follows,

$$x_k = Ax_{k-1} + \eta_k \tag{3.38}$$

$$y_k = C_k x_k + \nu_k \tag{3.39}$$

The relative POSE estimation and video see-through AR system model can be described using the following process flow and can be graphically represented using the block diagram in Figure 3-6 below:

1. Set up initial conditions including $\hat{x}_{1,1}$, Q_k, R_k, A, and $P_{1,1}$ and set the time step counter k=1, where $\hat{x}_{1,1}$ is the initial state estimate of the relative POSE parameters, Q_k and R_k are the disturbance and measurement noise covariance matrices respectively, and $P_{1,1}$ is the initial estimate covariance value.
2. Locate target object initial position on camera image plane through image processing detection techniques, noting that target object located will be compared against CAD model object library.
3. Start Recursive Estimation Loop
    a. Increment step counter, i.e. $k = k + 1$
    b. Capture image on camera image plane and through image processing techniques, locate each of the N feature points of the known target object, i.e. x_j^i and y_j^i that form the measured output y_k.
    c. Extended Kalman Filter Prediction Step
        i. Carry out prediction step of EKF to determine new state estimate for $\hat{x}_{k,k-1}$ and the estimate covariance $P_{k,k-1}$ based on the state estimate and estimate covariance of the prior time step.
    d. Extended Kalman Filter Measurement Update Step
        i. Linearization Step for each time step to find the measurement Jacobian C_k and predict the output $\hat{y}_k$ based on the state estimate of the current time step $\hat{x}_{k,k-1}$
        ii. Determine the Kalman Gain K

iii. Estimate Update step to refine the state estimate $\hat{x}_{k,k}$ and the estimate covariance $P_{k,k}$

e. Update the camera image plane feature point locations of virtual object image for each of the M points of the virtual object, i.e.,

$$x_{AR_j}^{i} = \frac{-f x_{AR_j}^{c}}{P_X z_{AR_j}^{c}} \tag{3.40}$$

$$y_{AR_j}^{i} = \frac{-f y_{AR_j}^{c}}{P_Y x_{AR_j}^{c}} \tag{3.41}$$

f. Render new virtual object on display based on updated virtual object feature point locations

4. End Recursive Estimation Loop

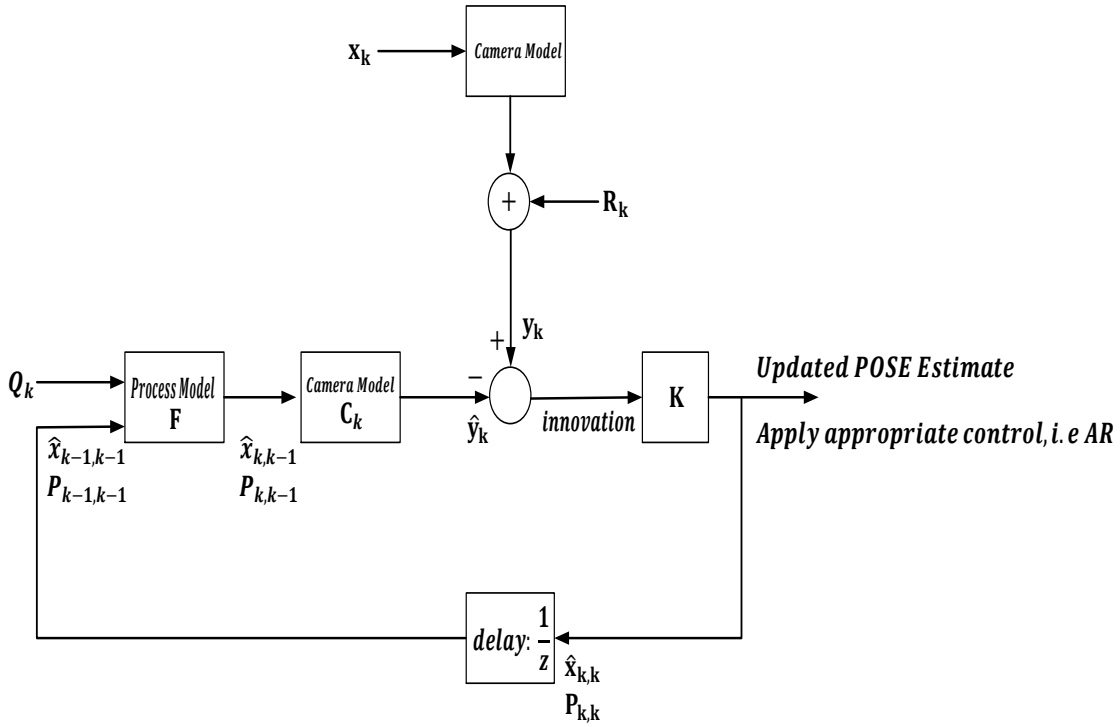

Figure 3-6: Block Diagram detailing the Relative POSE Estimation System noting that the control mechanism is completed through a comparison of the measured output y_k with the predicted/estimated output $\hat{y}_k$.

3.2.6 Initialization

Initialization of the EKF is important as incorrect parameter selection can easily result in a divergent outcome. For the EKF, there are four main parameters that require consideration as follows:

a. The initial state estimate $\hat{x}_{1,1}$ of the relative POSE parameters
$\left[X, Y, Z, \emptyset, \theta, \psi, \dot{X}, \dot{Y}, \dot{Z}, \dot{\emptyset}, \dot{\theta}, \dot{\psi}\right]^T$

b. The initial estimate covariance $P_{1,1}$

c. The disturbance covariance matrix Q_k

d. The measurement noise covariance matrix R_k

For the initial state estimate $\hat{x}_{1,1}$, care must be taken to ensure that the values of the relative POSE parameter selected are within the field of view of the system. For instance, if the initial state estimate is chosen such that any of the feature points of the target object are occluded, then convergence of the EKF may not be possible. The field of view of the camera will further constrain the system as it will restrict possible values of the roll angle $\emptyset$ about Z_C, and the yaw angle ψ about X_C. Furthermore, the initial state estimate should be selected such that it is in the neighbourhood of the target object and not occluded. Various approaches are proposed in literature such as [40], [41], and [42] to help deal with poor EKF initialization. For the general model, it is assumed that filter initialization occurs in a manner to produce a convergent solution. However the specific method used for initialization in this work will be discussed in Chapter 4.

The estimate covariance $P_{1,1}$ represents the confidence on the initial state estimate. Generally speaking if no a priori knowledge of the target object position is available, the value of $P_{1,1}$ would be set high indicating that the confidence in the estimates are low.

The disturbance covariance matrix Q_k selection is not simple as it represents the un-modeled system dynamics of the constant velocity model. For example, higher order terms of the relative POSE parameters such as acceleration and modeling error are encapsulated within Q_k. The matrix element values of the disturbance covariance matrix Q_k are generally selected to be larger for faster motion parameters and lower for slower motion parameters. Experimentation can be used to find a more optimal selection for Q_k. The measurement noise covariance matrix R_k represents the error in the image measurement system, which in this case is the camera of the mobile computing device. The matrix element values of the measurement noise covariance matrix R_k can typically be determined through experimentation and also through insight of the physical system intrinsic parameters. In this system, the measurement noise represents the error of the camera sensor in pixels and the measurement noise covariance can be determined through knowledge of the physical sensor's characteristics.

3.2.7 Effective Workspace

The concept of the effective workspace between the device and the target object is defined as the area in which the mathematical system model defined above can operate. As illustrated in Figure 3-7, the target object is occluded from the device's camera field of view, as it is physically behind the device. The constraints on the effective workspace can be intuitively determined by examining the system model.

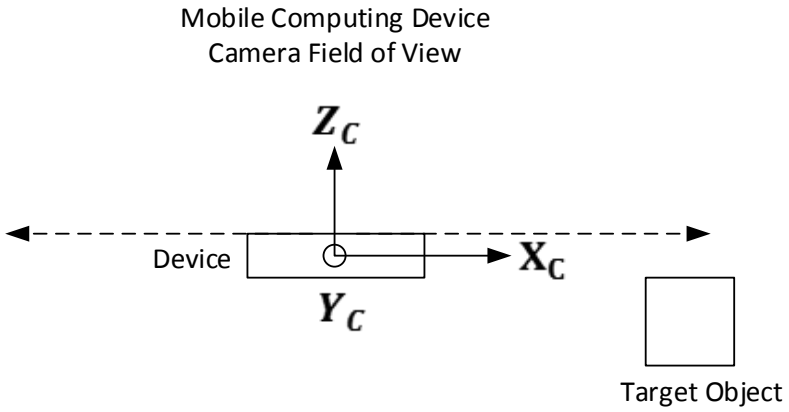

Figure 3-7: Illustration of how the target object can be occluded from the device's camera field of view if the relative roll angle $\emptyset$ is outside the camera's effective field of view.

There are no theoretical constraints in the relative X and Y directions as it is assumed that the device camera has the ability to form images on an object that is an arbitrarily large distance away. For the relative Z POSE parameter, it is known that the device and target object must have a positive distance, i.e. $Z > 0$, between them otherwise the target object would be physically behind the camera and would be occluded from view. As infinity focus was assumed for the device camera, there is no constraint for positive relative Z distances. This allows us to write the following three constraints for the Cartesian relative POSE parameters,

$$-\infty < X < \infty \tag{3.42}$$

$$-\infty < Y < \infty \tag{3.43}$$

$$0 < Z < \infty \tag{3.44}$$

For the relative POSE angles, constraints in the field of view will occur when the target object is at an angle that is greater than the device camera field of view. For the relative roll angle $\emptyset$ about Z, there are no occlusions or constraints. Additionally it was assumed that the camera has the ability to form images on the image plane for relative pitch θ and yaw ψ angles that fall along the camera frame Y_C and X_C axes respectively. Therefore as the relative pitch angle θ about Y and the relative yaw angle ψ about X increase above $\pm\frac{\pi}{2}$, the target object will be occluded from the device camera's field of view. This allows us to write the following three constraints for the Euler angle relative POSE parameters,

$$-\pi < \emptyset \leq \pi \tag{3.45}$$

$$-\frac{\pi}{2} < \theta < \frac{\pi}{2} \tag{3.46}$$

$$-\frac{\pi}{2} < \psi < \frac{\pi}{2} \tag{3.47}$$

3.3 Extension to Multiple Target Objects

Next the case for extending the current system model to include multiple target objects is considered. In this instance, a real time object tracking framework is required to be applied to each of the target objects to determine the relative POSE between the device and each target object and then render a unique virtual object for each target object based on the relative POSE data.

3.3.1 Assumptions Required for Multiple Target Objects

In order to extend the mathematical model and AR Framework to multiple target objects, a few additional assumptions are required from what was assumed and considered previously in Chapter 3.2.2:

a. It is assumed that there are $m > 1$ target objects, where m is a positive integer.

b. It is assumed that each target object is known and it is a rigid body with a defined number of feature points. The number of feature points for each $k \in \{1 \dots m\}$ target objects is defined by a set of points, where $4 \leq N \leq 6$, and their Cartesian coordinates in their respective object frames are perfectly known.

c. It is assumed that for each target object that no three feature points are collinear in the object frame.

d. The object feature points for all m target objects are distinguishable in the camera image and their perspective projection can be measured at each time step using image processing techniques.

e. The device and the m target objects are assumed to be independent with respect to their relative motions.

f. It is assumed that the relative 3D motion between the camera of the device and each of the target objects are smooth and can be approximated by a constant velocity model.

g. The number of feature points for each $k \in \{1 \dots m\}$ virtual objects are known and these virtual reality point positions are fixed relative to each of the target object.

h. It is assumed that the sampling rate for the discrete time system is constant.

3.3.2 Relative POSE Estimation Model for Multiple Target Objects

Extending the mathematical model to account for multiple target objects is trivial, as the procedure outlined in Chapter 3.2.5 is repeated for each target object at each time step. The relative POSE estimation and video see-through AR process flow is modified as follows with the pertinent changes detailed in **bold-face** text:

1. **For each *1…m* target objects set up initial conditions ($\hat{x}_{1,1}^{m}, Q_k^m, R_k^m, A^m$, and $P_{1,1}^{m}$)**

2. **For each *1…m* target objects**, **locate target object initial position on camera image plane through image processing detection techniques. It is noted that each of the *m* target objects located will be compared against CAD model object library**.
3. Start Recursive Estimation Loop
    a. Increment step counter, i.e. $k = k + 1$
    b. Capture image on camera image plane and through image processing techniques, locate each of the feature points for each known target object, i.e. x_j^i and y_j^i that form the measured output y_k^m.
    c. **Employ unique object identification techniques to ensure that the feature points are properly grouped to their respective objects and none are erroneously categorized to a different target object during the object tracking phase.**
    d. **For each *1…m* target objects**, **complete the following:**
        i. Complete the Extended Kalman Filter Prediction Step to determine new state estimate for $\hat{x}_{k,k-1}^m$ and the estimate covariance $P_{k,k-1}^m$
        ii. Complete the Extended Kalman Filter Measurement Update Step to find the measurement Jacobian C_k^m , predict the output $\hat{y}_k^m$ based on the state estimate of the current time step $\hat{x}_{k,k-1}^m$, determine the Kalman Gain K^m, and complete the Estimate Update step to refine the state estimate $\hat{x}_{k,k}^m$ and the estimate covariance $P_{k,k}^m$.
        iii. Update the camera image plane feature point locations for each virtual object.
        iv. Render new virtual object positions based on updated virtual object feature point locations.
4. End Recursive Estimation Loop

3.3.3 Considerations for Multiple Target Object Relative POSE Estimation

Generally speaking, the same parameters apply for model initialization as in the case of a single object as detailed in Chapter 3.2.6. One additional complication is that field of view of the device camera will play a more important role in the ability of the system to track multiple target objects. Specifically the field of view will limit the number of objects that can be physically tracked in real time due to the fact that the camera image plane is finite in size.

As well, care must be taken in the image processing techniques to ensure that unique object identification techniques are employed. This requirement exists in order to ensure that the image processing techniques do not erroneously skip to a different target object during the object tracking phase, which could lead to an inaccurate result. There are various approaches available to ensure unique object identification. Wagner [26] provides a good summary of various unique object identification approaches such as particle filters for motion and appearance estimation [43] [44] [45], or online learning techniques [46] [47] [48]. For the general model, it is assumed that unique object identification is incorporated into the image processing techniques. However, the specific method used for unique object identification will be discussed in Chapter 4. Figure 3-8 provides an example of how the feature points of multiple target objects can be simultaneously transformed into the camera image plane and it is noted that no update to the mathematical model is required.

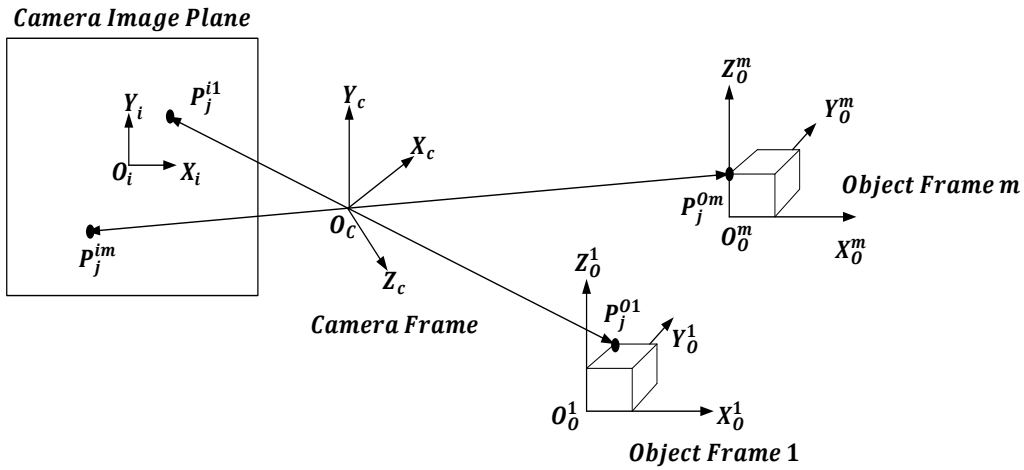

Figure 3-8: Depiction of transformation of feature points of multiple target objects into the camera image plane.

Chapter 4 SYSTEM IMPLEMENTATION

4.1 Overview and Requirements

The aim of this work is to physically implement a real-time video see-through AR application on a mobile computing device. This requires that object tracking and video see-through AR rendering will occur in a manner such that the AR image is smoothly rendered onto the display of the device. It is known that 20 frames per second represent the minimum requirement for our eyes and brain to receive animations without a sense of lag [49]. This requirement gives a bound for the minimum system performance. Therefore the system is required to operate within a minimum of 20 frames per second or more precisely complete the required operations within a 50ms period. Within this 50ms period, it is required to complete the following operations at a high level:

a. Complete image processing to locate the feature points of the known target object. The target object has pre-defined and known feature points that are distinguishable by the camera sensor and image processing techniques.
b. Complete relative POSE estimate to determine the six POSE parameters given the feature point inputs
c. Render a video see-through AR image that is overlaid on top of the current image on the view finder of the device's display and is based on a fixed position from the known target object.

This chapter is therefore broken down into an examination of the physical system selected, a discussion of the specific design and implementation details determined through the course of this work, an overview of the system model derived, extensions to multiple object tracking, and finally the verification and validation approaches used to quantify the system.

4.2 Physical System Model

4.2.1 Known Target Object Model: Credit Card

For the known target object model, there are three requirements that will help to guide the selection for this thesis. The first requirement is that the target object is known and it is a rigid body with a defined number of feature points that can be extracted from the object. The number of feature points is defined by a set of points where $4 \leq N \leq 6$ such that no three feature points are collinear. Secondly, it is required that the known target object feature points are distinguishable in the camera image plane and their perspective projection can be measured at each time step using image processing techniques. Finally, a target object is required to be selected that will allow for repeatability and reproducibility of results.

A credit card was selected as the known target object model where the corners are defined as the four feature points. This model selection also allows us to minimize computation since the number of feature points can be set to $N = 4$. It is noted that additional feature points will improve accuracy but will also increase computational complexity. The credit card physical dimensions are defined in the ISO 7810 standard [50] which defines the Physical Characteristics of Credit Card Sizes. The definition of the target object model is shown in Figure 4-1 below and the dimensions used in this research work are provided in Table 4-1 below.

Table 4-1: Known Target Object Model Physical Dimensions

Attribute	Dimension (mm)	Corresponding Axis in Object Frame
Length	85.6mm	X_O
Width	55.2mm	Y_O
Height	0.76mm	Z_O

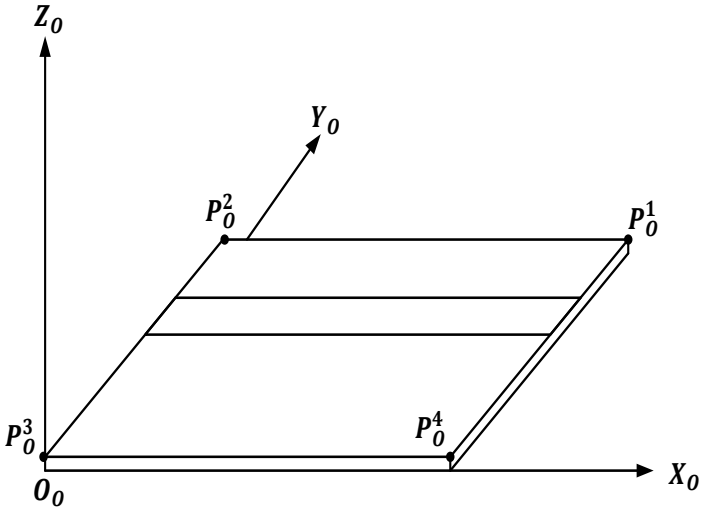

Figure 4-1: Known object model frame definition depicting the credit card and the feature point locations used in the relative POSE estimation.

4.2.2 Mobile Computing Device: BlackBerry Dev Alpha A

Since BlackBerry and the University of Waterloo contributed to the funding of this research, a device in the BlackBerry portfolio was used. The BlackBerry Dev Alpha A device [51] was selected for the purposes of this research due to its processing power (1.5GHz dual core microprocessor) and 8MP auto focus rear facing camera. It was felt that the processing power would be sufficient to implement the EKF, image processing, and virtual object rendering in real time and that the camera resolution would be sufficient to provide for distinguishable feature points. The physical or intrinsic parameters of both the display and camera of the device impacts the relative POSE estimation and these are discussed below.

4.2.3 Camera Physical Characteristics

The rear facing camera of the device is an 8MP auto focus camera and is an integrated camera module containing multiple parts including the camera lens, the CCD sensor, and a small micro motor or Voice

Coil Motor (VCM), which is used to adjust the auto focus of the camera effectively changing the focal length. For the purposes of this project implementation, the auto focus is fixed in the software allowing us to assume a fixed focal length ($f = 3.32mm$), which was obtained through calibration. Additionally the inter-pixel spacing of the camera was known ($P_X = P_Y = 1.4\mu m$) and easily obtained through calibration.

As well, the camera module has several lenses that will cause a level of refraction and distortion [9]. For the purposes of this project, the physical monocular camera model was modeled using a pinhole camera and the modeling error, i.e. distortion, is encapsulated into the camera measurement noise covariance matrix R_k parameter of the EKF. Additionally, it is assumed that the camera CCD sensor captures images with a global shutter, i.e. all parts of the image are recorded simultaneously. However it is known, as per Klein and Murray [66], that a CMOS sensor will record images with a rolling shutter such that the left edge of the scene will be recorded with some lag after the right edge of the scene resulting in a bias error.

Furthermore, it is noted that the benefit of using the EKF approach is that it allows for inclusion of modeling error into the parameters Q_k and R_k. In this case the assumptions of the constant velocity model and the pinhole camera model can be used and these modeling errors can be effectively encapsulated into the disturbance covariance matrix Q_k and the measurement noise covariance matrix R_k respectively.

4.2.4 Display Physical Characteristics

The display used in the BlackBerry Dev Alpha A device is based on a standard Liquid Crystal Display (LCD) technology comprising of composite pixels and a backlight. The RGB pixel array has a dimension of 760 x 1280 pixels. The origin of the display is at pixel location (1, 1) and is located in the top left corner of the display pixel plane. The optical center is assumed to be located at the exact center of the display, i.e. at display pixel plane location (380, 640). The feature points located through image processing are identified with respect to display pixel plane coordinates.

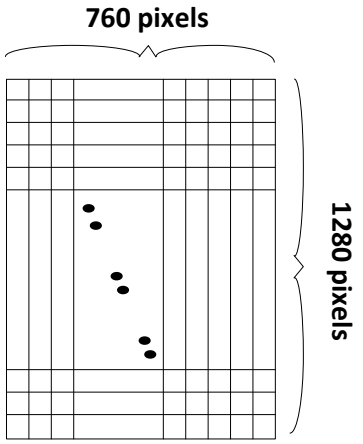

Figure 4-2: Device display pixel plane array noting that display origin is located at the top left corner.

4.3 Design

4.3.1 Design Assumptions

The following explicit assumptions are required to implement the AR application on the mobile computing device:

a. All of the assumptions from Chapter 3.2.2 apply unless otherwise noted.

b. The monocular camera of the device is considered to be fixed within the body of the device and has an associated coordinate camera frame C.

c. It is assumed that the monocular camera of the device can be modeled by the pinhole camera model and that the assumptions of global shutter apply.

d. It is assumed that the optical center of the display pixel plane is directly coupled to the camera image plane's origin O_I.

e. It is assumed that the camera sensor contains a different number of pixels than the display pixel plane. Therefore a transformation or mapping is required between the camera image plane and the display pixel plane.

4.3.2 Sampling Rates and Relative Motion Dynamics

Based on the hardware system design of the BlackBerry Dev Alpha A, the rear facing camera provides an image stream of up to 30 frames per second or every 33ms. Coupled with the requirement for completing the entire set of operations (image processing, POSE estimation, video see-through AR updates) within 20 frames per second or every 50ms, a suitable sampling rate must be selected.

Given that the BlackBerry Dev Alpha A device is a mobile handset running multiple and often unrelated processes/threads, it was felt that fixing the sampling rate somewhere between 33ms to 50ms period could potentially impact the performance. Therefore, it was decided to allow the sampling rate parameter to dynamically vary from what is normally a constant parameter. Noting that a constant velocity model of the relative motion dynamics is assumed, it becomes clear that the impact of the dynamic sampling rate causes the process model *A* to vary over each time step. The equation for the relative motion dynamics to account for a varying time step can be modified as follows,

$$\hat{x}_k = A_k \hat{x}_{k-1} + n_k \tag{4.1}$$

for the k^{th} time step where n_k is a vector of zero mean Gaussian disturbance noise with covariance Q_k. To consider a dynamic sampling period, the process model $A_k$is therefore modified as follows:

$$A_k = \begin{bmatrix} I_{6x6} & (t_k - t_{k-1})I_{6x6} \\ 0_{6x6} & I_{6x6} \end{bmatrix} \tag{4.2}$$

$$= \begin{bmatrix} I_{6x6} & \delta_k I_{6x6} \\ 0_{6x6} & I_{6x6} \end{bmatrix} \tag{4.3}$$

where I is the identity matrix and δ_k is the change in sampled time from the $k\text{-}1^{th}$ time step t_{k-1} to the k^{th} time step t_k. In Chapter 3, δ_k was constant since the sampling rate was constant. With a varied sampling rate δ_k will vary as shown above.

4.3.3 Image Processing to Locate the Target Object Feature Points

Image processing will be required to locate the feature points of the known target object. In this particular application the image stream provided to the viewfinder window is processed through the specialized image processing hardware and then passed to the microprocessor, where further processing is completed before being output to the display [52]. Image processing is then used to locate the target object feature points with respect to the display pixel plane.

In order to complete the required Image Processing operations, to locate the feature points on the display pixel plane, OpenCV [53] libraries were used. As OpenCV is an open source library with a focus on real time image processing, it was an ideal selection for inclusion in this project since it contains many of the required functions needed such as Gaussian blur, Sobel filter, and morphological operations. The Image Processing and Feature Point Location module developed using the OpenCV library can be broken down into two distinct phases: detection and tracking.

In the detection phase, the entire image is scanned and the required image processing functions are performed. There are two cases where the detection portion of the image processing algorithms is executed. Initially at the beginning of the operation where the device has no a priori knowledge of the object and its feature point locations in the image and secondly when the image processing component loses track of the object. In a real time system, entire image scanning can be a relatively expensive operation and is avoided where possible in this implementation.

In the tracking phase, only a subset region of the image is processed based on the location of the target object in the camera image plane and is typically referred to as windowing. Given that a constant velocity model is assumed for the relative motion dynamics, the use of windowing is considered as a valid approach [5]. It was found during trial runs that the tracking phase can vary from 12ms to 25ms, which is below the 33ms to 50ms requirement.

The selection of the credit card as the target object raised some challenges in the extraction of the feature points due to the fact that credit card corners are round. As a result, the image processing algorithm approach was to detect the four edges of the target object and then use the line intersections to extrapoloate the four object feature points. In the detection phase, the entire image was scanned whereas in the tracking phase a window around the entire target object was created. One benefit to this algorithmic approach is that multiple object tracking is easily handled, as each full target object is windowed separately making

unique object identification trivial. The algorithms for both the Detection and Tracking Phase of the image processing module are detailed more fully in Appendix C for reference.

Overall sources of error in the image processing approach used can be broken down as follows. The first source of error is due to the small difference in the physical object model versus the CAD object library where the credit card has rounded corners and the CAD object model assumes a rectangle. A second source of error is due to the camera sensor itself, which includes the typical sensor error along with the error caused due to the rolling shutter effect.

4.3.4 Display Pixel Plane to Camera Image Plane Transformation

As previously stated, the image processing techniques will locate the feature points of the target object with respect to the display pixel plane. To solve the relative POSE estimation problem, the feature points are required to be located in the camera image plane. Therefore the required transformation between the display pixel plane and the camera image plane needs to be determined.

As noted in Chapter 3.1.2, the image captured by the camera sensor is inverted from the image that is seen in the camera frame. The image that is rendered to the display is then corrected, i.e. re-inverted, to align with the image that is seen in the camera frame. Furthermore, it is assumed that there is no one-to-one mapping between the display pixel plane and the camera image plane, as the number of display pixels is different from the number of camera sensor pixels. Therefore, some scaling factor is also required. To determine the transformation and mapping from the display pixel plane to the camera image plane, the following assumptions are required:

a. Auto focus of the camera module is turned off, allowing us to assume a fixed focal length f.
b. The origin of display pixel plane corresponds to the optical center of the camera image plane.
c. The x and y axes of the camera image plane are defined in the horizontal direction of the image sensor.
d. Transformation of object feature points through the camera frame to the camera image plane is defined by the pinhole camera model described above.

The transformation of the feature points from the display pixel plane to the camera image plane can be defined through a function $h(x, y): \mathcal{R}^2 \rightarrow \mathcal{R}^2$, which is derived as follows:

1. The feature point pair (x_j^d, y_j^d) is obtained through Image Processing techniques and represents the feature point location in the display pixel plane. The feature point pair (x_j^d, y_j^d) is measured with respect to the top left corner of the display.
2. Express the feature point pair with respect to camera image plane coordinate system. A translation is required to represent the feature point pair with respect to the optical center or origin of the camera image plane (OC_x, OC_y). It is assumed that the optical center of the display pixel plane (360 pixels, 640 pixels) corresponds to the optical center of the camera image plane. Hence the transformation can be initially expressed as follows,

$$x_j^i = x_j^d - OC_x \tag{4.4a}$$

$$y_j^i = y_j^d - OC_y \tag{4.4b}$$

3. Invert the feature point pair, i.e. apply the image negation required from display pixel plane to camera image plane to correct for the operation performed by the specialized image processing hardware.
4. Scale the feature point pair x and y coordinates to account for the physical difference in size between the display pixel plane and the camera image plane. The scaling factors S_x and $\mathrm{S_y}$ are applied to the negated values in Equations 4.4a and 4.4b as follows,

$$x_j^i = S_x\left(-x_j^d + OC_x\right) \tag{4.5a}$$

$$\mathrm{y}_j^i = \mathrm{S_y}\left(-\mathrm{y}_j^d + \mathrm{OC_y}\right) \tag{4.5b}$$

5. Adjust the axes definition to match that of camera image plane. In this physical system with the above noted model definition of a right handed coordinate system, an inversion of the y axis coordinates is required as follows,

$$x_j^i = S_x\left(-x_j^d + OC_x\right) \tag{4.6a}$$

$$y_j^i = S_y\left(y_j^d - OC_y\right) \tag{4.6b}$$

where x_j^i, y_j^i are the Cartesian coordinates on the camera image plane,

x_j^d, y_j^d are the Cartesian coordinates on the display pixel plane,

OC_x, OC_y are the optical center coordinates of the display pixel plane (360, 640),

S_x, S_y are the scaling parameters from the display pixel plane to the camera image plane

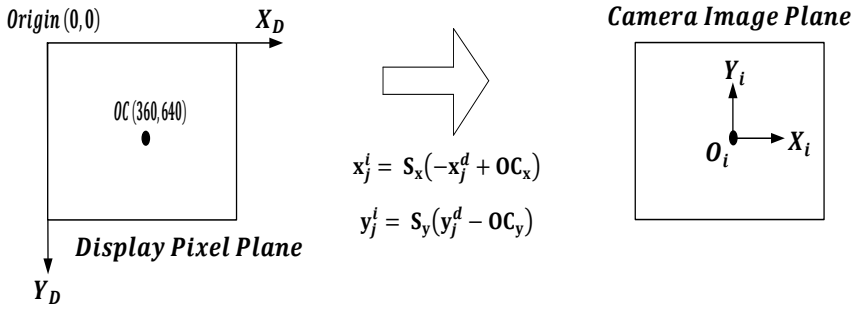

Figure 4-3: Transformation of feature points from the display pixel plane to the camera image plane.

It is noted that the transformation between the display and the camera sensor can be obtained through calibration tests, which require no a priori knowledge of the intrinsic parameters of both the display and the camera sensor. The calibration routine [54] used in this research is detailed in Appendix D. The scaling factor between the display pixel plane and camera image plane was found to be,

$$S_x = S_y = 0.7946 \tag{4.7}$$

This results in the following equations, which now gives the full definition for the transformation h from the display pixel plane to the camera image plane,

$$h(x_j^d, y_j^d) = \begin{bmatrix} x_j^i \\ y_j^i \end{bmatrix} \tag{4.8}$$

where

$$x_j^i = 0.7946\left(-x_j^d + 360\right) \tag{4.9a}$$

$$y_j^i = 0.7946\left(y_j^d - 640\right) \tag{4.9b}$$

4.3.5 Relative POSE Estimation Initialization

The relative POSE estimation to determine the six POSE parameters $W = [X, Y, Z, \emptyset, \theta, \psi]^T$ is based on the definitions and assumptions stated in Chapter 3. However, there are various elements within the EKF that are specific to the physical system and these require further discussion; specifically the initial conditions for the initial system state $\hat{x}_{1,1}$ and the initial estimate covariance $P_{1,1}$ along with values for the measurement noise covariance matrix R_k and the input dynamic disturbance noise covariance matrix Q_k.

There are various methods as noted in Chapter 3 that can improve filter initialization. These approaches were not pursued in this design, as the focus was on overall technical feasibility of the proposed system. Adding in a filter initialization element such as bootstrapping could be done by adjusting the initialization block without impact to the rest of the design and is something to be considered for future enhancements. For the case of this design, initial insight, experimentation, and simulation were used to generate a suitable set of values for these parameters. The initial state estimate $\hat{x}_{1,1}$ of the relative POSE parameters was set based on the assumption that the target object was located directly in front of the device with a small change in yaw. After some experimentation and trials, the initial state estimate $\hat{x}_{1,1}$ was set for the model as follows,

$$\hat{x}_{1,1} = [0, 0, 0.5, 0,0,0.3,0,0,0,0,0,0]^T \quad (4.10)$$

where the first three elements of initial state estimate $\hat{x}_{1,1}$ $X, Y,$ and Z are dimensioned in meters, the second three elements $\emptyset, \theta,$ and ψ are in radians, the next three elements $\dot{X}, \dot{Y},$ and $\dot{Z}$ are in meters/second, and the final three elements $\dot{\emptyset}, \dot{\theta},$ and $\dot{\psi}$ are in radians/second.

For the purposes of this work, it is assumed that the video see-through mobile AR application will be initialized to be in the neighbourhood of our initial state estimate. Therefore, our confidence in the initial state estimate is higher and as a result, the initial estimate covariance $P_{1,1}$ was therefore set to be low. Trials were conducted and the initial estimate covariance was set as follows,

$$P_{1,1} = 10^{-2}[I_{12x12}] \quad (4.11)$$

The disturbance covariance matrix Q_k encapsulates the error in the process model for each of the twelve parameters in the system state vector x_k. For the position elements of the disturbance covariance matrix, it was initially assumed that 95% of the time, i.e. 2σ, there would be a position error in the model of 0.5cm for the *X*, *Y*, and *Z* directions and 2° for the three Euler angles in one time step (20Hz). The position-based entries of the disturbance covariance matrix Q_k can then be determined by solving for σ^2. Similarly for the first derivative or velocity elements, it was assumed that 95% of the time, the velocity error in the model was 0.25cm/s for the *X*, *Y*, and *Z* directions and 1°/s for the three Euler angles in one time step (20Hz). As with above the velocity-based entries of the disturbance covariance matrix Q_k can be determined by solving for σ^2.

The measurement noise covariance matrix R_k represents the covariance of the measurement error. It was initially assumed that 95% of the time, i.e. 2σ, the measurement error due to noise will be within 4 display pixels. Similar to the disturbance covariance matrix, the entries of the measurement noise covariance matrix can be determined by solving for σ^2.

