Recursive Estimation of 3-D Motion Trajectories from Image Sequences Using Measurements of Feature Point Positions and Optical Flow

Lin Zhao

A project report submitted to
the Department of Electrical and Computer Engineering
in comformity with the requirements
for the degree of Master of Eng.

Queen's University
Kingston, Ontario, Canada

May, 1998

Abstract

This project presents a new method to compute three-dimensional trajectories from both optical flow and image plane position measurements derived from the estimation of second-order intensity variations in image sequences. It is based on Nagel's optical flow computation approach in which optical flow is obtained by minimizing the mean squared error differences between a second-order Taylor expansion of gray values from one frame and the observed gray values within the same window from the next frame. From his method, in the special case of a "gray value corner", and another special case of a "gray value extremum point", we have developed a new technique in which the measurement vector contains both position and image plane velocity. The measurement vector can be approximately expressed as a first-order Taylor expansion which is a function of parameters estimated from the measured gray value surface.

Our goal is to estimate the three dimensional trajectory and structure of a moving rigid object in an image sequence. The hybrid feature/flow-based recursive trajectory and structure estimation algorithm, proposed by Blostein and Chann, is reviewed in detail. This algorithm is used together with this new measurement equation we have derived in this project. It is shown that accuracy of the 3-D motion estimation is improved significantly. The effects of using the different values of parameters in the measurement equation to the 3-D motion are also demonstrated.

Acknowledgements

I would like to thank my project supervisor Dr. Steven Blostein for his great guidance, constant encouragement, and patience. Thanks to all my course instructors , Dr. Fairman, Dr. Bayoumi, Dr. Blostein, Dr. Zarowski, and others in our department, who taught me knowledge that had been used in my research. I am grateful to my friend, Mr Bin Zhou, who spent a lot of time to teach and help me a lot about LATEX. I also thank my parents for their encouragement and support.

Contents

iii

Chapter 1

Introduction

Background

Motion analysis is a very important subject in computer vision. It includes estimating the relative three-dimension(3-D) motion of a moving object with respect to the camera from given image sequences. A number of real-world problems require the estimation of 3-D motion. These include applications in video image compression, industrial automation and inspection, robot assembly, autonomous vehicle navigation, biomedical engineering and remote sensing.

In motion analysis, obtaining reliable information about the moving object is a fundamental and important problem. The information contains the 3-D time-varying position, orientation, translational and rotational velocity, which are extremely valuable for motion prediction. In this project, we focus on estimating the 3-D motion of a moving object using a single camera. The camera is assumed to be stationary and this moving object is a rigid-body that implies that the 3-D distance between any pair of points never varies with time.

Usually, there are two phases in a complete 3-D motion analysis system based on the use of measurement data obtained. The first phase is to compute the optical flow or/and establish the correspondence of discrete features. The second phase is to use these data to determine the motion parameters. Blostein and Chann had developed a hybrid feature/flow-based recursive 3-D trajectory and structure estima-

tion algorithm that focuses on the recovery of motion parameters. Unlike any other approaches, this algorithm is the first ever to make use of both image plane coordinates of discrete features and image plane velocity information and produce better estimation of motion parameters than that of a purely feature-based algorithm. This project addresses performance improvements for this hybrid algorithm.

Contribution

In our motion analysis research, we have derived the relationship between the measurement of image plane coordinates of feature points and image plane velocity. The relationship can be approximately expressed as a first order Taylor expansion in terms of the parameters of an estimated gray-level surface from pixel observations, and the error between estimated and actual parameters. It makes the joint computation of feature point coordinates and optical flow numerically tractable. We are also interested in how we can use the image measurements for estimating 3-D motion of a moving object from a sequence of images. We show that the estimation of these parameters of an estimated gray level surface is unbiased and its error covariance is known and does not change over time: We will show that the first-order term of this Taylor expansion about the estimated parameters in which the expection of the second term of right hand is equal to zero (Chapter 3). That means that the bias between the measurement vector and the first term of the right hand side of this equation, in terms of the estimation of these parameters, is equal to zero.

We will use this new measurement technique together with the hybrid feature/velocity-based recursive algorithm in estimating 3-D motion of a moving object. From the results of our simulations, we see obvious improvements in performance: smaller bias and variance, increased stability, and faster convergence under the conditions of random measured gray values, and a set of different gray value and image resolutions used in our experiments.

Organization

This project report is organized as: Chapter 2 provides the relevant background of 3-D motion estimation, a brief introduction of the Kalman filtering which will be used in the hybrid algorithm, and a detailed description of the hybrid feature/flow-based 3-D trajectory estimation algorithm. Chapter 3 presents details of the derivation that the measurement equation can be approximately expressed as a Taylor expansion, and discussion of the use of this equation. Chapter 4 shows the implementation details and the simulation results from our experiment. Chapter 5 forms the conclusions based on our experiment results.

Chapter 2

Background and the Hybrid algorithm of 3-D motion Estimation

In this chapter, we will briefly review relevant background material of 3-D motion estimation. Then we will give a detailed description of the hybrid feature/flow-based recursive 3-D trajectory and structure estimation algorithm. The background contains basic characteristics of the previous approaches to 3-D estimation from digital image sequences, the Kalman filter and its extended variations which are often used in many 3-D motion algorithms and also used in our research. The details of the hybrid algorithm include the imaging model, object model, motion model and system model. This hybrid recursive 3-D motion estimation algorithm is core to this project, and will be applied in Chapter 3 and Chapter 4.

2.1   Background of 3-D motion estimation

There are a number of different approaches developed for the recovery of 3-D motion information (position, translation and rotational velocity). As Chann notes[3], all of them can be classified into one of three main methods: feature-based, optical flow-based and direct. The following is a description of the basic features of these main approaches.

2.1.1   Feature-based Methods

Feature-based methods use a small number of salient feature-point coordinates on the image plane from the image sequences to track and estimate 3-D motion. The features are commonly chosen as corners, lines, and regions. The choice of features should not affect the formulation of the solution. The procedures of feature-based approaches contain three steps. The first is to extract the features of the moving object from one frame. The second step is to find those same features in the next frame to determine feature correspondence. Typically, the time interval between the first and second frame is very small. The final step is to estimate motion and structure. As we can see, obtaining these correspondences precisely is the key to feature-based methods. There are two kinds of feature-based methods used in motion estimation: batch and recursive.

Batch algorithms commonly put certain constraints on the object to obtain a set of linear or nonlinear equations in terms of the motion parameters. The minimum number of equations is at least as large as the number of motion parameters that need to be estimated. Because of the existence of noise in the process of estimation, the number of equations used is typically much more than the minimum in practice. The use of a large number of equations increases the computational load. In many real-time problems, recovery of 3-D motion parameters must be at real-time. In this case, most batch algorithms are not applicable because of the large computational requirements.

Recursive algorithms are widely used in real-time 3-D motion estimation because they provide higher efficiency in computation. A recursive algorithm using a dynamical system of equations requires only the previous state estimate and the current measurement to produce the current estimate. It can reduce the processing data significantly. The Kalman filter is the most commonly used recursive algorithm. We also use it in our project. A conceptual introduction of Kalman filtering will be given in the next section.

2.1.2 Optical-Flow-based methods

Unlike feature-based techniques, optical-flow-based methods recover 3-D motion information from the measurement of optical flow. The first step of optical-flow-based approaches is always to compute the optical flow field $\mathbf{u}(x_i, y_i) = [\ u(x_i, y_i)\quad v(x_i, y_i)\ ]^T$. The optical flow at the coordinate (x_i, y_i) is the instantaneous velocity of the gray level (brightness) pattern at this point. Because the optical flow equations can be algebraically expressed as the functions of the motion parameters, after computing the optical flow, 3-D motion parameters can be estimated. There is another difference between feature-based and optical-flow-based methods. Usually, feature-based approaches assume a stationary observer and a moving object, while optical-flow-based approaches commonly assume that an observer is moving within a static environment.

2.1.3 Direct methods

The third class are referred to as direct techniques. The characteristic of these methods is that there is no feature extraction and correspondence nor optical flow to be computed before recovery 3-D motion and structure. Instead, 3-D motion parameters can be obtained directly from the measurement of gray values at each pixel location.

2.2 Overview of Kalman filtering

We have mentioned previously that recursive algorithms are commonly applied in real-time problems for estimating 3-D motion because of the advantage of high computational efficiency. Kalman filtering is an important and extensively used recursive algorithms. The Kalman filter produces an optimal estimate of the system state based on all previous noisy measurements[5]. The hybrid algorithm used in this project is built on Kalman filtering. It is therefore necessary to give a general overview.

2.2.1 State space model

For a given dynamic physical system, in which the state-space model can be described by differential equation as:

$$\frac{d\mathbf{x}(t)}{dt} = f(\mathbf{x}(t), t) + \mathbf{w}(t) \tag{2.1}$$

where $\mathbf{x}(t)$ is the state of the system, $\mathbf{w}(t)$ is the process noise. The measurement equation is expressed as:

$$\mathbf{y}(t) = h(\mathbf{x}(t), t) + \mathbf{v}(t) \tag{2.2}$$

where $\mathbf{y}(t)$ is the measurement, $\mathbf{v}(t)$ is the measurement noise. The goal is to extract the system state $\mathbf{x}(t)$ from the noise added measurement $\mathbf{y}(t)$ and make the estimate, $\hat{\mathbf{x}}(t)$, close to $\mathbf{x}(t)$. The perfect estimate is $\hat{\mathbf{x}}(t)$ equal to $\mathbf{x}(t)$.

In a special case that both $f(\cdot)$ and $h(\cdot)$ are linear functions of the state variable, the plant model, then, can be written as:

$$\frac{d\mathbf{x}(t)}{dt} = F(t)\mathbf{x}(t) + \mathbf{w}(t) \tag{2.3}$$

$$\mathbf{y}(t) = H(t)\mathbf{x}(t) + \mathbf{v}(t) \tag{2.4}$$

The implementation of this plant model on a digital computer requires discretization of equations (2.3) and (2.4). It leads to:

$$\mathbf{x}_{k+1} = \Phi(k+1, k)\mathbf{x}_k + \mathbf{w}_k \quad \mathbf{w}_k \sim N(0, Q_k) \tag{2.5}$$

$$\mathbf{y}_k = H_k\mathbf{x}_k + \mathbf{v}_k \quad \mathbf{v}_k \sim N(0, R_k) \tag{2.6}$$

where $\Phi(k+1, k) = \exp[(t_{k+1} - t_k)F_k]$, and $\exp[\cdot]$ is the matrix exponential[5].

To unify the notation, we use $\mathbf{x}(k)$, $\mathbf{y}(k)$, $F(k)$ to express $\mathbf{x}_k$, $\mathbf{y}_k$, $\Phi(k+1, k)$ respectively. Then equations (2.5) and (2.6) can be rewritten as:

$$\mathbf{x}(k+1) = F(k)\mathbf{x}(k) + \mathbf{w}(k) \tag{2.7}$$

$$\mathbf{y}(k) = H(k)\mathbf{x}(k) + \mathbf{v}(k) \tag{2.8}$$

Next, we will introduce how the Kalman filter performs state estimation.

2.2.2　Discrete Linear Kalman Filter

The discrete linear Kalman filter is an elegant generalization of recursive least squares, and it is convenient for digital computer implementation. The Kalman filter estimates optimally the unknown system state from the noisy measurements.