Initial simulations and physical experiments were conducted. In these experiments, controlled motions were applied and were used to optimize the values for both the disturbance covariance matrix Q_k and the measurement noise covariance matrix R_k, as detailed in equations 4.12 and 4.13. A further examination of

the effects of changing these parameters and suitability of the parameter values is detailed further in Chapter 5.

$$Q_k = \begin{bmatrix} 2.25^{-5} I_{3x3} & 0_{3x3} & 0_{3x3} & 0_{3x3} \\ 0_{3x3} & 0.25\frac{\pi}{180} I_{3x3} & 0_{3x3} & 0_{3x3} \\ 0_{3x3} & 0_{3x3} & 2.44^{-2} I_{3x3} & 0_{3x3} \\ 0_{3x3} & 0_{3x3} & 0_{3x3} & \frac{\pi}{180} I_{3x3} \end{bmatrix} \tag{4.12}$$

$$R_k = 4[I_{8x8}] \tag{4.13}$$

4.3.6 Virtual Object Image Generation

In order to generate and render the video see-through virtual object in the display pixel plane, the position of the each of the virtual object's $M \geq 1$ feature points needs to be calculated with respect to the origin of the target object frame origin O_O. This operation can be done using the following three equations previously defined in Chapter 3 as follows,

$$P^C_{AR\,j} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + R^C_O(\emptyset, \theta, \psi) P^O_{AR\,j} \tag{4.14}$$

$$x^i_{AR\,j} = \frac{-f x^c_{AR_j}}{P_X z^c_{AR_j}} \tag{4.15}$$

$$y^i_{AR\,j} = \frac{-f y^c_{AR_j}}{P_Y z^c_{AR_j}} \tag{4.16}$$

A fixed offset for the virtual object from the object frame's origin was considered. However extension to multiple target objects was also considered and field of view or simply fitting all of the target objects and their respective virtual objects into the display of the device was felt to be a constraint. Therefore, the P^3_O point coinciding with the object frame origin O_O was selected so that the virtual object would be overlaid

on top of the target object, i.e. credit card. In order to render the virtual object onto the target object, OpenGL [55] libraries were used. As OpenGL is an open source library with a focus on rendering 2D and 3D graphics, it was selected for the approach of implementing the video see-through AR object.

In this project, the video see-through AR image is required to be overlaid on top of the camera video viewfinder image stream such that it remains in the fixed position and orientation relative to the target object. However due to implementation specifics of the BlackBerry Dev Alpha A software architecture (no writeable buffer exists from the camera service), a second image buffer was required to be created. This implementation approach of creating a second buffer allowed for the Image Processing and relative POSE estimation thread to be run separately from the AR Rendering Thread, which meant that the video see-through AR rendering steps did not count against the time budget of 33ms to 50ms since they were completed in parallel. Communication between these two process threads was handled via messages, which will aid in the execution efficiency. For this thesis, an image of a skeleton was selected as virtual object to be rendered and an image is shown in Figure 4-4 below.

Figure 4-4: Depiction of the skeleton frame AR image rendered onto the credit card object.

4.3.7 Effective Workspace

From a physical standpoint, any camera's field of view will be finite and will typically be conically shaped as illustrated in Figure 4-5 below. Therefore, it is expected that the relative *X, Y,* and *Z* POSE parameters and the effective workspace would be constrained to a finite distance. Additionally, for both the relative pitch angle θ about Y and for the relative yaw angle ψ about X, the camera's conical field of view will constrain the Effective Workspace. The constraints on the effective workspace for the relative POSE parameters will be unique for each physical system due to the construction, physical properties such as the lens, etc. and can be determined experimentally for each physical system.

However, for the purposes of this thesis, it is assumed that the camera used in the device has the ability to form images on the camera image plane, which are an arbitrarily large distance away, i.e. infinity focus. Additionally it is assumed that the camera has the ability to form images on the image plane for relative pitch θ and yaw ψ angles that fall along the camera frame Y_C and X_C axes respectively. Therefore, the effective workspace for this physical system can be defined as per Equations 3.42 through 3.47, which are restated below for completeness.

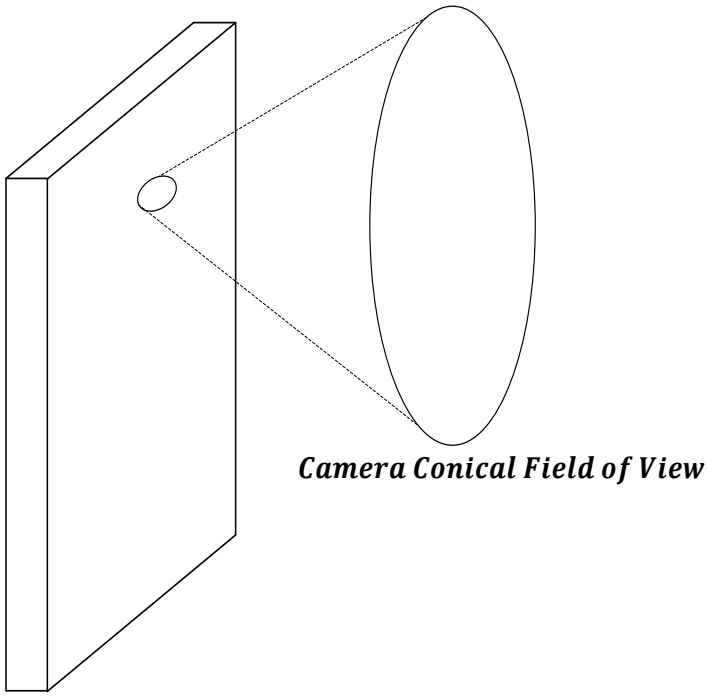

Figure 4-5: Device camera field of view will be practically constrained to a conical field of view.

$$-\infty < X < \infty \tag{3.42}$$

$$-\infty < Y < \infty \tag{3.43}$$

$$0 < Z < \infty \tag{3.44}$$

$$-\pi < \emptyset \leq \pi \tag{3.45}$$

$$-\frac{\pi}{2} < \theta < \frac{\pi}{2} \tag{3.46}$$

$$-\frac{\pi}{2} < \psi < \frac{\pi}{2} \tag{3.47}$$

As the focus of this work was directed towards proof of concept to implementing a video see-through mobile AR application using position based visual POSE estimation methods, maximizing the effective workspace was not an objective. Therefore the assumptions used above are deemed acceptable noting that the effective workspace area in practice is smaller than what is defined.

4.3.8 System Design Model

Using the knowledge and our model of the physical system along with the EKF, the physical system based on the BlackBerry Dev Alpha A device can be mathematically described as a linearized discrete time state space with the following definition,

$$x_k = A_k x_{k-1} + \eta_k \tag{4.17}$$

$$y_k = C_k x_k + \nu_k \tag{4.18}$$

where A_k is the process model described through a constant velocity model that is updated for each time step based on the time increment, C_k is the measurement Jacobian that is linearized about the current state estimate $\hat{x}_{k,k-1}$ for each time step, and η_k, ν_k represents the disturbance and measurement noise vectors respectively.

Now that the general model has been extended to this physical system, specifically with considerations to the varying sample rate and the display to camera transformation, a modified algorithm for the video see-through AR mobile application for the BlackBerry Dev Alpha A can be written. The updated system algorithm is provided below and can be graphically represented using the block diagram in Figure 4-6 below with the changes from the generalized model in **boldface** text**.**

1. Set up initial conditions including $\hat{x}_{1,1}, Q_k, R_k, A_k$, and $P_{1,1}$ and set the time step counter $k=1$
2. Locate target object initial position on camera image plane through image processing detection techniques, noting that the target object located will be compared against CAD model object library.
3. Start Recursive Estimation Loop
    a. Increment step counter, i.e. $k = k + 1$
    b. Determine sample rate time increment $\boldsymbol{\delta_k}$ between the $\boldsymbol{k^{th}}$ and $\boldsymbol{k\text{-}1^{th}}$ time steps
    c. Capture image on camera sensor and through image processing techniques, locate on the display pixel plane each of the $\boldsymbol{N = 4}$ feature point pairs of the known target object, i.e. $\boldsymbol{y_k^j = [x_j^d\ y_j^d],\ for\ j \in \{1,2,3,4\}}$.
    d. Using the function $\boldsymbol{h}$, transform measured feature point coordinates from display pixel plane to camera image plane and form the modified measured output $\boldsymbol{\tilde{y}_k}$ as follows,

$$\tilde{y}_k^j = h(x_j^d, y_j^d) = \begin{bmatrix} x_j^i \\ y_j^i \end{bmatrix} for\ j \in \{1,2,3,4\} \tag{4.19a}$$

$$\tilde{y}_k = \begin{bmatrix} x_1^i & y_1^i & \cdots & x_4^i & y_4^i \end{bmatrix}^T \tag{4.19b}$$

    e. **Update the Process Model $\boldsymbol{A_k}$ for each time step**,

$$A_k = \begin{bmatrix} I_{6x6} & \delta_k I_{6x6} \\ 0_{6x6} & I_{6x6} \end{bmatrix} \tag{4.20}$$

f. Extended Kalman Filter Prediction Step

i. Carry out prediction step of EKF to determine new state estimate for $\hat{x}_{k,k-1}$ and the estimate covariance $P_{k,k-1}$ based on the state estimate and estimate covariance of the prior time step,

$$\hat{x}_{k,k-1} = A_k \hat{x}_{k-1,k-1} \tag{4.21}$$

$$P_{k,k-1} = A_k P_{k-1,k-1} {A_k}^T + Q_k \tag{4.22}$$

g. Extended Kalman Filter Measurement Update Step

i. Linearization Step for each time step to find the measurement Jacobian C_k and predict the output $\hat{y}_k$ based on the state estimate of the current time step $\hat{x}_{k,k-1}$,

$$C_k = \frac{\partial G(x)}{\partial X}\Big|_{x=\hat{x}_{k,k-1}} \tag{4.23}$$

$$\hat{y}_k = C_k \hat{x}_{k,k-1} \tag{4.24}$$

ii. Determine the Kalman Gain K,

$$K = P_{k,k-1}\, C_k^T \left(R_k + C_k P_{k,k-1} C_k^T\right)^{-1} \tag{4.25}$$

iii. Estimate Update step to refine the state estimate $\hat{x}_{k,k}$ and the estimate covariance $P_{k,k}$ noting that the modified measured output $\tilde{y}_k$ is included in the calculation of the *innovation*,

$$\hat{x}_{k,k} = \hat{x}_{k,k-1} + K(\tilde{y}_k - \hat{y}_k) \quad (4.26)$$

$$P_{k,k} = P_{k,k-1} - KC_kP_{k,k-1} \quad (4.27)$$

h. Update the camera image plane feature point locations of virtual object image for each of the M points of the virtual object

i. Render new virtual object on display based on updated virtual object feature point locations

4. End Recursive Estimation Loop

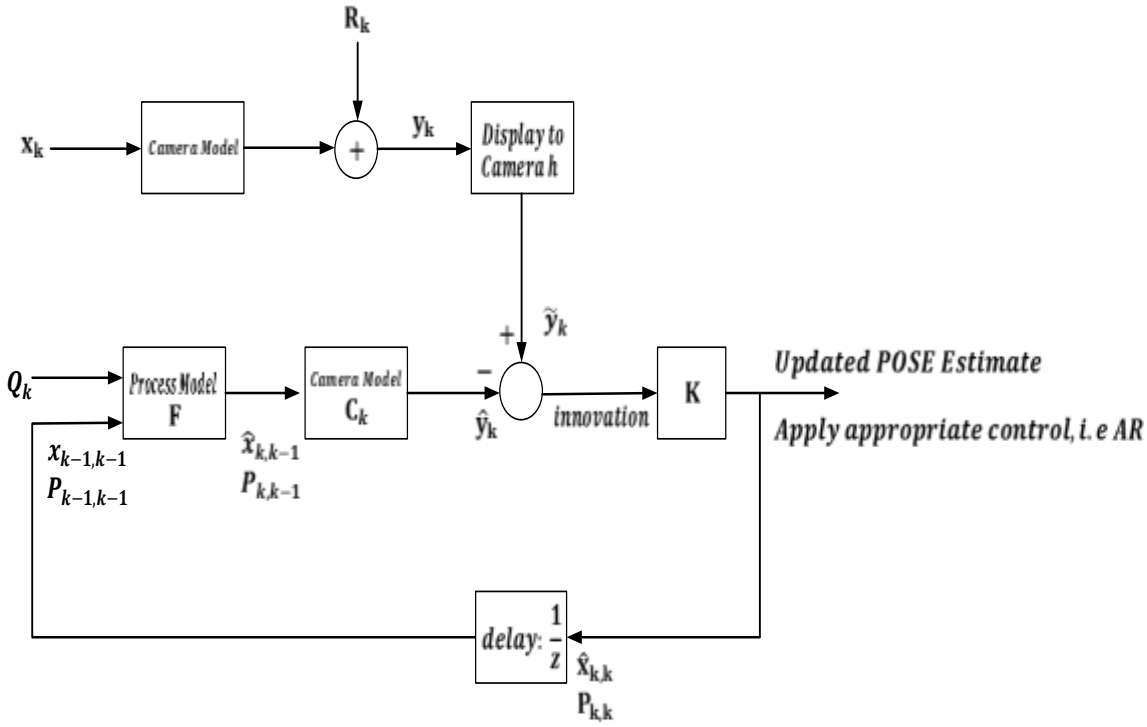

Figure 4-6: System block diagram detailing the implementation of a video see-through AR system on BlackBerry Dev Alpha A device.

4.3.9 Implementation Architecture

The design was implemented in C/C++ using the BlackBerry 10 Native Software Development kit. Version control was handled using the open source Tortoise SVN repository [56]. The required OpenCV and OpenGL libraries were also included for the implementation. The architecture of the implementation [57] is based on multi-threaded architecture and is shown in Figure 4-7 below.

Trials were completed on both the image processing and POSE estimation portions of the implementation and the image processing module was found to take anywhere between 12ms to 25ms for each time step. The relative POSE estimation module was consistently benchmarked to take approximately 1ms. As a separate rendering thread/module is used for the video see-through virtual object, it was executed in parallel to the Image Processing and relative POSE estimation portions of the algorithm and constantly renders the latest virtual object image based on the current relative POSE data. As a result, the image rendering portion of the system design does not end up impacting the time budget of 50ms. Therefore it is concluded that that the system can operate without a sense of lag for one known target object.

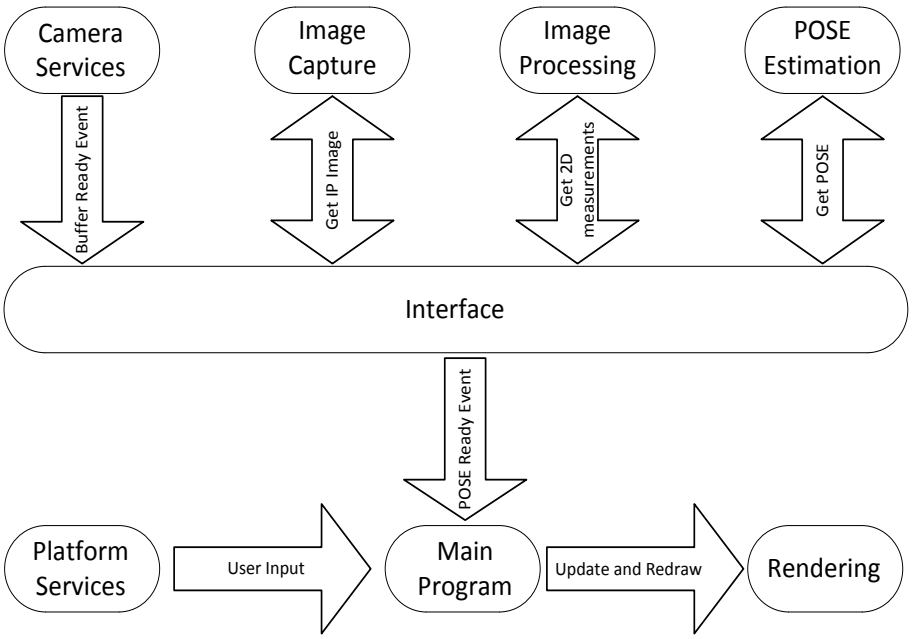

Figure 4-7: Software architecture implementation for video see-through mobile AR application.

4.4 Design Extension and Scalability

4.4.1 Multiple Objects

Extending the design to operate on multiple known objects requires only minor changes to the overall implementation. Since the software architecture is based on multiple threads, the framework exists to simply launch additional state machines that manage the object tracking, relative POSE estimation, and AR rendering. There are two primary challenges for this particular approach. The first challenge is to ensure that the state machines track unique objects and the second challenge is with the increased processing time and its impact to the smoothness in the AR rendering. To resolve both challenges, the detection portion of the image processing algorithm was modified to scan the entire region of the display for known objects and secondly to include comparative logic between all state machines to ensure that each state machine is tracking a unique object.

Additionally the POSE estimation algorithm does not require any changes. However it is noted that different process models A_k could be used for each target object to reflect that there would likely be different relative motion dynamics for each target object. The procedure outlined above in Chapter 4.3.8 is identical except for the following notable cases.

1. Initialization, real-time tracking via the EKF, and AR rendering is completed in parallel for each $1 \ldots m$ target objects. It is noted that a different process model A_k can be employed.
2. Unique object identification techniques need to be employed.

4.4.2 Multiple Known Object Models

In order to extend the video see-through mobile AR design to multiple and unique object models, it is a matter of adjusting the image processing module to contain a 3D CAD library of different object models. The Detection portion of the image processing module would then need to be updated accordingly to properly map the feature points located to a specific 3D object model in the library. Once the mapping to

the different object models is completed during the Detection Phase, the Tracking Phases of the image processing module would execute without any required changes. The algorithmic process flow would not require any changes.

4.4.3 Different Device Models

Finally the POSE estimation and video see-through AR framework can be extended to different mobile computing devices or alternate platforms with the assumption that the device contains a display and a opposite facing camera to the display. In order to apply the relative POSE estimation and virtual object generation algorithms to a different device, only a select few parameters need to be updated, as shown in Table 4-2 below. It is also noted that all of the above assumptions would still apply and aside from some intrinsic parameters, the system model architecture remains unchanged.

Table 4-2: Model Parameters Unique to Device Implementation – BlackBerry Dev Alpha A Specific Parameters Shown

Attribute	Variable	Device Value	Determination Method
Camera Focal Length	f	3.32 mm	A priori knowledge and/or camera calibration
Camera Inter-pixel Spacing in X	P_x	1.4µm/pixel	A priori knowledge and/or camera calibration
Camera Inter-pixel Spacing in Y	P_y	1.4µm/pixel	A priori knowledge and/or camera calibration
Total Display Pixels in X direction		720 pixels	Published parameter
Total Display Pixels in Y direction		1280 pixels	Published parameter
Display Pixel Plane X-Optical Center	OC_x	360 pixels	Published parameter
Display Pixel Plane Y-Optical Center	OC_y	640 pixels	Published parameter
Display to Camera Scaling Parameter in X, Y	$S_x,\ S_y$	0.7946	Experimentation through camera calibration routines
Covariance Matrices	Q_k, R_k	See Chapter 4.3.5	Optimized through experimentation

4.5 Verification and Validation Approaches

Given the above system implementation, a great deal of consideration was made to determine the best way to verify and validate the system was to break it down into two basic parts. The first part in the verification, as detailed in Chapter 5, was to investigate the position based visual POSE estimation method in a MATLAB based simulation environment to ensure that the algorithm functioned as intended and also to explore its robustness. The next step was to validate the overall video see-through mobile AR system implementation using the BlackBerry Dev Alpha A device. The validation, as detailed in Chapter 6 was conducted using a ground truth measurement system that could independently validate the effectiveness and accuracy of the system.

Chapter 5 SYSTEM VERIFICATION

In order to verify the system and evaluate the position based visual POSE estimation mathematical model proposed, simulations were completed in MATLAB assuming a virtual work space defined as the area of relative motion between the device and the target object. Controlled inputs and conditions were used to assess the suitability of this model to examine different motion models, the effect of noise, and optimization of the parameters. In this chapter, the verification approach is presented along with the test procedure used. Various simulation results are presented including the effect of changing initial conditions, motion models, and parameter selection. Finally conclusions are presented to discuss the results and the suitability of the proposed design.

5.1 System Verification Approach

In order to verify and evaluate the effectiveness of the position based visual POSE estimation based video see-through AR system, known and controlled inputs were used in a MATLAB simulation environment to assess the effectiveness of the system. As noted in Chapter 4, the video see-through AR system with a relative motion between a mobile computing device and a target object can be mathematically defined as a linearized discrete state space model as follows,

$$x_k = A_k x_{k-1} + \eta_k \tag{4.17}$$

$$y_k = C_k x_k + \nu_k \tag{4.18}$$

In order to complete an initial verification of the position based visual POSE estimation based system, simulations will be used to create or construct a known motion model over time, i.e. x_k^{actual}, run the relative POSE estimation algorithms detailed in Chapter 4, and complete a comparison between the system's predicted system state $\hat{x}_k$ and the truth data x_k^{actual}. Simulations will then be used to show the robustness of this mathematical model to different physical motion models such as constant velocity, constant acceleration, and arbitrary motions. Effects of noise, sampling rates, and varying initial conditions such as

the disturbance covariance matrix Q_k and the measurement noise covariance matrix R_k will also be examined.

5.2 System Verification Setup

5.2.1 Assumptions

The following assumptions were made when completing the simulation analysis with a known or constructed relative motion model x_k^{actual} between the device and the target object:

a. It is assumed that the mathematical system model presented in Chapter 4.4.7 is applicable including all of the previously stated assumptions.

b. The target object was selected to be the credit card model with $N = 4$ feature points with the object definition as provided per Figure 4-1 and Table 4-1

c. It is assumed that the relative motion model x_k^{actual} between the device and the target object is known precisely, i.e. constructed via simulation. The measurement error ν_k is also known and is injected into the system.

d. It is assumed that the sampling rate of the system is fixed, resulting in a static process model A_k

5.2.2 Test Procedure and Initial Conditions

The general approach in the verification is to set both the motion model x_k^{actual} and the measurement noise ν_k so that the measured output y_k can be calculated. The position based visual POSE estimation algorithms will then be run and both the state $\hat{x}_k$ and the output $\hat{y}_k$we will be estimated. The percent error in the state estimate will then be calculated for each time step to assess the accuracy of the mathematical system model. The following test procedure was used in the verification analysis:

1. Construct motion model or relative POSE vector x_k^{actual} based on test case, e.g. constant velocity, constant acceleration, arbitrary, including the initial actual relative POSE of the target object. The initial position was chosen to be relatively far from the initial state estimate $\hat{x}_{1,1}$ to be the following noting that the pertinent dimensions of the Cartesian elements are in meters and meters/second and the Euler angle elements are in radians and radians/second.

$$x_1^{actual} = [1, 1, 1, 0.1, 0.1, 0.1, 0, 0, 0, 0, 0, 0]^T \quad (5.1)$$

2. Set up initial conditions including $\hat{x}_{1,1}, Q_k, R_k, A, \delta_k$, and $P_{1,1}$ and set the time step counter $k=1$

   a. Where δ_k is the sample period used in the relative POSE estimation. As the system minimum performance requirement is 20Hz, this value will be used in the verification analysis, and this gives us the following definition of δ_k,

$$\delta_k = 50ms \quad (5.2)$$

   b. Where $\hat{x}_{1,1}$ is the initial state estimate of the relative POSE parameters and is set to the following based on the work completed in Chapter 4:

$$\hat{x}_{1,1} = [0, 0, 0.5, 0, 0, 0.3, 0, 0, 0, 0, 0, 0]^T \quad (4.10)$$

   c. Where $P_{1,1}$ is an initial estimate covariance value and is defined as per Equation 4.11.

   d. Where Q_k, R_k are the disturbance and measurement noise covariance matrices as defined by Equations 4.11 and 4.12.

   e. Where the process model A is defined assuming a fixed sample rate as per Equation 3.27.

3. Start Recursive Estimation Loop

   a. Increment step counter, i.e. $k = k + 1$

b. Based on the constructed motion model or relative POSE vector x_k^{actual}, calculate the actual output y_k assuming a known or injected level of measurement noise ν_k to locate each of the N feature point pairs of known target object, i.e. (x_j^i, y_j^i) as follows,

$$x_{j\,k}^{i} = S_x * \frac{-f\left(X + C_\emptyset C_\theta x_j^O + (C_\emptyset S_\theta S_\psi - S_\emptyset C_\psi) y_j^O + (C_\emptyset S_\theta C_\psi + S_\emptyset S_\psi) z_j^O\right)}{P_X\left(Z - S_\theta x_j^O + C_\emptyset S_\psi y_j^O + C_\theta C_\psi z_j^O\right)} + \nu_k \tag{5.3}$$

$$y_{j\,k}^{i} = S_y * \frac{-f\left(Y + S_\emptyset C_\theta x_j^O + (S_\emptyset S_\theta S_\psi + C_\emptyset C_\psi) y_j^O + (C_\emptyset S_\theta S_\psi - S_\emptyset C_\psi) z_j^O\right)}{P_Y\left(Z - S_\theta x_j^O + C_\emptyset S_\psi y_j^O + C_\theta C_\psi z_j^O\right)} + \nu_k \tag{5.4}$$

where ν_k is known and represents the measurement error in pixels on the camera image plane and where,

$$\tilde{y}_k = \begin{bmatrix} x_{1\,k}^i & y_{1\,k}^i & \cdots & x_{4\,k}^i & y_{4\,k}^i \end{bmatrix}^T \; for\; j \in \{1,2,3,4\} \tag{5.5}$$

c. Extended Kalman Filter Prediction Step

   i. Carry out prediction step of EKF to determine new state estimate for $\hat{x}_{k,k-1}$ and the estimate covariance $P_{k,k-1}$ based on the state estimate and estimate covariance of the prior time step,

d. Extended Kalman Filter Measurement Update Step

   i. Linearization Step for each time step to find the measurement Jacobian C_k and predict the output $\hat{y}_k$ based on the state estimate of the current time step $\hat{x}_{k,k-1}$,

   ii. Determine the Kalman Gain K,

   iii. Estimate Update step to refine the state estimate $\hat{x}_{k,k}$ and the estimate covariance $P_{k,k}$,

4. End Recursive Estimation Loop

5. Calculate the percent error between the constructed motion model x_k^{actual} and the predicted state estimates $\hat{x}_k$,

$$x\_err_k = \frac{|x_k^{actual} - \hat{x}_{k,k}|}{x_k^{actual}} * 100 \quad (5.5)$$

6. Plot Results

5.3 Initial Model Verification Using A Constant Velocity Motion Model

5.3.1 Constant Velocity Model

Initial model simulations using a constant velocity (CV) relative motion to represent the actual motion was examined first. This was done to ensure initial confidence in our system implementation. For the purposes of the simulation, the CV model defined in Equation 3.25 can be simplified further, as the time increment δ_k and velocity $\dot{x}_k$ are constant and the disturbance noise η_k is set to 0. This provides us with the following expression for the constructed motion model x_k^{actual} as follows,

$$x_k^{actual} = x_{k-1}^{actual} + \delta_k v_k \quad (5.6)$$

In selecting a particular motion model, there were several considerations and observations. First of all, care was taken for all simulations to ensure that the relative motion between the device and the target object stayed within the constraints of the assumed and previously defined effective workspace of the system model, as per Equations 3.42 through 3.47, and that motion occurred on all six relative POSE parameters.

During simulations, it was observed that the relative error for the Euler angle relative POSE parameters went very high for motions that crossed the axis, i.e. angle equal to zero. This follows naturally from our definition of the error in Equation 5.5, due to a divide by zero error, and will be discussed in Chapter 5.3.3.

One additional consideration was the measurement noise vector v_k, which is taken in relation to the camera image measurement. For the purposes of the initial model simulations, the measurement noise v_k was defined in the MATLAB simulation environment as follows,

$$v_k = s * \text{randn}( ) \tag{5.7}$$

which represents a Gaussian distributed noise signal where the standard deviation is $s = \sigma$. For the case of $s = 1$, the standard deviation is $\sigma = 1$. This selection means that approximately 95% of the noise values lie between ± 2 camera image plane pixels. In other words, this means that a measurement error on the camera image plane of approximately ± 2 pixels for the case in which the measurement noise has a standard deviation $\sigma = 1$ is expected. Given the display to camera transformation h, which is approximately equal to 0.8, this translates to noise or error of ± 2.5 display plane pixels. This translates to roughly a 0.7% measurement error in the *X* direction and a 0.4% error in the *Y* directions given the 720 by 1280 pixel dimensions on the display.

Therefore for the purposes of discussion, the following constant velocity model $\dot{x}_k$ in Equation 5.8 will be examined as the baseline motion model for all comparative analysis unless otherwise specified. It is also noted that the measurement noise vector v_k with a standard deviation of $\sigma = 1$ will be used.

$$\dot{x}_k = \begin{bmatrix} 0.1 & -0.01 & 0.02 & \frac{5\pi}{180} & \frac{3\pi}{180} & \frac{1\pi}{180} \end{bmatrix}^T \tag{5.8}$$

5.3.2 Simulation Test Results

Using the initial conditions x_1^{actual} defined in Equation 5.1 and the actual motion model described in Equation 5.8 to form the relative POSE vector x_k^{actual}, the system model was simulated for a duration of 25 seconds and the results are shown in Figures 5-1 and 5-2 below.

As can be seen from the two figures below, the predicted relative POSE estimates using a CV motion tracks the actual relative POSE motion quite well with minimal error. Some high initial levels of overshoot were observed. However the levels were not unreasonable and is to be expected as the EKF algorithm initially converges. System tracking to within 10% of the actual value by 20 time steps (or 1s), which indicates a fast level of convergence. Additionally, choice of initial conditions such as x_1^{actual} and its

difference from the initial state estimate $\hat{x}_{1,1}$ will impact how quickly the model converges. Therefore an initial conclusion can be made that the system model functions as expected. However the system's robustness to other parameters will be explored further below.

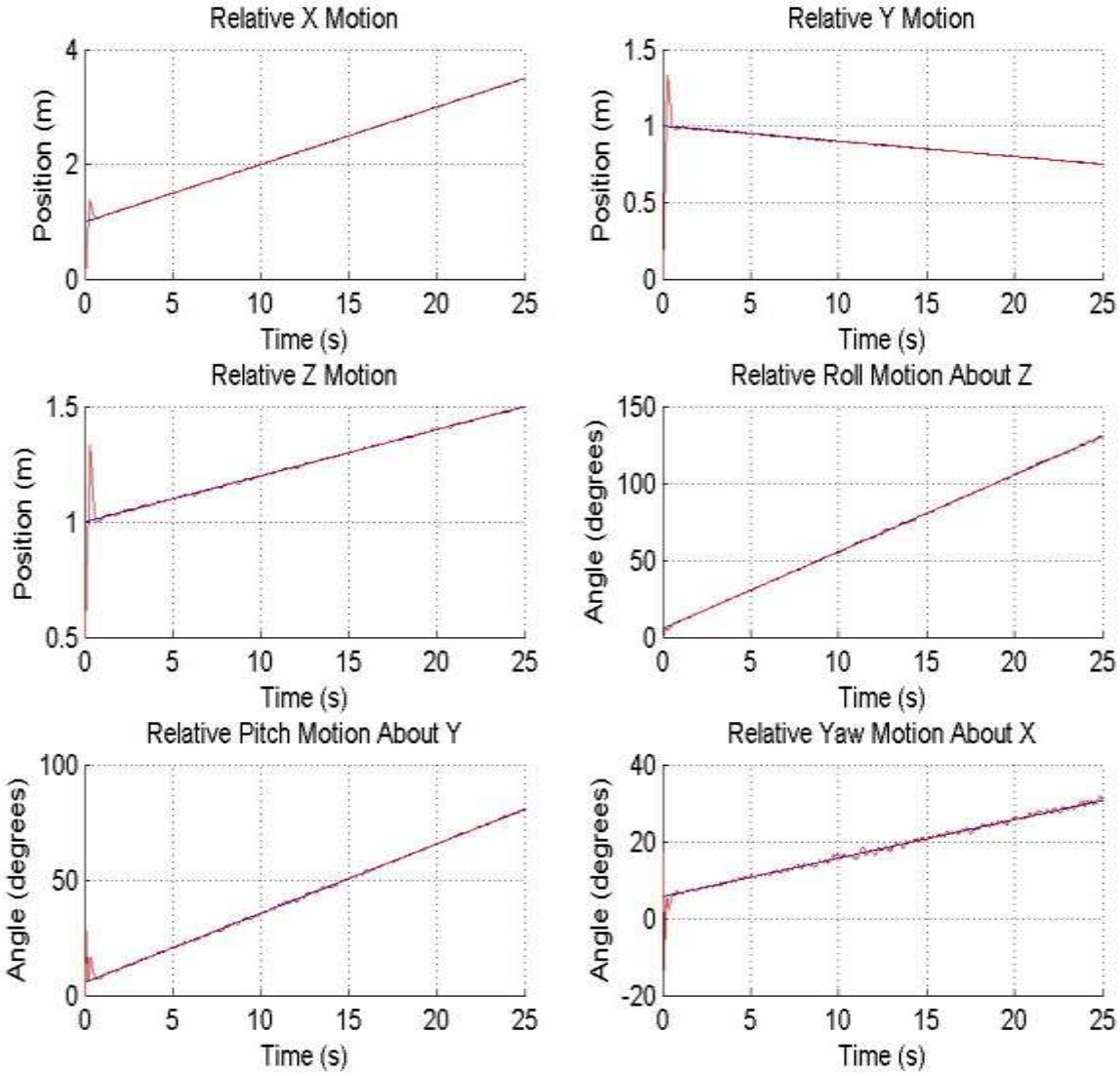

Figure 5-1: Depiction of the ideal (blue) and predicted relative (red) motion on all six axes for the CV case with measurement noise vector v_k with a standard deviation of $\sigma = 1$.

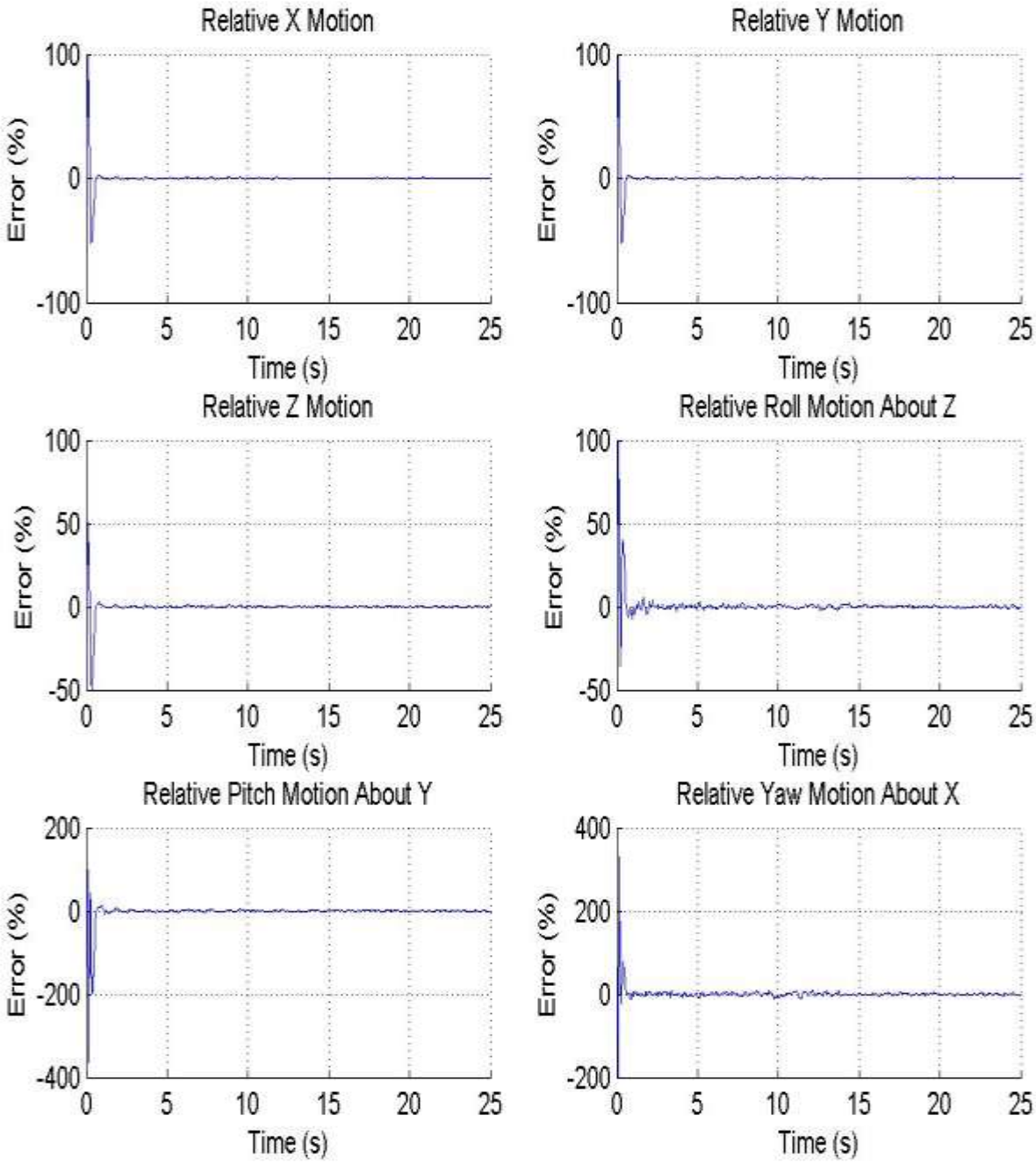

Figure 5-2: Depiction of the relative percent error on all six axes for the CV case with measurement noise vector ν_k with a standard deviation of $\sigma = 1$.

5.3.3 Effect of Changing Initial Position

As noted above, the actual initial position x_1^{actual} and its relative distance to the initial state estimate $\hat{x}_{1,1}$can impact the performance. Using the CV motion model described in Equation 5.8 and the initial conditions detailed in Chapter 5.2.2 along with the measurement noise v_k with a standard deviation of $\sigma = 1$, two cases for different values of x_1^{actual} will be detailed below.

The first case will detail the impact of having the initial state estimate $\hat{x}_{1,1}$ be very close in proximity to the actual initial state x_1^{actual}. The following actual initial state was also selected to illustrate the effects of a zero value for the relative *Y* parameter and the results are graphically detailed in Figures 5-3 and 5-4 below,

$$x_1^{actual} = [0.2, 0.02, 0.55, 0.1, 0.1, 0.35, 0, 0, 0, 0, 0, 0]^T \quad (5.9)$$

As can be seen from the two figures below, it is observed that having the initial state estimate $\hat{x}_{1,1}$ very close proximity to the actual initial state x_1^{actual} results in a lower error, as is to be expected. It is also noted that a high relative percent error component exists when a zero value occurred at 2.05s or time step 41.

Examining the simulation data for this relative *Y* zero value point, it is seen that the estimate for the relative *Y* at the zero value point is $2.2*10^{-5}$ m versus the actual result of $-1.4*10^{-17}$ m. Even though both values are small and relatively close to each other, the relative error calculation becomes skewed due to the effective divide by zero for this time step caused by the definition of Equation 5.5. As a result of possible divide by zero cases for the relative error calculation, all motion models were chosen to avoid zero values for the purposes of presentation.

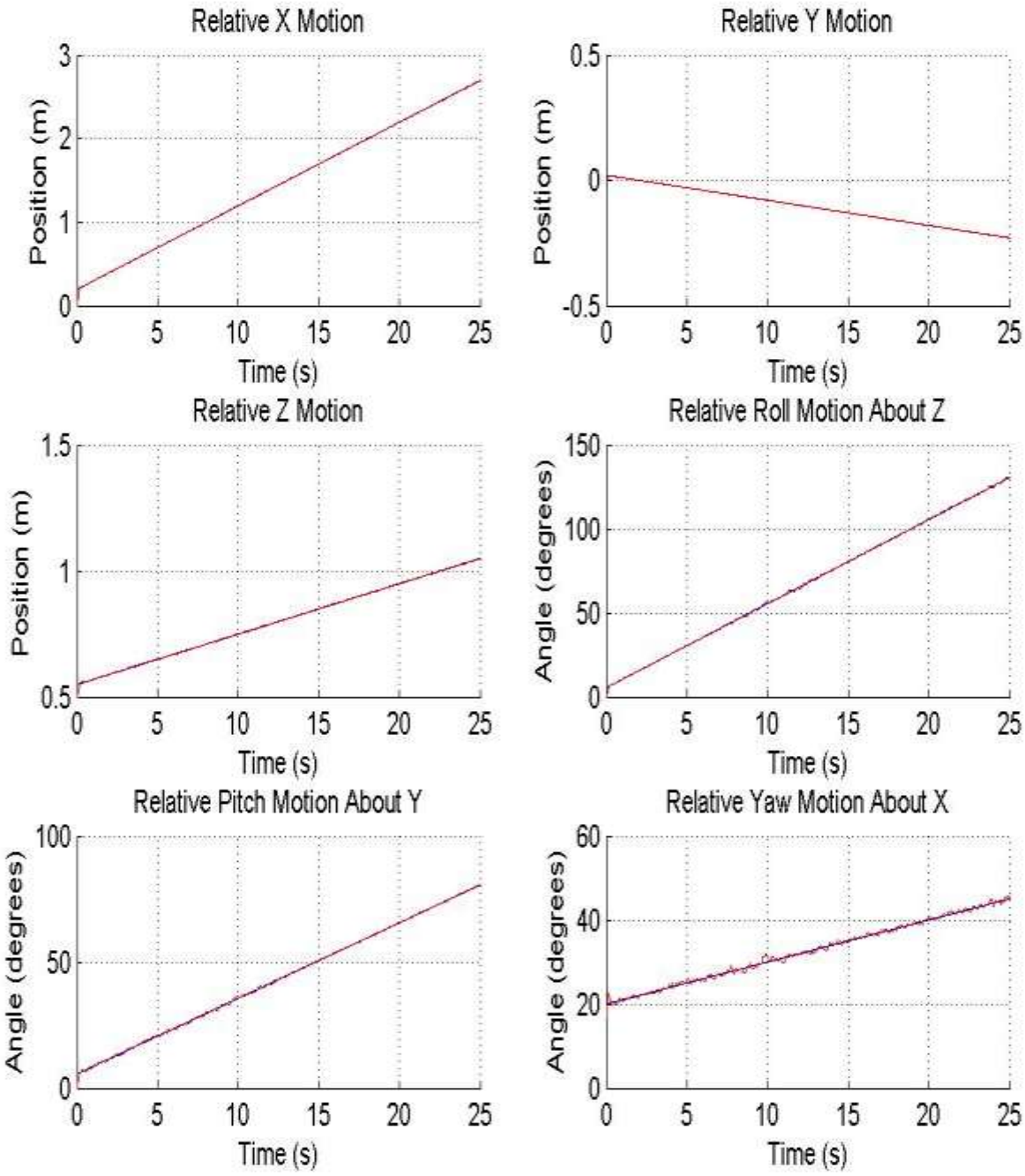

Figure 5-3: Depiction of the ideal (blue) and predicted relative (red) motion on all six axes for the CV case with the relative *Y* parameter containing a zero value point.

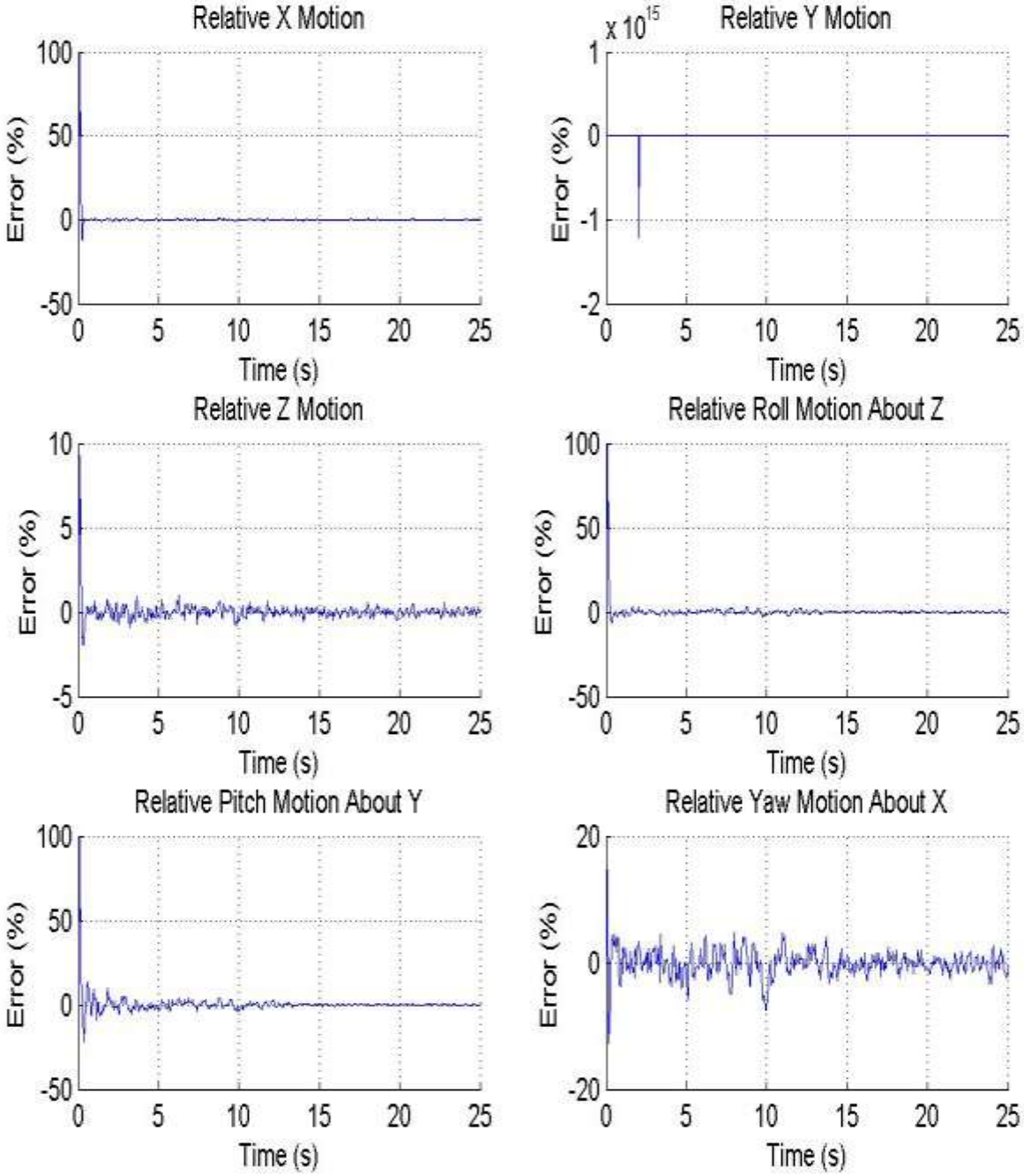

Figure 5-4: Illustration of how error can be artificially skewed due to zero value points and the definition of the relative error calculation.

The final case examined was to illustrate the effect of having a large relative distance and orientation between the actual initial state x_1^{actual} and initial state estimate $\hat{x}_{1,1}$. The selected initial state x_1^{actual} used in the verification is detailed in Equation 5.10 and the resulting plots are shown in Figures 5-5 and 5-6 below,

$$x_1^{actual} = [10, 10, 10, 1, 0.3, 0.3, 0, 0, 0, 0, 0, 0]^T \tag{5.10}$$

As can be seen from the two figures below it is observed that the error is larger and also ±10% convergence for the system takes longer in the case for a large gap between the initial actual state x_1^{actual} and initial state estimate $\hat{x}_{1,1}$. The initial overshoot is also quite high, but this is also to be expected. All of these results are to be expected. While the system does not settle or track to within ±10% of the actual value due to a higher error in the relative yaw ψ parameter, the system looks to converge or properly track around 10-11 seconds, which is a large increase from our prior simulation results shown above. The relative pitch θ parameter also has a large average error due to the initial overshoot. Once the relative pitch θ parameter settles or converges the error is smaller relative to the relative yaw ψ parameter.

Table 5-1 presents the error and setting time observations from these simulations. It is noted that these results are only presented to illustrate the trend of how initial position can impact both the error and convergence time. The results are variable based on many parameters such as noise, initial conditions, motion models, etc. Additionally, the outlier data point for the relative *Y* error zero axis crossing was removed as it artificially skewed the data due to how the error calculation is defined. As is expected, a close initial state estimate to the actual initial position will result in lower error and a large difference will result in higher error. Convergence of the system will also be affected by the difference between the initial state estimate and the actual initial position. It is noted that a faster convergence time to within ±10% would have occurred if there was no zero axis crossing event for the relative *Y* parameter.

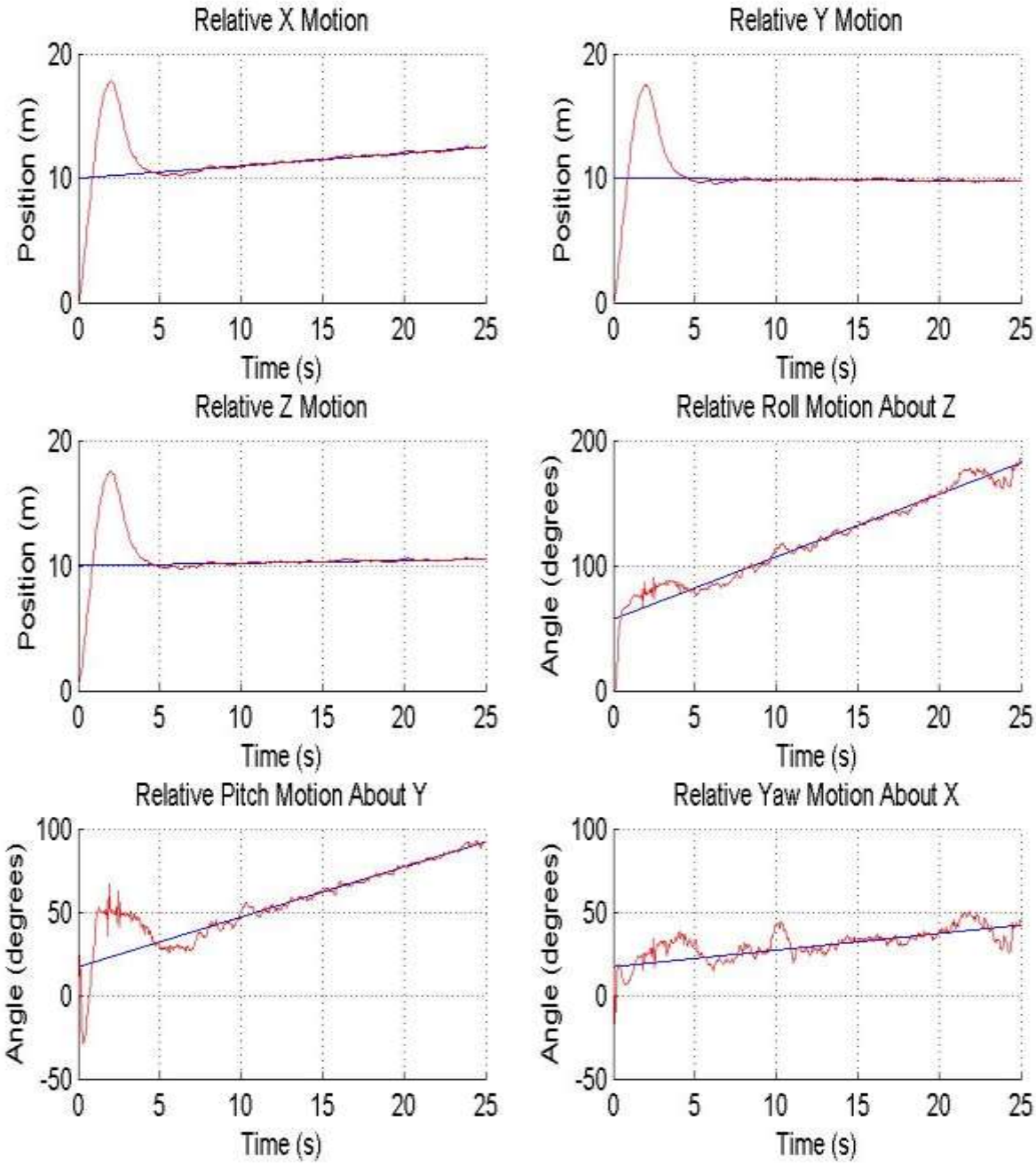

Figure 5-5: Illustration of the increased convergence time and overshoot caused by large difference between initial position x_1^{actual} (blue) and initial state estimate $\hat{x}_{1,1}$ (red).

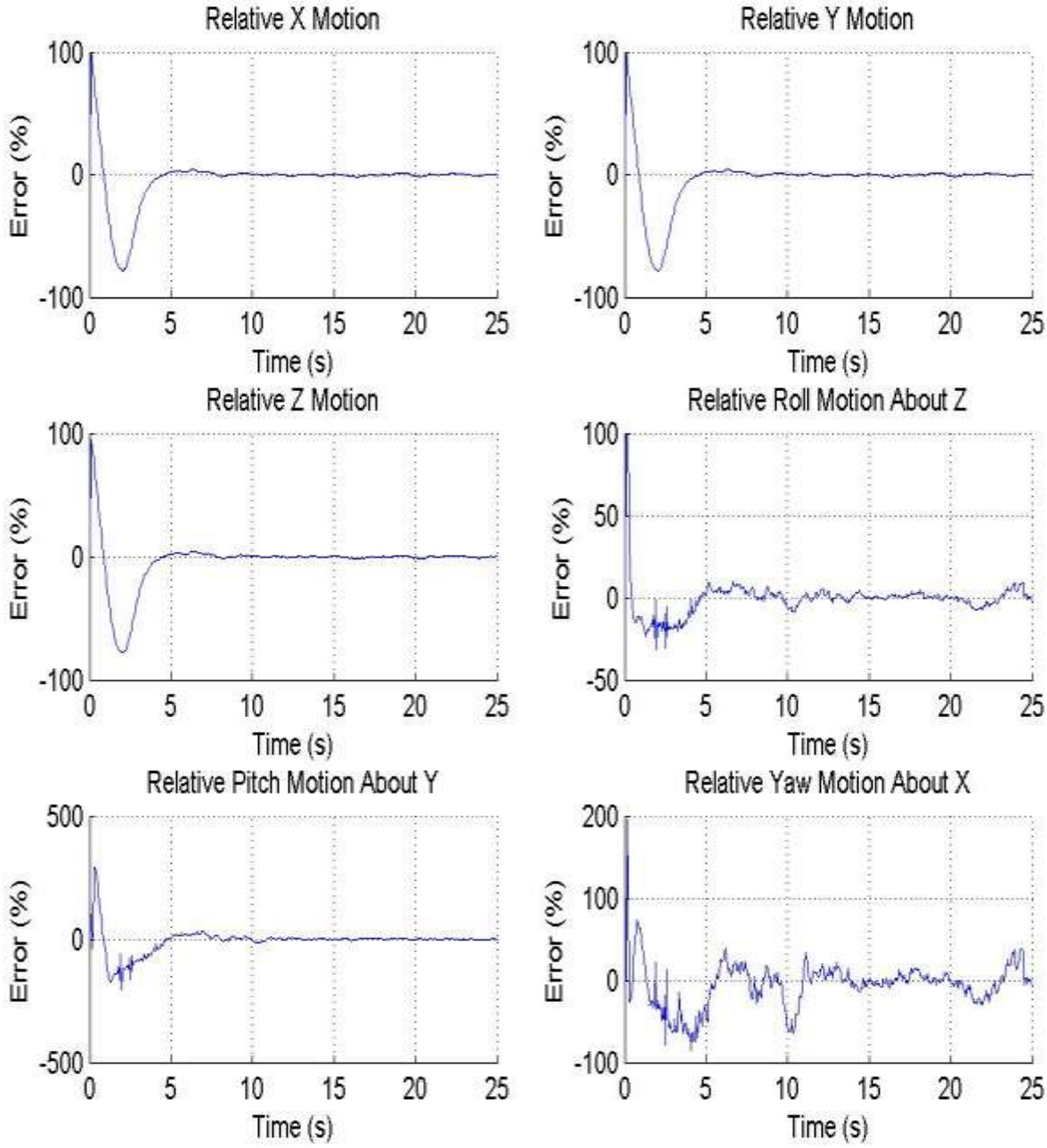

Figure 5-6: Illustration of the increased initial system error caused by large difference between initial position x_1^{actual} and the initial state estimate $\hat{x}_{1,1}$.

Table 5-1: Impact of Initial Position Estimate and its Proximity from the Actual Initial Position

	Percent Error							Convergence to within +/- 10%	
Initial Position Estimate	X	Y	Z	$\emptyset$	θ	ψ	**Average Error**	**# time steps**	**Total time (s)**
Baseline	1.04	1.06	0.84	1.44	3.66	4.19	2.04	20	1
"close"	0.53	1.49	0.25	0.94	1.44	1.5	1.02	45	2.25
"far"	7.66	7.6	7.6	5.52	22.19	19.11	11.64	All but yaw by 212	10.6

5.4 Model Verification Using Alternate Motion Models

Now that an initial verification of the system model has been completed, the impact of using alternate motion models to derive x_k^{actual} will be examined. The purpose of these alternate motion model experiments is to verify the assumption that the relative motion dynamics can be modeled as a CV model when the actual relative motion does not behave in a constant velocity fashion. Two categories of alternate motion models were considered: constant acceleration (CA) and an arbitrary motion models. Various motions were simulated and the results are detailed below for discussion.

First the CA case was examined. A generic CA motion model can be described through the following two equations,

$$\dot{\mathrm{x}}_k^{actual} = \dot{\mathrm{x}}_{k-1}^{actual} + \delta_k \ddot{x}_k \tag{5.11}$$

$$\mathrm{x}_k^{actual} = \mathrm{x}_{k-1}^{actual} + \delta_k \dot{\mathrm{x}}_{k-1}^{actual} \tag{5.12}$$

Various CA models were simulated and the following model of the acceleration $\ddot{x}_k$ is presented for discussion since it contains an acceleration factor in all six axes. As previously, the relative motion was kept to the constraints of the effective workspace detailed in Equations 3.42 through 3.47 and the initial state x_1^{actual} was defined as per Equation 5.1. It is assumed that the measurement noise vector ν_k is defined as per Equation 5.7 and has a standard deviation of $\sigma = 1$.

$$\ddot{x}_k = \begin{bmatrix} 0.1 & -0.01 & 0.01 & \frac{2\pi}{180} & \frac{1.05\pi}{180} & \frac{0.8\pi}{180} \end{bmatrix}^T \tag{5.13}$$

Simulations were completed based on a relative known CA model and the results are detailed in Figures 5-7 and 5-8 below. Examining the two figures below, reasonable tracking performance is seen with an initial overshoot that is to be expected as the filter converges. However, it can be concluded that this system, which uses a CV motion model for the relative motion dynamics, is able to properly track or estimate the relative POSE of an object when a relative CA motion is experienced.

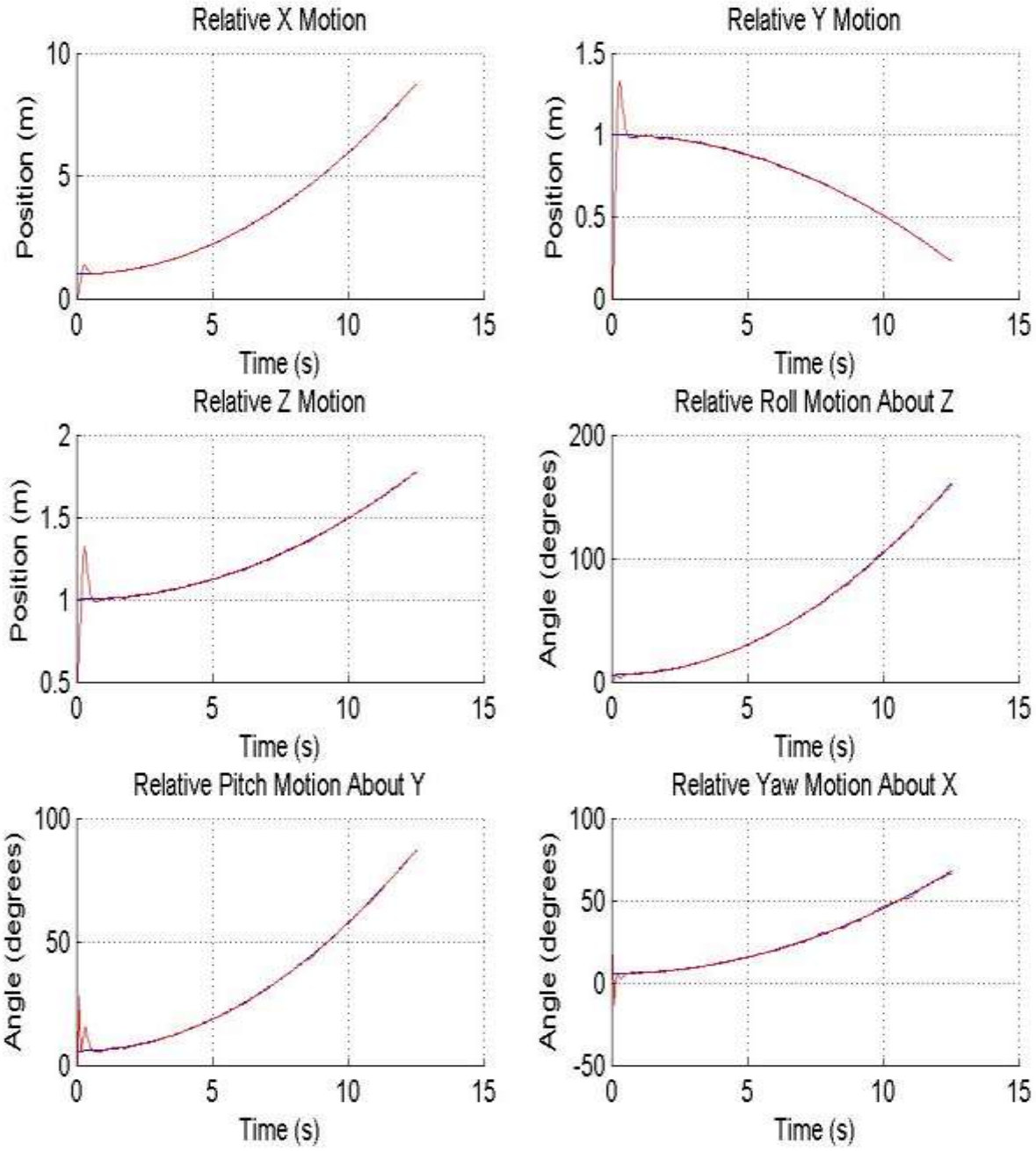

Figure 5-7: Depiction of the ideal (blue) and predicted relative (red) motion on all six axes for the CA case with measurement noise vector ν_k with a standard deviation of $\sigma = 1$.

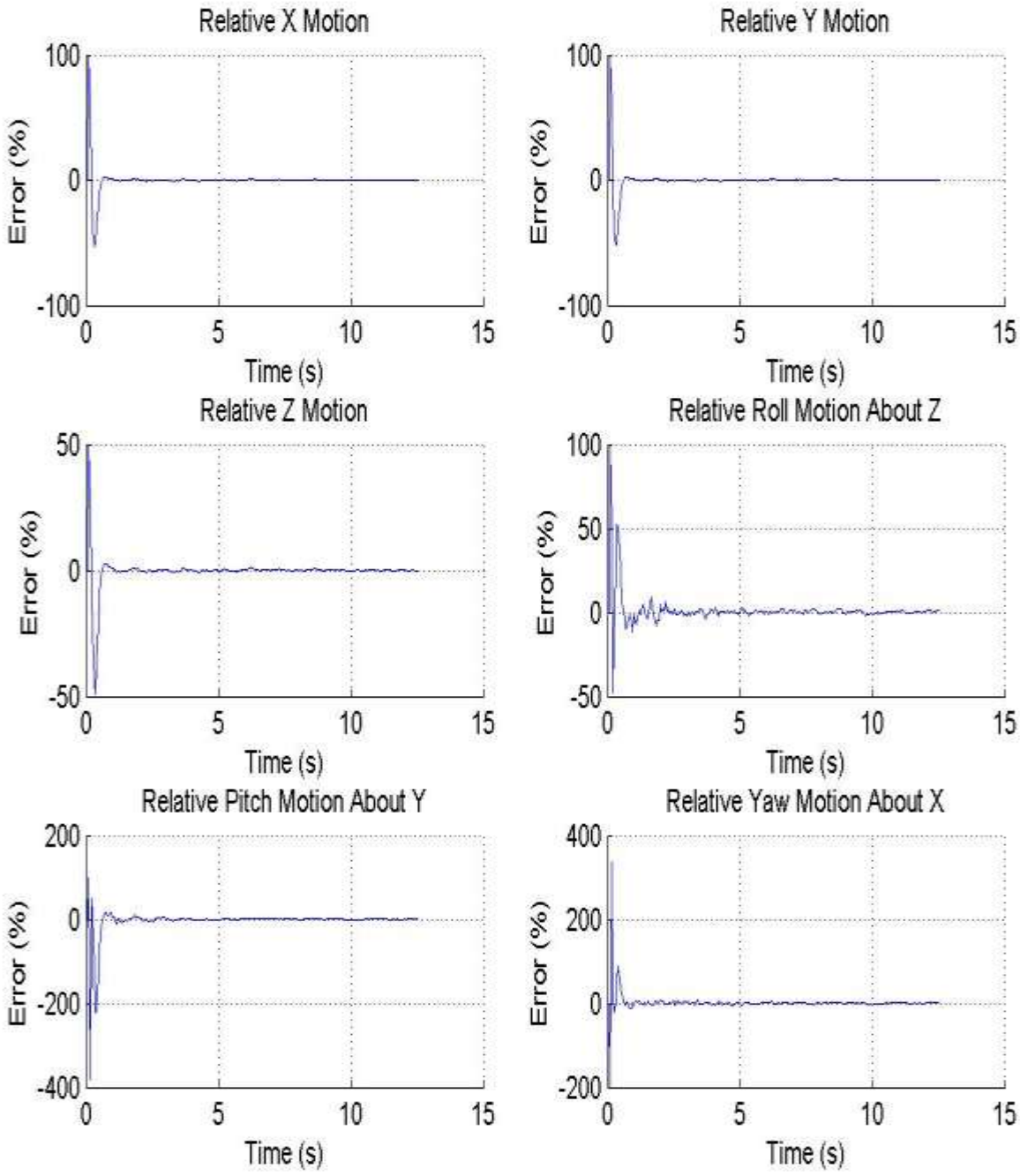

Figure 5-8: Percent error plot for the CA motion case for all six relative POSE parameters.

Next, the case in which the actual relative motion dynamics of the system follows an arbitrary motion is examined. An arbitrary motion model can be described as follows,

$$\mathrm{x}_k^{actual} = \mathrm{x}_{k-1}^{actual} + \delta_k \dot{x}_{arbitrary_k} \tag{5.14}$$

where $\dot{x}_{arbitrary_k}$ is arbitrarily determined for each time step. Various arbitrary motions were simulated and the following model is presented for discussion, as it excites possible positive or negative motion on all six axes for each time step with the express purpose of showing how this system can react to inflection points. The motion was also chosen such that it falls within the constraints of our effective workspace. As was done previously, it is assumed the measurement noise vector ν_k is defined per Equation 5.7 with a standard deviation of $\sigma = 1$ for this verification activity. The arbitrary motion model $\dot{x}_{arbitrary_k}$ is defined as follows in Equation 5.15, noting that the MATLAB rand() function was used,

$$\dot{x}_{arbitrary_k} = \begin{bmatrix} rand(1) * 0.1 + \sin\left(\frac{k}{10}\right) \\ 0.1 + rand(1) * \sin\left(\frac{k}{3}\right) \\ \cos\left(rand(1) * \frac{k}{12}\right) \\ \sin\left(rand(1) * \frac{k}{15}\right) \\ rand(1) * 0.15 \\ rand(1) * 0.5 * \sin\left(\frac{k}{3}\right) \end{bmatrix} \tag{5.15}$$

The simulation results for the above arbitrary motion model are presented in Figures 5-9 and 5-10 below, noting that reasonable tracking performance is observed. These results are consistent with the CV and CA motions trialed above, noting that the relative error is approximately the same between the three different motion models. As well the overshoot in all three cases is quite low. It should be noted that the system does not achieve convergence to within ±10% due to higher errors in the relative yaw ψ parameter. However it appears to reach a steady state quite quickly after around three seconds with some amount of error.

The relative error and convergence time for all three of the motion models are detailed in Table 5-2 below. As was the case for Table 5-1, it is noted that this table is used for qualitative purposes as different motion models and conditions such as noise levels will yield different results. However, the trends are shown, and these results allow us to conclude that the CV motion model is a suitable selection for the process model of the relative motion dynamics.

Table 5-2: Effect of System for Different Motion Models on the System

	Percent Error							**Convergence to within +/-10%**	
Motion Model	X	Y	Z	$\emptyset$	θ	ψ	**Average Error**	**# time steps**	**Total time (s)**
CV	1.04	1.06	0.84	1.44	3.66	4.19	2.04	20	1
CA	1.84	1.87	1.42	2.88	6.95	5.27	3.37	20	1
Arbitrary	2.58	2.7	3.25	2.89	7.8	13.68	5.48	64 for all but yaw Yaw N/A	3.2 NA

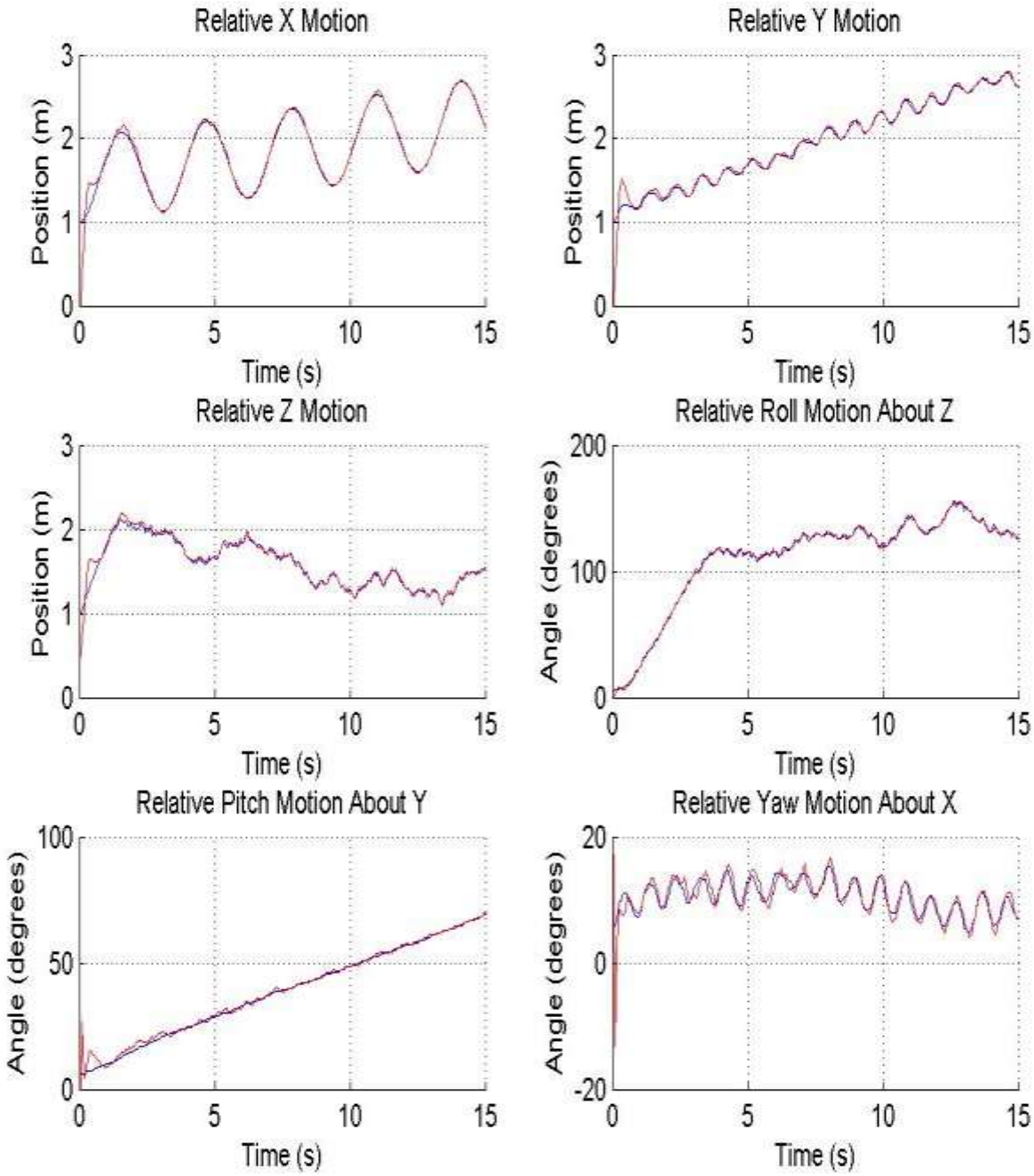

Figure 5-9: Depiction of the ideal (blue) and predicted relative (red) motion on all six axes for an arbitrary motion case with measurement noise vector ν_k with a standard deviation of $\sigma = 1$.