Suppose there is a discrete system, whose plant model is described in equations (2.7) and (2.8). The Kalman filter equations are

Time Update:

$$
\begin{aligned}
\hat{\mathbf{x}}(k+1|k) &= F(k)\hat{\mathbf{x}}(k|k) & (2.9)\\
P(k+1|k) &= F(k)P(k|k)F^T(k) + Q(k) & (2.10)
\end{aligned}
$$

Measurement Update:

$$
\hat{\mathbf{x}}(k+1|k+1) = \hat{\mathbf{x}}(k+1|k) + K(k+1)[Y(k+1) - H(k+1)\hat{\mathbf{x}}(k+1|k)] \quad (2.11)
$$

$$
\begin{aligned}
P(k+1|k+1) &= [I - K(k+1)H(k+1)]P(k+1|k)\\
&= [I - K(k+1)H(k+1)]P(k+1|k)[I - K(k+1)H(k+1)]^T\\
&\quad + K(k+1)R(k+1)K^T(k+1) \quad P(0|0) = P(0) \quad (2.12)
\end{aligned}
$$

$$
\begin{aligned}
K(k+1) &= P(k+1|k)H^T(k+1)\\
&\quad \times [H(k+1)P(k+1|k)H^T(k+1) + R(k+1)]^{-1} \quad (2.13)
\end{aligned}
$$

Note that we prefer using the latter form of equation (2.12) to ensure the positive definiteness and symmetry of the error covariance matrix. In the formulation of the linear Kalman filter, if there is no prior knowledge about the process noise and measurement noise, $\mathbf{w}_k$ and $\mathbf{v}_k$ are assumed to be temporally white Gaussian noise:

$$
E\{\mathbf{w}_i\mathbf{w}_j^T\} = \begin{cases} Q_k & i = j \\ 0 & i \neq j \end{cases} \quad (2.14)
$$

$$
E\{\mathbf{v}_i\mathbf{v}_j^T\} = \begin{cases} R_k & i = j \\ 0 & i \neq j \end{cases} \quad (2.15)
$$

without loss of generality, $\mathbf{w}$ and $\mathbf{v}$ are often assumed to be uncorrelated:

$$E\{\mathbf{w}_i\mathbf{v}_j^T\} = 0 \quad \forall i,j \tag{2.16}$$

The following is a summary of an algorithm that performs the Kalman filtering equations (2.7) to (2.16).

1. Initialize the step index $k = 0$

2. Initialize the state estimate vector $\hat{\mathbf{x}}(0|0)$

3. Initialize the covariance matrix $P(0|0) = P(0)$ of the state estimate error

4. Compute the next covariance of the state prediction error $P(k+1|k+1)$ using equation (2.10)

5. Compute the Kalman gain $K(k+1)$, using equation (2.13)

6. Compute the next covariance of state estimator error $P(k+1|k+1)$, using equation (2.12)

7. Compute the next predicted state $\hat{\mathbf{x}}(k+1|k)$, using equation (2.9)

8. Compute the measurement residual, $\Delta\mathbf{y}(k+1|k) = \mathbf{y}(k+1) - H(k+1)\hat{\mathbf{x}}(k+1|k)$

9. Compute the state estimate $\hat{\mathbf{x}}(k+1|k+1)$, using equation (2.11)

10. Increment k by 1

11. Output the state estimate

12. Return to step 4

With each filter cycle, the estimate is updated and improved by incorporating a new measurement vector.

2.2.3 Kalman Filtering for Nonlinear systems

In the case of systems not describable as a linear model, its plant model is given as:

$$\mathbf{x}(k+1) \;=\; f(\mathbf{x}(k), k) + \mathbf{w}(k) \tag{2.17}$$

$$\mathbf{y}(k) \;=\; h(\mathbf{x}(k), k) + \mathbf{v}(k) \tag{2.18}$$

where $f(\cdot)$ and $h(\cdot)$ are deterministic nonlinear functions depending on the system state. In order to apply Kalman filtering to this nonlinear problem, we need to linearize the nonlinear equations about the current estimate state $\hat{\mathbf{x}}(k|k)$. To achieve this, we only introduce the extended Kalman filter (EKF) and iterated extended Kalman filter (IEKF), which are used in our project.

Extended Kalman Filter (EKF)

To obtain linearized process and measurement equations, one can apply the first-order Taylor expansion about $\hat{\mathbf{x}}(k|k)$, the current state estimate at time k. The EKF is different from a "linearized Kalman filter" which is linearized about some predetermined reference trajectory. One advantage of using the EKF is the avoidance of a predetermined reference trajectory. The EKF is only suitable for small state deviations $\tilde{\mathbf{x}}(k|k) = \mathbf{x}(k) - \hat{\mathbf{x}}(k|k)$ and $\tilde{\mathbf{x}}(k+1|k) = \mathbf{x}(k+1) - \hat{\mathbf{x}}(k+1|k)$.

After linearization of a nonlinear plant model, there is a little change from the basic Kalman filter equations. The prediction of state estimation equation (2.9), is replaced by

$$\hat{\mathbf{x}}(k+1|k) = \hat{\mathbf{x}}(k|k) + \int_{t_k}^{t_{k+1}} f(\mathbf{x}, \tau) d\tau \tag{2.19}$$

The measurement update equation (2.11), is replaced by

$$\hat{\mathbf{x}}(k+1|k+1) = \hat{\mathbf{x}}(k+1|k) + K(k+1)\{\mathbf{y}(k+1) - h[\hat{\mathbf{x}}(k+1|k)]\} \tag{2.20}$$

The state coupling matrix

$$F_k = \left.\frac{df(\mathbf{x}_k, k)}{d\mathbf{x}_k}\right|_{\mathbf{x}_k = \hat{\mathbf{x}}_k} \tag{2.21}$$

is the Jacobian of $f(\cdot)$ with respect to the state vector at $\hat{\mathbf{x}}(k)$.

The input-to-output coupling matrix

$$H_k = \left. \frac{dh(\mathbf{x}_k, k)}{d\mathbf{x}_k} \right|_{\mathbf{x}_k = \hat{\mathbf{x}}_k} \tag{2.22}$$

is the Jacobian of $h(\cdot)$ with respect to the state vector at $\hat{\mathbf{x}}(k)$.

Iterated Extended Kalman Filter (IEKF)

The iterated extended Kalman filter is developed based on the assumption that when $\mathbf{y}(k+1)$ is obtained, the filtered estimates, $\hat{\mathbf{x}}(k+1|k+1)$, are, in general, better than the predicted estimates, $\hat{\mathbf{x}}(k+1|k)$. That is, one might achieve better results by repeating the linearization of the measurement function about the latest estimate until a predefined condition is satisfied. The IEKF uses an iterator instead of equation (2.20) in the EKF. This iterator is given by

$$\eta_{i+1} = \hat{\mathbf{x}}(k+1|k) + K_{k+1} \{ \mathbf{y}_{k+1} - h(\eta_i) - H_{k+1}[\hat{\mathbf{x}}(k+1|k) - \eta_i] \} \quad i = 0, 1, \cdots \tag{2.23}$$

where $\eta_0 = \hat{\mathbf{x}}(k+1|k)$, and $\eta_l = \hat{\mathbf{x}}(k+1|k+1)$. Obviously, when l is equal to 1, the IEKF reduced to the EKF.

2.3 The Hybrid Feature/Optical-flow-based Recursive Algorithm

In previous sections of this chapter, we have introduced the basic characteristics of feature-based and optical-flow-based techniques and the fundamental differences between them. In feature-based approaches, discrete features are extracted and tracked over time; in optical-flow-based approaches, 2-D velocities of gray value patterns are computed for every frame in an image sequence.

Blostein and Chann[1][2][3] have developed a recursive algorithm, which makes use of both feature and optical flow measurements, to estimate 3-D motion trajectory and structure based on Broida *et al.*'s pure feature-based algorithm[4]. Since this project is a continuation of Blostein and Chann's research, a detailed description of

this algorithm is necessary. In this algorithm, it is assumed that a stationary video camera is used to observe a single moving rigid-body over time. The goal is to recover the 3-D trajectory of this object from an extended image sequence captured by the video camera. The imaging model, object model, motion model, and system plant model will be examined in order.

2.3.1 Imaging model

The pin-hole camera model combined with the perspective projection is used. It is sufficient for modelling the imaging process. Suppose that a coordinate frame is attached to the camera, with its origin at the focal point and its x-axis coinciding with the optical axis. The image plane is parallel to the x-y plane and is located at $Z = f$, where $f > 0$, is the focal length of the camera. Let $P = (X, Y, Z)$ be a 3-D point and $p = (x, y)$ be its projection onto the image plane via the perspective projection [1][2][3][11][12].

$$p = \begin{bmatrix} x \\ y \end{bmatrix} = \frac{f}{Z} \begin{bmatrix} X \\ Y \end{bmatrix} + \begin{bmatrix} n_x \\ n_y \end{bmatrix} \tag{2.24}$$

where n_x and n_y are additive noise terms, the random error caused mainly from spatial quantization. Without loss generality, a unity focal length is assumed, (i.e., $f = 1$).

2.3.2 Object Model

In order to observe a moving object, whose structure is known, through a stationary camera, two coordinate systems are used. One is the camera-centred coordinate system (CCCS) with the z-axis pointing in the direction of the optical axis. Another is the object-centred coordinate system (OCCS), which is a moving frame attached to the object.

Let the position vector of O_o in the CCCS be given by

$$S_R(t) = [\ X_R(t) \ \ Y_R(t) \ \ Z_R(t) \ ]^T \tag{2.25}$$

and the position vector of the i-th feature point in the OCCS be given by

$$S_{oi} = [\ X_{oi} \ \ Y_{oi} \ \ Z_{oi} \ ]^T \tag{2.26}$$

12

then the position vector of this feature point in the CCCS is given by

$$S_i(t) = S_R(t) + R(t)S_{oi} \tag{2.27}$$

where $R(t)$ is the rotation matrix that aligns the OCCS with the CCCS. The quantity, $S_R(t)$ is the origin of OCCS, and is not directly observable. Because of the assumption of 3-D object rigidity, S_{oi} is not dependent on time.

2.3.3   Motion model

In rigid-body motion there is no relative motion of points in or on the rigid-body. The points must always maintain a fixed position relative to one another and all the points move with the body as a whole. Motion of a rigid-body is completely characterized by translational and rotational components. The moving position of each point in or on the moving rigid-body can be represented by the same translational and rotational transformation of the point from its initial position.

Translational motion

A linear model is used to describe the translation motion. The assumption is that the object is moving at a constant velocity $T = [\ T_x \quad T_y \quad T_z\ ]^T$. The centre of rotation is not affected by the rotational motion of the object. Suppose its initial location at time $t = t_0$ is known. Then its position at any time t is given by

$$S_R(t) = S_R(t_0) + (t - t_0)T \tag{2.28}$$

Rotational Motion

It is also assumed that this moving object is rotating about an axis though O_o at a constant rotational velocity $\omega = [\ \omega_x \quad \omega_y \quad \omega_z\ ]^T$. The centre of rotation, O_o, is not affected by the rotational motion of the object. The orientation of the OCCS with respect to the CCCS can be defined as the 3-D rotation which must align the OCCS with the CCCS. The unit quaternion is used to express the orientation of a rotating

13

object in time.

$$\mathbf{q}(t) = \exp[(t - t_0)\Omega(\omega)]\mathbf{q}(t_0) \qquad (2.29)$$

where

$$\mathbf{q}(t) = [\ q_1(t) \quad q_3(t) \quad q_3(t) \quad q_4(t)\ ]^T \quad ||\mathbf{q}(t)|| = 1 \qquad (2.30)$$

which is a 4-parameter vector representing 3 degrees of freedom, and

$$\Omega(\omega) = \frac{1}{2} \begin{bmatrix} 0 & \omega_z & -\omega_y & \omega_x \\ -\omega_z & 0 & \omega_x & \omega_y \\ \omega_y & -\omega_x & 0 & \omega_z \\ -\omega_x & -\omega_y & -\omega_z & 0 \end{bmatrix} \qquad (2.31)$$

where $[\ \omega_x \quad \omega y \quad \omega_z\ ]^T$ is the angular velocity vector.