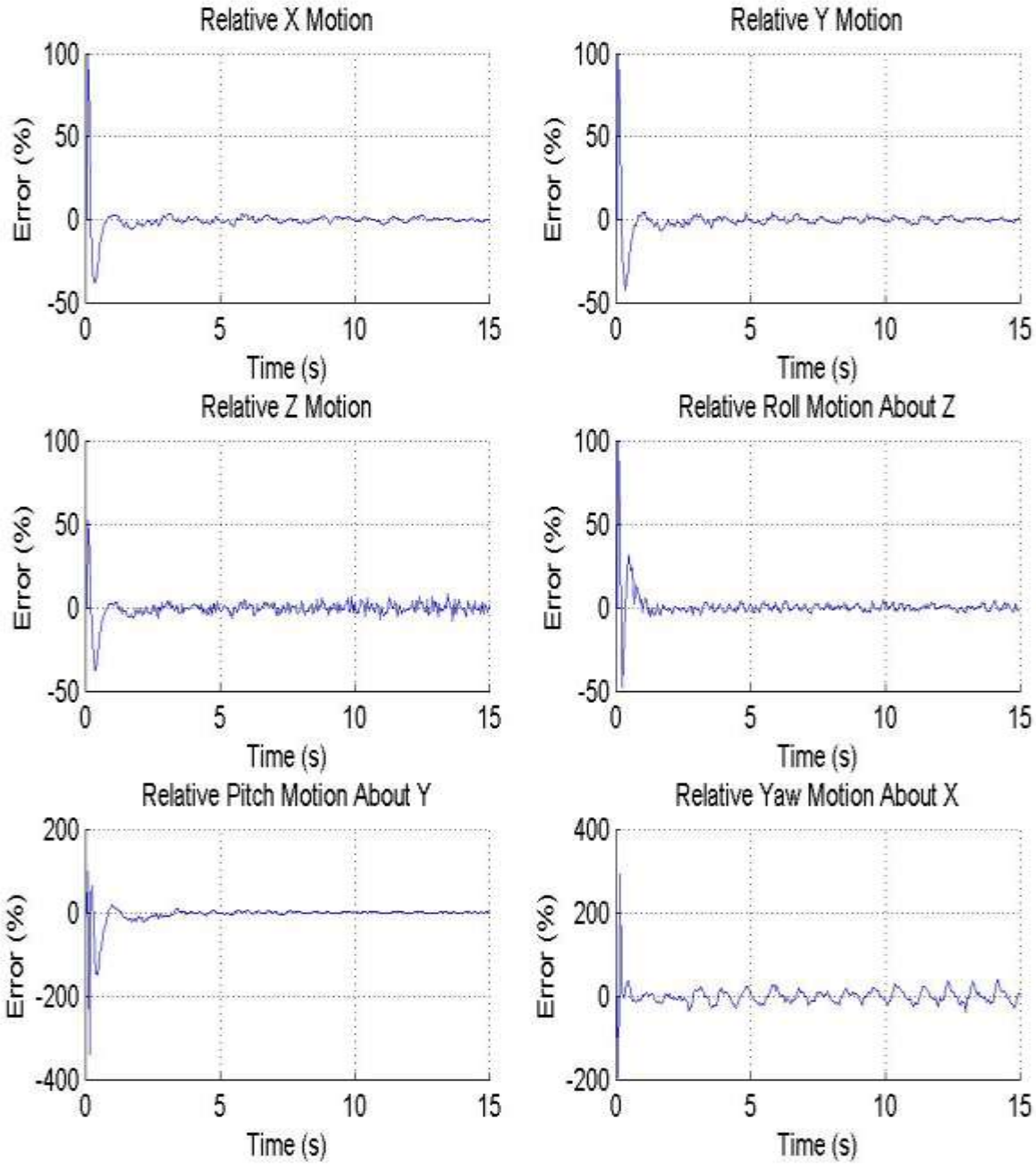

Figure 5-10: Percent error plot for the arbitrary motion case for all six relative POSE parameters.

5.5 Effects of Measurement Noise

Next the effects of measurement noise on the system model will be examined with the motivation that additional measurement noise can be introduced through both the camera sensor and the image processing techniques. As the measurement noise level increases, degradation in the system model performance and accuracy is expected. The baseline CV motion model for x_k^{actual} detailed in Equation 5.8 will be used to show the effects of noise on this system model. Table 5-3 provides a reference to the injected noise level for the measurement system and how it relates to the physical system in terms of pixel measurement error. Figures 5-11 through 5-14 illustrate the effect of increasing noise on the system.

Table 5-3: Relationship of Measurement Noise v_k to Measurement Error

Noise v_k	Measurement Error in Camera Pixels	Measurement Error in Display Pixels	Measurement Error in *X* direction (display)	Measurement Error in *Y* direction (display)
$\sigma = 1$	± 2	± 2.5	0.7%	0.4%
$\sigma = 4$	± 8	± 16	4.4%	2.5%
$\sigma = 7$	± 14	± 28	7.8%	4.4%
$\sigma = 8$	± 16	± 32	8.9%	5%

As is shown in the figures below, the increase in measurement noise will increase the relative error in the system model, which is to be expected. This result can be theoretically verified by restating Equation 4.26 as follows,

$$\hat{x}_{k,k} = \hat{x}_{k,k-1} + K(\tilde{y}_k - \hat{y}_k) \tag{4.26}$$

Where the modified measured output $\tilde{y}_k$ is comprised of the actual motion, the display to camera transformation h, and the measurement noise v_k. The above equation can therefore be expanded to show the direct relationship between the measurement noise and the predicted state estimate $\hat{x}_{k,k}$ is as follows,

$$\hat{x}_{k,k} = \hat{x}_{k,k-1} + K\left(h * y_k^{actual} + v_k - \hat{y}_k\right) \tag{5.17}$$

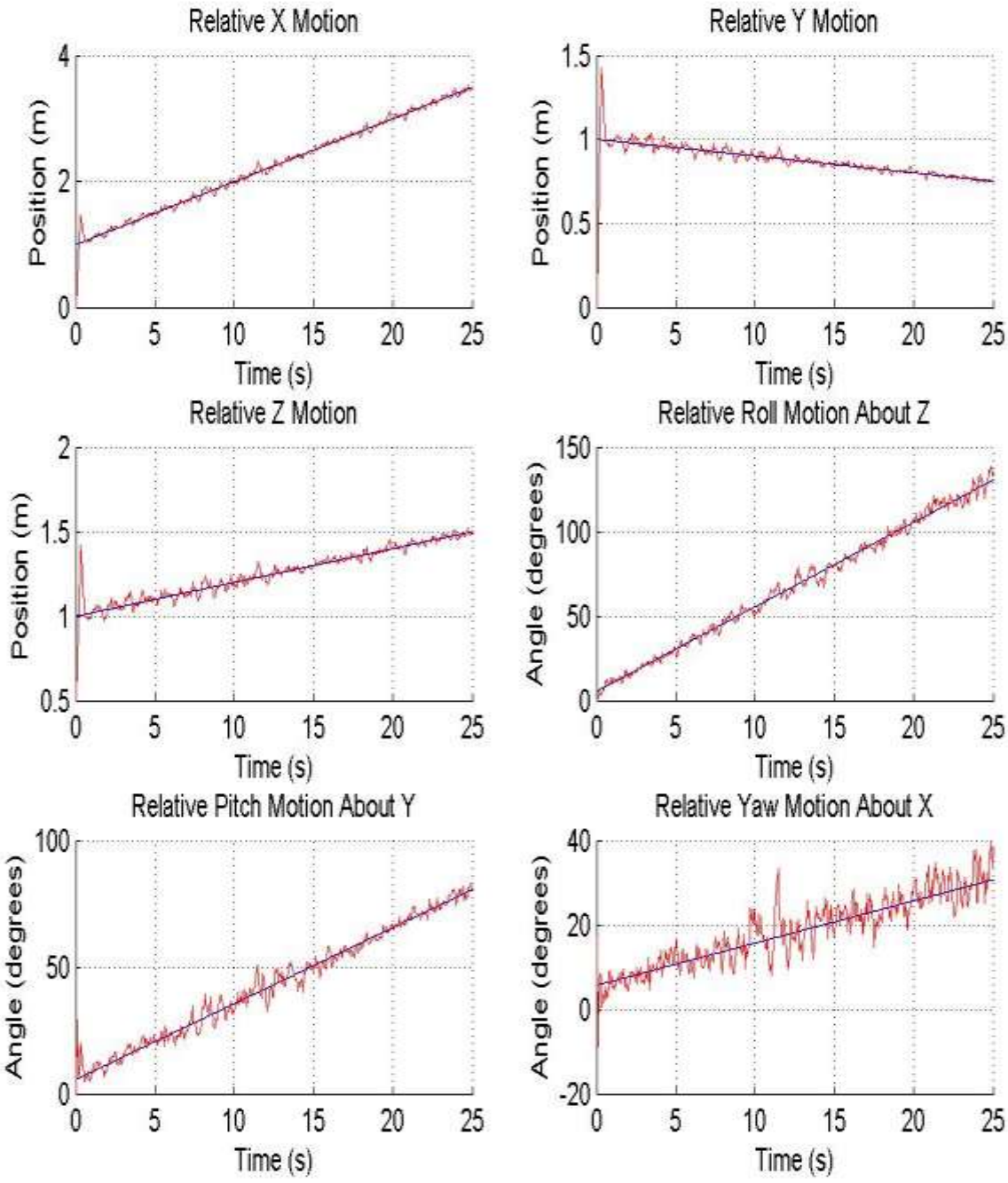

Figure 5-11: Depiction of the ideal (blue) and predicted relative (red) motion on all six axes for the CV case with measurement noise vector v_k with a standard deviation of $\sigma = 7$.

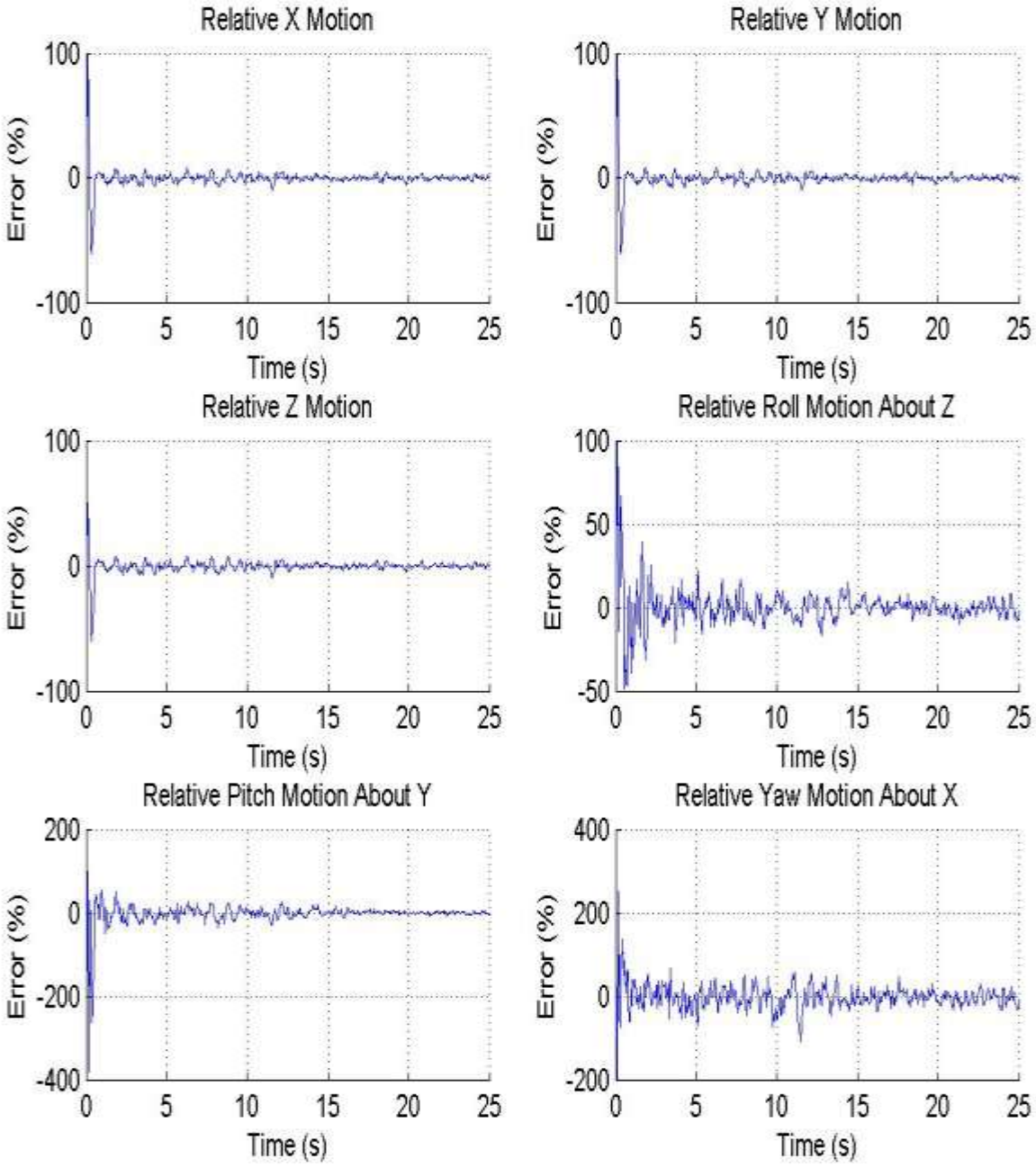

Figure 5-12: Percent error plot for the CV motion case for all six relative POSE parameters having a measurement noise vector ν_k with a standard deviation of $\sigma = 7$.

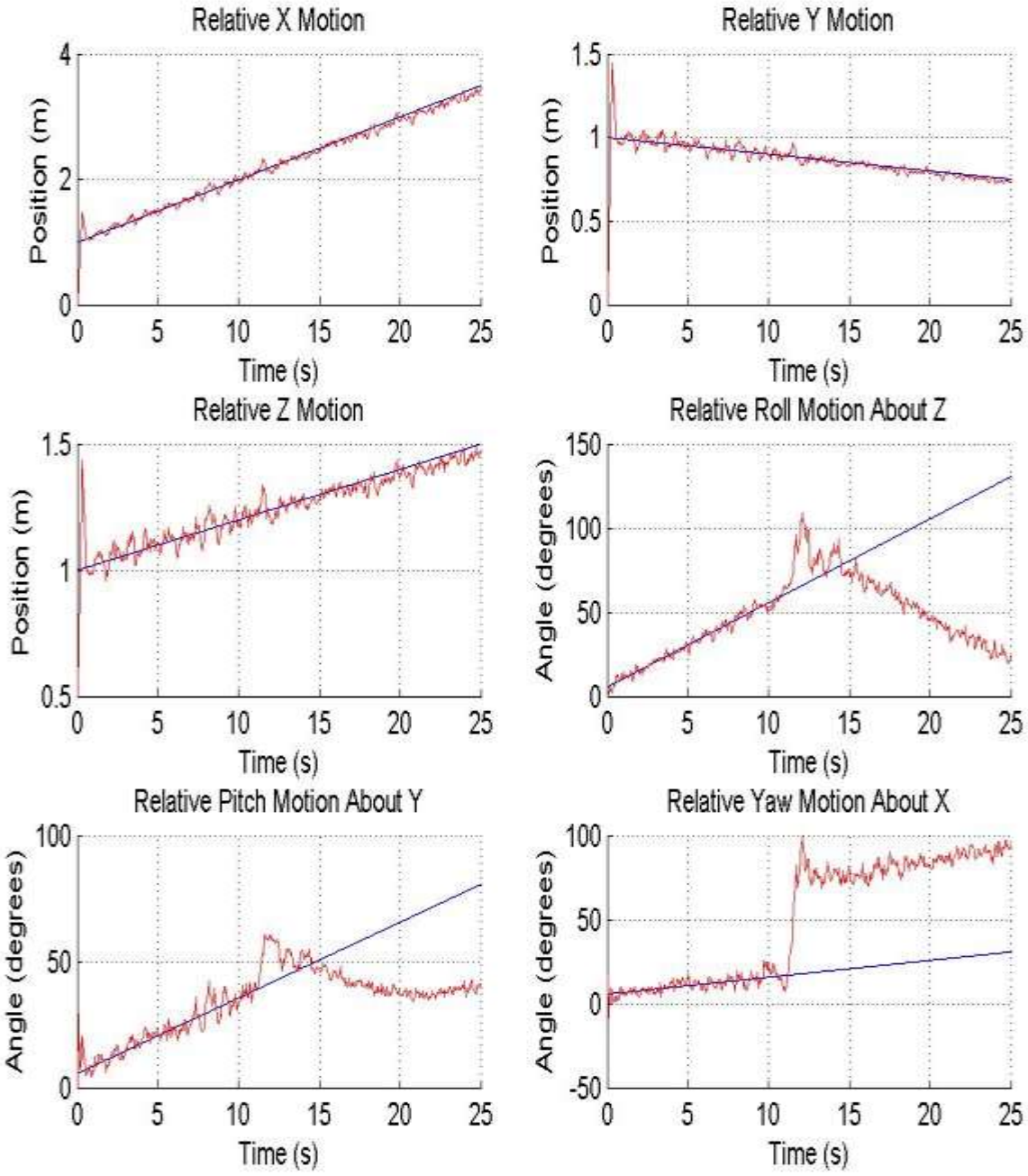

Figure 5-13: Depiction of the ideal (blue) and predicted relative (red) motion on all six axes for the CV case with measurement noise vector ν_k with a standard deviation of $\sigma = 8$.

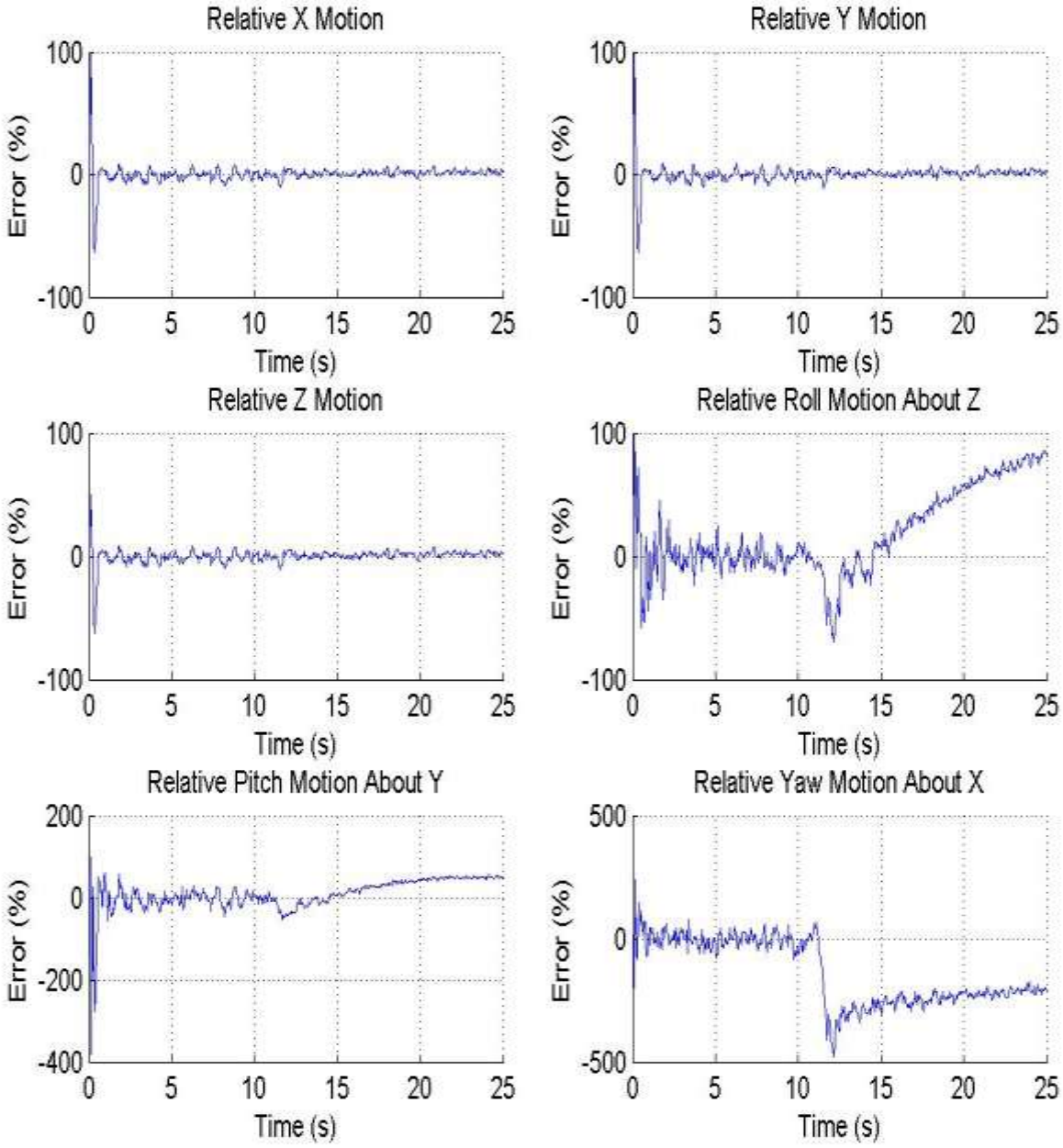

Figure 5-14: Percent error plot for the CV motion case for all six relative POSE parameters having a measurement noise vector ν_k with a standard deviation of $\sigma = 8$.

Also it was observed, and is reflected in Figures 5-11 and 5-13, that the initial system overshoot is unaffected by the varying noise levels. Additionally, convergence or tracking of the system to $\pm 10\%$ of the actual value is affected negatively as the noise increases. One additional important observation is that as the measurement noise increases, it can negatively reduce the effective workspace of the system. Reviewing Figure 5-13 it is observed that the system stops tracking the actual relative motion properly for the relative Euler angles for the CV motion case with a measurement noise vector ν_k with a standard deviation of $\sigma = 8$. As the relative position and orientation increases, each pixel on the display represents a larger physical distance of the target object and the workspace. The higher level of measurement noise therefore has a more pronounced effect on the ability of the system to maintain tracking as is shown by Figure 5-13. It should be noted that the above results are illustrative and the results will vary due to the actual relative motion, the noise distribution, and other initial conditions. Table 5-4 qualitatively shows the trend of how the relative error increases as the measurement error increases.

Table 5-4: Impact of Increasing Measurement Noise on the System

	Percent Error							Convergence to within +/-10%	
Noise ν_k	X	Y	Z	$\emptyset$	θ	ψ	Average	# time steps	Total time (s)
$\sigma = 1$	1.04	1.06	0.84	1.44	3.66	4.19	2.04	20	1
$\sigma = 4$	1.87	1.93	1.66	3.72	6.9	12.03	4.68	289 for all but yaw Yaw N/A	14.45 N/A
$\sigma = 7$	2.76	2.85	2.54	6.09	10.47	20.26	7.49	289 for all but yaw Yaw N/A	14.45 N/A

5.6 Effects of Changing the Sampling Rate

As noted in Chapter 3 and in Equation 3.26, a higher sampling rate should reduce the dynamic modeling error. This result is clear, as a reduction in the sample period will reduce the apparent change in the velocity between frames.

$$Dynamic\ Model\ Error \propto \delta_k * |\dot{x}_{actual} - \dot{x}_{model}| \quad (3.26)$$

Also for this physical system, the lower sampling rate of 20Hz is bound due to the requirement for smoothness in the rendering as noted in Chapter 4. The upper bound of the sampling rate is 30Hz due to the system limitation of the camera video stream being limited to 30fps. For the purposes of the verification activities, the baseline CV motion model detailed in Equation 5.8 for x_k^{actual} will be used to show the effects of the two endpoint sampling rates on this system model.

Various simulations were conducted with the two sampling rates (20Hz, 30Hz) for a couple of cases of varying measurement noise vector v_k and the results are detailed in Table 5-5 below. A clear relationship between relative system error and the sampling rate is seen, which matches the theoretical prediction. As the sampling rate increases a decrease in the overall relative error is observed, which is to be expected based on the theory. Convergence to within ±10% is approximately the same with differences accounted for due to the random nature of the noise signal causing minor shifts. Table 5-5 qualitatively details the trend of how the relative error decreases with an increasing sample rate. However, it is also seen that the differences in error between the two sampling rates are minimal, i.e. less than 1% difference. This also allows us to conclude that the selection of a dynamically varying sample rate between 20Hz to 30Hz is acceptable from a system design perspective, as the error incurred from a change in sampling rates is negligible.

Table 5-5: Impact of Different Samplings Rate on the System

		Percent Error							±10% convergence	
Sampling Rate (Hz)	**Noise v_k**	X	Y	Z	$\emptyset$	θ	ψ	**Avg**	**# time steps**	**Total time (s)**
20	$\sigma = 1$	1.04	1.06	0.84	1.44	3.66	4.19	2.04	20	1
30	$\sigma = 1$	0.82	0.83	0.69	1.18	3.22	3.65	1.73	22	0.73
20	$\sigma = 7$	2.76	2.85	2.54	6.09	10.47	20.26	7.49	289 for all but yaw Yaw N/A	14.45 N/A
30	$\sigma = 7$	2.54	2.61	2.4	5.22	9.67	17.56	6.67	477 for all but yaw Yaw N/A	15.9 N/A

5.7 Impact of Updating Q_k and R_k

Next the effects of varying the measurement noise covariance matrix R_k and the disturbance covariance matrix Q_k will be examined. As noted previously in Chapter 3, the disturbance covariance matrix Q_k represents the un-modeled system dynamics of the CV model with larger values of Q_k being used for faster motion parameters and conversely smaller values of Q_k selected for slower motion. This statement is theoretically verified through a brief examination of Equations 3.26, which defines the dynamic model error, and Equation 4.22, which sets the estimate covariance $P_{k,k-1}$,

$$P_{k,k-1} = A_k P_{k-1,k-1} {A_k}^T + Q_k \tag{4.22}$$

A faster relative motion will have more apparent change in velocity between two subsequent frames, and the result is that CV motion model will have larger error in predicting the state estimate. This implies that there would be less confidence in state estimates for cases of fast relative motion. In order to account for these cases of reduced confidence in our state estimate, the estimate covariance $P_{k,k-1}$ would therefore need to be increased. This increase in the estimate covariance implies that larger entrywise values of Q_k would be required for faster motions. Conversely, smaller entrywise values of Q_k are required for slower motions.

The effect of varying the measurement noise covariance matrix R_k can be predicted through an examination of its definition. Recall that v_k is defined as the measurement noise vector of zero mean Gaussian noise R_k. As the entries of the covariance matrix R_k increase, the distribution of the measurement noise vector v_k will also widen due to the entrywise increase in R_k. Hence an increase in the error to the predicted state estimate $\hat{x}_{k,k}$ is expected. Conversely as the entries of the covariance matrix R_k decrease, a decrease in error is expected to the predicted state estimate $\hat{x}_{k,k}$.

Simulations were completed to verify the theoretical effects of changing R_k and Q_k. Given that that the effects of Q_k can vary based on the actual motion, simulations were completed with both the baseline CV motion model for x_k^{actual} detailed in Equation 5.8 and also the arbitrary motion model as detailed in Equations 5.14 and 5.15. For the initial conditions, the initial actual state x_1^{actual} as per Equation 5.1 was used, the initial state estimate $\hat{x}_{1,1}$ was defined per Equation 4.10, and the measurement noise vector v_k

with a standard deviation of $\sigma = 1$ was used. The baseline values of R_k and Q_k used are defined as per Equations 4.12 and 4.13. Various combinations were simulated to verify the effects of modifying both R_k and Q_k and the following four cases will be detailed for qualitative purposes,

a. R_k decreased and Q_k unchanged
b. R_k increased and Q_k unchanged
c. R_k unchanged and Q_k increased
d. R_k unchanged and Q_k decreased

Table 5-6 below provides a reference to how the entrywise values of R_k and Q_k were modified relative to the baseline definition listed in Equations 4.12 and 4.13 for each of the four test cases.

Table 5-6: R_k and Q_k Modification Table

	Increase – Multiplication Factor	Decrease – Multiplication Factor
$\boldsymbol{R_k}$	2^4	2^{-4}
$\boldsymbol{Q_k}$	10^2	10^{-2}

Various simulation experiments were conducted and some of the results of the various motion cases are provided in Figures 5-15 through 5-19 for illustration. Tables 5-7 and 5-8 summarizes the results noting that these results are qualitative only to show trends and will vary based on initial conditions, noise, actual motion model, etc.

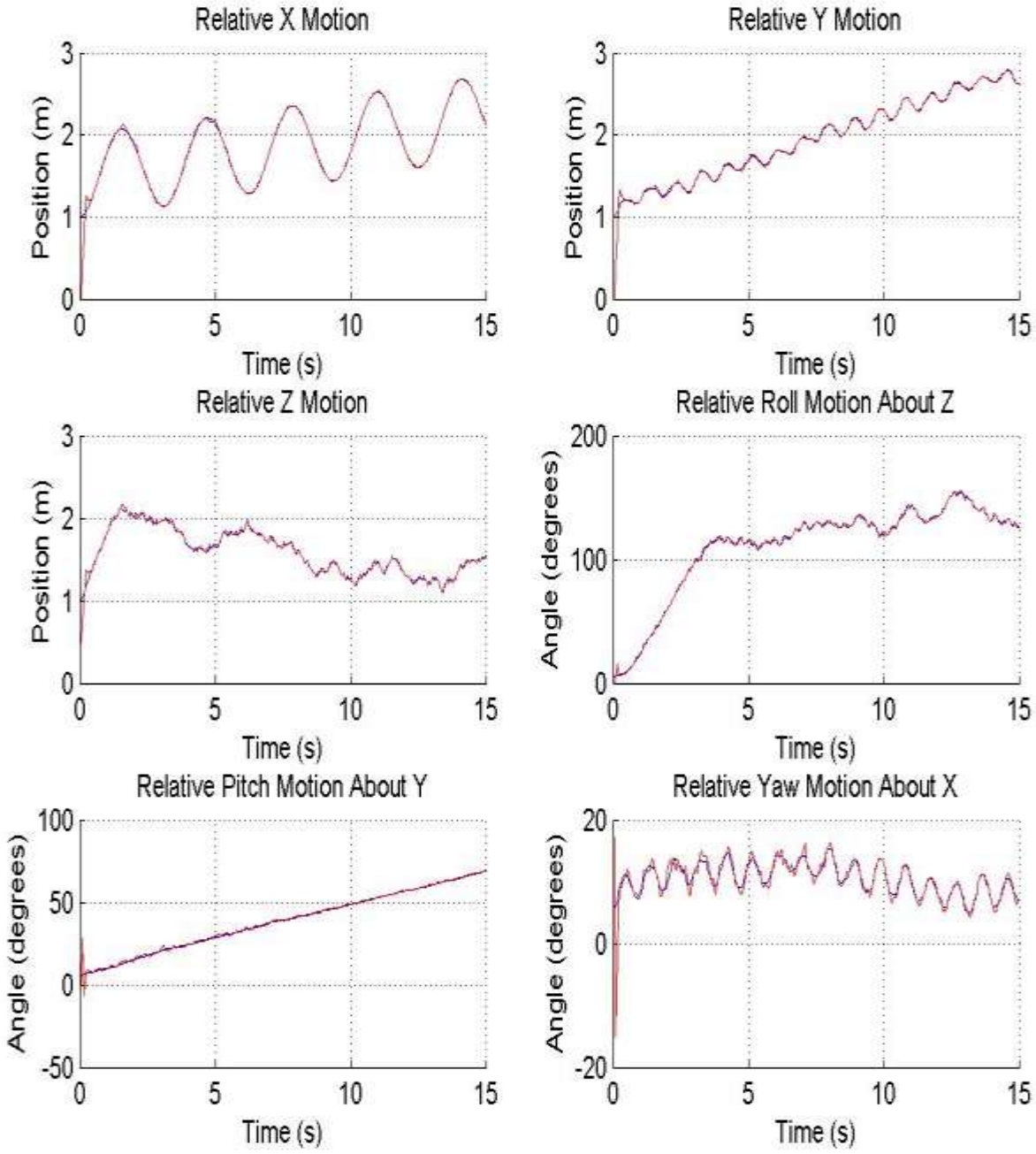

Figure 5-15: Depiction of the ideal (blue) and predicted relative (red) motion on all six axes for the arbitrary motion case with an entrywise decrease to R_k and Q_k unchanged, and a measurement noise vector v_k with a standard deviation of $\sigma = 1$.

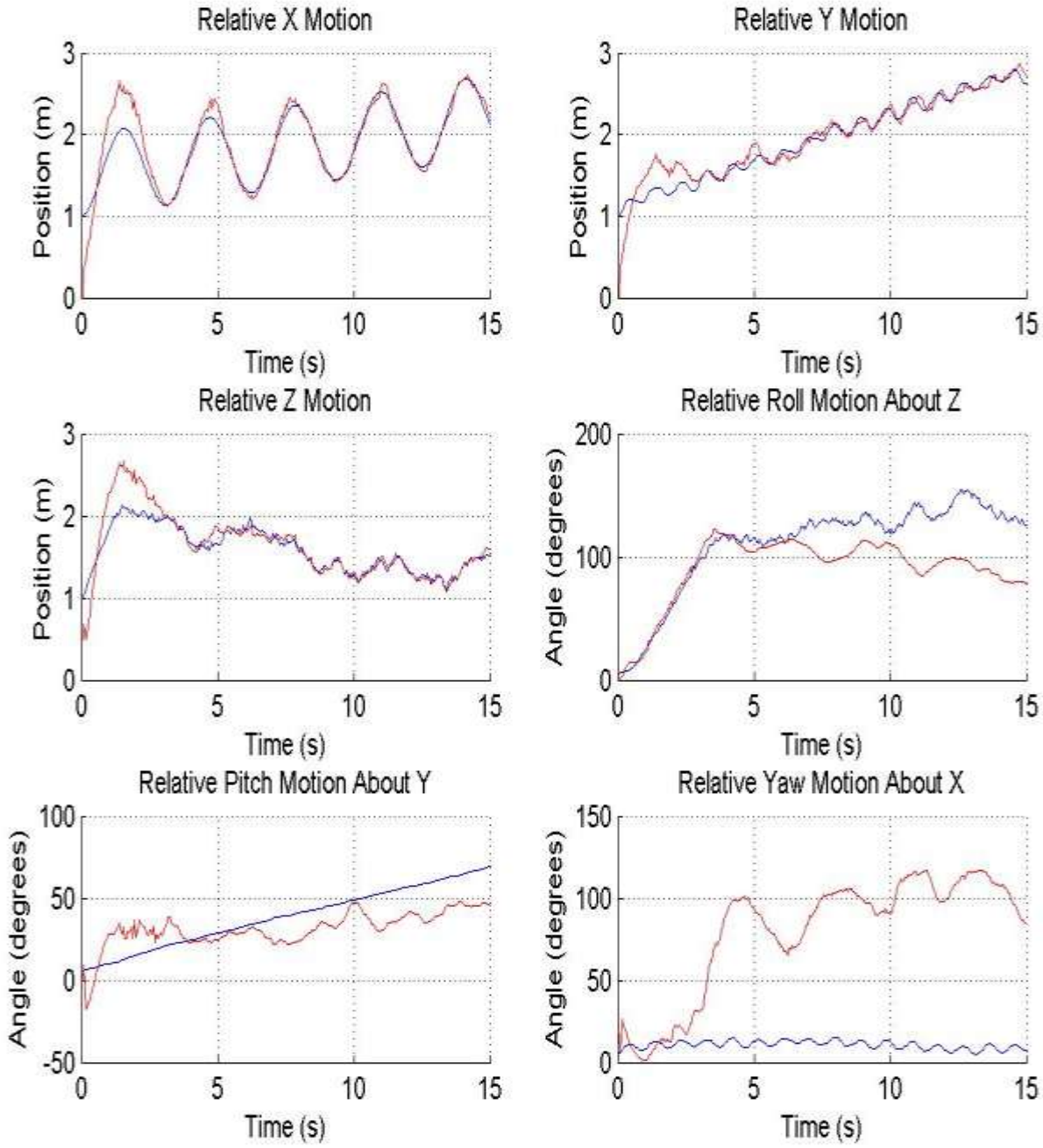

Figure 5-16: Depiction of the ideal (blue) and predicted relative (red) motion on all six axes for the arbitrary motion case with an entrywise increase to R_k and Q_k unchanged, and a measurement noise vector v_k with a standard deviation of $\sigma = 1$.

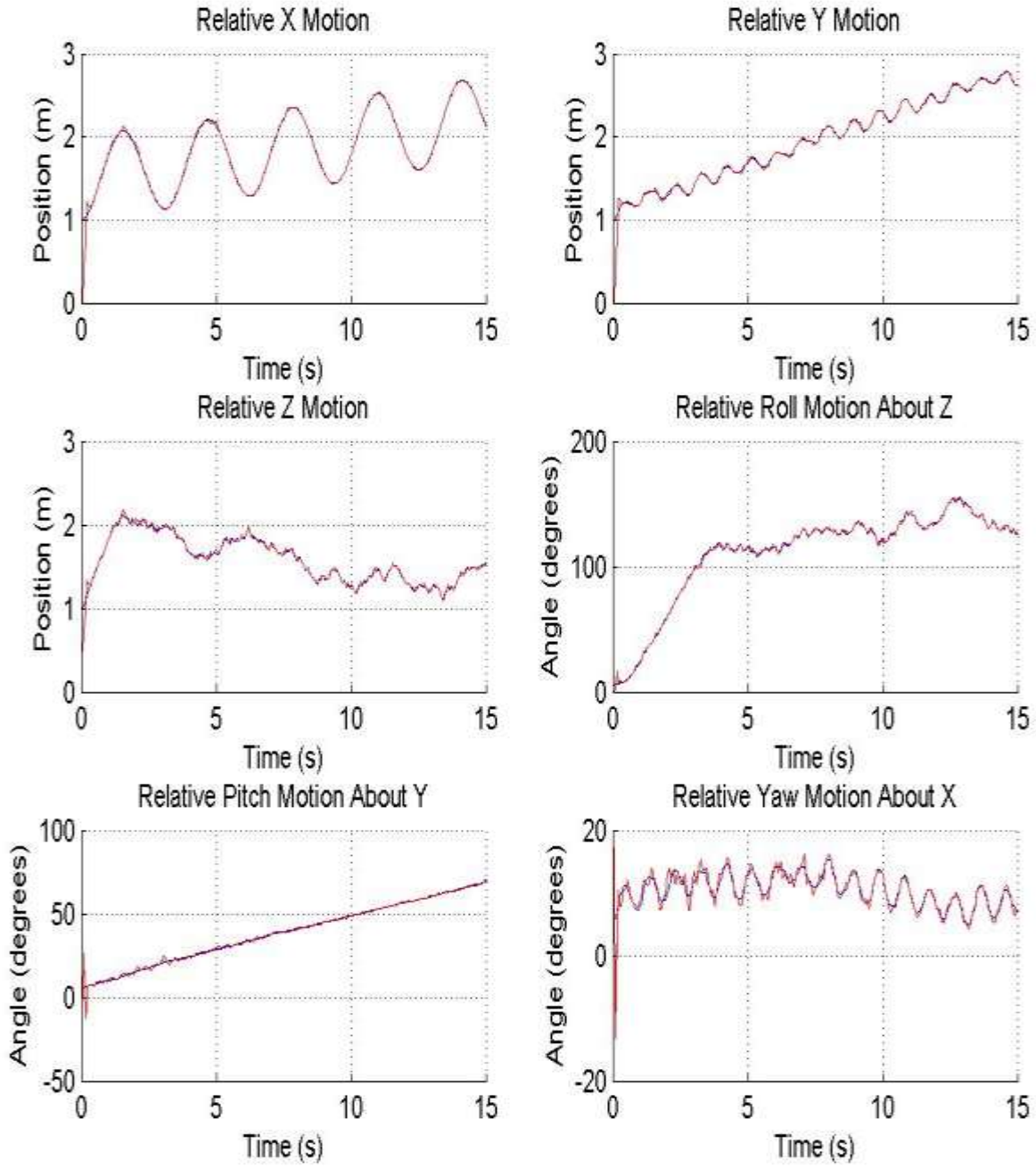

Figure 5-17: Depiction of the ideal (blue) and predicted relative (red) motion on all six axes for the arbitrary motion case with R_k unchanged and an entrywise increase to Q_k, and a measurement noise vector v_k with a standard deviation of $\sigma = 1$.

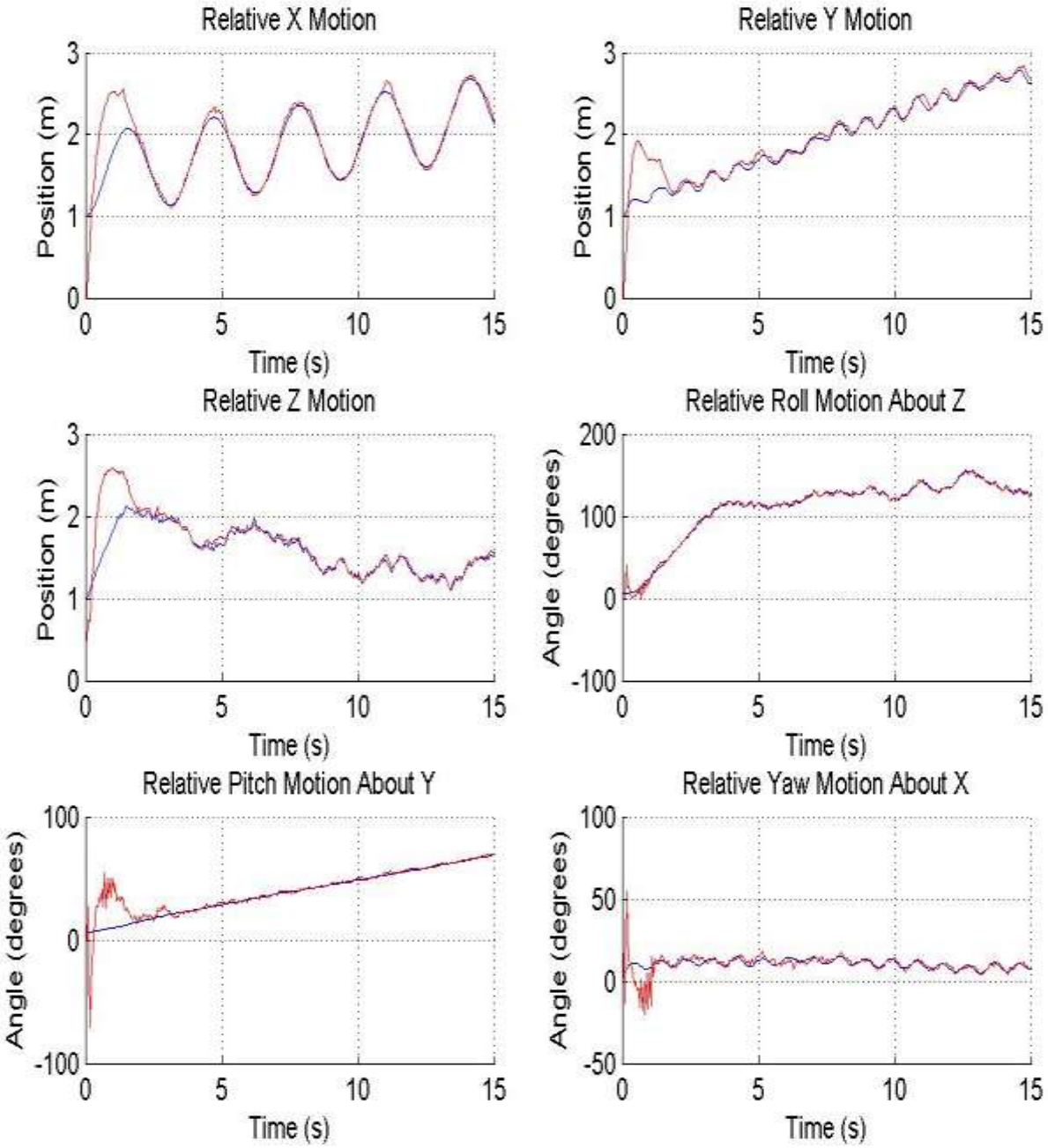

Figure 5-18: Depiction of the ideal (blue) and predicted relative (red) motion on all six axes for the arbitrary motion case with R_k unchanged and an entrywise decrease to Q_k, and a measurement noise vector v_k with a standard deviation of $\sigma = 1$.

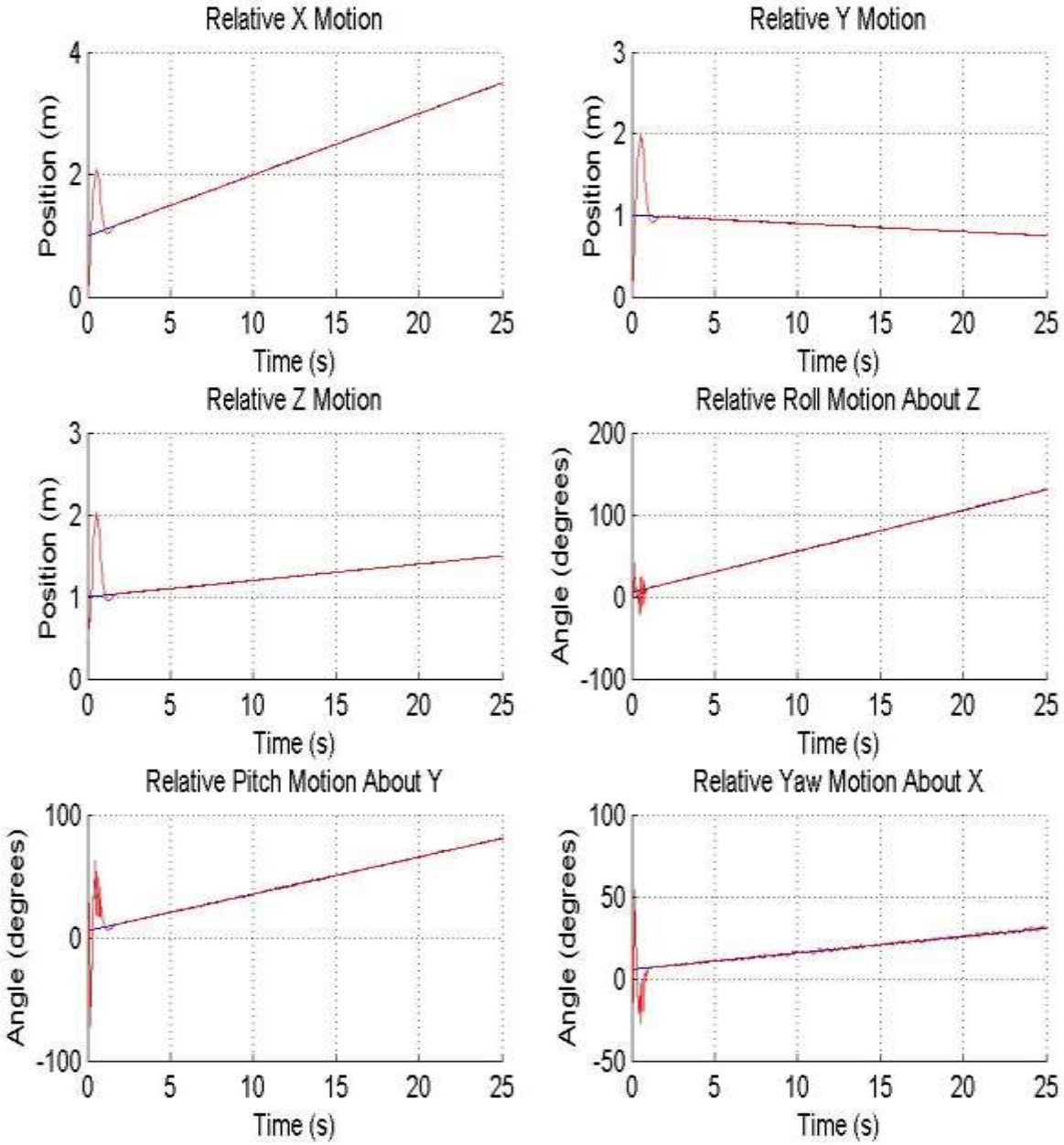

Figure 5-19: Depiction of the ideal (blue) and predicted relative (red) motion on all six axes for the CV motion case with R_k unchanged and an entrywise decrease to Q_k, and a measurement noise vector v_k with a standard deviation of $\sigma = 1$.

Table 5-7: Impact of Varying the Q_k and R_k Covariance Matrices for the CV Motion Case

Baseline CV Motion Case	X	Y	Z	∅	θ	ψ	**Average**
Total Average Percent Error (%)	1.04	1.06	0.84	1.44	3.66	4.19	2.04
Peak Absolute Overshoot (% error)	100	100	50.05	100	363.26	330.64	173.99
R_k decreased, Q_k unchanged	X	Y	Z	∅	θ	ψ	**Average**
Total Average Percent Error (%)	0.95	0.97	0.75	2.06	3.98	5.41	2.35
Peak Absolute Overshoot (% error)	100	100	50.05	260.2	470.51	360.92	223.69
R_k increased, Q_k unchanged	X	Y	Z	∅	θ	ψ	**Average**
Total Average Percent Error (%)	1.19	1.19	1.12	1.76	6.65	6.28	3.03
Peak Absolute Overshoot (% error)	100	100	54.36	100	389.3	362.48	184.36
R_k unchanged, Q_k increased	X	Y	Z	∅	θ	ψ	**Average**
Total Average Percent Error (%)	1.08	1.1	0.89	3.89	5.82	7.16	3.32
Peak Absolute Overshoot (% error)	100	100	54.39	429.4	954.26	330.44	328.08
R_k unchanged, Q_k decreased	X	Y	Z	∅	θ	ψ	**Average**
Total Average Percent Error (%)	3	3.02	2.77	6.11	15.06	11.79	6.96
Peak Absolute Overshoot (% error)	113.53	115.93	114.1	539.14	1261.7	828.62	495.51

Table 5-8: Impact of Varying the Q_k and R_k Covariance Matrices for the Arbitrary Motion Case

Baseline Arbitrary Motion Case	X	Y	Z	∅	θ	ψ	**Average**
Total Average Percent Error (%)	2.58	2.7	3.25	2.89	7.8	13.68	5.48
Peak Absolute Overshoot (% error)	100	100	52.38	100	338.6	290.84	163.64
R_k decreased, Q_k unchanged	X	Y	Z	∅	θ	ψ	**Average**
Total Average Percent Error (%)	1.57	1.81	2.67	3.41	6.24	10.69	4.4
Peak Absolute Overshoot (% error)	100	100	52.4	216.9	416.41	316.05	200.29
R_k increased, Q_k unchanged	X	Y	Z	∅	θ	ψ	**Average**
Total Average Percent Error (%)	6.66	6.56	6.86	19.67	46.53	725.16	135.24
Peak Absolute Overshoot (% error)	100	100	62.91	100	386.95	2216.4	494.37
R_k unchanged, Q_k increased	X	Y	Z	∅	θ	ψ	**Average**
Total Average Percent Error (%)	1.54	1.77	2.73	3.98	9.33	11.91	5.21
Peak Absolute Overshoot (% error)	100	100	52.38	364.52	859.29	294.48	294.48
R_k unchanged, Q_k decreased	X	Y	Z	∅	θ	ψ	**Average**
Total Average Percent Error (%)	6.58	6.53	6.86	6.68	34.99	26.76	14.73
Peak Absolute Overshoot (% error)	100	100	68.51	496.65	1234.9	564.1	427.4

As can be seen from the above table and figures, an entrywise increase or decrease to the measurement covariance matrix R_k leads to a direct effect on the relative error as predicted theoretically. For the case of an entrywise decrease to R_k, the average error is only slightly larger than the baseline case for the CV motion case, i.e. 2.35% versus 2.04% baseline. However the error is decreased on the Cartesian elements while slightly increased on the Euler angles. For the arbitrary motion case, an overall error reduction is observed (4.4% versus 5.48% baseline) with a small increase in error in the baseline arbitrary motion case for the roll angle ∅**.** Since the measurement noise vector v_k is set to a standard deviation of $\sigma = 1$ and a baseline value of $R_k = 4[I_{8x8}]$, it is postulated that the baseline value of R_k, which represents 95% of the error being within 4 display pixels, is close to the noise error floor. Hence entrywise decreases to R_k from the baseline will have minimal impact.

For the case of an entrywise increase of the measurement covariance matrix R_k, an increase in error is observed for both the CV (2.04% to 3.03%) and arbitrary motion (5.48% to 135.24%) cases. For the arbitrary motion case the increase to R_k results in a loss of convergence for the three relative Euler angles. One additional effect of increasing the elements of the measurement covariance matrix R_k is that it will tend to reduce the value of the Kalman gain K. A reduction in the Kalman gain means that the *innovation* and the measured output values y_k will have a lessened impact on the value for the state estimate $\hat{x}_{k,k}$ and that the state process model A_k will drive $\hat{x}_{k,k}$ as shown in Equations 4.25 and 4.26 below.

$$K = P_{k,k-1}\, C_k^T \left(R_k + C_k P_{k,k-1} C_k^T\right)^{-1} \tag{4.25}$$

$$\hat{x}_{k,k} = \hat{x}_{k,k-1} + K(\tilde{y}_k - \hat{y}_k) \tag{4.26}$$

For the case when the EKF is tracking a motion that is similar to a constant velocity, the process model A_k will be able to suitably predict the state estimate $\hat{x}_{k,k}$ with a reasonable level of error as seen in Table 5-7 (R_k increased, Q_k unchanged case). However with the arbitrary motion case, filter convergence may be difficult if the changes in relative motion are too large as was seen in Figure 5-16 and reflected numerically in Table 5-8 (R_k increased, Q_k unchanged case).

Next the case for which an entrywise increase to the disturbance covariance matrix Q_k is examined. For the "slower" CV motion case, an increase in error is seen when Q_k is increased, i.e. 3.32% versus 2.04% baseline. With the faster" arbitrary motion case, an entrywise increase to Q_k results in a decrease in error, which matches the theory, i.e. 5.21% versus 5.48% baseline. Furthermore, for the relative pitch θ angle motion an increase in error is observed. This result is also to be expected as the relative pitch motion is smooth and changes more "slowly". Additionally, while the peak overshoot is greater than the baseline case for both the CV and arbitrary motions, it can be seen through an examination of the figures for the arbitrary motion that an entrywise increase to Q_k will result in much better convergence. This can be clearly seen through Figures 5-9 (baseline case), 5-17 (entrywise increase to Q_k), and 5-18 (entrywise decrease to Q_k) where initial tracking performance is much better.

For an entrywise decrease to the disturbance covariance matrix Q_k, an increase in overall error is observed for the arbitrary motion case (14.73% versus 5.48% baseline) as expected. The overall error is greater for the "faster" motion of the arbitrary error case, as expected. For the CV motion case, the overall

error also increased (6.96% versus 2.04% baseline). However, the majority of this error is due to the initial convergence phase, as seen in Figure 5-19 above. Using MATLAB it was found that the average error once ±10% convergence was attained was less than 1%. This result is also seen through the higher peak overshoot values noted in Table 5-7 above.

Based on an overall review of the above results, it is seen that the baseline values for Q_k and R_k provide the best balance between overall system error, amount of initial overshoot, and error once convergence to within ±10% has been maintained.

5.8 Discussion of Results

Through simulation the effects of changing various parameters in this system were examined and the system was verified to be reasonably robust. First and foremost, it was shown that the CV model is an appropriate selection for the relative motion dynamics through its ability to handle multiple actual motion models. The error seen is below 6% for various motion model cases and this performance is deemed acceptable.

As well, the system was shown to be tolerant to a relatively high amount of noise. In practice, it is expected that the measurement error due to the camera sensor and image processing techniques will be on the order of a few pixels and the system demonstrated robustness to tens of pixel measurement noise. Noise levels up to $\sigma = 8$ can cause loss of convergence tracking for the EKF. With the 8MP camera used in this design, it is unlikely that large errors due to the camera sensor will be observed. However errors due to the image processing techniques could vary and this aspect needs to be examined during the validation experiments.

The simulations have verified the theoretical effects of changing the sample rate, i.e. increasing sample rate decreases the overall error. However these simulations also show us that allowing for a varied sample rate between 20Hz and 30Hz will not drastically impact the system error. Less than 1% difference in error would be incurred by this changing sampling rate, which is viewed as acceptable. Therefore the approach taken for this system to use a dynamically varied sample rate is also verified.

Simulations have also shown that the design selection used for both the disturbance noise Q_k and the measurement noise covariance matrix R_k strikes an appropriate balance between error, overshoot, and convergence. While the parameters could be varied to suit the test specific conditions, it is felt that the parameters chosen provide the best overall balance and performance.

The effects of initial conditions particularly how close in proximity the initial state estimate $\hat{x}_{1,1}$ is to the initial actual state x_1^{actual} will dramatically affect the overall performance. Using a pre-defined initial state estimate $\hat{x}_{1,1}$, as is done for this system, will cause higher error and overshoot, and increase the time to attain ±10% convergence or tracking if the proximity to the initial starting point is large. It should be noted that most numerical methods will work well provided that the initial starting point is close to the solution. As this design is a proof of concept, more sophisticated initialization techniques were not pursued. However, these elements can easily be added to the design as noted in Chapters 3 and 4.

It is therefore concluded through simulation verification that the defined system and the parameter selection used will provide us with reasonable overall performance for tracking the relative position and orientation between the device and the target object. The next step, as detailed in Chapter 6, will be to independently validate the physical system implementation using a ground truth measurement system to determine the effectiveness and accuracy of the physical system implementation.

Chapter 6 SYSTEM VALIDATION

Now that the system has been designed and mathematically verified through simulation, validation is to be completed on the physical system to determine the system's performance and accuracy. As the video see-through mobile AR application measures the relative POSE between the BlackBerry Dev Alpha A and a target object, an independent measurement system is required to ensure that the relative POSE values produced by the BlackBerry device are accurate. In order to show the accuracy of this video see-through AR application, a measurement system with a higher level of precision and accuracy will be used to determine the *ground truth.* Here *ground truth* is defined as the set of measurements that are known to be much more accurate than the measurements generated by the BlackBerry Dev Alpha A device. This chapter details the validation approach and methods used to compare the ground truth data with the data collected by the BlackBerry. Additionally the results from physical experiments are presented for various target object motions along with a discussion of the results. Finally conclusions of the system validation experiments are provided.

6.1 General Approach

In order to validate the relative POSE and video see-through AR system on the BlackBerry Dev Alpha A device, a method is required to assess whether the relative POSE estimation computed by the device is correct to some small level of measurement uncertainty. The difficulty lies in the fact that the device collects relative POSE data between itself and the target object. Therefore some method is needed to independently validate this relative POSE data. An approach to validate the relative POSE data is to fix the position of the device and use a fixed position ground truth measurement system to measure the relative 3D motion of the target with respect to both the device and the ground truth measurement system. Then through some mathematical transformations, the motion of the target object relative to its starting position can be computed for both the device and also from the ground truth measurement system. A comparative analysis of the data can then be completed to validate the implemented physical system.

6.1.1 Ground Truth Measurement System Model

As previously noted in Chapter 3, the transformation of the j^{th} feature point in the object frame to the device's camera frame can be expressed by using the following homogeneous matrix transformation,

$$\begin{bmatrix} x_j^C \\ y_j^C \\ z_j^C \\ 1 \end{bmatrix} = T_O^C \begin{bmatrix} x_j^O \\ y_j^O \\ z_j^O \\ 1 \end{bmatrix} \tag{6.1}$$

where

$$T_O^C = \begin{bmatrix} & X_O^C \\ R_O^C & Y_O^C \\ & Z_O^C \\ 0_{1x3} & 1 \end{bmatrix} \tag{6.2}$$

Before proceeding further, a mathematical model describing the relationship from the object to the ground truth measurement system is required. The relative position of the object frame O with respect to the ground truth measurement system G can be defined as the vector,

$$P_G = \left[X_O^G, Y_O^G, Z_O^G\right]^T \tag{6.3}$$

The relative orientation of the object frame O with respect to the ground truth measurement system G is defined through the roll α, pitch β, and yaw γ angles and can be defined through a rotation matrix as follows,

$$R_O^G = R_{zC}(\alpha)R_{yC}(\beta)R_{xC}(\gamma) \tag{6.4}$$

where

$$R_O^G = \begin{bmatrix} C_\beta C_\gamma & S_\alpha S_\beta C_\gamma - C_\alpha S_\gamma & C_\alpha S_\beta C_\gamma + S_\alpha S_\gamma \\ C_\beta S_\gamma & S_\alpha S_\beta S_\gamma + C_\alpha C_\gamma & C_\alpha S_\beta S_\gamma - S_\alpha C_\gamma \\ -S_\beta & S_\alpha C_\beta & C_\alpha C_\beta \end{bmatrix}, where\ C_\alpha = cos\,\alpha, S_\alpha = sin\,\alpha\,, etc \quad (6.5)$$

Therefore a transformation of the j^{th} feature point in the object frame O with respect to the ground truth measurement system frame G can be written by the homogeneous transformation matrix as follows,

$$\begin{bmatrix} x_j^G \\ y_j^G \\ z_j^G \\ 1 \end{bmatrix} = T_O^G \begin{bmatrix} x_j^O \\ y_j^O \\ z_j^O \\ 1 \end{bmatrix} \quad (6.6)$$

where T_O^G is defined as follows,

$$T_O^G = \begin{bmatrix} & & X_O^G \\ & R_O^G & Y_O^G \\ & & Z_O^G \\ 0_{1x3} & & 1 \end{bmatrix} \quad (6.7)$$

6.1.2 Validation Approach

Next the case is considered where the device is in a fixed position. In this instance, the relative motion between the device and the target object is simply the dynamic motion of the target object itself. The initial static position of the target object prior to any motion and its coordinate frame O can be used to represent an effective world coordinate frame. Therefore the origin of the object frame O_O prior to any motion can also be defined as the origin of the world frame as follows,

$$W_O = O_O, at\ time\ t = 0 \quad (6.7)$$

Thus, for a dynamic motion of the target object assuming a fixed device position, the motion of the target object can also be expressed with respect to the origin of the world coordinate frame origin W_O, i.e.

from the target object's starting point. The relative POSE to a new location O_k at time step k from the world frame origin W_O can be determined by the expression in Equations 6.8 and 6.9 below,

$$\tilde{T}_{O_k}^{W_O} = \left(T_{W_O}^{C}\right)^{-1} T_{O_k}^{C}, for\ k = 0\ \ldots n\ total\ time\ steps \quad (6.8)$$

$$\bar{T}_{O_k}^{W_O} = \left(T_{W_O}^{G}\right)^{-1} T_{O_k}^{G}, for\ k = 0\ \ldots n\ total\ time\ steps \quad (6.9)$$

where the homogenous transformation matrix $\tilde{T}_{O_k}^{W_O}$ represents the relative POSE from the world frame origin W_O to some new location O_k for the device and the homogenous transformation matrix $\bar{T}_{O_k}^{W_O}$ represents the relative POSE from the world frame origin W_O to some new location O_k for the ground truth measurement system. As per the definition of the homogenous transformation matrix of O_k with respect to W_O, the homogenous transformation matrix $T_{O_k}^{W_O}$, is defined as follows,

$$T_{O_k}^{W_O} = \begin{bmatrix} & X_{O_k}^{W_O} \\ R_{O_k}^{W_O} & Y_{O_k}^{W_O} \\ & Z_{O_k}^{W_O} \\ 0_{1x3} & 1 \end{bmatrix} \quad (6.10)$$

where the three Cartesian elements $X_{O_k}^{W_O}$, $Y_{O_k}^{W_O}$, and $Z_{O_k}^{W_O}$ represent the change in position from W_O to O_k with respect to the object frame O as defined in Figure 4-1. Similarly the rotation matrix $R_{O_k}^{W_O}$ represents the $\mathcal{R}^3$ space rotation from W_O to O_k about the object frame O in the Z, Y, and X axes order.

Therefore successive points in the motion of the target object can be represented a vector of these homogenous transformation matrices as shown in Equation 6.11,

$$T_O^{W_O} = \begin{bmatrix} T_{O_1}^{W_O} & T_{O_2}^{W_O} & \ldots & T_{O_n}^{W_O} \end{bmatrix} \quad (6.11)$$

where the first element in the vector represents the change in POSE from W_O to O_1, the k^{th} element represents the change in POSE from W_O to O_k, etc. In this manner, a motion model of the target object for both of these two independent systems (device and ground truth measurement system) can be constructed.

This constructed motion model represents the homogeneous transformation matrix of the target object with respect to the world frame origin W_O for each time step. The six element relative POSE vector $W_O^{W_O}$ relative to the world frame origin W_O for each time step, as defined by Equation 6.12 below, can be constructed by decomposing the homogenous transformation matrix to determine the six element POSE vectors.

$$W_O^{W_O} = \begin{bmatrix} W_{O_1}^{W_O} & W_{O_2}^{W_O} & \dots & W_{O_n}^{W_O} \end{bmatrix} \tag{6.12}$$

where the relative POSE vector generated by the device is defined as,

$$\widetilde{W}_{O_j}^{W_O} = \begin{bmatrix} X_{O_j}^{W_O} & Y_{O_j}^{W_O} & Z_{O_j}^{W_O} & \emptyset_{O_j}^{W_O} & \theta_{O_j}^{W_O} & \psi_{O_j}^{W_O} \end{bmatrix}^T \tag{6.13}$$

and the relative POSE vector generated by the ground truth measurement system is defined as,

$$\bar{W}_{O_j}^{W_O} = \begin{bmatrix} X_{O_j}^{W_O} & Y_{O_j}^{W_O} & Z_{O_j}^{W_O} & \alpha_{O_j}^{W_O} & \beta_{O_j}^{W_O} & \gamma_{O_j}^{W_O} \end{bmatrix}^T \tag{6.14}$$

A comparison can then be completed between the two relative POSE vectors from the two systems and therefore a validation of the results generated by the device can be completed as illustrated in Figure 6-1 below.

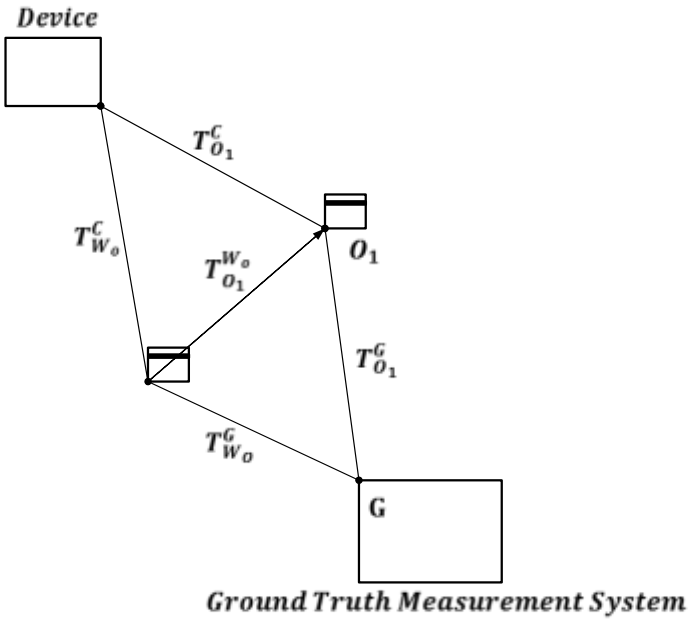

Figure 6-1: Relative motion of the target object from W_O to O_1 can be expressed through $T_{O_1}^{W_o}$.

6.2 Test Setup and Procedure

6.2.1 Ground Truth Measurement System

In order to complete the validation activities a suitable ground truth measurement system is required. There are various criteria in selecting the ground truth measurement system and these are listed below:

a. 6DOF real time tracking for multiple markers
b. Sample rate larger than the device (30 fps)
c. High level of precision and accuracy
d. Relatively low system cost

As a result of these criteria, the Natural Point OptiTrack V100:R2 system was selected due to its ability to complete real time marker tracking, its frame rate of 100 fps, and its ability to track markers down to sub-millimeter movements with repeatable accuracy [58].

6.2.2 Validation Experimental Test Procedure

The following test procedure was used to complete the validation experiments to show the effectiveness of the system model implementation on the BlackBerry Dev Alpha A device:

a. Mount four OptiTrack markers for each target object as detailed in Figure 6-2 and locate such that two markers lie along the X_O axis of the object frame, two markers lie along the Y_O axis of the object frame, and the intersection of these markers occur at the origin of the object frame O_O.

b. Setup physical test system as shown in Figure 6-3 below noting that BlackBerry Dev Alpha A device position is fixed to a tri-pod and OptiTrack cameras are set in static position.

c. Setup BlackBerry Dev Alpha A device such that feature points and relative POSE are output as debug statements to the console of the BlackBerry 10 Native SW Development Kit environment.

d. Calibrate OptiTrack system using OptiTrack calibration wand.

e. Commence recording motion on the OptiTrack system and also initiate ImageProc application on the BlackBerry Dev Alpha A device. Recorded motion will be the time increments for each time step, and the relative POSE estimates $W = [X, Y, Z, \emptyset, \theta, \psi]^T$.

f. With the BlackBerry Dev Alpha A device kept in a static position, move target objects (*1 to N*) in various motions in the workspace.

g. Stop recording on OptiTrack system and also stop ImageProc application on the BlackBerry Dev Alpha A device.

h. Store OptiTrack recorded data and BlackBerry Dev Alpha A data to files in a readable format for MATLAB.

i. Import data in MATLAB and complete required analytical analysis.

The test equipment used is detailed in Appendix E and the target object models used are provided in Appendix F.