Note that the rotation matrix R can be expressed in terms of q,

$$R = \begin{bmatrix} q_1^2 - q_2^2 - q_3^2 + q_4^2 & 2(q_1q_2 - q_3q_4) & 2(q_1q_3 + q_2q_4) \\ 2(q_1q_3 + q_2q_4) & -q_1^2 + q_2^2 - q_3^2 + q_4^2 & 2(q_2q_3 - q_1q_4) \\ 2(q_1q_3 - q_2q_4) & 2(q_2q_3 + q_1q_4) & -q_1^2 - q_2^2 + q_3^2 + q_4^2 \end{bmatrix} \qquad (2.32)$$

In (2.32) the dependence on time is suppressed to simplify the notation. All quantities are time-varying.

2.3.4   System Plant Model

3-D motion system consists of a rigid object translating and rotating continuously over time. Measurements are obtained at uniform-spaced time intervals. The system is modelled as equations (2.1) and (2.2). The problem is to estimate the system state from the measurements. Discretization is required before implementation on digital computer. Since both $f(\cdot)$ and $h(\cdot)$ are nonlinear functions, a linear Kalman filter may not be applied. The extended Kalman filter (EKF) and iterated extended Kalman filter (IEKF) are chosen in this project due to this nonlinear filtering problem. The state coupling matrix $F(k)$ and the input-output coupling matrix $H(k)$ are obtained by using equation (2.21) and (2.22) respectively.

The State Variables

In a good model which describes the system, the state vector must contain sufficient information to characterize the behaviour of the system. In this algorithm, the state

14

vector consists of the image plane coordinates of the OCCS origin, the translational velocity, the unit quaternion representing the orientation of the object, the rotational velocity, and the normalized coordinates of the feature points in the OCCS.

$$
x(t) = \begin{bmatrix} x_R(t)/z_R(t) \\ y_R(t)/z_R(t) \\ \dot{X}_R(t)/z_R(t) \\ \dot{Y}_R(t)/z_R(t) \\ \dot{Z}_R(t)/z_n(t) \\ q_1 \\ q_2 \\ q_3 \\ q_4 \\ \omega_x \\ \omega_y \\ \omega_z \\ X_{o1}/z_R(t) \\ Y_{o1}/z_R(t) \\ Z_{o1}/z_R(t) \\ \vdots \\ X_{oM}/z_R(t) \\ Y_{oM}/z_R(t) \\ Z_{oM}/z_R(t) \end{bmatrix} = \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \\ s_6 \\ s_7 \\ s_8 \\ s_9 \\ s_{10} \\ s_{11} \\ s_{12} \\ s_{13} \\ s_{14} \\ s_{15} \\ \vdots \\ s_{10+3M} \\ s_{11+3M} \\ s_{12+3M} \end{bmatrix} \tag{2.33}
$$

In (2.33) M is the number of the feature points.

The normalization factor $z_R(t)$ is used to eliminate the unknown scaling factor. In equation (2.33), we denote states s_1–s_5 as the translational states, states s_6–s_{12} as the rotational states, and states s_{13}–s_{12+3M} as the structural states.

Measurements of feature point position and optical flow

In a feature-based algorithm, measurements are normally image plane coordinates of the feature point. Blostein and Chann have made a first step toward integrating the feature-based and optical-flow-based approaches to 3-D motion estimation. They have established the measurement vector that consists of feature point position and its corresponding optical flow. However, they have considered the feature points and optical flow measurements to be statistically independent. In practice, these

measurements are all derived from a single sensor, a video camera. In this report, we propose a new measurement model that takes the correlation between feature point and optical flow measurements into account.

The measurements of feature point can be obtained by using the perspective projection.

$$\mathbf{y}_k = \begin{bmatrix} x_1 \\ y_1 \\ \vdots \\ x_M \\ y_M \end{bmatrix} = \begin{bmatrix} X_1/Z_1 \\ Y_1/Z_1 \\ \vdots \\ X_M/Z_M \\ Y_M/Z_M \end{bmatrix} \tag{2.34}$$

where M is the number of the feature points. The measurement function is highly non-linear in the state variables. It can be shown by examining the image plane coordinates of the i-th feature point.

$$x_i = \frac{X_i}{Z_i} = \frac{X_R + R_1 S_{oi}}{Z_R + R_3 S_{oi}} \tag{2.35}$$

$$y_i = \frac{Y_i}{Z_i} = \frac{Y_R + R_2 S_{oi}}{Z_R + R_3 S_{oi}} \tag{2.36}$$

where R_1, R_2, and R_3 are the first, second, third rows of the rotation matrix R, respectively (see equation (2.32)).

Dividing the right hand side of (2.35) and (2.36) by Z_R, equations (2.35), and (2.36) can be expressed in terms of the state variables [3],

$$x_i = \frac{s_1 + a_i}{1 + c_i} \tag{2.37}$$

$$y_i = \frac{s_2 + b_i}{1 + c_i} \tag{2.38}$$

where

$$\begin{aligned} a_i &= (q_1^2 - q_2^2 - q_3^2 + q_4^2)\frac{X_{oi}}{Z_R} + 2(q_1 q_2 - q_3 q_4)\frac{Y_{oi}}{Z_R} + 2(q_1 q_3 - q_2 q_4)\frac{Z_{oi}}{Z_R} \\ &= (s_6^2 - s_7^2 - s_8^2 + s_9^2)s_{10+3i} + 2(s_6 s_7 - s_8 s_9)s_{11+3i} \\ &\quad + 2(s_6 s_8 + s_7 s_9)s_{12+3i} \end{aligned} \tag{2.39}$$

16

$$
\begin{aligned}
b_i &= 2(q_1q_2 + q_3q_4)\frac{X_{oi}}{Z_R} + (-q_1^2 + q_2^2 - q_3^2 + q_4^2)\frac{Y_{oi}}{Z_R} + 2(q_2q_3 - q_1q_4)\frac{Z_{oi}}{Z_R} \\
&= 2(s_6s_7 + s_8s_9)s_{10+3i} + (-s_6^2 + s_7^2 - s_8^2 + s_9^2)s_{11+3i} \\
&\quad + 2(s_7s_8 - s_6s_9)s_{12+3i}
\end{aligned}
\tag{2.40}
$$

$$
\begin{aligned}
c_i &= 2(q_1q_3 - q_2q_4)\frac{X_{oi}}{Z_R} + 2(q_2q_3 + q_1q_4)\frac{Y_{oi}}{Z_R} + (-q_1^2 - q_2^2 + q_3^2 + q_4^2)\frac{Z_{oi}}{Z_R} \\
&= 2(s_6s_8 - s_7s_9)s_{10+3i} + 2(s_7s_8 + s_6s_9)s_{11+3i} \\
&\quad + (-s_6^2 - s_7^2 + s_8^2 + s_9^2)s_{12+3i}
\end{aligned}
\tag{2.41}
$$

Equation (2.34) is measurement equation of Broida et $al.$'s algorithm[4], which consists of only feature point coordinates. Bolstein and Chann modified this equation by adding the measurement of optical flow at the feature point[1][2][3]. The purpose of adding optical flow to the measurements is to provide more information to the Kalman filter such that the Kalman filter might produce a more accurate state estimate. Differentiating the image coordinates (x_i, y_i) with respect to time yields the optical flow equations:

$$
u_i = \frac{dx_i}{dt} = \frac{d}{dt}\left(\frac{X_i}{Z_i}\right) = \frac{\dot{X}_i}{Z_i} - \frac{X_i\dot{Z}_i}{Z_i^2} = \frac{\dot{X}_i}{Z_R} \cdot \frac{Z_R}{Z_i} - \frac{X_i}{Z_R} \cdot \frac{Z_R}{Z_i} \cdot \frac{\dot{Z}_i}{Z_R} \cdot \frac{Z_R}{Z_i}
\tag{2.42}
$$

$$
v_i = \frac{dy_i}{dt} = \frac{d}{dt}\left(\frac{Y_i}{Z_i}\right) = \frac{\dot{Y}_i}{Z_i} - \frac{Y_i\dot{Z}_i}{Z_i^2} = \frac{\dot{Y}_i}{Z_R} \cdot \frac{Z_R}{Z_i} - \frac{Y_i}{Z_R} \cdot \frac{Z_R}{Z_i} \cdot \frac{\dot{Z}_i}{Z_R} \cdot \frac{Z_R}{Z_i}
\tag{2.43}
$$

without losing generality, the focal length is assumed to be unity. Equations (2.42), (2.43) need to be expressed in terms of state variables.

Obviously, we need to determine the velocity of the ith feature point with respect to the camera, i.e., $\dot{S}(t) = [\ \dot{X}_i\ \ \dot{X}_i\ \ \dot{X}_i\ ]^T$. Differentiating (2.27) with respect to time, yields

$$
\frac{d}{dt}S_i(t) = \frac{d}{dt}S_R(t) + \frac{dR(t)}{dt}S_{oi}
\tag{2.44}
$$

The time derivative of the rotation matrix is given by

$$
\frac{d}{dt}R(t) = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} R(t)
\tag{2.45}
$$

using the relationship

$$\frac{Z_R}{Z_i} = \frac{Z_R}{Z_R + R_3 S_{oi}} = \frac{1}{1 + \frac{R_3}{Z_R} \cdot S_{oi}} \tag{2.46}$$

We substitute (2.44), (2.45), and (2.46) into (2.42), and (2.43)[3]:

$$u_i = \frac{s_3 + c_i s_{11} - b_i s_{12}}{1 + c_i} - \frac{(s_5 + b_i s_{10} - a_i s_{11})(s_1 + b_i)}{(1 + c_i)^2} \tag{2.47}$$

$$v_i = \frac{s_4 - c_i s_{10} + a_i s_{12}}{1 + c_i} - \frac{(s_5 + b_i s_{10} - a_i s_{11})(s_2 + b_i)}{(1 + c_i)^2} \tag{2.48}$$

where a_i, b_i, and c_i are defined in (2.39), (2.40), (2.41) respectively. Now, the new measurement vector $\mathbf{y}_k$, which includes the coordinates (positions) and velocities (optical flow) of the feature points, is established as:

$$\mathbf{y}_k = \begin{bmatrix} x_1 \\ y_1 \\ \vdots \\ x_M \\ y_M \\ u_1 \\ v_1 \\ \vdots \\ u_M \\ v_M \end{bmatrix} \tag{2.49}$$

The individual elements of the Jacobian matrix H_k can be obtained by differentiating (2.37), (2.38), (2.47), (2.48) with respect to each state variable. This was obtained by Blostein and Chann [1][2][3].

2.4   Summary

In this chapter, we have introduced the basic characteristics of three main categories methods for 3-D motion estimation and their differences. We have introduced the concept of the Kalman filter with some its variations used in our project and the procedures of performing Kalman filtering. Finally, we have described the hybrid feature/flow-based recursive 3-D motion estimation algorithm in detail.