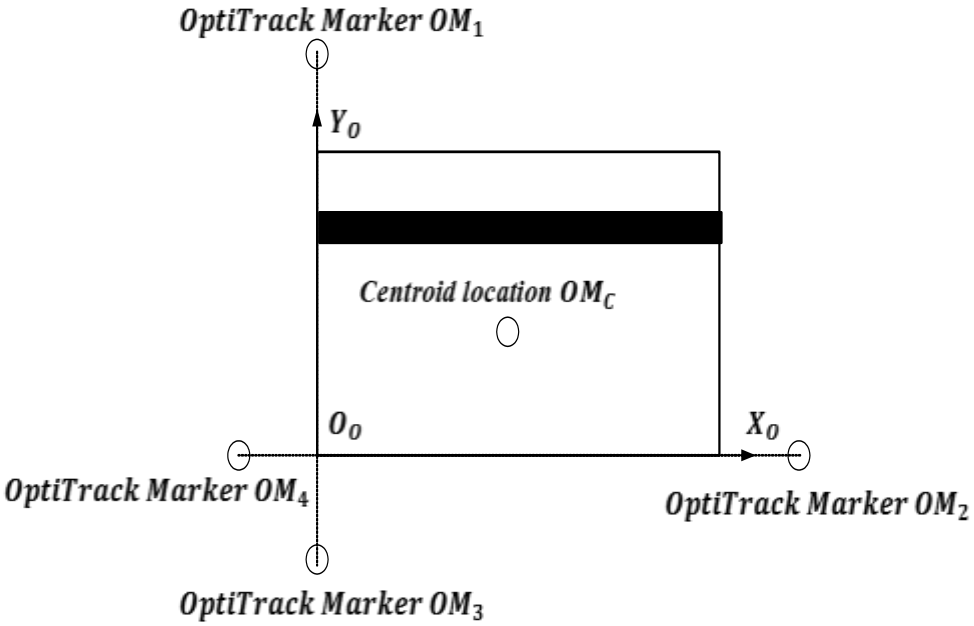

Figure 6-2: Depiction of OptiTrack markers including a representation of the OptiTrack marker Centroid relative to target object and its associated frame.

Table 6-1: OptiTrack Marker Positioning For Object Models

Target Object	Distance Between Markers – X direction	Distance Between Markers – Y direction
Credit Card Model #1	19.6 cm	13.5cm
Credit Card Model #2	20.7cm	13.1cm

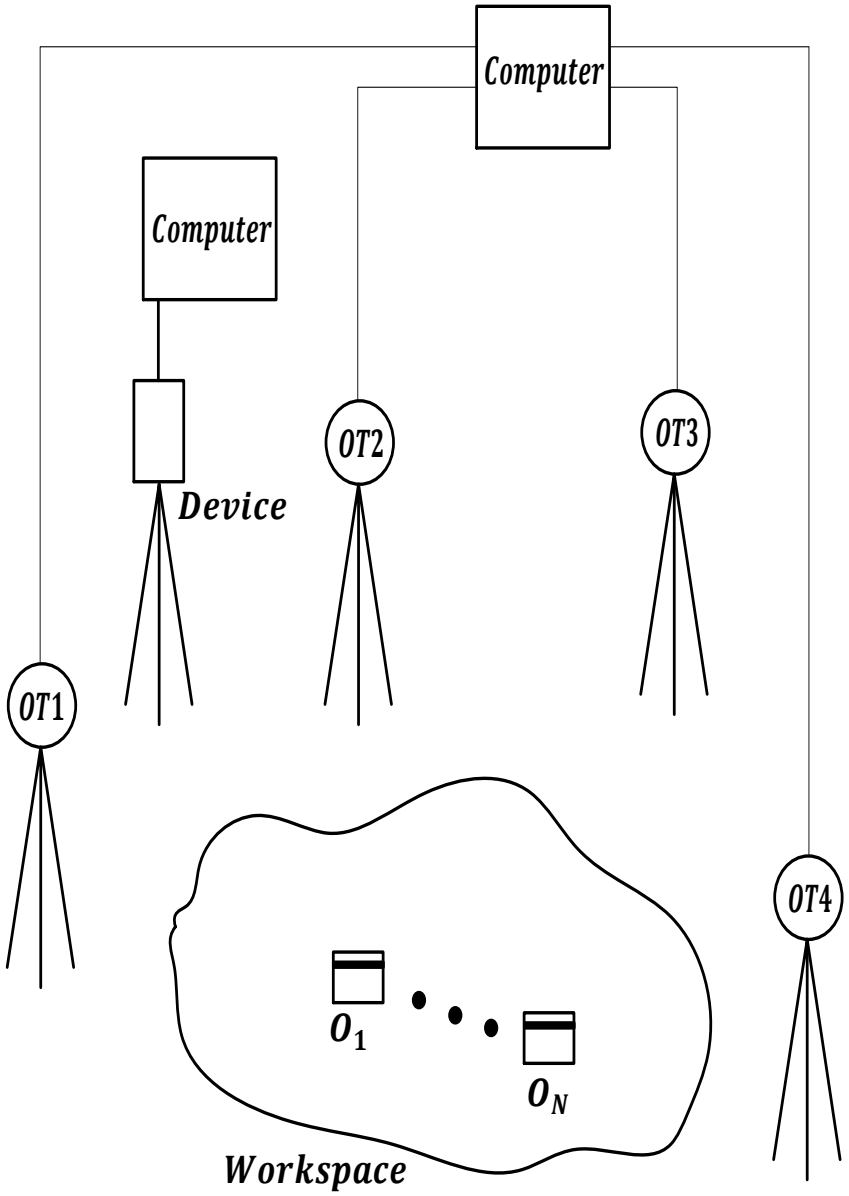

Figure 6-3: Test setup used in completing the validation experiments.

6.3 Procedure for Analysis of Results

6.3.1 Procedure for Device Analysis

Once the relative POSE data is collected from both the device and the ground truth measurement system, the data must be converted into the required homogenous transformation matrices. For the device, this can be simply done by reading in the relative POSE data generated by the device for each time step and creating the homogeneous transformation matrix $\tilde{T}_{O_k}^{W_O}$ as per Equation 6.8 along with a vector of transformation matrices $\tilde{T}_{O}^{W_O}$ as per Equation 6.11 and the relative POSE vector for the device $\widetilde{W}_{O}^{W_O}$ as per Equations 6.12 and 6.13.

$$\tilde{T}_{O}^{W_O} = \left[\tilde{T}_{O_1}^{W_O} \quad \tilde{T}_{O_2}^{W_O} \quad \ldots \quad \tilde{T}_{O_n}^{W_O}\right] \tag{6.15}$$

$$\widetilde{W}_{O}^{W_O} = \left[\widetilde{W}_{O_1}^{W_O} \quad \widetilde{W}_{O_2}^{W_O} \quad \ldots \quad \widetilde{W}_{O_n}^{W_O}\right] \tag{6.16}$$

6.3.2 Procedure for Analysis of Ground Truth Measurement System Data

As noted above, the data collected from the ground truth measurement system is with respect to the Centroid OptiTrack markers as shown in Figure 6-2 above. At each time step, the OptiTrack measurement system records the following data:

a. The six degree of freedom relative position and orientation of the OptiTrack markers' Centroid $\mathrm{OM_C}$ with respect to the OptiTrack world frame. The Centroid is defined as the geometric center of the four OptiTrack markers and is different than the object frame origin. It is noted that OptiTrack Centroid OM_C is at a statically fixed offset from the object frame origin O_O

b. OptiTrack Marker locations

In order to compare the relative motion of the object from its initial starting position, a method is required to transform the data from the OptiTrack system to construct the required homogeneous transformation matrix $\bar{T}_{O_k}^{W_O}$. In order to complete this calculation, the transformation from the object frame with respect to the ground truth measurement system frame T_O^G needs to be determined.

Different methods for constructing the homogeneous transformation matrix T_O^G were investigated. One possible approach is to construct the object frame by taking the intersection of the two vectors $\overrightarrow{OM_2OM_4}$ and $\overrightarrow{OM_1OM_3}$ created by the marker positions to determine the object frame origin O_O. The object frame can then be constructed by finding the unit vectors $\hat{x}_o$, $\hat{y}_o$, and $\hat{z}_o$, which can be used to construct the rotation matrix of the object frame with respect to the marker Centroid. The homogenous transformation matrix of the object frame with respect to the marker Centroid frame can then be constructed using this rotation matrix and the Cartesian offset between the Centroid location OM_C of the object frame origin O_O. Therefore, the required homogeneous transformation matrix $T_{O_k}^G$ can be constructed for each time step as follows,

$$T_{O_k}^G = T_{Centroid_k}^G T_O^{Centroid} \tag{6.17}$$

However, some error in the OptiTrack measurements was observed due to the system confusing the trackable marker set, resulting in bad correspondences of the data for some of the OptiTrack marker data points. Therefore, another method was pursued to construct the required homogeneous transformation matrix T_O^G. As previously noted, the OptiTrack marker Centroid location OM_C is statically offset from the object frame origin O_O. Therefore as per the *general principal of relativity* [59], the object frame origin O_O, the four OptiTrack markers, and the OptiTrack marker Centroid location OM_C will all experience the same relative motion for any given 3D motion of the target object since they are all statically located within a common inertial reference frame. However, relativity theory also requires that the spatial axes of the object and OptiTrack marker Centroid reference frames exactly coincide with each other.

Therefore, assuming that the spatial axes of the object frame O coincides with that of the OptiTrack marker Centroid frame, a modified expression for the homogenous transformation matrix T_O^G for each time step k can be written as follows,

$$T_{O_k}^G = T_{Centroid_k}^G \tag{6.18}$$

where the homogeneous transformation matrix $T^G_{Centroid_k}$ of the Centroid with respect to the ground truth measurement system is defined by Equation 6.19, as follows,

$$T^G_{Centroid_k} = \begin{bmatrix} & & X^G_{Centroid_k} \\ R^G_{Centroid_k} & & Y^G_{Centroid_k} \\ & & Z^G_{Centroid_k} \\ 0_{1x3} & & 1 \end{bmatrix} \tag{6.19}$$

As noted above, the OptiTrack system provides all of the required information at each time step to construct the homogenous transformation matrix $T^G_{Centroid_k}$, i.e. the six degree of freedom relative position and orientation of the OptiTrack markers' Centroid OM_C with respect to the OptiTrack world frame. Since the required measurements for the ground truth measurement system will not be with respect to the object frame origin, a new effective world frame origin is required for just the ground truth measurement system. As the target object and its inertial reference frame was held at rest for the initial phase of the motion experiments, the world frame origin for the OptiTrack system can be defined as the initial starting position of the Centroid marker location,

$$U_O = OM_{C_O},\ at\ time\ t = 0 \tag{6.20}$$

The update to the world frame origin reference for the ground truth measurement system therefore requires minor updates to the various expressions needed to complete the validation. An updated expression for $\bar{T}^{W_O}_{O_k}$ based on the ground truth measurement system's world origin U_O, i.e. the OptiTrack marker Centroid initial static position, can be written as follows,

$$\bar{T}^{U_O}_{O_k} = \left(T^G_{U_O}\right)^{-1} T^G_{Centroid_k}, for\ k = 0\ \ldots p\ total\ time\ steps \tag{6.21}$$

Similarly the vector of the homogeneous transformation matrices and the relative POSE vector of the object with respect to the ground truth measurement system's world origin U_O, can be written as follows,

$$\bar{T}^{U_O}_{O} = \left[\bar{T}^{U_O}_{O_1} \quad \bar{T}^{U_O}_{O_2} \quad \ldots \quad \bar{T}^{U_O}_{O_p}\right] \tag{6.22}$$

$$\bar{W}_O^{U_O} = \left[\bar{W}_{O_1}^{U_O} \quad \bar{W}_{O_2}^{U_O} \quad \ldots \quad \bar{W}_{O_p}^{U_O}\right] \tag{6.23}$$

It was determined through experimentation that the spatial axes of the object frame O and the Centroid frame are not exactly aligned. The relationship between the two frames is shown in Figure 6-4. A correction to the OptiTrack marker Centroid frame, as detailed in Table 6-2 below, was applied so that the spatial axes of the object frame and the Centroid frame exactly coincide with each other. Once the spatial axes transformation was completed, the homogenous transformation matrix $T_{Centroid_k}^{G}$ of the Centroid with respect to the ground truth measurement system can be constructed and no changes to the above expressions in Equations 6.19, 6.21, 6.22, and 6.23 are required.

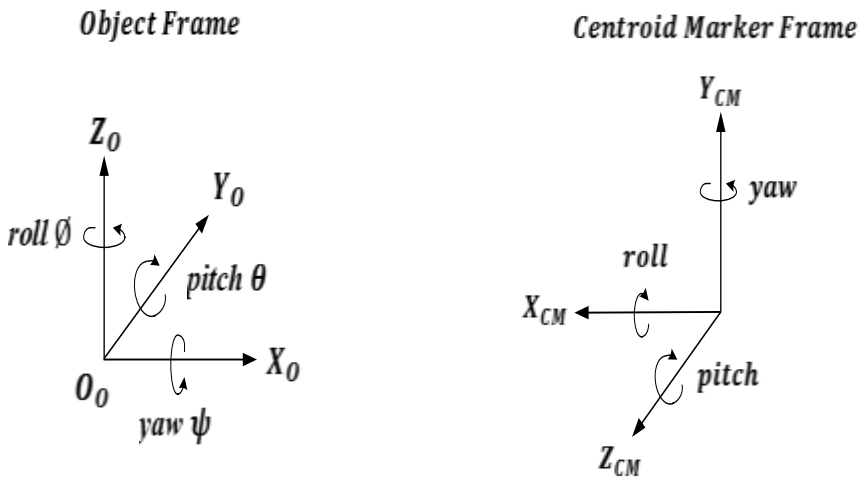

Figure 6-4: Spatial axes frame representation for object frame O and OptiTrack Marker Centroid frame noting that pitch is negated in OptiTrack system.

Table 6-2: Object Frame to Centroid Frame Conversion Summary

Axis	Object Frame	OptiTrack Centroid Frame
X	X	$-X$
Y	Y	$-Z$
Z	Z	Y
$roll\ about\ Z\ (\emptyset)$	$roll\ about\ Z\ (\emptyset)$	yaw
$pitch\ about\ Y\ (\theta)$	$pitch\ about\ Y\ (\theta)$	$pitch$
$yaw\ about\ X\ (\psi)$	$yaw\ about\ X\ (\psi)$	$-roll$

6.3.3 Limitations and Assumptions

When completing the analysis between the device and the ground truth measurement system, comparison is required between the relative POSE vector for the device (Equation 6.16) and the relative POSE vector for the ground truth measurement system (Equation 6.23). The two equations are restated below for quick reference and can be constructed based on the measurement data collected.

$$\widetilde{W}_O^{W_O} = \left[\widetilde{W}_{O_1}^{W_O} \quad \widetilde{W}_{O_2}^{W_O} \quad \dots \quad \widetilde{W}_{O_n}^{W_O}\right] \tag{6.16}$$

$$\bar{W}_O^{U_O} = \left[\bar{W}_{O_1}^{U_O} \quad \bar{W}_{O_2}^{U_O} \quad \dots \quad \bar{W}_{O_p}^{U_O}\right] \tag{6.23}$$

There are several additional limitations that arise from the physical system and the experimental test setup. A quick summary of these limitations are listed below. First of all, it is noted that there are a different number of time steps for each measurement system, i.e. n time steps for the device and p time steps for the ground truth measurement system. Secondly, the frame/sampling rate of the device varies between 20Hz and 30Hz compared to a 100Hz sampling rate of the OptiTrack system. Additionally, the two measurement systems are not directly coupled and there is no trigger to synchronize the start and end for the two sets of measurements, as both systems are manually operated. Finally some error in the OptiTrack measurements was observed due to the system confusing the trackable marker set resulting in bad correspondences of the data for some data points.

To account for the limitation of the two measurement systems not being synchronized, the relative POSE vectors $\widetilde{W}_O^{Wo}$ and $\bar{W}_O^{Uo}$ need to be manually aligned. The data alignment can be completed by applying an offset to the device measurement data set. As the initial and final positions were kept static, application of the offset is deemed as a feasible and reasonable approach. It should be noted that application of this manual time offset or shift will induce some level of error to the calculations. Furthermore, the duration recorded also differs for each system. Once the static time offset is applied to the device relative POSE vector data, both the device and ground truth relative POSE vector data sets will be truncated to ensure that the same span and duration of data is available for comparison.

Additionally, as the sampling rates between the device and the OptiTrack systems differ, the data points for the relative POSE vectors listed in Equations 6.16 and 6.23 cannot be directly compared. As the frequency of measurements is at least three times higher on the OptiTrack system, a simple interpolation of the OptiTrack data will be completed and is assumed to be suitable for comparison. It is noted that this interpolation will be completed after the time offset and truncation is applied.

For the device, at time step k, the relative POSE vector of the object from its initial position is $\widetilde{W}_{O_k}^{Wo}$. The data at device time step k corresponds to some physical measurement time denoted as Ω. With respect to the physical measurement time Ω, locate the OptiTrack relative POSE vector values that correspond to the largest time value just less than Ω and the smallest time value just greater than Ω. For simplicity, these two ground truth measurement time steps will be denoted as τ and $\tau + 1$ where,

$$\tau \leq \Omega \leq \tau + 1 \tag{6.24}$$

The OptiTrack data can then be interpolated through the following definition to create a corresponding a target object motion POSE vector $\bar{\bar{W}}_{O_k}^{Uo}$ for device time step k and also for each $k \in \{1 \dots n\}$ device time steps as follows,

$$\bar{\bar{W}}_{O_k}^{Uo} = \frac{\bar{W}_{O_\tau}^{Uo} + \bar{W}_{O_{\tau+1}}^{Uo}}{2}, for\ time\ \ \Omega \text{ seconds} \tag{6.25}$$

$$\bar{\bar{W}}_O^{Uo} = \left[\bar{\bar{W}}_{O_1}^{Uo} \quad \bar{\bar{W}}_{O_2}^{Uo} \quad \dots \quad \bar{\bar{W}}_{O_n}^{Uo}\right] \tag{6.26}$$

In order to measure the performance of the system, the absolute error can be constructed for each $k \in \{1 \dots n\}$ device time steps by directly subtracting the device generated target object motion POSE vector from the interpolated target object motion POSE vector of the ground truth measurement system,

$$W\_Error_k = \bar{\bar{W}}_{O_k}^{U_O} - \widetilde{W}_{O_k}^{W_O} \tag{6.27}$$

For the case of error in the raw OptiTrack measurements due to the system confusing the trackable marker set, these outliers will be manually removed from the raw data set. While it is unfortunate and not desired to adjust the ground truth measurement system data, these outliers create multiplicative error in the resulting relative POSE vectors and need to be handled. As the sampling frequency of the ground truth measurement system is at least three times larger than that of the device, removing these outlier data points is therefore viewed as acceptable since there is a tolerance to reducing samples without impacting the results.

6.4 Experimental Test Results

Physical experiments were conducted keeping both the device and the ground truth measurement system static and applying a smooth motion to the target object(s). Various motions were completed and the following five particular cases will be detailed as follows:

a. Single object motion in object frame Z direction
b. Single object motion in random direction – case 1
c. Single object motion in random direction – case 2
d. Single object motion in random direction – case 3
e. Two object motion in random directions

The experimental results are detailed in Figures 6-5 through 6-16 below and a summary of the results is detailed in Tables 6-3 for the single object motion cases and 6-4 below for the two object motion case.

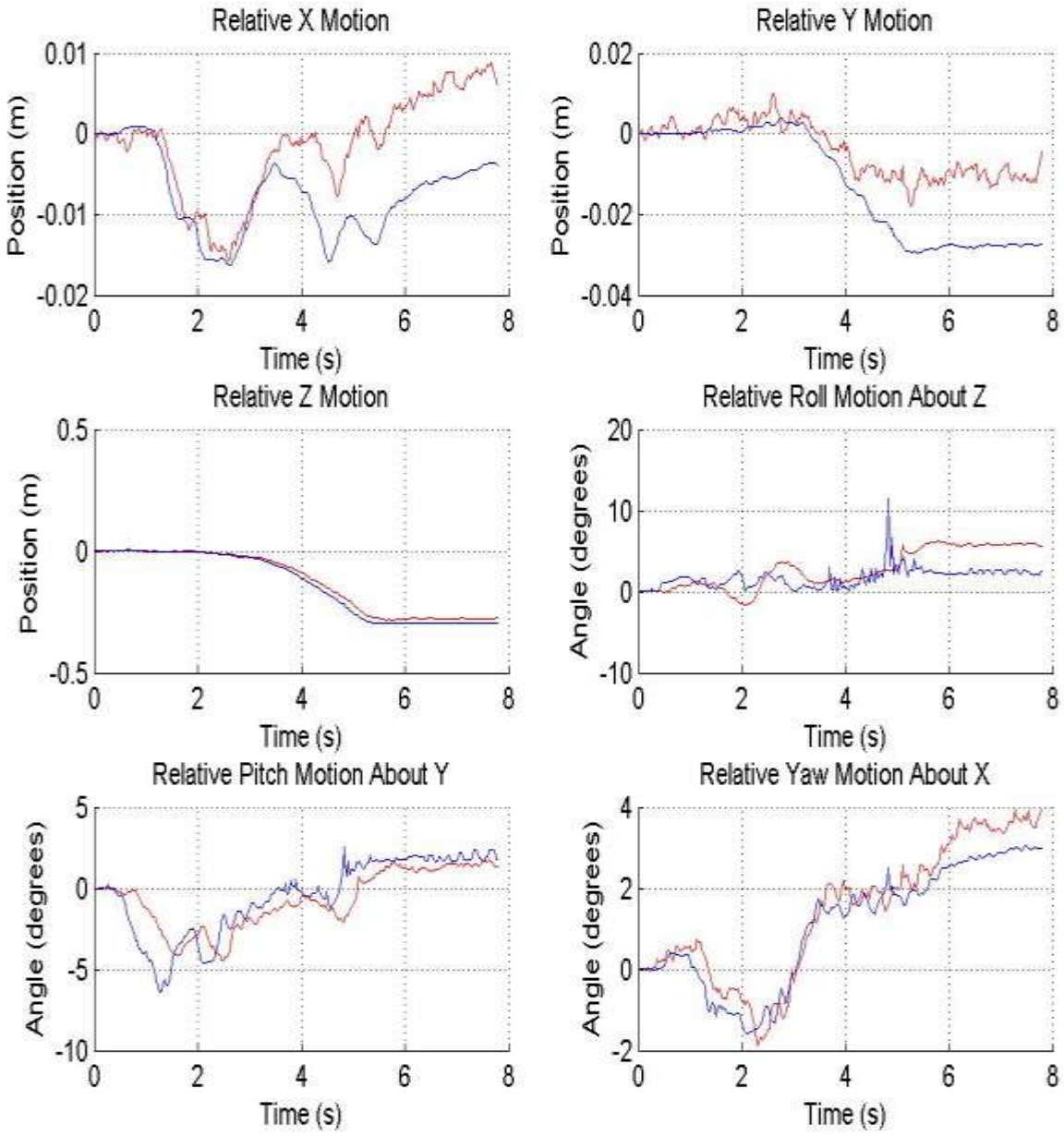

Figure 6-5: Depiction of the target object motion relative to its initial position, as measured by the ground truth measurement system $\bar{\bar{W}}_{O_k}^{U_O}$ (blue) and the device $\widetilde{W}_{O_k}^{W_O}$ (red) for a *Z*-axis motion case.

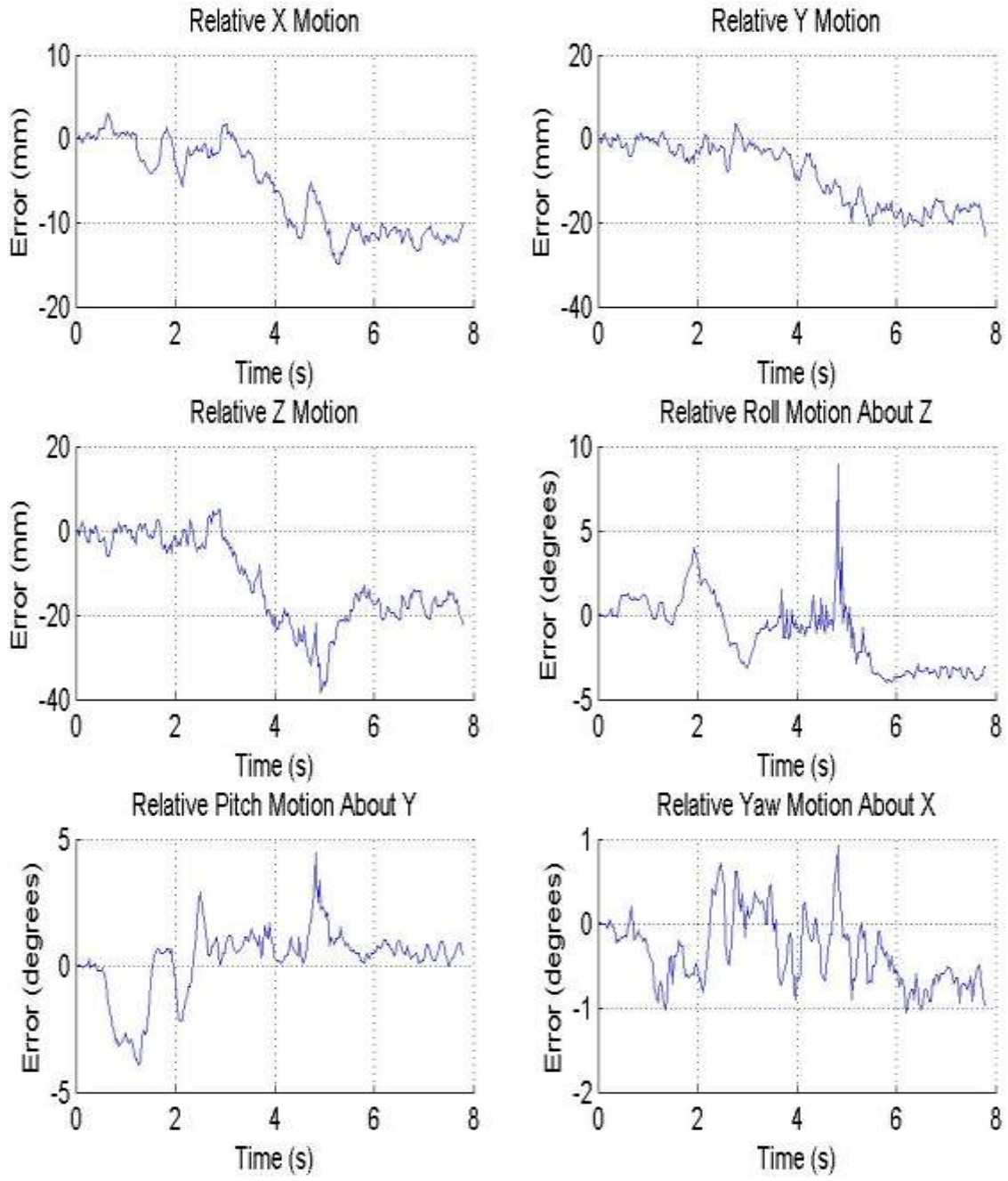

Figure 6-6: Absolute error calculation W_Error_k for the *Z*-axis motion case detailed in Figure 6-5.

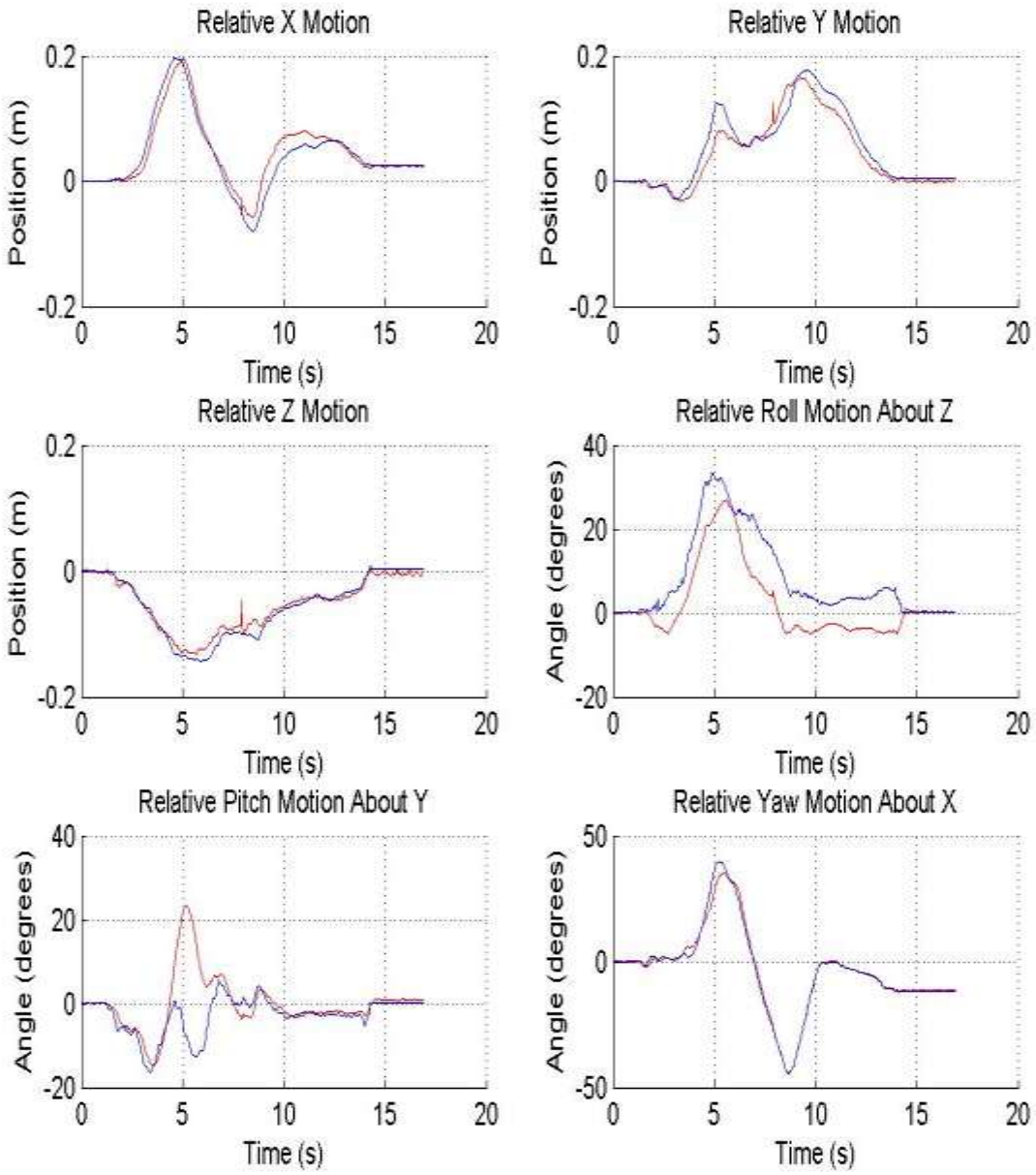

Figure 6-7: Depiction of the target object motion relative to its initial position, as measured by the ground truth measurement system $\bar{\bar{W}}_{O_k}^{U_O}$ (blue) and the device $\tilde{W}_{O_k}^{W_O}$ (red) for the first random motion case.

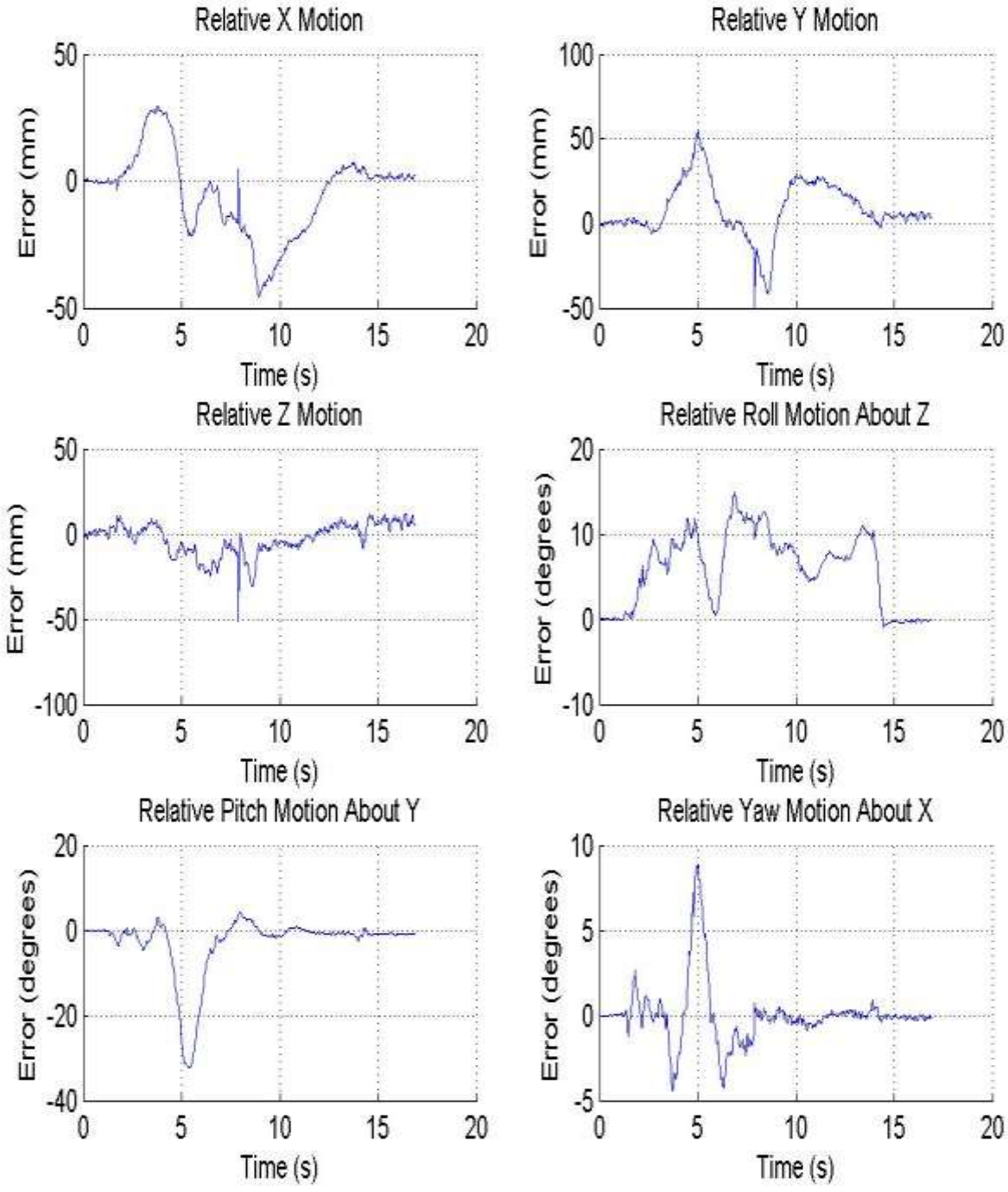

Figure 6-8: Absolute error calculation W_Error_k for the first random motion case detailed in Figure 6-7.

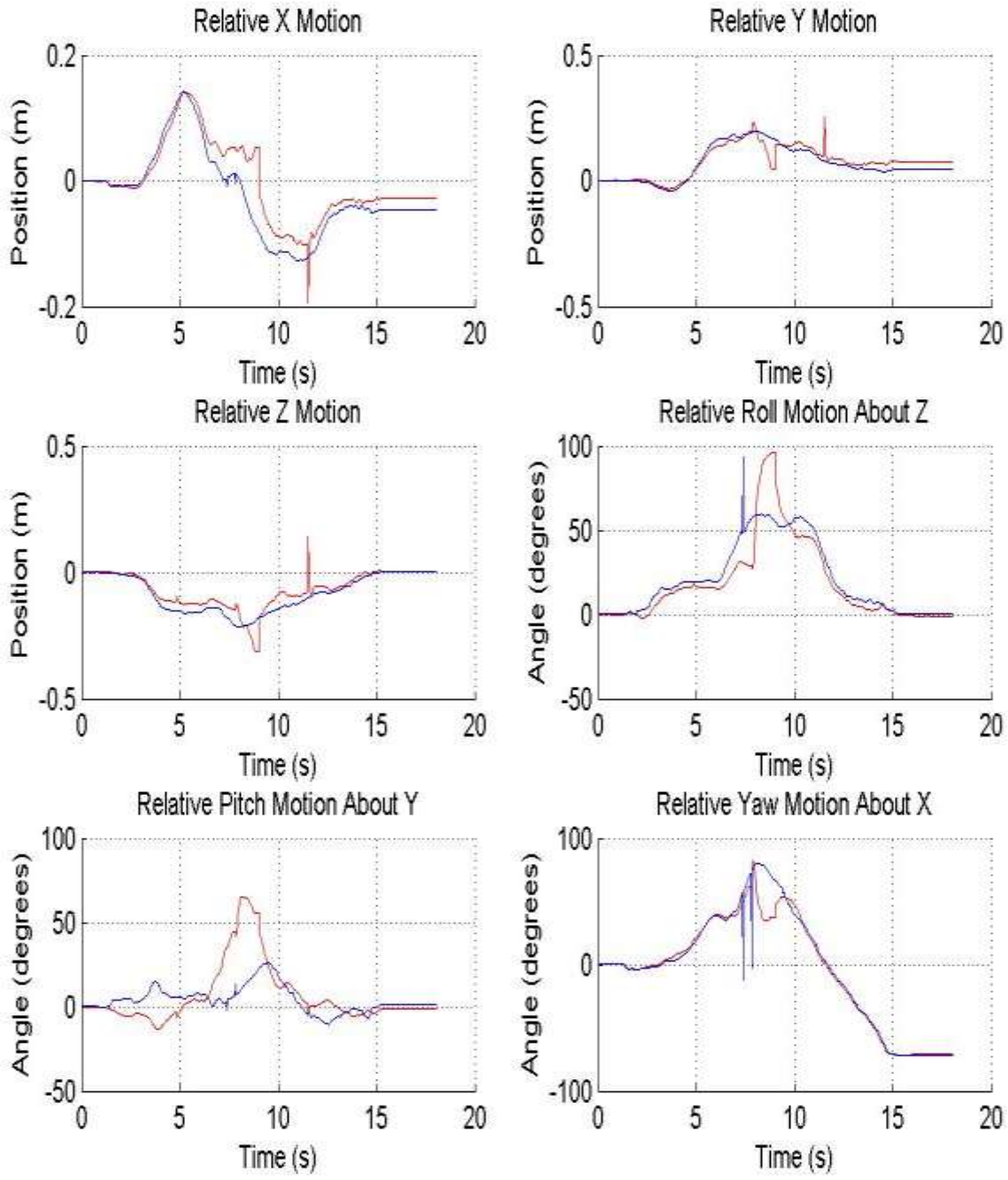

Figure 6-9: Depiction of the target object motion relative to its initial position, as measured by the ground truth measurement system $\bar{\bar{W}}_{O_k}^{U_O}$ (blue) and the device $\tilde{W}_{O_k}^{W_O}$ (red) for a second random motion case.

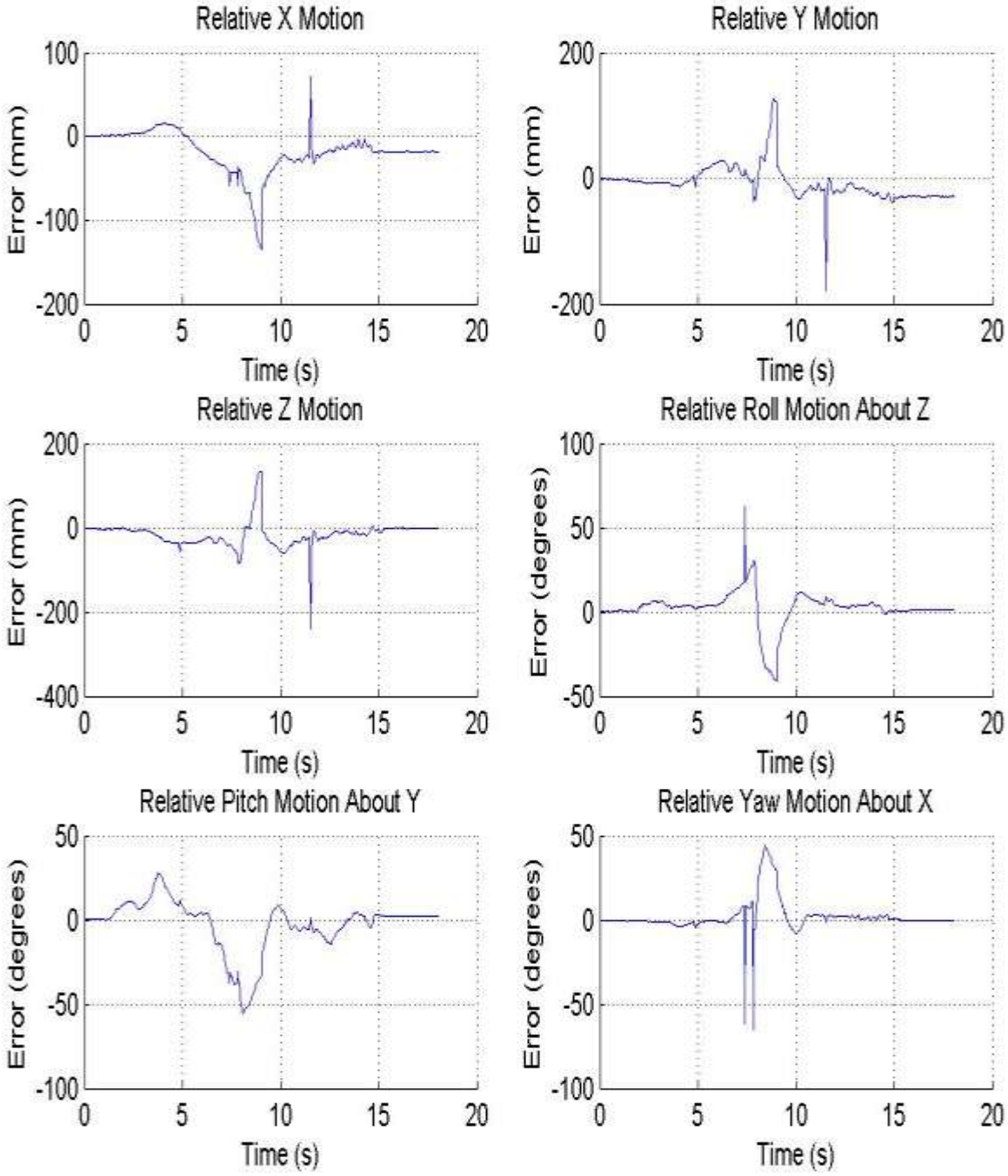

Figure 6-10: Absolute error calculation W_Error_k for the second random motion case detailed in Figure 6-9.

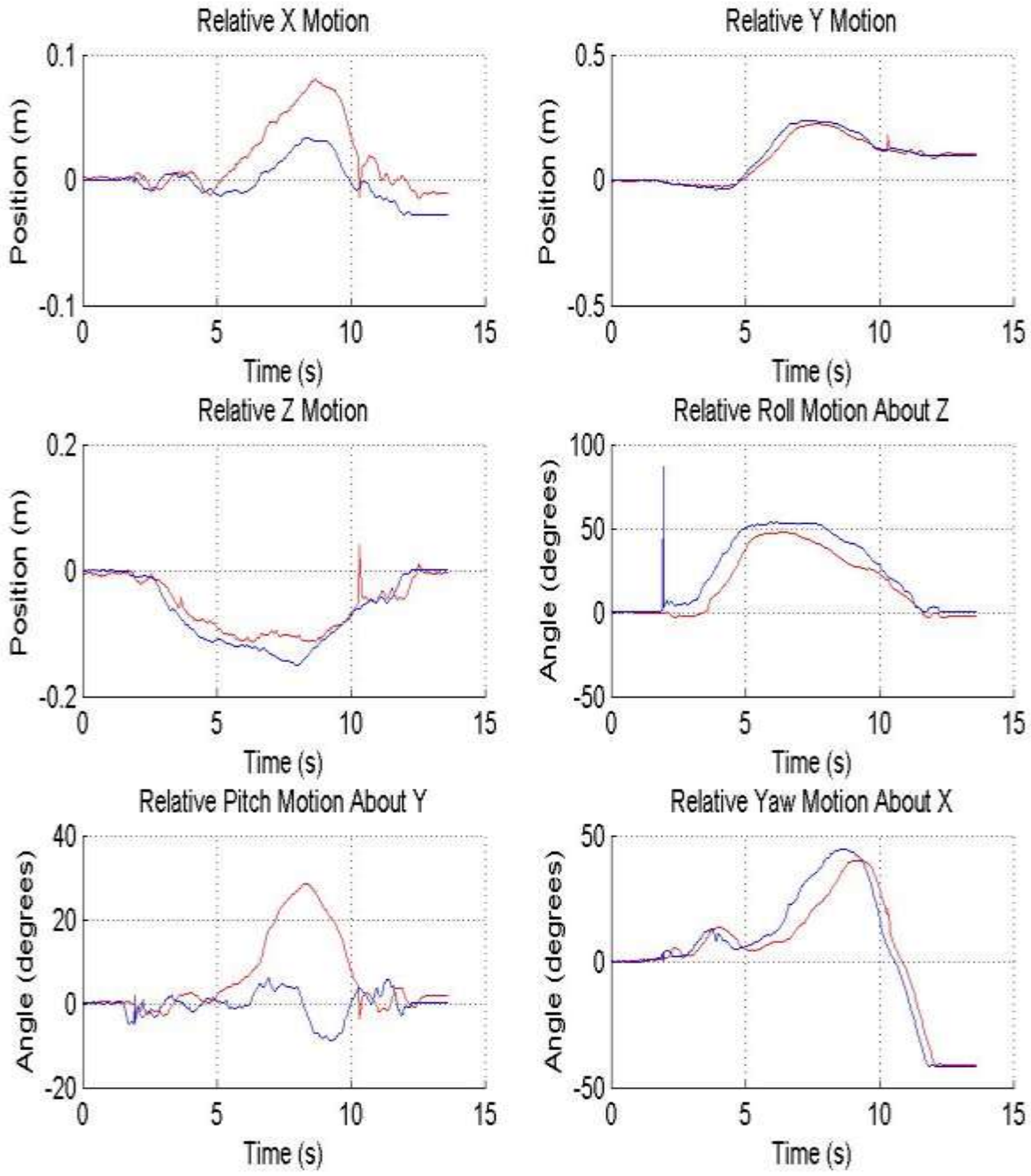

Figure 6-11: Depiction of the target object motion relative to its initial position, as measured by the ground truth measurement system $\bar{\bar{W}}_{O_k}^{U_O}$ (blue) and the device $\tilde{W}_{O_k}^{W_O}$ (red) for a third random motion case.

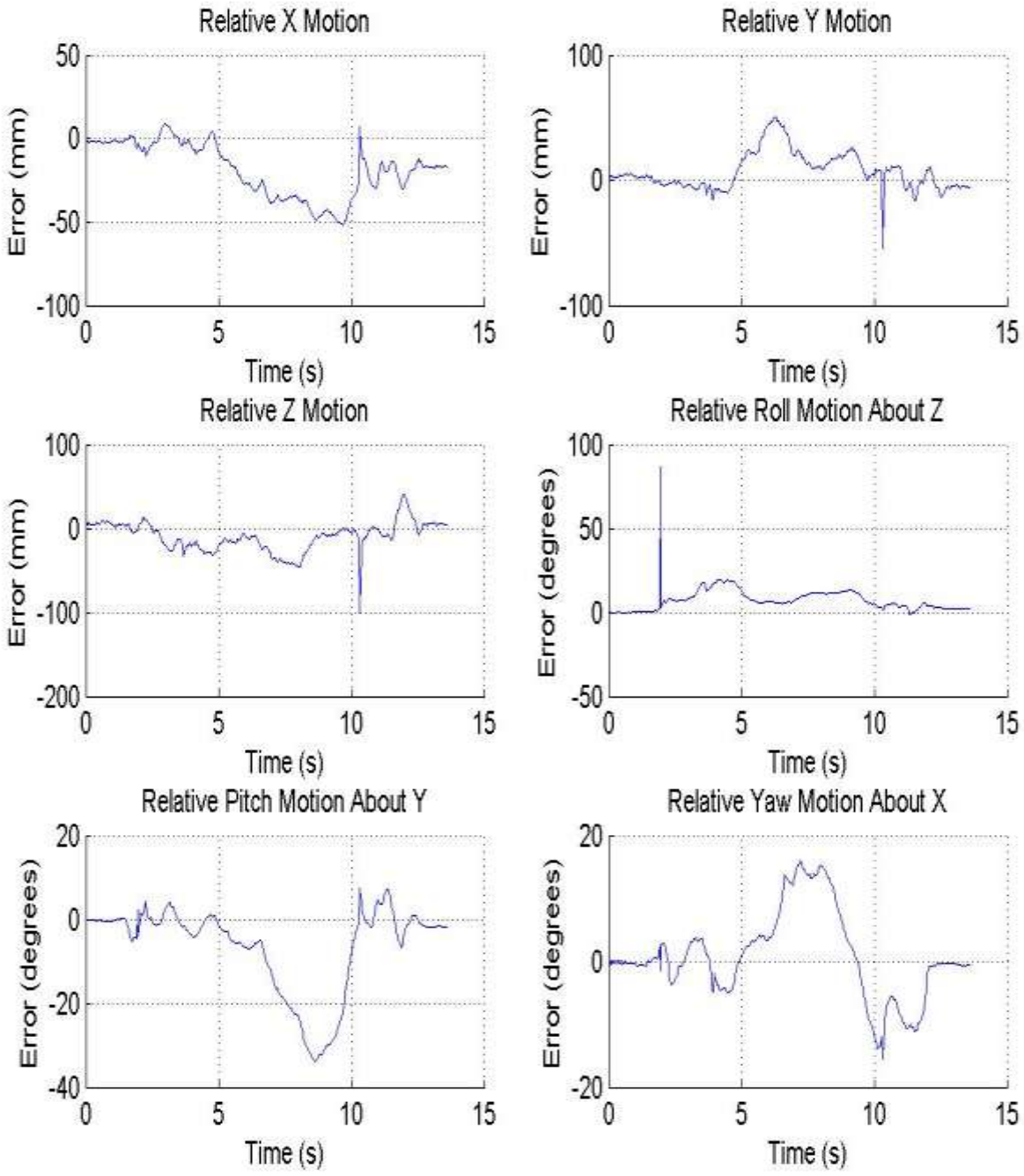

Figure 6-12: Absolute error calculation W_Error_k for the third random motion case detailed in Figure 6-11.

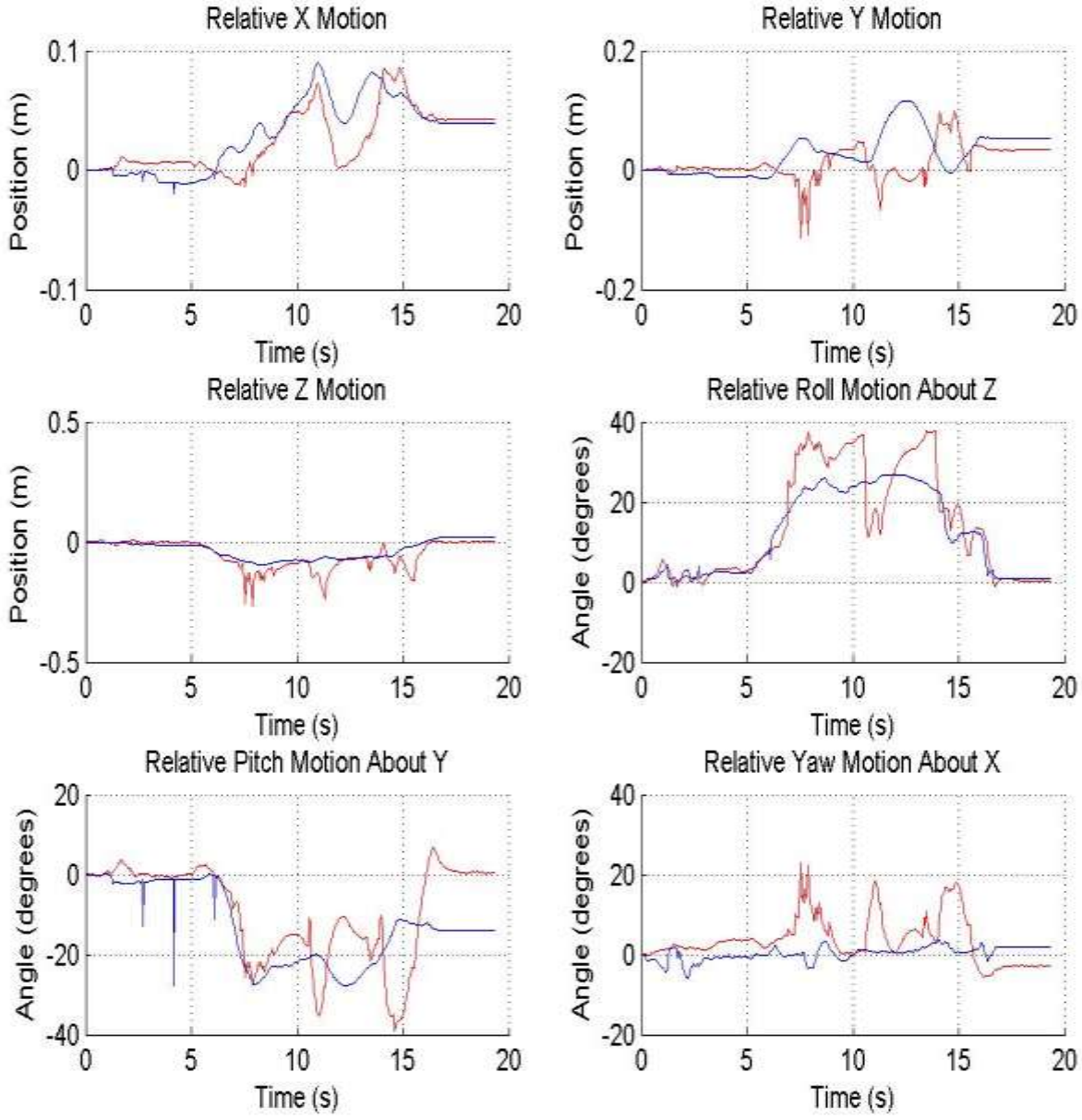

Figure 6-13: Depiction of the first target object motion relative to its initial position, as measured by the ground truth measurement system $\bar{\bar{W}}_{O_k}^{Uo}$ (blue) and the device $\widetilde{W}_{O_k}^{Wo}$ (red) for a two object random motion case.

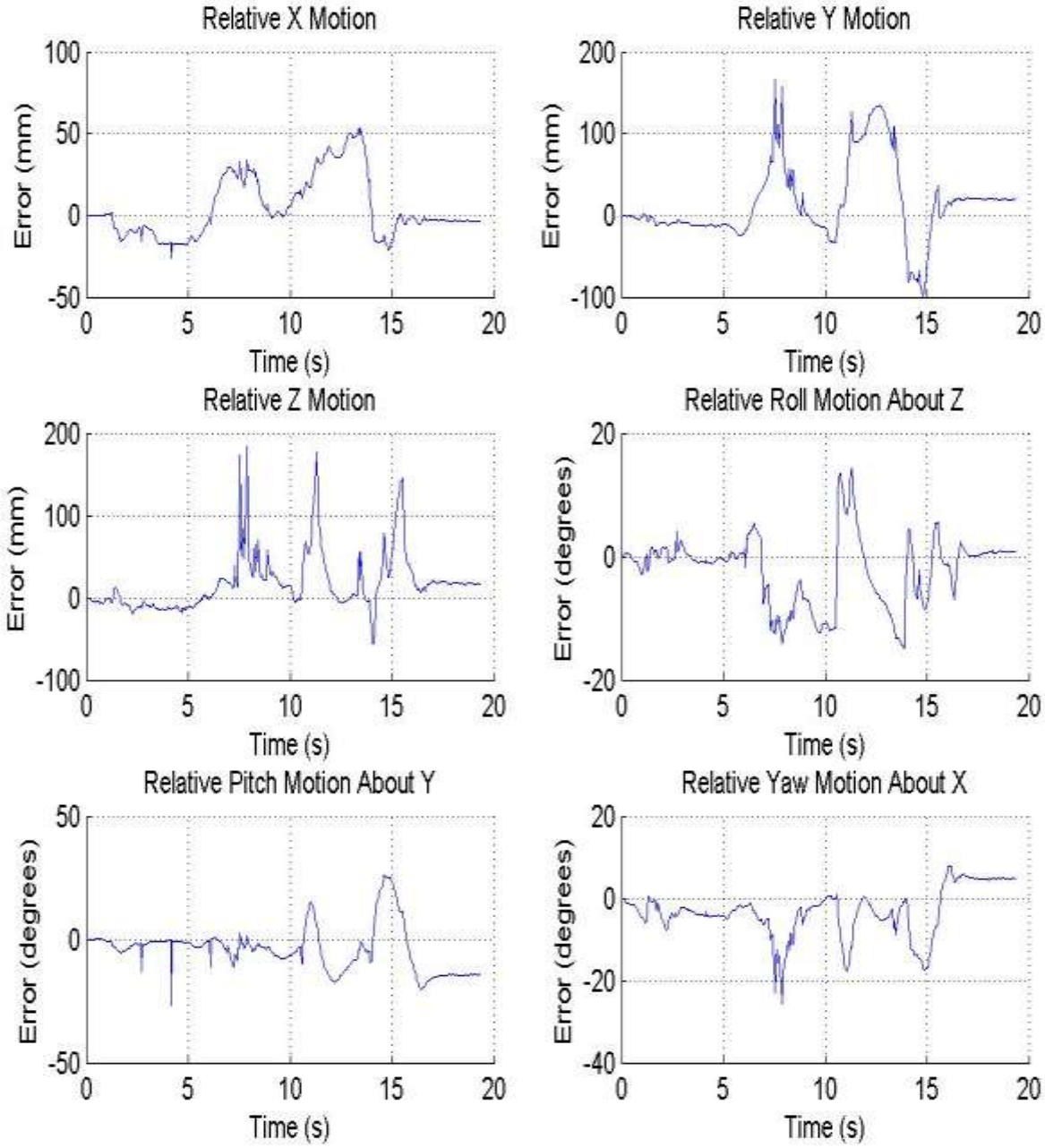

Figure 6-14: Absolute error calculation W_Error_k for the first object in a two object random motion case as detailed in Figure 6-13.

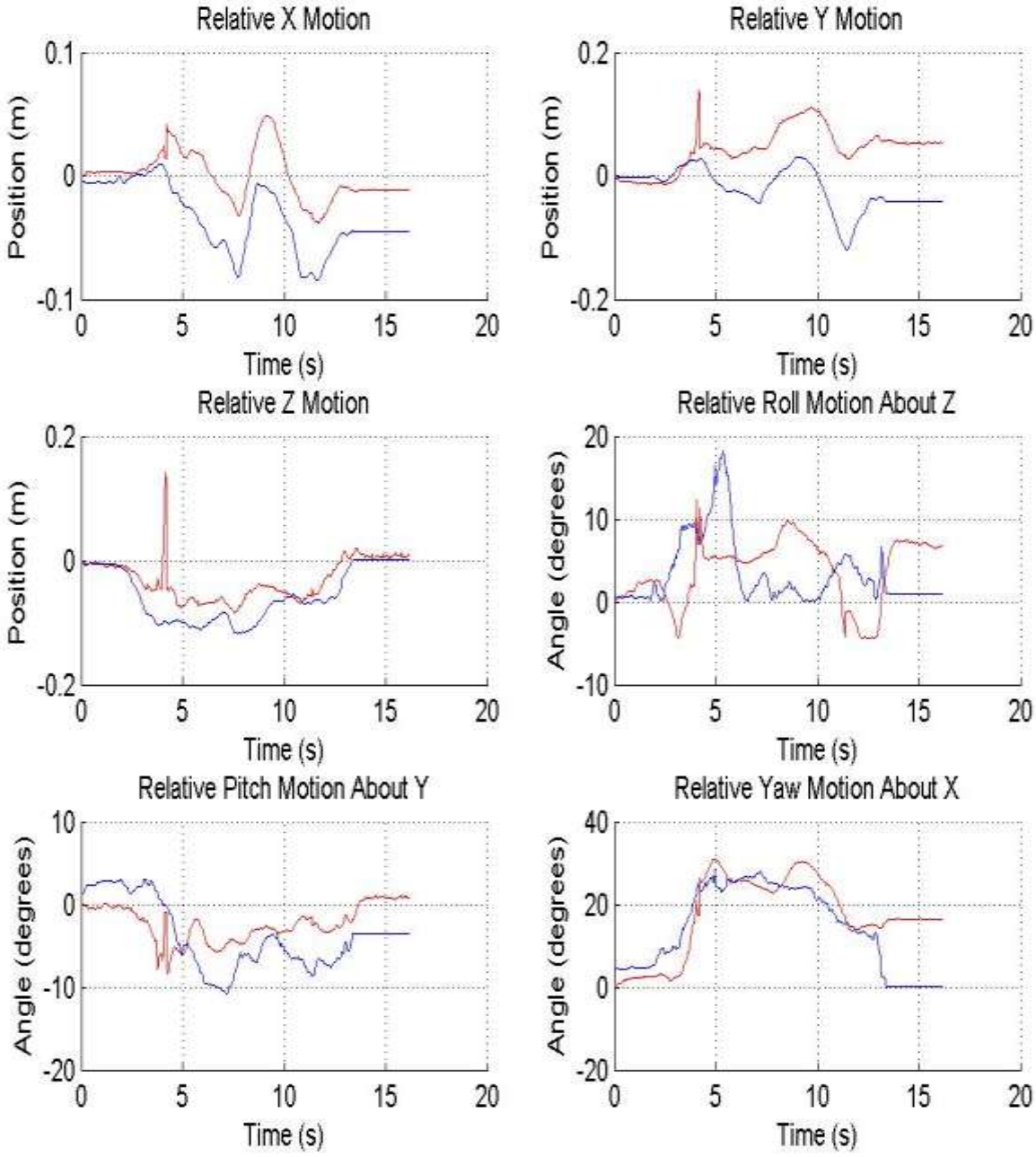

Figure 6-15: Depiction of the second target object motion relative to its initial position, as measured by the ground truth measurement system $\bar{\bar{W}}_{O_k}^{U_O}$ (blue) and the device $\widetilde{W}_{O_k}^{W_O}$ (red) for a two object random motion case.

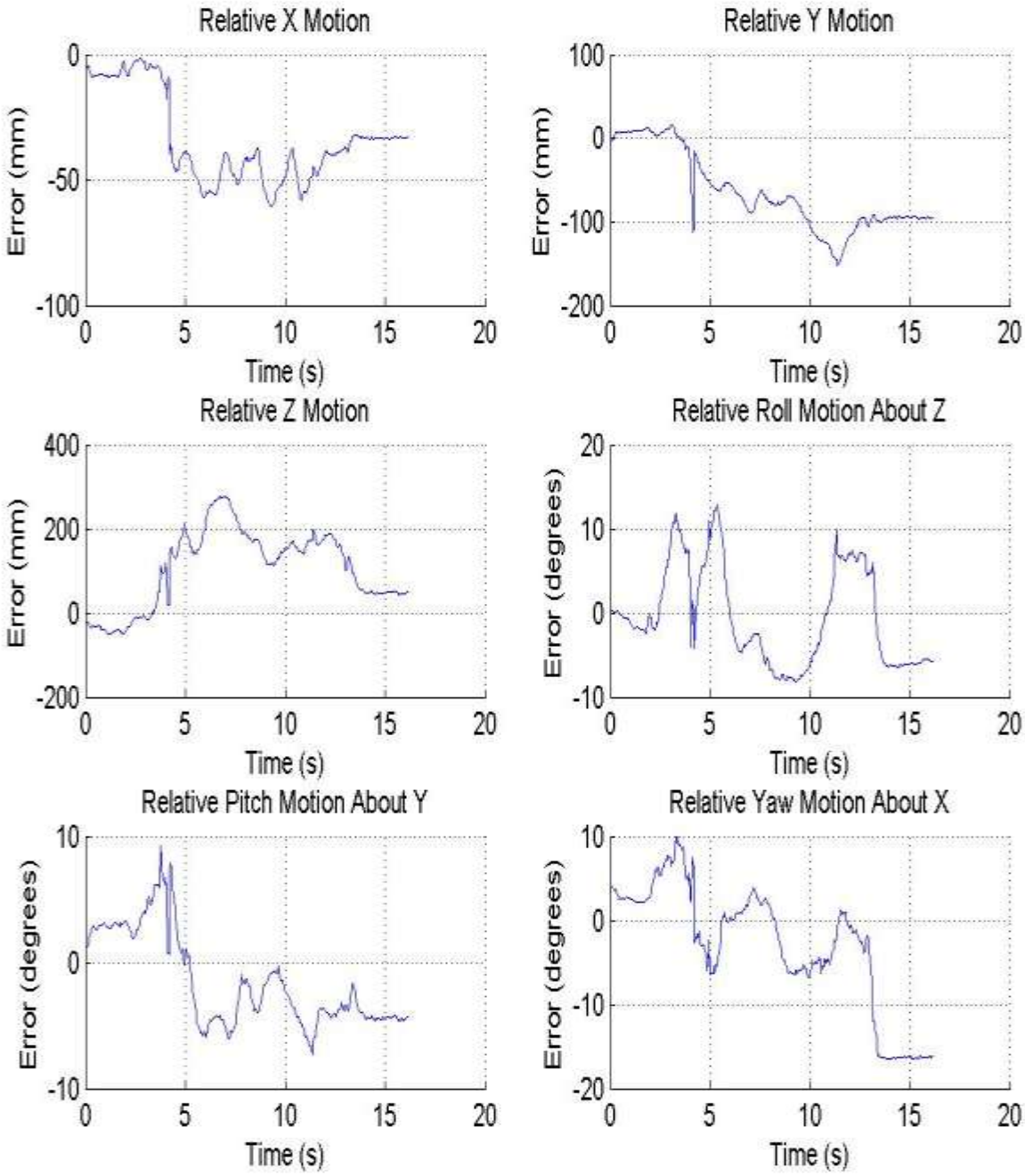

Figure 6-16: Absolute error calculation W_Error_k for the second object in a two object random motion case as detailed in Figure 6-15.

Table 6-3: Single Object Motion Validation Experimental Results

Motion Case	**Criteria**	X	Y	Z	$\emptyset$	θ	ψ
Z-axis motion (1.5s offset)	Mean Abs Error (mm or °)	6.13	10.61	19.14	1.86	1.29	0.62
	Max Abs Error (mm or °)	14.93	23.01	38.48	8.89	4.48	1.06
	% within 10mm or 5°	60.26	56.41	43.16	99.57	100	100
	% above mean error	47.01	44.88	53.41	45.3	33.77	50
Random motion case 1 (5.4s offset)	Mean Abs Error (mm or °)	12.3	13.37	7.36	6.01	3.41	0.94
	Max Abs Error (mm or °)	45.81	54.05	51.31	14.95	32.38	8.82
	% within 10mm or 5°	54.64	52.86	79.29	37.67	87.78	96.45
	% above mean error	43.2	42.6	37.87	57.2	16.96	25.05
Random motion case 2 (1.8s offset)	Mean Abs Error (mm or °)	21.41	19.53	21.58	6.27	9.78	4.03
	Max Abs Error (mm or °)	135.01	177.39	240.56	62.65	55.17	65.16
	% within 10mm or 5°	26.8	35.49	42.88	67.73	49.91	85.58
	% above mean error	32.16	44.36	43.07	26.62	29.21	15.53
Random motion case 3 (3s offset)	Mean Abs Error (mm or °)	19.14	11.73	14.89	7.17	7.49	5.11
	Max Abs Error (mm or °)	51.53	54.44	101.56	86.92	33.83	15.95
	% within 10mm or 5°	37.65	60.39	49.89	38.63	61.86	63.33
	% above mean error	41.81	32.76	38.39	39.86	23.96	36.19
Average	Mean Average Error (mm or °)	14.75	13.81	15.74	5.32	5.49	2.7

Table 6-4: Two Object Motion Validation Experimental Results

Motion Case	Criteria	X	Y	Z	$\emptyset$	θ	ψ
Two Object Motion – first object (1.5s offset)	Mean Abs Error (mm or °)	14.71	34.60	24.89	4.24	7.70	5.3
	Max Abs Error (mm or °)	53.47	166.23	183.65	14.82	27.01	25.68
	% within 10mm or 5°	45.22	21.31	30.28	63.94	48.80	66.53
	% above mean error	43.82	27.69	25.9	39.64	41.24	28.49
Two Object Motion – second object (1.5s offset)	Mean Abs Error (mm or °)	33.27	67.39	25.38	5.04	3.64	6.01
	Max Abs Error (mm or °)	60.22	152.67	238.23	12.9	9.27	16.52
	% within 10mm or 5°	25	18.98	37.73	42.82	83.57	54.17
	% above mean error	62.27	58.8	43.75	57.18	53.7	34.26
Average	Mean Average Error (mm or °)	23.99	51	25.14	4.64	5.67	5.66

6.5 Discussion of Experimental Results

Validation experiments were completed to examine the effects of various motions. Based on the procedures outlined above in Chapter 6.3, the motion depicted in the above figures is of the motion of the target object relative to its static initial starting position. The motions depicted are based on measurements taken by both the device and the ground truth measurement system.

For the single object motion case, the device was generally able to track the motion of the target object, as seen in Figures 6-5 through 6-12. The mean absolute error seen in the Cartesian elements was approximately 15mm compared with 4° for the three Euler angles. For the Z-axis motion case, depicted in Figures 6-5 and 6-6 and detailed in Table 6-3, the mean absolute error observed was the smallest of all of the test cases. In this test case, the device was able to track the target object motion relatively well with a little higher error seen in the Y and Z axes.

For the first random motion case of a single object, depicted in Figures 6-7 and 6-8, an overall increase in error compared with the Z-axis motion case was observed. More pronounced error is seen in the roll

$\emptyset$ and pitch θ Euler angles. For the roll $\emptyset$ angle, the mean absolute error is more general, but it is deemed acceptable given its low value (6.01°). For the pitch θ angle, the mean absolute error appears to be more pronounced during cases for "faster" periods of motion occurring around the 5-7s duration. Examining Figure 6-7 further, the error on the pitch θ angle occurs during a relatively rapid change in motion. For this motion case, the majority of the error was kept to within 10mm or 5°, as seen in Table 6-3 above.

For the second random motion case of a single object, depicted in Figures 6-9 and 6-10, similar tracking results were seen to that of the first random motion case although with larger error. Larger error was seen on all six axes during cases of "faster" periods of motion. Given that sharper motions were completed in this motion case, higher error was seen. Additionally the maximum absolute error calculation is skewed due to likely data outliers on all axes for either the device or the OptiTrack system. With these points removed, the maximum absolute error will be reduced across the board.

For the third random motion case of a single object, depicted in Figures 6-11 and 6-12, generally acceptable tracking is seen on four of the six axes with poorer performance on the X axis and pitch θ angle. For the X axis motion, a general and broad error is seen compared with the pitch θ angles, which appears to lose convergence between 6-10s. It is theorized that the loss of convergence for the pitch θ angle likely skews the relative X axis motion result due to the nature of the $\left(T_{W_O}^{C}\right)^{-1}T_{O_k}^{C}$ calculation for the device. Data outliers or spikes on the Z axis and roll $\emptyset$ angle result in a higher skewed maximum absolute error calculation.

Generally speaking, the error observed in the single object motion case is viewed as acceptable with a few caveats. The system model is observed to have some issues dealing with "faster" motions, so an examination of the entrywise values of the disturbance covariance matrix Q_k should be completed. Specifically the values should be increased to account for "faster" periods of motion. Additionally, the error observed may be due to the use of the constant velocity model for the relative motion dynamics. Given the higher observed error during periods of more rapid and "faster" motion, consideration should be given to using a constant acceleration or higher order model for the relative motion dynamics. One of the stated limitations by various authors [21] [61] [62] of the EKF as pertaining to AR applications is that lower order motion models, such as the constant velocity model, could see some lag in human motion.