Chapter 3

Measurement of Position and Velocity from Second-order Intensity Derivatives

We have introduced the hybrid measurement structure of feature point position and optical flow which is a highly nonlinear function in terms of the state variables defined in Chapter 2. We are also interested in how the optical flow and its coordinate on the image plane are related to the observed gray values obtained from a digital camera. In other words, we try to compute the optical flow and position on the image plane from the spatiotemporal gray value patterns of image intensity. To make our results tractable, we will then derive an approximation equation of this measurement that is a function of the second- order intensity derivatives in image sequences. First, we will briefly introduce the basic concepts of differential techniques for computing optical flow. Then, we will describe and discuss Nagel's approach[6] that computes optical flow from second-order intensity variations of an observed gray value pattern. Based on Nagel's method, we will show that the measurement vector, which includes optical flow and its coordinate position on the image plane, can be computed by estimating the second-order derivative of the gray value pattern from a sequence of images. Finally, we will discuss the applications of this approach to the hybrid feature/flow-based algorithm proposed by Blostein and Chann[1][2].

3.1 Differential Optical Flow techniques

It is well known that the measurement of optical flow is a fundamental problem in the processing of time-varying image sequences. The local computation of optical flow is an approximation to the 2-D field, which is a projection of the 3-D velocities of surface points onto an imaging surface, from spatiotemporal models of image intensity. There are many methods proposed for computing optical flow: differential techniques, region-based matching, energy-based methods, and phase-based techniques[8]. Differential techniques are used in our project, in which optical flow is computed from spatiotemporal intensity derivatives in image sequences.

3.1.1 First-order Differential Methods

First-order differential techniques use first-order derivatives based on image translation. The spatiotemporal model[8][9][11][12], which is time-varying, is expressed as:

$$g(\mathbf{x}, t) = g(\mathbf{x} - \mathbf{u}t, 0) \tag{3.1}$$

where $\mathbf{x} = [\ x\quad y\ ]^T$ is the coordinate in the x-y plane, and $\mathbf{u} = [\ u\quad v\ ]^T$ is the optical flow of the point.

It is assumed that the motion must be "small". In other words, the time interval between two frames must be very small. Then, from the Taylor series expansion, or more generally from the assumption that the temporal derivative $dg(\mathbf{x}, t)/dt = 0$, we can easily derive the gradient constant equation as:

$$\nabla g(\mathbf{x}, t)\mathbf{u} + g_t(\mathbf{x}, t) = 0 \tag{3.2}$$

or directly

$$g_x(\mathbf{x}, t)u + g_y(\mathbf{x}, t)v + g_t(\mathbf{x}, t) = 0 \tag{3.3}$$

where $\nabla g(\mathbf{x}, t) = [\ g_x(\mathbf{x}, t)\quad g_y(\mathbf{x}, t)\ ]^T$, is the spatial gradient vector field over the image plane, and $g_t(\mathbf{x}, t)$ is the partial time derivative of $g(\mathbf{x}, t)$. Usually, the notation is simplified to:

$$\nabla g(\mathbf{x}, t) \quad = \quad \nabla g = [\ g_x\quad g_y\ ]^T$$

$$g_t(\mathbf{x}, t) \;=\; g_t$$

As we can see from equation (3.2), there are two unknown components of $\mathbf{u}$ constrained by one linear equation. Therefore, we need more constraints to solve this problem.

3.1.2 Second-order Differential Methods

Second-order differential approaches use second-order derivatives of $g(\mathbf{x}, t)$ to constrain optical flow components. It can be derived directly from equations (3.1), (3.3)[8][9][12] that,

$$g_{xx}u + g_{yx}v + g_{tx} \;=\; 0 \tag{3.4}$$

$$g_{xy}u + g_{yy}v + g_{ty} \;=\; 0 \tag{3.5}$$

with $g_{xy} = g_{yx}$.

Equations (3.4), (3.5) can be rewritten as

$$\begin{bmatrix} g_{xx} & g_{xy} \\ g_{xy} & g_{yy} \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} + \begin{bmatrix} g_{tx} \\ g_{ty} \end{bmatrix} = 0 \tag{3.6}$$

where the symmetric matrix of second derivatives is called the Hessian of $g(\mathbf{x}, t)$, expressed as H. Assuming $d\nabla g(\mathbf{x}, t)/dt = 0$, optical flow can be computed by using equation (3.6) alone or together with equation (3.3). Second-order approaches can recover 2-D velocity in most cases, but cannot solve for $\mathbf{u}$ where the "aperture problem" exists in local neighbourhood where $DetH$ is equal to zero. A computational method has been proposed in which the aperture problem can be avoided in most cases. For the detail of this approach, the interested reader is referred to [9].

Differential techniques compute 2-D velocity under the assumption that $\mathbf{u}$ is constant across at least two different areas with linear independent values for the gradient ∇g or/and H (i.e. $\nabla\nabla g$). However, for the situation of abrupt change of $\mathbf{u}$, for instance, shadow edges and corners, which are commonly chosen as features in 3-D motion analysis, differential methods of linear modelled gray value variations, which are too simplistic around edges or corners, are not suitable in this kind of situation.

Nagel had proposed a minimization approach that can deal with this kind of situation for solving for a 2-D velocity vector $\mathbf{u}$. We will discuss this in the next section.

3.2 Estimating 2-D Velocity with Local Minimization

Nagel developed a local approach to inter-frame displacement estimation [6]. The basic idea is to assume that $g(\mathbf{x}_0, t_1)$ is observed at $\mathbf{x}_0$ in frame 1, and $g(\mathbf{x}_0, t_2)$ is observed at $\mathbf{x}_0$ in frame 2. A local environment around $\mathbf{x}_0$ is assumed to have been displaced between t_1 and t_2 by a vector $\mathbf{u} = (u, v)^T$. To determine the vector $\mathbf{u}$, we must minimize the squared differences $\sum[g(\mathbf{x}_0, t_2) - g(\mathbf{x} - \mathbf{u}, t_1)]^2$. If the time interval is small enough, $\mathbf{u}$ remains well within the chosen environment around $\mathbf{x}_0$. This minimization produces two coupled nonlinear equations for the two unknown components of the displacement vector $\mathbf{u}$. In a special case which known as a "gray value corner"[6], these coupled equations can be simplified to obtain a closed-form solution. We have discovered another special case of a "gray value extremum point" when we simulated the gray level surface of the feature point. In that case, we can also simplify these equations to get the closed-form as described in the following sections. The solution of the two unknown components of vector $\mathbf{u}$ is in terms of the estimated parameters of the second-order Taylor expansion modelled for the observed gray level surface.

3.2.1 Parameters of the Observed Gray Value Surface Model

Assume that in an small observed window, the origin $\mathbf{x}_0 = [\ x_0 \quad y_0\ ]^T$ of the co-ordinate system is the center of the small area of interest, i,e., $\mathbf{x} = [\ x \quad y\ ]^T$, the observed gray level surface can be approximately expressed by a second-order Taylor expansion[6][11]

$$
\begin{aligned}
g(\mathbf{x}) \;=\;\; & g(\mathbf{x}_0) + g_x x + g_y y + \frac{1}{2}g_{xx}x^2 \\
& + g_{xy}xy + \frac{1}{2}g_{yy}y^2 + \varepsilon
\end{aligned}
\tag{3.7}
$$

where g_x, g_y are partial derivatives of $g(\mathbf{x})$ with respect to x and y. g_{xx}, g_{xy} and g_{yy} are the second partial derivatives, and ε is random noise. Without loss of generality, let the initial position $\mathbf{x}_0 = (0,0)$ and the gray level surface be modelled as

$$
\begin{aligned}
f(x,y) &= f_0 + f_x x + f_y y + \frac{1}{2} f_{xx} x^2 \\
&\quad + f_{xy} xy + \frac{1}{2} f_{yy} y^2
\end{aligned}
\tag{3.8}
$$

Then, the measurement or observed gray level $g(x_i, y_i)$ can be modelled as

$$
g(x_i, y_i) = f(x_i, y_i) + n_g \quad n \sim N(0, \sigma_g^2)
\tag{3.9}
$$

with $x_i = (i-1)\mathrm{mod}(2k+1) - k, -k \le x_i \le k$, and $y_i = k - (i-1)\mathrm{mod}(2k+1), -k \le y_i \le k$, where k is an integer, $i = 1, 2, \ldots, N$ and $N = (2k+1)^2$, the total number of raster points in the observed small window, which is shown in Figure 3.1. For example, $k = 1$ corresponds to 3×3 window, and $k = 2$ corresponds to 5×5 window.
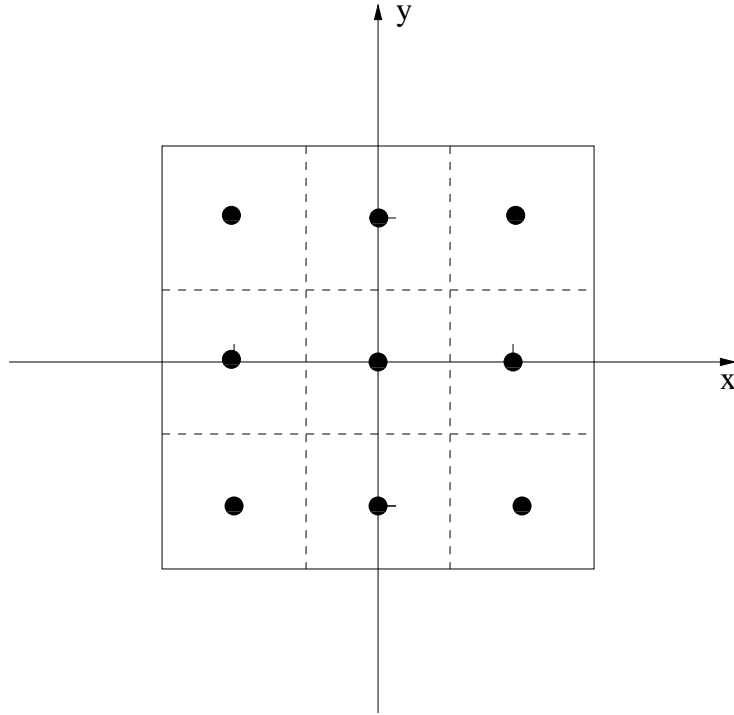
Figure 3.1: The observed window for estimating a gray-value surface

Then,

$$
\mathbf{G} = \mathbf{A}\mathbf{f} + \mathbf{n}_g
\tag{3.10}
$$

23

where

$$\mathbf{G} = [g(x_1, y_1), g(x_2, y_2), \ldots, g(x_N, y_N)]^T$$

$$\mathbf{A} = \begin{bmatrix} 1 & x_1 & y_1 & \frac{1}{2}x_1^2 & x_1y_1 & \frac{1}{2}y_1^2 \\ 1 & x_2 & y_2 & \frac{1}{2}x_2^2 & x_2y_2 & \frac{1}{2}y_2^2 \\ \vdots & & & & & \\ 1 & x_N & y_N & \frac{1}{2}x_N^2 & x_Ny_N & \frac{1}{2}y_N^2 \end{bmatrix}$$

and

$$f = [f_0, f_x, f_y, f_{xx}, f_{xy}, f_{yy}]^T$$

For a given estimate of f, the squared error between $\mathbf{G}$ and $\mathbf{Af}$ is

$$\begin{aligned} \mathbf{e}^2 &= \operatorname{tr}[(\mathbf{G} - \mathbf{Af})\mathbf{W}(\mathbf{G} - \mathbf{Af})^T] \\ &= (\mathbf{G} - \mathbf{Af})^T\mathbf{W}(\mathbf{G} - \mathbf{Af}) \end{aligned} \tag{3.11}$$

where $\mathbf{W}$ is a known weight matrix given by $\mathbf{W} = 1/\sigma_g^2\mathbf{I}$. Then, setting

$$\frac{\partial \mathbf{e}^2}{\partial \mathbf{f}} = 2\mathbf{A}^T\mathbf{W}(\mathbf{G} - \mathbf{Af}) = 0 \tag{3.12}$$

The least-squares estimate is

$$\hat{\mathbf{f}} = (\mathbf{A}^T\mathbf{WA})^{-1}\mathbf{A}^T\mathbf{WG} \tag{3.13}$$

which is the maximum-likelihood unbiased estimate. Under Gaussian noise assumptions,

$$\hat{\mathbf{f}} \sim N[\mathbf{f}, (\mathbf{A}^T\mathbf{WA})^{-1}] \tag{3.14}$$

i.e., $E\{\boldsymbol{\varepsilon}_f\} = 0$, where $\boldsymbol{\varepsilon}_f = \mathbf{f} - \hat{\mathbf{f}}$. Note that $\mathbf{A}^T\mathbf{WA}$ is called the Gram or Grammian matrix and the error covariance matrix of the parameter is the inverse of the Gram matrix[7], i.e.,

$$cov(\boldsymbol{\varepsilon}_f) = (\mathbf{A}^T\mathbf{WA})^{-1} \tag{3.15}$$

which is a constant. For the window size of 3×3, which is used in our simulation, $cov(\varepsilon_f)$ is given as,

$$Cov(\boldsymbol{\varepsilon}_f) = \sigma_g^2 \begin{bmatrix} 40 & 0 & 0 & -48 & 0 & 0-48 \\ 0 & 12 & 0 & 0 & 0 & 0 \\ 0 & 0 & 12 & 0 & 0 & 0 \\ -48 & 0 & 0 & 144 & 0 & 0 \\ 0 & 0 & 0 & 0 & 18 & 0 \\ -48 & 0 & 0 & 0 & 0 & 144 \end{bmatrix}$$

3.2.2 Computing 2-D velocity by minimizing mean squared error

Assuming that $g(\mathbf{x}_0, t_1)$ is observed at location $\mathbf{x}_0$ in frame 1 at time t_1, and $g(\mathbf{x}_0, t_2)$ is observed at the same location in frame 2 at time t_2, a local environment around $\mathbf{x}_0$ is displaced between time t_1 and time t_2 by the optical flow vector $\mathbf{u} = [\ u \quad v\ ]^T$. To determine $\mathbf{u}$, we want $\sum [g(\mathbf{x}, t_2) - g(\mathbf{x} - \mathbf{u}, t_1)]^2$, summed over all $\mathbf{x}$ from the local environment around $\mathbf{x}_0$, to be minimized. The time interval is very small so that $\mathbf{u}$ can remain well within the chosen small area.

Letting $\mathbf{f}_1 = [\ f1_0 \quad f1_x \quad f1_y \quad f1_{xx} \quad f1_{xy} \quad f1_{yy}\ ]$ be the parameters of the gray-value surface at time t_1, then the summed squared error is equal to

$$\sum [g(\mathbf{x}, t_2) - f1_0 - f1_x(x - u) - f1_y(y - v) - \frac{1}{2}f1_{xx}(x - u)^2$$
$$-f1_{xy}(x - u)(y - v) - \frac{1}{2}f1_{yy}(y - v)^2]^2 \tag{3.16}$$

The axes of the local coordinate system for the environment around $\mathbf{x}_0$ should be aligned with the principal curvature directions such that $f1_{xy} = 0$. Taking partial derivatives with respect to u and v, and setting them to zero, yields

$$\sum [f1_x + f1_{xx}(x - u)][g(\mathbf{x}, t_2) - f1_0 - f1_x(x - u)$$
$$-f1_y(y - v) - \frac{1}{2}f1_{xx}(x - u)^2 - \frac{1}{2}f1_{yy}(y - v)^2] = 0 \tag{3.17}$$

and

$$\sum [f1_y + f1_{yy}(y - v)][g(\mathbf{x}, t_2) - f1_0 - f1_x(x - u)$$
$$-f1_y(y - v) - \frac{1}{2}f1_{xx}(x - u)^2 - \frac{1}{2}f1_{yy}(y - v)^2] = 0 \tag{3.18}$$

Simplication of equation (3.17) and (3.18) yields

$$[f1_x - f1_{xx}u][\bar{g}_2 - \bar{g}_1 + u(f1_x - \frac{1}{2}f1_{xx}u)$$
$$+v(f1_y - \frac{1}{2}f1_{yy}v)] + \overline{x^2}f1_{xx}[f2_x - (f1_x - f1_{xx}u)] = 0 \tag{3.19}$$

and

$$[f1_y - f1_{yy}v][\bar{g}_2 - \bar{g}_1 + u(f1_x - \frac{1}{2}f1_{xx}u)$$
$$+v(f1_y - \frac{1}{2}f1_{yy}v)] + \overline{y^2}f1_{yy}[f2_y - (f1_y - f1_{yy}v)] = 0 \tag{3.20}$$

where $\bar{g}_2$ and $\bar{g}_1$ are the average gray value observed at time t_2 and t_1, respectively. $\bar{x^2}$ and $\bar{y^2}$ are the average of all the raster points' coordinates. Now, let $\mathbf{x}_0$ be the position of a gray value corner, where the axes of the local coordinate system align with the principal coordinate system. The gradient passes through a maximum at $\mathbf{x}_0$, i.e.,

$$f1_x = \text{extremum} \quad \text{and} \quad f1_y = 0$$
$$f1_{xx} = 0, \quad f1_{yy} = \text{extremum} \neq 0 \tag{3.21}$$

By using these conditions to simplify this two coupled nonlinear equations, in (3.20) we end up with

$$u \;\; = \;\; -(\bar{g}_2 - \bar{g}_1 - \frac{1}{2}f1_{yy}v^2)/f1_x \tag{3.22}$$
$$v \;\; = \;\; -f2_y/f1_{yy}$$

which are in closed-form in terms of the parameters of the modelled gray level surface. If one needs a numerical solution, the coupled nonlinear equation can be solved directly to obtain the vector $[\; u \quad v \;]^T$. We remark that it is not required to use gray value corner characteristics to extract features. However, we are interested in a closed-form solution, from which we can derive what we need.

3.3 Estimating the coordinate of the 2-D Velocity on x-y Plane

We have expressed the closed-form solution of u and v derived by Nagel, in terms of the parameters of the gray level surface model at time t_1 and t_2, as shown in the previous section.

Now, we try to find the position of this gray value corner on the image plane, which is assumed to be the feature point of the object. Returning to the model of a gray value surface, equation (3.8), taking partial derivatives with respect to u and v, and using the gray value corner characteristics

$$\frac{\partial}{\partial x}f(x,y) \;\; = \;\; \text{extremum} = M$$

$$\frac{\partial}{\partial y} f(x, y) = 0 \tag{3.23}$$

we obtain the position for gray value corner feature points as

$$x = \frac{f2_{yy}(M - f2_x) + f2_{xy}f2_y}{f2_{xx}f2_{yy} - (f2_{xy})^2} \tag{3.24}$$

$$y = -\frac{f2_{xy}(M - f2_x) + f2_{xx}f2_y}{f2_{xx}f2_{yy} - (f2_{xy})^2} \tag{3.25}$$

The choice of the feature point as a "gray value corner" can be used in measuring the feature point coordinates and optical flow, but it has a weakness. In the course of Nagel's derivation, for the purposes of simplifying that coupled nonlinear equations, Nagel imposed a condition that the axes of the local coordinate system must align with the principal curvature so that f_{xy} is equal to zero. Of course, one could achieve this by rotating the local coordinate system a certain angle. After computing the optical flow vector and the position of the feature point with respect to the local coordinate system, we need to rotate the local coordinate system back to align with the camera-centered coordinate system (CCCS). This will result in difficulty in measuring the feature point position and optical flow because the rotational motion of the moving object is changing over time.

To overcome this problem, we fix the local coordinate system and align it with the CCCS. We then exploit the characteristics of another special case of a "gray value extremum point" to compute the feature point optical flow and its coordinate as described next.

Now, we discuss the situation in which there exists a brightest or darkest point in a local area. In other words, there exists a extremum point on the gray level surface in a small area of interest. We select this point as the feature point and try to find the displacement vector $\mathbf{u} = [\ u \quad v\ ]^T$ and its location $\mathbf{x} = [\ x \quad y\ ]^T$ in the observed small window. It is not convenient in practical applications to rotate the axes of the local coordinate system for the environment around $\mathbf{x}_0$ to align with the principal curvature direction to force f_{xy} equal to zero. We will therefore keep this term in our derivation to compute optical flow $\mathbf{u}$ and its coordinate.

Taking partial derivatives of equation (3.16) with respect to u and v, and setting them to zero,

$$\sum[f1_x + f1_{xy}(y-v) + f1_{xx}(x-u)][g(\mathbf{x},t_2) - f1_0 - f1_x(x-u)$$
$$-f1_y(y-v) - \frac{1}{2}f1_{xx}(x-u)^2 - f1_{xy}(x-u)(y-v) - \frac{1}{2}f1_{yy}(y-v)^2] = 0$$

(3.26)

and

$$\sum[f1_y + f1_{xy}(x-u) + f1_{yy}(y-v)][g(\mathbf{x},t_2) - f1_0 - f1_x(x-u)$$
$$-f1_y(y-v) - \frac{1}{2}f1_{xx}(x-u)^2 - f1_{xy}(x-u)(y-v) - \frac{1}{2}f1_{yy}(y-v)^2] = 0$$

(3.27)

Simplication of Equations (3.26) and (3.27) yields a coupled nonlinear equation

$$[f1_x - f1_{xy}v - f1_{xx}u][\bar{g}_2 - \bar{g}_1 + u(f1_x - \frac{1}{2}f1_{xx}u)$$
$$+v(f1_y - \frac{1}{2}f1_{yy}v) - f1_{xy}uv] + \bar{x}^2 f1_{xx}[f2_x - (f1_x - f1_{xy}v - f1_{xx}u)] = 0$$

(3.28)

and

$$[f1_y - f1_{xy}u - f1_{xx}v][\bar{g}_2 - \bar{g}_1 + u(f1_x - \frac{1}{2}f1_{xx}u)$$
$$+v(f1_y - \frac{1}{2}f1_{yy}v) - f1_{xy}uv] + \bar{y}^2 f1_{yy}[f2_y - (f1_y - f1_{xy}u - f1_{yy}v)] = 0$$

(3.29)

We let $dg = \bar{g}_2 - \bar{g}_1$, which approximates the partial derivative of the gray value with respect to time. Now, letting $\mathbf{x}_0$ denote the position of a gray value extremum point projecting onto the image plane, by using the condition $f1_x = f1_y = 0$, $f1_{xx}, f1_{xy}, f1_{yy} \neq 0$, Equation (3.28) and (3.29) become

$$-[f1_{xx}u + f1_{xy}v][dg - \frac{1}{2}f1_{xx}u^2 - f1_{xy}uv - \frac{1}{2}f1_{yy}v^2]$$
$$+\bar{x}^2 f1_{xx}[f2_x + f1_{xy}v + f1_{xx}u] = 0$$

(3.30)

28

and

$$-[f1_{xy}u + f1_{yy}v][dg - \frac{1}{2}f1_{xx}u^2 - f1_{xy}uv - \frac{1}{2}f1_{yy}v^2]$$
$$+\bar{y}^2 f1_{yy}[f2_y + f1_{xy}u + f1_{yy}v] = 0 \qquad (3.31)$$