Another source of the observed error is likely due to the image processing component of the system. As noted in Chapter 4, the feature extraction algorithms complete edge detection for each of the four edges of the credit card and then extrapolate the intersections of these four lines to determine the four object

feature points. Since the 2D image is more sensitive to changes in the X, Y, and roll $\emptyset$ parameters than it is for changes to the depth parameters (Z, pitch θ , and yaw ψ) [39], it is likely that a larger error will be observed in the extracted feature points due to changes in the relative depth parameters. Furthermore, as the entire credit card is windowed during the *tracking phase* of the image processing step, the orientation of the credit card image in the display pixel plane will affect the processing time required for each time step. For example, when the credit card image in the display pixel plane is relatively aligned with the X and Y axes of the display pixel plane, then the *tracking phase* window will take up a smaller portion of the overall display. However, when the credit card image orientation is skewed, a much larger sized *tracking phase* window will be required meaning that more processing time will be required for this case. Therefore, the additional processing time due to a skewed credit card image orientation will result in a larger time step between samples causing the dynamic model error to increase further, as defined in Equation 3.26, which is restated below for reference.

$$Dynamic\ Model\ Error \propto \delta_k * |\dot{x}_{actual} - \dot{x}_{model}| \tag{3.26}$$

Finally the camera sensor and the measurement noise will contribute to further error when completing the feature point extraction. There is some small amount of error due to the sensor and there will be additional error due to the assumption of global shutter. As the physical camera CCD sensor will take images with a rolling shutter such that the left edge of the scene will be recorded with some lag after the right edge of the scene, some bias error will be introduced. It is therefore theorized that making adjustments to the disturbance covariance matrix Q_k, moving to a higher order relative motion dynamics model like a CA model, and making improvements to the image processing techniques, including compensation for the rolling shutter, should provide for improvements that will alleviate some of the performance issues observed.

Next, the multiple object motion case is examined. In this test case, the two target objects were moved simultaneously in the workspace and the relative motion for each target object with respect to their own initial starting position was computed. The results are shown in Figures 6-13 through 6-16. Generally speaking the results are noticeably worse than in the single object motion cases examined. Based on a visual review of the data, rough tracking results are observed. However, the mean absolute Cartesian element error is much higher (33% average) than the single object case (15% average), as can be seen

through a review of Tables 6-3 and 6-4. The mean absolute Euler angle is around 5°, which is considered acceptable and is in line with the results seen for the single object case.

For both target objects, a higher level of noise and spikes are seen on the device's measurements and is clearly seen in the error plots of Figures 6-14 and 6-16. From Figures 6-13 and 6-15, noisiness in the device's measurements are concluded to be a contributor to the degraded tracking performance, which appears in the absolute mean error calculations (see Table 6-4). These spikes or outliers also have the effect of improperly increasing the absolute maximum peak error measurements. The increased measured noisiness on the measurements indicates that an entrywise increase to the measurement covariance matrix R_k may be required. Additionally the image processing techniques would need to be optimized to better handle multiple object cases, as the device's data input quality was much worse in the two object motion cases. Apart from the error due to the image processing algorithms used, it is noted that two separate windows for each of the credit cards will be required in the *tracking phase*, resulting in a larger cycle time for the two object case. This cycle time increase directly correlates to an increase in the dynamic modelling error, as per Equation 3.26.

In addition to the issues noted with the image processing, the other recommendations from the single object case also apply to the multiple object case. Moving to an increased value of the disturbance covariance matrix Q_k should help to reduce the error during "faster" periods of motion. Moving to a constant acceleration or higher order model for the relative motion dynamics will aid the system's performance to random relative motions.

Based on the results from the validation experiments, it is therefore concluded that the video see-through mobile AR application based on position based visual POSE estimation methods was successfully implemented for the single object real-time tracking case. The tracking performance is concluded to be sufficient with a level of error that is tolerable for this proof of concept. To improve overall performance, it is recommended that changes be investigated in the values for the disturbance covariance matrix Q_k and the measurement covariance matrix R_k. Moving the process model A_k to a constant acceleration model from a constant velocity model is also recommended, as it will improve the systems performance to "faster" motions typically done by humans. Finally, it is recommended that the image processing techniques be further refined to provide better performance for both speed to allow for less error in the multiple object case and overall accuracy.

Chapter 7 CONCLUSIONS AND FUTURE WORK

7.1 Conclusions

This thesis investigated the design, verification, and validation of a video see-through AR application for a mobile computing device using position based visual POSE estimation methods. In addition, a framework was presented that allows extension to multiple objects, multiple object models, and other mobile computing platforms.

The general framework for the video see-through mobile AR application was proposed detailing the mathematical system model and the solution using position based visual POSE estimation methods. Specifically a solution using the EKF was provided including considerations for the relative motion dynamics, initialization, and the effective workspace of this system. Extension of this framework to multiple target objects was given along with considerations to ensure unique object identification in the tracking phase.

The general framework was then applied to a physical system. The design specifics to implement the video see-through mobile AR application on a BlackBerry Dev Alpha A mobile computing device was given, including a modified system model due to the physical characteristics of the device's camera and display. The architecture of the physical system was provided along with details for completing the Image Processing using the OpenCV libraries and video see-through AR rendering using the OpenGL libraries. Details around extending this system model to multiple objects, multiple object models, and multiple mobile computing devices were given.

To verify the video see-through mobile AR system model based on the position based visual POSE estimation methods, a suite of simulations were completed to assess the system models performance. Various motions models were applied with varying parameters such as initial position, noise, sampling rates, and disturbance and noise covariance values. The proposed system was verified and concluded to have reasonable overall performance given the simulation results along with a conclusion that the parameters were suitably selected for this system.

To validate the video see-through mobile AR application, an independent ground truth measurement system was used. In these validation experiments, the device and the ground truth measurement system were held in a static position and a smooth motion was applied to the target object in 3D space over time. The relative position and orientation was measured by both the device and the ground truth measurement system and this data was transformed to represent the relative motion of the target object from its initial starting point. The implemented system was concluded to have acceptable performance in the single object real-time tracking case. Multiple objects saw degraded performance, likely due to the image processing techniques used. Overall, it is recommended that changes to the disturbance covariance matrix Q_k and the measurement covariance matrix R_k be examined to improve performance due to "faster" motions and higher measurement noise levels seen in the multiple object experiments. Additionally it is recommended that a constant acceleration or higher order model process model be adopted, as it will better respond to the more natural and rapid motions generated by humans. Finally it is recommended that the image processing techniques be further refined to provide improved overall accuracy in general and to also increase execution speed to allow for less error in the multiple object case.

7.2 Contributions

As noted previously and summarized here, the primary contribution of this research is the first implementation of a video see-through mobile AR application using position based visual POSE estimation methods for real-time tracking of multiple objects using a single monocular camera. At the present time, all other currently published approaches in this field use large numbers of feature points and tracking approaches that are computationally expensive in contrast to the position based visual POSE estimation method. Given that mobile phones often run various other applications and services like the cellular radio, WiFi, BlueTooth, etc., minimization of the video see-through mobile AR application's computational complexity is an important consideration to allow for true real-time tracking performance in a mobile computing environment when there is a competition for resources. Additionally this work provides the details required to extend this work to multiple object models and also to multiple mobile computing device platforms.

7.3 Future Work

From the work presented in the prior chapters, there are two major areas where future work can be conducted. The first is in optimization of the system model and the second is to refine the methods used in validation.

There are several areas of system model optimization that can be pursued to improve upon the performance and expansion of scope. The first improvement would be to optimize the image processing of the detection and tracking of the target object feature points. Additional work should be done to improve upon the robustness of the measurements, which may require an algorithmic change in the image processing techniques. Investigation should also be pursued to implement a rolling shutter compensation similar to what was done by Klein and Murray [66] in the PTAM approach. Additionally, consideration should be given to augmenting the feature point extraction accuracy with a co-operative approach between time steps, such as with bundle adjustment [66]. However care should be taken to minimize computational expense, as it could affect real-time tracking performance.

Additionally, the current image processing and feature point detection/identification implementation was benchmarked to execute on the order of 12ms to 25ms. This in contrast to the 1ms needed to complete the relative POSE estimation step. Recall that to ensure real-time tracking implementation, both the image processing and relative POSE estimation steps need to complete in less than 50ms to ensure no visible lag occurs in the video see-through AR implementation. It is noted that due to the architecture design, the video see-through AR rendering operations can be removed from the execution time budget, as it is done in parallel to image processing and POSE estimation. In reviewing the current execution time budget, it appears that the system could only reasonably manage real time parallel tracking of two to three target objects before lag is seen by the user. Therefore further work should be done to optimize the time required for the image processing techniques, as it will increase the number of target objects that can be tracked in parallel.

A second area of model improvement and optimization would be to improve the capabilities of object detection and tracking. An ideal enhancement would be to include the capability to detect and track unknown target objects. Tribou [60] has done some work in this area and applying these methods would improve upon the system model versatility. Furthermore, the system model should be extended to also include a 3D CAD library of various object models. The target object model of the credit card was selected

for this work to allow for reproducibility and repeatability of results. However, the 3D CAD library should be extended to include objects of interest to the application in question, e.g. medical training, etc. Additionally the system model should be made more robust with respect to unique object identification techniques such as outlined in Chapter 3 and summarized by Wagner [24].

One final system model optimization would be to enhance the overall model to change the relative motion dynamics model to a constant acceleration or higher order model and to update the disturbance covariance matrix Q_k and the measurement covariance matrix R_k, as previously stated. Additionally investigation should be done to examine a dynamically varying disturbance covariance matrix Q_k that adjusts as the sample rate varies. Recall that a faster sample rate implies that the CV model will have less error in predicting the motion so a smaller value of Q_k would be required. Therefore as the sample rate varies, the disturbance covariance matrix Q_k should be dynamically adjusted lower for faster sample rates and higher for slower sample rates. It is felt that with these areas of model improvements, the items noted during the validation experiments would be addressed.

The second area for further research is to further refine the methods used in validation. As noted previously, one limitation of the current approach is that the device and ground truth measurement system are not synchronized in the recording of data. As a result, a manual time offset needed to be applied in the data analysis in order to compare the results, which induces some level of error. To eliminate this source of error, investigation should be done to implement some triggering that would allow the data from both the device and ground truth measurement system to be synchronized. Possible options include using the PC to trigger both device and ground truth measurement system applications.

Another option would be to investigate offline photogrammetric methods for validation. In such an approach, experiments could be conducted using just the device and the target object(s). The video see-through AR images could then be recorded onto the device along with the relative POSE data and downloaded so that offline image analysis and comparison could be performed. Such an approach would eliminate the issues seen due to aligning and synchronizing the data.

Additionally, work should be completed to relate the absolute position error of the target object from its initial starting position to an error value on the device's display. As the relative POSE between the device and the target object varies, the perception of the scale of the error will shift accordingly. To determine the error with respect to the display for each time step, the position error vector could be added to the previously determined relative POSE vector to create an adjusted relative POSE vector. The object feature points can then be transformed into the display pixel plane using the adjusted relative POSE vector

and Equations 3.5 and 4.9 for each time step. From here, a comparison to the extracted feature points could easily be completed to determine the error relative to the display.

In closing, it has been shown that position based visual POSE estimation methods can be successfully applied to a video-see through AR real-time tracking application for a mobile computing device. By investigating the improvements noted above, the video see-through mobile AR application using position based visual POSE estimation methods would have greater applicability with respect to real-time applications that require 3D knowledge of the surrounding environment, such as surgical training, equipment assembly and servicing, and search and rescue applications.

Appendix A MATHEMATICAL PRELIMINARIES

A.1 Rotation Matrices

Matrices, which define in the $\mathcal{R}^3$ space rotation with an angle θ about X, Y, and Z axes are described below:

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\,\theta & -sin\,\theta \\ 0 & sin\,\theta & cos\,\theta \end{bmatrix} \tag{A.1}$$

$$R_y(\theta) = \begin{bmatrix} cos\,\theta & 0 & sin\,\theta \\ 0 & 1 & 0 \\ -sin\,\theta & 0 & cos\,\theta \end{bmatrix} \tag{A.2}$$

$$R_z(\theta) = \begin{bmatrix} cos\,\theta & -sin\,\theta & 0 \\ sin\,\theta & cos\,\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{A.3}$$

Then by a simple matrix multiplication, the combined rotational matrix can be obtained about the three Cartesian axes as follows,

$$R_{zyx}(\emptyset, \theta, \psi) = R_z(\emptyset)R_y(\theta)R_x(\psi) \tag{A.4}$$

Appendix B MEASUREMENT JACOBIAN

The measurement Jacobian C_k is linearized about the current state estimate $\hat{x}_{k,k-1}$ for each time step. It is derived as the partial derivative of $G(x_k)$ with respect to the six relative POSE parameters $W = [X, Y, Z, \emptyset, \theta, \psi]$,

$$G(x_k) = -f\left[\frac{x^c_{1_k}}{P_X z^c_{1_k}} \quad \frac{y^c_{1_k}}{P_Y z^c_{1_k}} \quad \frac{x^c_{2_k}}{P_X z^c_{2_k}} \quad \frac{y^c_{2_k}}{P_Y z^c_{2_k}} \quad \cdots \quad \frac{x^c_{N_k}}{P_X z^c_{N_k}} \quad \frac{y^c_{N_k}}{P_Y z^c_{N_k}}\right]^T \tag{B.1}$$

$$C_k = \left.\frac{\partial G(x)}{\partial X}\right|_{x=\hat{x}_{k,k-1}} \tag{B.2}$$

where the measurement Jacobian C_k^j is defined for each camera image plane feature point pair as follows,

$$C_k^j = \left[\frac{\partial x_j^i}{\partial X} \quad \frac{\partial x_j^i}{\partial X} \quad \frac{\partial x_j^i}{\partial Z} \quad \frac{\partial x_j^i}{\partial \emptyset} \quad \frac{\partial x_j^i}{\partial \theta} \quad \frac{\partial x_j^i}{\partial \psi} \quad \frac{\partial y_j^i}{\partial X} \quad \frac{\partial y_j^i}{\partial Y} \quad \frac{\partial y_j^i}{\partial Z} \quad \frac{\partial y_j^i}{\partial \emptyset} \quad \frac{\partial y_j^i}{\partial \theta} \quad \frac{\partial y_j^i}{\partial \psi}\right]^T \tag{B.3}$$

The measurement Jacobian generated in MATLAB can be algebraically simplified further as follows,

$$\left[\frac{\partial x_j^i}{\partial X} \quad \frac{\partial x_j^i}{\partial Y} \quad \frac{\partial x_j^i}{\partial Z} \quad \frac{\partial x_j^i}{\partial \emptyset} \quad \frac{\partial x_j^i}{\partial \theta} \quad \frac{\partial x_j^i}{\partial \psi}\right]^T = -\frac{f}{P_X}\frac{1}{T_{13}+Z} *$$

$$\left[1 \quad 0 \quad -\frac{x_j^C}{z_j^C} \quad (-S_\emptyset T_{11} + C_\emptyset T_{12}) \quad \left(C_\emptyset T_{13} + \frac{x_j^C}{z_j^C} T_{11}\right) \quad \left(-C_\emptyset S_\theta T_{12} + S_\emptyset (S_\theta T_{11} + C_\theta T_{13}) + \frac{x_j^C}{z_j^C} C_\theta T_{12}\right)\right]^T \tag{B.4}$$

$$\left[\frac{\partial y_j^i}{\partial X} \quad \frac{\partial y_j^i}{\partial Y} \quad \frac{\partial y_j^i}{\partial Z} \quad \frac{\partial y_j^i}{\partial \emptyset} \quad \frac{\partial y_j^i}{\partial \theta} \quad \frac{\partial y_j^i}{\partial \psi}\right]^T = -\frac{f}{P_Y}\frac{1}{T_{13}+Z} *$$

$$\left[0 \quad 1 \quad -\frac{y_j^C}{z_j^C} \quad (C_\emptyset T_{11} + S_\emptyset T_{12}) \quad \left(S_\emptyset T_{13} + \frac{y_j^C}{z_j^C} T_{11}\right) \quad \left(-S_\emptyset S_\theta T_{12} - C_\emptyset (S_\theta T_{11} + C_\theta T_{13}) + \frac{y_j^C}{z_j^C} C_\theta T_{12}\right)\right]^T \tag{B.5}$$

where

$$T_{11} = [C_\theta \quad S_\theta S_\psi \quad S_\theta C_\psi] \begin{bmatrix} x_j^O \\ y_j^O \\ z_j^O \end{bmatrix} \tag{B.6}$$

$$T_{12} = [0 \quad -C_\psi \quad S_\psi] \begin{bmatrix} x_j^O \\ y_j^O \\ z_j^O \end{bmatrix} \tag{B.7}$$

$$T_{13} = [-S_\theta \quad C_\theta S_\psi \quad C_\theta C_\psi] \begin{bmatrix} x_j^O \\ y_j^O \\ z_j^O \end{bmatrix} \tag{B.8}$$

Appendix C IMAGE PROCESSING ALGORITHMS

C.1 Detection Algorithm

1. Down sample the whole image by factor of 4
2. Find all the edge pixels
    a. Apply 7 by 7 Gaussian smooth kernel
    b. Apply 3 by 3 Sobel kernel, store the result in separate memory
    c. Calculate magnitude of the gradient
    d. Binary threshold at value 10
3. Apply morphological closing twice to get solid shapes
4. Find all contours in the binary image
    a. For each contour, apply approximation with distance of 12
    b. If the approximation has 4 vertices, break and assume this contour is the object
5. Find bounding box of the approximated object, and enlarge the bounding box by 60 pixels on each side
6. The bounding box is the tracking window.

C.2 Tracking Algorithm

1. If detection window is more than 450 by 700, down sample the image
2. Find line segments
    a. Find the connected components in the detection window
        i. Find all edge pixels
            1. Apply 7 by 7 Gaussian smooth kernel
            2. Apply 3 by 3 Sobel kernel, store the result in separate memory
            3. Calculate magnitude of the gradient
            4. Binary threshold at value 10
        ii. Flood-fill the image from 4 corners
    b. Find contour of each of the connect components, the neighbouring pixels are also neighbours in the returned list

c. Sort connected components by the distance of the component's centroid to the center of the detection window
d. For each contour, remove corner pixels
    i. For each pixel, select the two pixels that are 20 positions away from the current pixel
    ii. Calculate the area of the triangle formed by the three points
    iii. If area is greater than 100, then the current pixel is sitting on a corner
e. The remaining pixels form straight line segments. Associate each line segment with the originating contour
f. For each segment, connect the segments that are close together
g. Remove segments that are too short in comparison to other segments

3. Fit line equations for each of the line segments
4. Find feature points from line
    a. Determine a quadrilateral
        i. Starting from an empty set, add batches of lines to the set. Each batch of lines belongs to the same contour
        ii. Compute all intersection as lines are added to the set
        iii. Compute the convex polygon formed by the intersections
        iv. Remove convex vertices that are not on the corner using the same method as described above
        v. If the convex polygon is not a quadrilateral, continue to process next batch of lines
    b. Find black strip within quadrilateral
        i. For each pair of edges, divide each edge into 10 steps
        ii. Connect a line between corresponding step, and sub-sample the image along the line
        iii. If more than 85% of the sub-sampled pixels are below 50 in intensity, a black strip is found
        iv. If found, copy the vertices of the convex polygon to the feature points in order
    c. Continue fetching lines until no line to fetch

Appendix D CAMERA CALIBRATION

The assumption of a pinhole camera model for the camera used in the device will introduce some level of distortion and error. Fortunately a large amount of this error can be eliminated through calibration and additionally the scaling factors S_x and S_y in the display to camera transformation function h can be determined. There are various automated techniques available [63] [64] along with manual methods that allow for both the error to be calibrated and for the scaling factor to be determined. In general the techniques involve taking images of known input patterns such as chessboards from set distances that allow for the required parameters to be accurately determined.

As the project was implemented using the OpenCV libraries, use of the OpenCV camera calibration routines [54] was ideal, as it allowed for the correction the images in real time thereby minimizing both the radial and tangential distortion errors [9]. Radial distortion can occur due to a variety of reasons but most commonly occur due to the symmetry of the lens and is classified as a barrel or pincushion distortion. This distortion can be corrected using Brown's distortion model [65] as follows,

$$x_{corrected} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6 + \ ... + k_n r^{2n}),\ n = 1 \ldots \infty \qquad (D.1)$$

$$y_{corrected} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6 + \ ... + k_n r^{2n}),\ n = 1 \ldots \infty \qquad (D.2)$$

where x and y represent the display pixel plane coordinates, $x_{corrected}$ and $y_{corrected}$ represents the corrected output display coordinates, k_n represents the radial distortion coefficients, and r is defined as follows:

$$r = \sqrt{(x - x_{corrected})^2 + (y - y_{corrected})^2} \qquad (D.3)$$

Tangential distortion is caused due to the fact that the lenses are not perfectly parallel to the camera CCD sensor and can also be corrected using Brown's distortion model [62] as follows,

$$x_{corrected} = x + \left(2p_1 xy + p_2(r^2 + 2x^2)\right) \qquad (D.4)$$

$$y_{corrected} = y + (2p_2 xy + p_1(r^2 + 2y^2)) \qquad (D.5)$$

where p_1 and p_2 are defined as the tangential distortion coefficients and r is defined as above. This gives five distortion parameters, which OpenCV can use in real time to correct the image taken by the camera sensor.

$$Distortion_{coefficients} = [k_1 \quad k_2 \quad p_1 \quad p_2 \quad p_3] \tag{D.6}$$

Following the calibration routines, 16 pictures of a 6x8 chessboards were taken at various angles. Each chessboard space was 2.87cm by 2.87cm. The pictures were dimensioned 720 by 1280 pixels, which correlates to the display pixel plane. Since the calibration routines are operating on a known target input with respect to the display pixel plane image, the scaling factor from the display to camera can also be determined. Using the OpenCV camera calibration tool, the following parameters were determined.

$$Distortion_{coefficients} = \begin{bmatrix} 0.24004484937784151 \\ -5.8501954862571743 \\ 0 \\ 0 \\ 41.511647147457438 \end{bmatrix}^T \tag{D.7}$$

$$\frac{f*S_x}{P_x} = \frac{f*S_y}{P_y} = 1884.2751480305226 \ m/pixel \tag{D.8}$$

$$OC_x = 360 \tag{D.9}$$

$$OC_y = 640 \tag{D.10}$$

As there is a priori knowledge of the system; specifically that the camera focal length is fixed and is $f = 3.32mm$, and the camera inter-pixel spacing parameters $P_x = P_{y=}1.4\mu m/pixel$, this allows us to determine the display to camera scaling parameters S_x and S_y as follows.

$$S_x = S_y = 0.7946 \tag{D.11}$$

Appendix E TEST EQUIPMENT

The following test equipment was required to complete the validation experiments using the device and a ground truth measurement system:

a. Natural Point OptiTrack V100:R2 Measurement System, quantity 4 cameras
b. Vista Attaris Tripod, quantity 5
c. OptiTrack Calibration Wand
d. OptiTrack Markers, quantity 8
e. BlackBerry Dev Alpha Device, BBPIN: 29D7590F
f. Credit Card Object Models – see appendix E
g. Computer, quantity 2
h. Software: OptiTrack Tracking Tools Version 2.5.0
i. BlackBerry 10 Native SW Development Kit
j. Tortoise SVN Code Depository
k. MATLAB 7.1 Student and MATLAB 13 Student SW Packages

Appendix F TARGET OBJECT MODELS

The following two figures shown below represent the Known Object Credit Card Models used in the validation experiments and are presented here for completeness.

Figure F-1: Known credit card object model used in single object validation experiments.

Figure F-2: Known credit card object model used in multiple object validation experiments and was used as the second object model.

REFERENCES

[1] Zhou F, Duh H BL, and M Billinghurst. Trends in Augmented Reality Tracking, Interaction and Display:A Review of Ten Years of ISMAR, *ISMAR '08 Proceedings of the 7th IEEE/ACM International Symposium on Mixed and Augmented Reality*, pp. 193-202, 2008.

[2] Caudell TP, and D Mizell. Augmented Reality: An Application of Heads-Up Display Technology to Manual Manufacturing Processes, *Proceedings of the Twenty-Fifth Hawaii International Conference on System Sciences*, Volume II, pp. 659-669, 1992.

[3] Feiner S, Macintyre B, and D Seligmann. Knowledge-Based augmented reality, *Communications of the ACM - Special issue on computer augmented environments: back to the real world*, Volume 36 Issue 7, pp. 53-62, 1993.

[4] Hutchison S, GD Hager, and PI Corke. A Tutorial on Visual Servo Control, *IEEE Transactions on Robotics and Automation*, Volume 12, Issue 5, pp 651-670, 1996.

[5] L Weiss L, Sanderson AC, and CP Neuman. Dynamic sensor-based control of robots with visual feedback, *IEEE Journal of Robotics and Automation*, Volume 3, Issue 5, pp. 404-417, 1987.

[6] Wilson W, Hulls C, and G Bell. Relative End-Effector Control Using Cartesian Position Based Visual Servoing, *IEEE Transactions on Robotics and Automation*, Volume 12, Issue 5, pp. 684-696, 1996.

[7] Chaumette F and S Hutchinson. Visual servo control, Part I: Basic approaches, *IEEE Robotics and Automation Magazine*, Volume 13, Issue 4, pp. 82-90, 2006.

[8] Chaumette F and S Hutchinson. Visual servo control, Part II: Advanced approaches, *IEEE Robotics and Automation Magazine*, Volume 14 Issue 1, pp. 109-118, 2007.

[9] Forsyth D, and J Ponce. *Computer Vision: A Modern Approach*, Prentice Hall Inc., 2003.

[10] Kalman RE. A new approach to linear filtering and prediction problems, *Journal of Basic Engineering*, Volume 82, Issue 1, pp. 35-45, 1960.

[11] Wagner D, and D Schmalstieg. Making Augmented Reality Practical on Mobile Phones, Part 1, *IEEE Computer Graphics and Applications*, Volume 29, Issue 3, pp. 12-15, 2009.

[12] Wagner D, and D Schmalstieg. Making Augmented Reality Practical on Mobile Phones, Part 2, *IEEE Computer Graphics and Applications*, Volume 29, Issue 4, pp. 6-9, 2009.

[13] Oui WW, Ng EGW, and RU Khan. An Augmented Reality's framework for Mobile, *2011 International Conference on Information Technology and Multimedia (ICIM)*, pp. 1-4, 2011.

[14] Kato H and M Billinghurst. Marker tracking and HMD calibration for a video-based augmented reality conferencing system, *(IWAR '99) Proceedings. 2nd IEEE and ACM International Workshop on Augmented Reality*, pp. 85-94, 1999.

[15] C Geiger, B Kleinjohann, C Reimann and D Stichling. Mobile AR4ALL, *Proceedings. IEEE and ACM International Symposium on Augmented Reality*, pp. 181-182, 2001.

[16] Mohring M, Lessig C, and O Bimber, Video See-Through AR on Consumer Cell Phones, *ISMAR '04 Proceedings of the 3rd IEEE/ACM International Symposium on Mixed and Augmented Reality*, pp. 252-253, 2004.

[17] Wagner D, and D Schmalstieg. Artoolkit on the pocketpc platform, *IEEE International 2003Augmented Reality Toolkit Workshop*, pp. 14-15, 2003.

[18] Schall G, Grabner H, Graber M, Wolhart P, Schmalstieg D, and H Bischof. 3D Tracking in Unknown Environments Using On-Line Keypoint Learning for Mobile Augmented Reality, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 2008. CVPRW '08, pp. 1-8, 2008.

[19] Gherghina, A, Olteanu A, and N Tapus. A Marker-Based Augmented Reality System for Mobile Devices, *Roedunet International Conference (RoEduNet)*, 2013 11th, pp. 1-6, 2013.

[20] Wagner D, and D Schmalstieg. ARToolKitPlus for Pose Tracking on Mobile Devices, *Proceedings of 12th Computer Vision Winter Workshop (CVWW'07)*, pp. 139-146, 2007.

[21] Lepetit V and P Fua. Monocular Model-Based 3D Tracking of Rigid Objects: A Survey, *Foundations and Trends in Computer Graphics and Vision*, vol. 1, no. 1, pp. 1-89, Oct. 2005.

[22] Wagner D, Langlotz T, and D Schmalsteig. Robust and unobtrusive marker tracking on mobile phones, In *ISMAR 2008: Proceedings of the 7th IEEE/ACM International Symposium on Mixed and Augmented Reality*, pp. 121-124, 2008.

[23] Wagner D., Reitmayr G, Mulloni A, Drummond T, and D Schmalsteig. 2008. Pose tracking from natural features on mobile phones. *ISMAR 2008: Proceedings of the 7th IEEE/ACM International Symposium on Mixed and Augmented Reality*, pp. 125–134, 2008.

[24] Lowe D. Distinctive image features from scale-invariant keypoints, *Int. Journal of Computer Vision*, Volume 60, Issue 2, pp. 91-110, 2004.

[25] Ozuysal M, Fua, P, and V Lepetit. Fast keypoint recognition in ten lines of code, *IEEE Conference on Computer Vision and Pattern Recognition*, 2007. CVPR '07, pp. 1-8, 2007.

[26] Wagner D, Schmalstieg D, and H Bischof. Multiple Target Detection and Tracking with Guaranteed Framerates on Mobile Phones, *ISMAR 2009: 8th IEEE International Symposium on Mixed and Augmented Reality*, pp. 57-64, 2009.

[27] Skrypnyk I and DG Lowe. Scene Modelling, Recognition and Tracking with Invariant Image Features, *ISMAR '04:Proceedings of the Third IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'04)*, pp. 110–119, 2004.

[28] Bay H., Tuytelaars T, and LV Gool. SURF: Speeded up robust features, *Computer Vision and Image Understanding 10 (2008)*, pp. 346–359, 2008.

[29] Takacs, G, Chandrasekhar V, Gelfand N, Xiong Y, Chen WC, Bismpigiannis T, Grzeszczuk R, Pulli, K., and B Girod. Outdoors Augmented Reality on Mobile Phone using Loxel-Based Visual Feature Organization, *MIR '08 Proceedings of the 1st ACM international conference on Multimedia information retrieval*, pp. 427-434, 2008.

[30] Ta, D.-N., Chen, W.-C., Gelfand, N., Pulli, K., SURFTrac: Efficient Tracking and Continuous Object Recognition using Local Feature Descriptors, *Conference on Computer Vision and Pattern Recognition (CVPR'09),* pp. 2937-2943, 2009

[31] Yang X and KT Cheng. LDB: An Ultra-Fast Feature for Scalable Augmented Reality on Mobile Devices, *2012 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pp. 49-57, 2012.

[32] Mikolajczyk K and C Schmid. A Performance Evaluation of Local Descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 27, 10 (Oct.2005), pp. 1615-1630. 2005.

[33] Klein G, and D Murray. Parallel tracking and mapping for small ar workspaces, *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR),* pp. 225–234, 2007.

[34] H Durrant-Whyte and T Bailey. Simultaneous localization and mapping (SLAM): part I. *IEEE Robotics & Automation Magazine*, 13(2):99{110, 2006.

[35] Verbelen T, Simoens P, De Turck F, and B Dhoedt. A component-based approach towards mobile distributed and collaborative PTAM, *2012 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pp. 329-330, 2012.

[36] MW Spong and M Vidyasagar. *Robot dynamics and control*. Wiley India Pvt. Ltd., 2008.

[37] Heath TL. *The thirteen books of Euclid's Elements*, Dover Publications, 1956.

[38] Brogan W. *Modern Control Theory, 3rd Edition*, Prentice Hall Inc., 1991.

[39] Wang W, and Wilson W. 3D relative position and orientation estimation using Kalman filtering for robot control, *1992 IEEE International Conference on Robotics and Automation*, 1992. Proceedings, pp. 2638–2645, 1992.

[40] Janabi-Sharifi F, and M Marey. A Kalman-Filter-Based Method for Pose Estimation in Visual Servoing, *IEEE Transactions on Robotics*, Volume 26, Issue 5, pp. 939-947, 2010.

[41] Ficocelli M, and F Janabi-Sharifi. Adaptive filtering for pose estimation in visual servoing, *2001 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2001 Proceedings, pp. 19–24. 2001.

[42] Lippiello V, Siciliano B, and L Villani. Adaptive extended Kalman filtering for visual motion estimation of 3D objects, *Control Engineering Practice*., Volume 15, Issue 1, pp. 123–134, 2007.

[43] Cai Y, Freitas, N, and JJ Little. Robust Visual Tracking for Multiple Targets, *ECCV2006*, pp. 107-118, 2006.

[44] Yang C, Duraiswami R, and L Davis. Fast Multiple Object Tracking via a Hierarchical Particle Filter, *Tenth IEEE International Conference on Computer Vision*, ICCV 2005, Volume 1, pp. 212-219, 2005.

[45] Yu T, and Y Wu. Collaborative Tracking of Multiple Targets, *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2004*, Volume 1, pp.834-841, 2004.

[46] Yuan X, and SZ Li. Learning Feature Extraction and Classification for Tracking Multiple Objects: A Unified Framework, *In Proc. of the IEEE International Conference on Video and Signal Based Surveillance (AVSS'06)*, pp. 22-27, 2006.

[47] Grabner M, Grabner H, and H Bischof. Real-time tracking via on-line boosting. *In Proc. of British Machine Vision Conference (BMVC)*, Volume I, pp. 47-56, 2008.

[48] Pernkopf F. Tracking of Multiple Targets Using On-Line Learning for Appearance Model Adaptation, *IEEE Transactions on Systems, Man, and Cybernetics*, Volume 38, Issue 6, pp. 1465-1475, 2008.

[49] Card SK, Moran TP, and A Newell. *Psychology of Human-Computer Interaction*, Lawrence Erlbaum Associates, Inc. 1983.

[50] ISO/IEC 7810:2003 *Identification cards – physical characteristics.*

[51] BlackBerry, *BlackBerry 10 Dev Alpha A and B – What Developers Need to Know*, http://devblog.blackberry.com/2012/10/blackberry-10-dev-alpha-a-b-specs, 2012.

[52] Gonzales R, and R Woods. *Digital Image Processing*, Third Edition, Prentice Hall Inc., 2008.

[53] OpenCV.org, *Open Source Computer Vision*, http://opencv.org/, 2013.

[54] OpenCV Development Team, *Camera Calibration with OpenCV*, http://docs.opencv.org/doc/tutorials/calib3d/camera_calibration/camera_calibration.html/, 2013.

[55] OpenGL.org, *OpenGL: The Industry's Foundation for High Performance Graphics*, http://www.opengl.org/, 2013.

[56] TortoiseSVN team, *TortoiseSVN Open Source Subversion (SVN) client for Software Version Control*, http://tortoisesvn.net/, 2013.

[57] Chen YS. Augmented Reality with Extended Kalman's Filter System Documentation, *University of Waterloo*, 2013.

[58] Natural Point Inc., *V100:R2 – An affordable camera for mocap and tracking applications*, http://www.naturalpoint.com/optitrack/products/v100-r2/indepth.html, 2013.

[59] Einstein A. *Relativity: The Special and the General Theory. A Popular Exposition. Authorized Translation by Robert W. Lawson, Third Edition*, Methuen & Co. Ltd, 1920.

[60] Tribou M. Recovering Scale in Relative Pose and Target Model Estimation Using Monocular Vision, *University of Waterloo*, 2009.

[61] Azuma R. "A survey of augmented reality," in Computer Graphics, SIGGRAPH Proceedings, pp. 1–8, 1995.

[62] Ravela S, Draper B, Lim J, and R Weiss, "Adaptive tracking and model registration across distinct aspects," in International Conference on Intelligent Robots and Systems, pp. 174–180, 1995.

[63] Bouguet JY, CalTech University. *Camera Calibration Toolbox for Matlab – References*, http://www.vision.caltech.edu/bouguetj/calib_doc/htmls/ref.html/, 2010.

[64] Bouguet JY, CalTech University. *Camera Calibration Toolbox for Matlab*, http://www.vision.caltech.edu/bouguetj/calib_doc/, 2010.

[65] Brown DC. Decentering Distortion of Lenses, *Photometric Engineering*, Volume XXXII, No. 3, pp. 444-462, 1966.

[66] Klein G, and D Murray. Parallel tracking and mapping on a camera phone, *Proceedings of the 2009 8th IEEE International Symposium on Mixed and Augmented Reality (ISMAR '09)*, pp. 83-86, 2009.