Assume a gray value surface model $f(\mathbf{x}, t)$ as Equation (3.8), with $f(\mathbf{x} - \mathbf{u}dt, t_1) = f(\mathbf{x}, t_2)$, i.e., the intensity at spatial location $\mathbf{x}$ at time t is the same at spatial location $(\mathbf{x} - \mathbf{u}t)$ at time t_0[8][9][10][11], where $dt = t_2 - t_1$ is very small. Expanding this function about $\mathbf{x}$ in a second-order Taylor series yields

$$f(\mathbf{x}, t_2) - f(\mathbf{x}, t_1)$$
$$\approx \begin{bmatrix} f1_x \\ f1_y \end{bmatrix}^T \begin{bmatrix} u \\ v \end{bmatrix} + \begin{bmatrix} u \\ v \end{bmatrix}^T \begin{bmatrix} f1_{xx} & f1_{xy} \\ f1_{xy} & f1_{yy} \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$
$$= f1_x u + f1_y v + \frac{1}{2}f1_{xx}u^2 + f1_{xy}uv + \frac{1}{2}f1_{yy}v^2 \qquad (3.32)$$

The left hand side of equation (3.32) is the approximate partial derivative with respect to time when dt is very small. If this local model holds around this extremum point, then $f1_x = f1_y = 0$, and we get

$$dg - \frac{1}{2}f1_{xx}u^2 - f1_{xy}uv - \frac{1}{2}f1_{yy}v^2 = 0 \qquad (3.33)$$

Substituting Equation (3.33) into Equations (3.30) and (3.31) yields

$$f1_{xx}u + f1_{xy}v = -f2_x \qquad (3.34)$$
$$f1_{xy}u + f1_{yy}v = -f2_y \qquad (3.35)$$

then,

$$u = \frac{f2_y f1_{xy} - f2_x f1_{yy}}{f1_{xx}f1_{yy} - f1_{xy}^2} \qquad (3.36)$$

$$v = \frac{f2_x f1_{xy} - f2_y f1_{xx}}{f1_{xx}f1_{yy} - f1_{xy}^2} \qquad (3.37)$$

The position of the extremum point in the observed window is easily obtained under the condition

$$\frac{\partial}{\partial x}f(x, y) = 0$$
$$\frac{\partial}{\partial y}f(x, y) = 0$$

would result in

$$x = \frac{f2_y f2_{xy} - f2_x f2_{yy}}{f2_{xx} f2_{yy} - (f2_{xy})^2} \tag{3.38}$$

$$y = \frac{f2_x f2_{xy} - f2_y f2_{xx}}{f2_{xx} f2_{yy} - (f2_{xy})^2} \tag{3.39}$$

x, y are computed at time t_2.

3.4 Computing the Measurement Vector of Position and Optical Flow

In the previous section, the optical flow vector $\mathbf{u} = [\ u \quad v\ ]^T$ and the position vector $\mathbf{x} = [\ x \quad y\ ]^T$ have been described in terms of the parameters of the modelled gray level surface, i.e., as functions of $\mathbf{f}$. Let $\mathbf{Y}$ be the vector $[\ x \quad y \quad u \quad v\ ]^T$. By using a Taylor expansion about the estimated $\hat{\mathbf{f}}$

$$\mathbf{Y}(\mathbf{f}) \approx \mathbf{Y}(\hat{\mathbf{f}}) + \mathbf{J}|_{\mathbf{f}=\hat{\mathbf{f}}} \cdot \varepsilon_{\hat{\mathbf{f}}} \tag{3.40}$$

where $\mathbf{Y}(\hat{\mathbf{f}}) = [\ \hat{x} \quad \hat{y} \quad \hat{u} \quad \hat{v}\ ]^T$, and $\mathbf{J}$ is a Jacobian matrix defined as $\mathbf{J} = \partial \mathbf{Y}/\partial \mathbf{f}|_{\mathbf{f}=\hat{\mathbf{f}}}$. Here, $\varepsilon_{\mathbf{f}} = \mathbf{f} - \hat{\mathbf{f}}$ is a Gaussian random vector with $\varepsilon_{\mathbf{f}} \sim \mathbf{N}[\mathbf{0}, (\mathbf{A^T W A})^{-1}]$, $\varepsilon = \mathbf{Y}(\mathbf{f}) - \mathbf{Y}(\hat{\mathbf{f}}) = \mathbf{J} \cdot \varepsilon_{\mathbf{f}}$ is also a random vector, and

$$E(\varepsilon) = E(\mathbf{J} \cdot \varepsilon_f) = \mathbf{J} E(\varepsilon_f) = 0 \tag{3.41}$$

$$\begin{aligned} cov(\varepsilon) &= cov(\mathbf{J}|_{\mathbf{f}=\hat{\mathbf{f}}} \cdot \varepsilon_{\mathbf{f}}) \\ &= E[\mathbf{J}|_{\mathbf{f}=\hat{\mathbf{f}}} \cdot \varepsilon \cdot \varepsilon^T \mathbf{J}^T|_{\mathbf{f}=\hat{\mathbf{f}}}] \\ &= \mathbf{J}|_{\mathbf{f}=\hat{\mathbf{f}}} E(\varepsilon_{\mathbf{f}} \cdot \varepsilon_{\mathbf{f}}^T) \mathbf{J}^T|_{\mathbf{f}=\hat{\mathbf{f}}} \end{aligned} \tag{3.42}$$

Note that $E(\varepsilon_{\mathbf{f}} \cdot \varepsilon_{\mathbf{f}}^T) = (\mathbf{A^T W A})^{-1}$, which is the error covariance matrix of the parameters $\mathbf{f}$. Although $\varepsilon_{\mathbf{f}}$ is unknown, $cov(\varepsilon_{\mathbf{f}})$ remains invariant to $\mathbf{f}$.

If we need to compute $\mathbf{Y}$, we have to determine $\varepsilon_{\mathbf{f}}$. We can use the Cholesky decomposition to factor $cov(\varepsilon_{\mathbf{f}}) = \mathbf{S} \cdot \mathbf{S}^T = \mathbf{S I S}^T$, where $\mathbf{S}$ is a lower triangular matrix, the matrix square root of $(\mathbf{A^T W A})^{-1}$, and $\mathbf{I}$ is the identity matrix. Then,

$$\mathbf{Y}(\mathbf{f}) = \mathbf{Y}(\hat{\mathbf{f}}) + \mathbf{J} \cdot \mathbf{S} n' \tag{3.43}$$

where $n' \sim N(0,1)$. We have shown that $\mathbf{Y}(\hat{\mathbf{f}})$, the estimate of $\mathbf{Y}(\mathbf{f})$, is unbiased. The error covariance is $\mathbf{J} \cdot cov(\boldsymbol{\varepsilon}_{\mathbf{f}})\mathbf{J}^T$. Because the value of $\mathbf{J}|_{\mathbf{f}=\hat{\mathbf{f}}}$ is based on the estimated $\mathbf{f}$, the different $\hat{\mathbf{f}}$ at different times yields different $\mathbf{J}|_{\mathbf{f}=\hat{\mathbf{f}}}$, while $cov(\boldsymbol{\varepsilon}_{\hat{\mathbf{f}}})$ is always the same. In the next section, we assess how the new measurement error covariance equation impacts the 3-D trajectory estimation results.

3.5 Summary and Discussion

We have shown that the measurement vector of feature point position and optical flow is a function of the estimated vector $\hat{\mathbf{f}}$, which includes the parameters of the measured spatiotemporal gray value pattern from image sequences. We have shown that if we properly choose the moving object's feature points as a "gray value corner" or "gray value extremum point", we could obtain the measurement vector by using this new method of computing feature point location and the optical flow. We have mentioned in Chapter 2 that the feature-based methods are based on feature extraction, followed by feature correspondence. Optical-flow-based methods depend upon computing optical flow, for the recovery of 3-D motion information. As we can see, obtaining reliable measurements of feature point position on the image plane, as well as optical flow is very important for feature-based and optical-flow-based methods, respectively.

Blostein and Chann had compared and generalized the strengths and weaknesses of feature-based and optical-flow-based methods. Based on their research into 3-D motion estimation, they had built a frame work of integrating the measurement of feature point and of optical flow and developed the hybrid feature/flow-based recursive 3-D trajectory estimation algorithm. The remaining task, which is how to obtain the measurements of feature point position and optical flow on the image plane, forms the new contribution in this project report. A new method of measuring feature point position and optical flow from image sequences has been developed in Section 3.3, completing the hybrid feature/flow-based algorithm for use in practical applications. From our research, we have also found that the measurement error is zero mean, but the measurement errors are correlated. The values of this error covariance matrix are

controlled by the gray-value resolution (see equations (3.40), (3.41), (3.42), (3.15)).

In summary, we have introduced the basic concept of differential techniques for computing optical flow. We have presented Nagel's approach of estimating optical flow. Based on his approach, we have developed a new method of computing measurement vectors of feature point position and optical flow in the special case of "gray value corners", we have also found another special case, "gray value extremum points" and derived algorithms for computing optical flow as well as the coordinate of feature point on the image plane. It is significant that if there exist gray-value "extremum points" on the moving object, these points are selected as feature points. Finally, we have discussed the application of this new measurement method in the hybrid feature/flow-based algorithm for 3-D motion estimation.

Chapter 4

Implementation Details and Simulation Results

In Chapter 2, we have introduced the basic concepts of recursive estimation and Kalman filtering, and presented details of the hybrid feature/optical-flow-based recursive 3-D motion estimation algorithm. As we can see from Section 2.2, the application of Kalman filtering requires the information of the process noise covariance matrix, the measurement error covariance matrix, a reasonably accurate estimate of the initial state vector and an initial estimate for the error covariance matrix. In this chapter, we will present detailed simulation results of the algorithm.

4.1 The Synthetic Testbed

Blostein and Chann had developed a flexible synthetic testbed for evaluating the hybrid feature/flow-based 3-D trajectory estimation algorithm. Our simulation is based on this hybrid 3-D recursive algorithm. We can use the synthetic testbed developed in [3] in our experiments. The numerical computation package MATLAB[13] is used in the implementation of the entire system. A global configuration file is applied to make the modification of simulation parameters convenient. This synthetic system, used in our experiment, contains a stationary "pin-hole" camera and a cube of $3 \times 3 \times 3$. All distances are measured in terms of focal lengths(FL). There are four arbitrarily non-coplanar vertices chosen as the feature points ($M = 4$), which are assumed to be

"gray value extremum points". The trajectory of the object is generated using the kinematic equations associated with the motion model. The propagation of the state vector in time is obtained by means of numerical integration (second- and third-order Runge-Kutta formulae in MATLAB).

4.2   Simulating Error Measurements

In Broida *et al*'s algorithm[4], the measurement includes only the positions of the observed feature points. The sources of error are due to spatial quantization and feature extraction. Error measurements are simulated by perturbing the error free values with random quantities.

$$p_i(\varepsilon) = \ p_i + r_p \quad r_p \sim N(0, \sigma_p^2 I) \tag{4.1}$$

where p_i is defined as error-free position measurement of the ith feature point, and r_p is modelled as white Gaussian noise, where σ_p is determined by the spatial (image) resolution. In [4], the measurement error covariance matrix R_k is a constant diagonal matrix

$$R_k = \begin{bmatrix} \sigma_p^2 & & & & & 0 \\ & \ddots & & & & \\ & & \sigma_p^2 & & & \\ & & & \sigma_p^2 & & \\ & & & & \ddots & \\ 0 & & & & & \sigma_p^2 \end{bmatrix} \tag{4.2}$$

In our project, the error measurements are based on equation (3.40). There are two sources of noise that control the values of the error vector and error covariance matrix: one is the spatial resolution, the other is the gray-value resolution, which is the main factor to influence the accuracy of the measurements. The following equation used is the error measurement of ith feature point.

$$Y_i(\varepsilon) = Y_i + J \cdot \varepsilon_f \tag{4.3}$$

where $Y_i = [\ p_i \quad u_i\ ]^T$, the error-free measurement and $J \cdot \varepsilon_f$ is the measurement error. Next, we will present how to simulate the error measurements.

First, we need to measure the gray values at the coordinate of the feature point in an observed small window, sized 3×3 in our experiment (see Figure 3.1). At time $t = t_k - dt$, we obtain the estimate $\mathbf{f}_1$. Then, we repeat the process at time $t = t_k$ to obtain the estimate $\mathbf{f}_2$. The relation between time interval dt and Δt is given by

$$dt = c \cdot \Delta t \tag{4.4}$$

(see Figure 4.1), where $\Delta t = t_{k+1} - t_k$ is the measurement time interval for vector Y: $\Delta t = 1$ is used in our simulation, and the scale factor c can be chosen from 0.01 to 0.05. We assume that the observed gray value surface is a surface that has a extremum point or "corner point". Considering the spatial (image) resolution, the standard deviation of the inter-pixel distance, in terms of the focal length is σ_p (given in Table 4.1). The measured position of the extremum point of the gray value surface projecting onto the x-y plane, which is also the position of the feature point, is given as

$$p_i(\varepsilon) = \begin{bmatrix} x_{io} \\ y_{io} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} + n_p \quad n_p \sim N(0, \sigma_p^2 I) \tag{4.5}$$

where $\begin{bmatrix} 0 & 0 \end{bmatrix}^T$ is the origin of that observed window. For the purposes of examining the performance of this hybrid algorithm with the error measurement (see equation (3.41)) under a wide range of conditions, the measured gray values are simulated to be random from 1 to 255 in our implementation. The following equation is used in simulating the gray value surface.

$$
\begin{aligned}
f(x, y | x_0, y_0, \sigma_p) &= f_0 + f_x(x - x_0) + f_y(y - y_0) + \frac{1}{2} f_{xx}(x - x_0)^2 \\
&\quad + f_{xy}(x - x_0)(y - y_0) + \frac{1}{2} f_{yy}(y - y_0)^2
\end{aligned}
\tag{4.6}
$$

where (x_0, y_0) is the position of the "gray value extremum point" or "gray value corner", which is very close to the origin $(0, 0)$ of the small window (see Equation (4.5)). We use equation (4.6) to assign gray values to the nine raster points of the observed small window.

Since the motion of the object and brightness of the background may be changing all the time, the surface parameter $\mathbf{f}$ of equation (4.6) should be changing over time. However, this time-variation is not convenient for simulation. For simplicity,

a Gaussian surface is used in our simulation. Obviously, the extremum point is the peak point of the Gaussian surface. The following equation is used in simulation

$$g(x, y | x_0, y_0, \sigma_p) = \frac{1}{2\pi\sigma^2} \exp\left\{-\frac{1}{2\sigma^2}\left[(x - x_0)^2 + (y - y_0)^2\right]\right\} \qquad (4.7)$$

By simulating randomized measurements of gray values, the time-varying surface parameter can be simulated. Then, from equations (3.9), (3.13), and least squares estimation, we will obtain an estimate of $\hat{f}$, so that the Jacobian matrix with respect to the parameters of the gray value surface can be computed. The gray value resolution, σ_g, is given in Table 4.2. The next problem is to simulate the random vector ε_f, representing the estimation error of **f**. Since the error covariance of **f**, $cov(\varepsilon_f)$, is known and time-invariant, we use Cholesky decomposition to factorize it.

$$cov(\varepsilon_f) = S \cdot S^T = SIS^T \qquad (4.8)$$

where S is a lower triangular matrix, the square root of $cov(\varepsilon_f)$, and I is the identity matrix. Finally, we obtain the measurement error:

$$J \cdot \varepsilon_f = J|_{f=\hat{f}} \cdot S \cdot \mathbf{n}' \quad \mathbf{n}' \sim N(0, I) \qquad (4.9)$$

To examine how different spatial and gray-value resolutions affect the estimation of state variables, a set of different spatial and gray-value resolutions are used in our experiments (See Table 4.1, and Table 4.2). The image plane, as in [1][2][3], is chosen to be a unit square (measured in terms of focal lengths). Its origin is located at the centre. The spatial resolution is easily obtained for a unit square image plane, for example, the spatial resolution of 256×256 pixels is $\sigma_p = 1/256 \approx 0.004$. The gray value resolution is obtained by converting the uniformly distributed error to the normal distribution, for example, 8-bit gray value resolution achieves $x + \sigma_g, \sigma_g = \sqrt{[0.5 - (-0.5)]^2/12} \approx 0.289$, where x is the measured gray value.

4.3 Measurement Error Covariance Matrix

The value of the measurement error covariance matrix affects the estimation accuracy of the Kalman filter. The following will show that how it influences the behaviour

36

spatial resolution (pixels)	σ_p (in focal lengths)
256×256	0.004
512×512	0.002
1024×1024	0.001

Table 4.1: Spatial resolutions for experiments

gray-value resolution (bit)	σ_g
8	0.2887
10	0.0722
12	0.018

Table 4.2: gray-value resolutions for experiments

of Kalman filtering. First, we introduce two matrix inversion lemmas, which will be used here.

Lemma 1:

$$(A_{11} + A_{12}A_{22}^{-1}A_{21})^{-1} = A_{11}^{-1} - A_{11}^{-1}A_{12}(A_{22} + A_{21}A_{11}^{-1}A_{12})^{-1}A_{21}A_{11}^{-1} \tag{4.10}$$

Lemma 2:

$$A_{11}^{-1}A_{12}(A_{22} + A_{21}A_{11}^{-1}A_{12})^{-1} = (A_{11} + A_{12}A_{12}^{-1}A_{21})^{-1}A_{12}A_{22}^{-1} \tag{4.11}$$

where A_{11}, A_{22}, A_{12}, A_{21} are $n \times n$, $m \times m$, $n \times m$, $m \times n$ matrices respectively. Let $A_{11}^{-1} = P(k+1|k+1)$, $A_{12} = H^T(k)$, $A_{22} = R(k)$, then, the measurement update of the estimation error covariance equation (2.12), and the Kalman gain equation (2.12) can be rewritten as:

$$P(k+1|k+1) = [P(k+1|k)^{-1} + H^T(k+1)R^{-1}(k+1)H(k+1)]^{-1} \tag{4.12}$$

$$K(k+1) = P(k+1|k+1)H^T(k+1)R^{-1}(k+1) \tag{4.13}$$

From equation (4.13), we can see that the Kalman gain is inversely proportional to the measurement error covariance $R(k+1)$, and directly proportional to the estimation error covariance $P(k+1|k+1)$. If we have error-free measurements (which is impossible), i.e., $R(k+1) = 0$, and after substituting into equation (4.13), we obtain

$$K(k+1) = P(k+1|k)H^T(k+1)[H(k+1)P(k+1|k)H^T(k+1)]^{-1} = H^{-1}(k+1) \tag{4.14}$$

and equation (2.11) becomes

$$\hat{x}(k+1|k+1) = x(k+1|k) + H^{-1}(k+1)[y(k+1) - H(k+1)\hat{x}(k+1|k)] = x(k+1|k+1)$$

$$(4.15)$$

which is the perfect estimate.

A small measurement error will increase the Kalman gain and the estimated state, $\hat{x}(k+1|k+1)$, will weight the measurement more heavily. On the other hand, a large measurement error will decrease the Kalman gain and the estimate state will weight the measurement less, and the state estimate will depend more on the system model. From the discussion above, we can see that accurate estimation measurement error is very important for Kalman filtering. The measurement error covariance is implemented based on the equation given as

$$R_k = J \cdot cov(\varepsilon_f) J^T \qquad (4.16)$$

Unlike that of Broida *et al*'s implementation of R_k which is a constant diagonal matrix for simplicity, our measurement is time-varying because of the time-varying $J|_{f=\hat{f}}$, even though $cov(\varepsilon_f)$ is time-invariant.

4.4   Process Noise Covariance Matrix

The process noise covariance matrix is also important to the overall performance of the Kalman filter. From equations (2.10), (2.12), (2.13), we can see that a large $Q(k)$ will increase $P(k+1|k+1)$ and $K(k+1)$, then, the state estimation will weight the measurements more heavily. The existence of large $Q(k)$ implies that the plant (state) equation is not accurate in describing the system and the predicted state is not reliable (see equation(2.9)), and the state estimate will be forced to depend more on the measurements (see equation (2.11)). From the discussion about the effects of the measurement error and process error on the Kalman filter, we can see that the Kalman filter can trade off its Kalman gain automatically based on the value of the measurement error and process error.

In our experiments as in [3], the process noise covariance matrix $Q(k)$ is implemented as a constant diagonal matrix with the diagonal element value of 5×10^{-6}.

4.5 Initial State Estimation

In section 2.2, we have listed the steps of an algorithm for performing Kalman filtering. The first, second, and third steps are to initialize the step index $k = 0$, the state estimate vector, $\hat{x}(0|0)$, and the covariance matrix, $P(0|0) = P(0)$, of the state estimate error. The most common way of simulating initial state estimate is to perturb the true initial states with random quantities. In some practice, however, it is not feasible because the amount of error added to each individual element in the true initial vector must not be too large for obtaining reliable initial values.

As in [3], we implement the initial state estimate as

$$\hat{x}_o(i) = [1 \pm (1 + r)e]x_o(i) \quad i = 1, \ldots, 8, 13, \ldots, 12 + 3M \qquad (4.17)$$

where r is a random number which is uniformly distributed in the interval $(0, 1)$, and e is the magnitude of error. It should be noted that there is no noise added to the quaternion state of the initial vector because of the assumption that the OCCS always initially aligns to the CCCS. The value of e is set as $e = 0.2$ in our experiments, which implies the error in the initial state estimate in the range of 20%–40%.

The next problem is to determine the initial error covariance matrix $P(0|0) = P(0)$. This is not difficult because the value of error added initial state vector is known. The Initial values of $P(0)$ is implemented as

$$P_0 = d^2 I \qquad (4.18)$$

where I is $(12 + 3M) \times (12 + 3M)$ matrix and d is the largest component in vector $|\hat{\mathbf{x}}_0 - \mathbf{x}_0|$. We want to examine the performance of Kalman filter when the error in initial state values is controlled inside the range of 20%–40%.

4.6 Simulation Results

4.6.1 Experiment Design

The experiments that we have designed are similar to [3]. Monte Carlo trials are used in our simulations. The number of Monte Carlo trials is set as 30. Each trials consists of 100 image frames, and the initial frame is not included. There are three different experiments to be performed. Experiment 1 compares our hybrid algorithm using measurements of feature point positions and optical flow from image sequences with that of Broida's feature-based algorithm[4]. Experiment 2 evaluates the filter performance using different spatial resolutions and gray value resolutions when the object moves at a constant translational velocity and a constant rotational velocity in which the motion model is accurately described. Experiment 3 examines the filter behaviour when there is a slight deviation from the motion model due to translational acceleration.

The bias and variance of the estimated states is the criteria used to measure the performance of the Kalman filter. Suppose we have obtained the value of one particular state variable $\hat{x}(t)$ at time t. The bias is calculated as

$$b = \frac{1}{R} \sum_{i=1}^{R} \hat{x}_i(t) - x(t) \tag{4.19}$$

and the variance is calculated as

$$v = \frac{1}{R} \sum_{i=1}^{R} [\hat{x}_i(t) - x(t)]^2 \tag{4.20}$$

where R is the total number of Monte Carlo trials, $x(t)$ is the true value of this particular state variable at time t, and the subscript i denotes the trial number. Since the central issue of this project is 3-D trajectory estimation, the states of position (s_1 and s_2), translational velocity (s_3-s_5), and rotational velocity (s_{10}-s_{12}) are extremely important. The bias and variance results presented later in this Chapter will include these states only. Note that all the results corresponding to the position and translational velocity states have been scaled by $Z_R(t)$, the true depth of the object rotation centre.

Both the extended Kalman filter(EKF) and iterated extended Kalman filter(IEKF) have been used in our experiment. Only the results of the IEKF are shown because the performance of IEKF is slightly better than that of EKF based on our simulation results.

4.6.2 Experiment 1: Algorithm Comparison

In this experiment, we compared our hybrid feature/flow-based algorithm using measurement of feature point positions and optical flow from image sequence with that of Broida's purely feature-based algorithm. The main difference between these two algorithms is that Broida's algorithm uses only feature point position measurements on the image plane while our hybrid algorithm makes use of both image plane position and optical flow measurements. The motion parameters used in these two algorithms are the same, given in Table 4.3. The spatial resolution used is 512×512 pixels in measuring the feature point on the image plane for both algorithms. The gray value resolution used is 10 bits for our hybrid algorithm only. Because Broida's algorithm does not use gray value resolution in measuring the feature point positions, the object is moving at a constant translational velocity and a constant rotational velocity.

Initial position $S_R(t_0)(FL)$	$\begin{bmatrix} -6 & 10 & 28 \end{bmatrix}^T$
Translational velocity T(FL/frame)	$\begin{bmatrix} 0.05 & -0.1 & -0.2 \end{bmatrix}^T$
Rotational velocity ω(radians/frame)	$\begin{bmatrix} 0.03 & 0.04 & 0.05 \end{bmatrix}^T$

Table 4.3: Motion parameters for Experiment 1 and 2

The simulation results are shown in Figure 4.1 (bias) and 4.2 (variance). The solid line represents the result obtained by using our hybrid algorithm and the dashed line represents the result obtained by using Broida's algorithm. From the simulation results, we can see the obvious improvements achieved by the hybrid algorithm. The estimates of using hybrid algorithm converge more quickly than that of using Broida's, especially the translational velocity states. We have also experimented with different gray value resolutions. Even using a gray value resolution of 8 bits the hybrid

algorithm still performs better than Broida's algorithm.

4.6.3   Experiment 2: Constant Translation and Rotation

In this experiment, the motion of the object is assumed to be constant translation and rotation. Table 4.3 gives the motion parameters used in this experiment.

We have carried out this experiment as two parts: Experiment 2.1 and Experiment 2.2. In the former, we have selected an arbitrary value of spatial resolution (see Table 4.1) and varied the gray-value resolution (see Table 4.2) to see the effects of changing gray-value resolution on the state estimates. In the latter, we have arbitrarily chosen a gray-value resolution and varied the spatial resolution. The purpose of this experiment is to examine the effects of using different spatial and gray-value resolutions on the state estimates. The choice of spatial and gray-value resolution affects the measurement error which will be propagated into the state estimate (see equation (4.9), and discussion in section 4.3).

The simulation results of Experiment 2.1 are shown in Figure 4.3 (bias) and 4.4 (variance), where the spatial resolution 512×512 pixels is fixed and three different values of gray-value resolution are used: the solid line, dashed line, and dash-dot line represent 8-bit, 10-bit, and 12-bit resolution, respectively. The performance of using these three different gray-value resolutions with same spatial resolution are all stable and convergent. Not surprisingly, the behaviour of 8-bit resolution is worse than that of 10-bit and 12-bit resolution. The behaviour of 10-bit is better and the performance of 12-bit is the best. We have experimented with all pairs of spatial resolutions and gray-value resolutions, and found that by fixing a spatial resolution, the performance of 12-bit is the best. This is due to the measurement error covariance matrix $J \cdot cov(\varepsilon_f) J^T$, where the value of $cov(\varepsilon_f)$ is controlled mainly by the gray value resolution (see Equation (3.15)).

The simulation results of Experiment 2.2 are shown in Figure 4.3 (bias) and 4.4 (variance), this time, the gray-value resolution of 10-bit is fixed with three different spatial resolution. Similarly, the performance of 256×256 is good, that of 512×512 pixels is better, and that of 1024×1024 pixels is the best. Since the results using
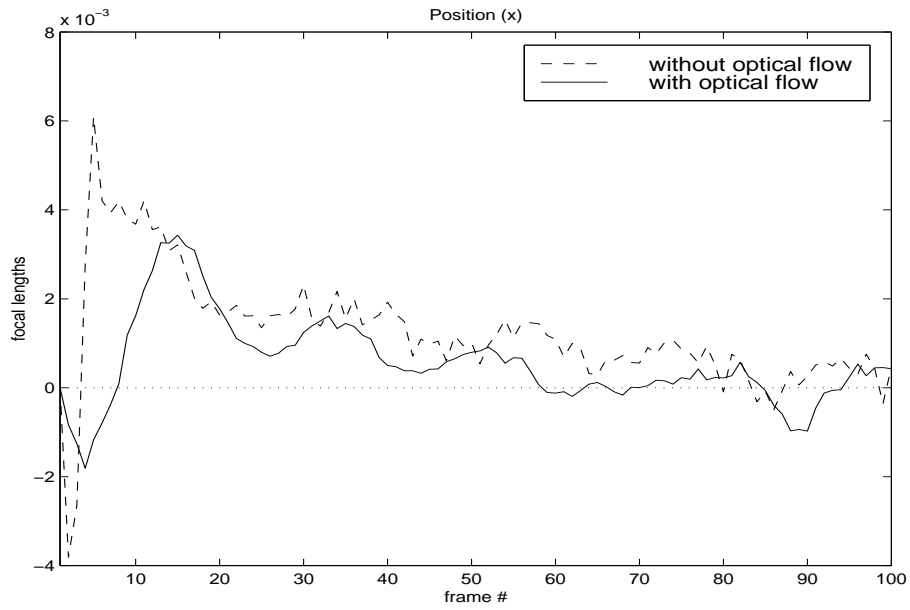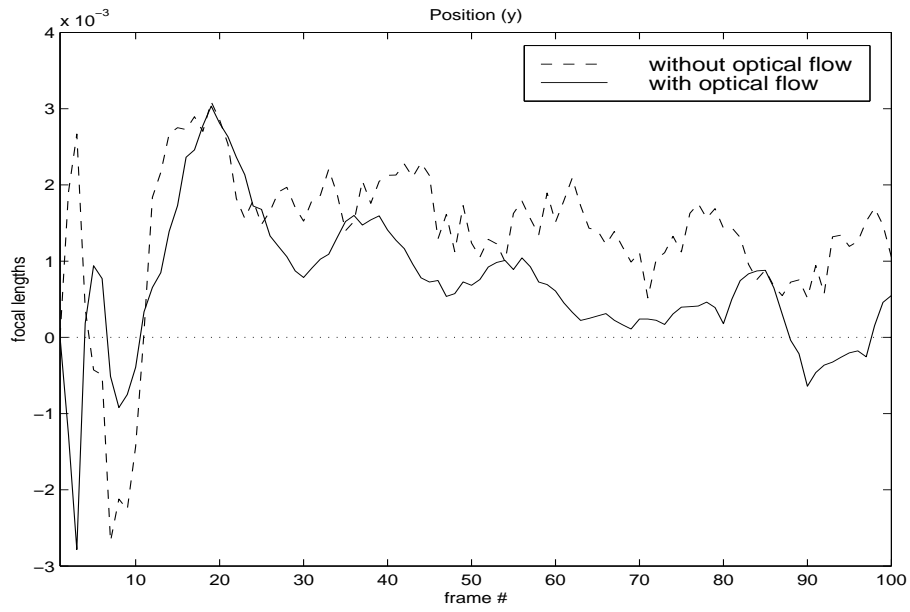
Figure 4.1: Experiment 1: Results of Monte Carlo simulations (bias)
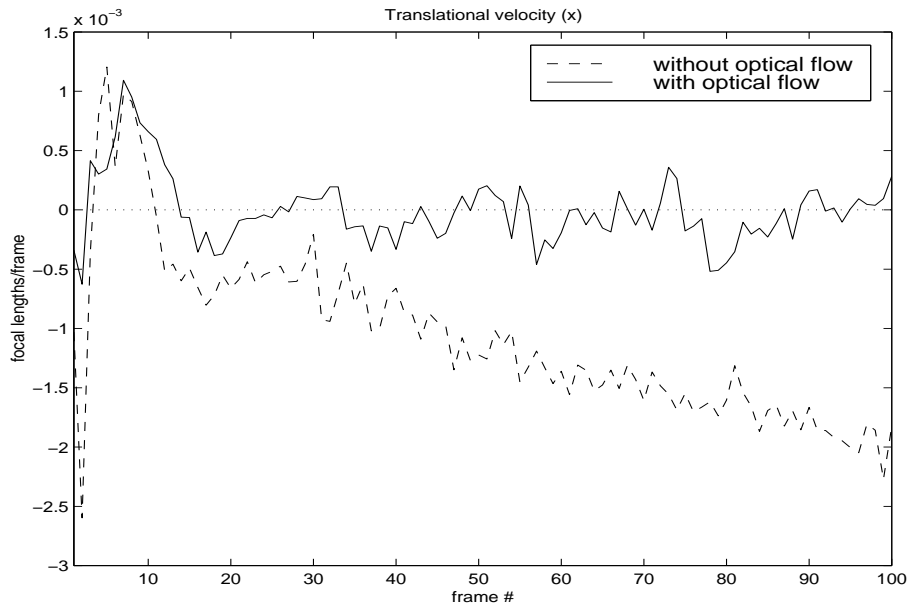
(c)
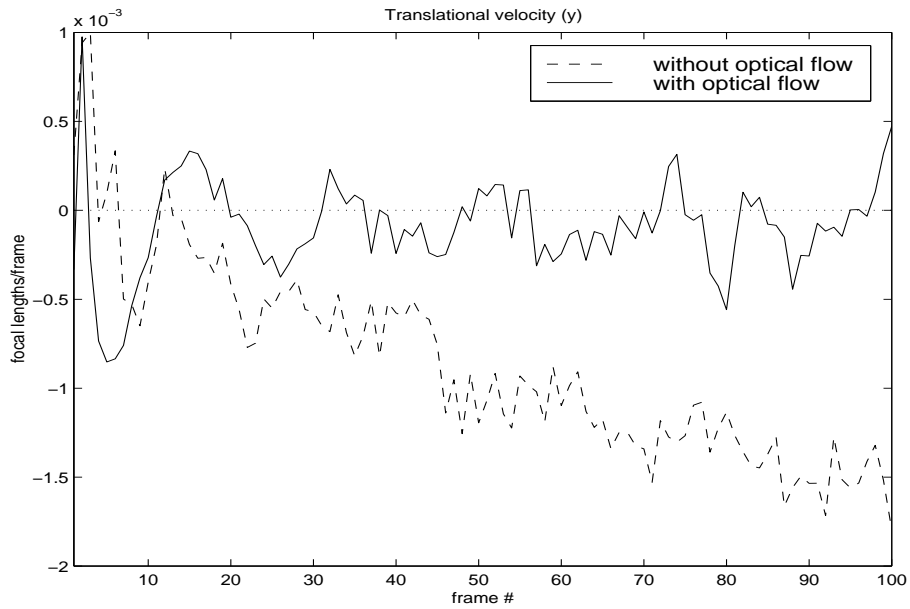
(d)

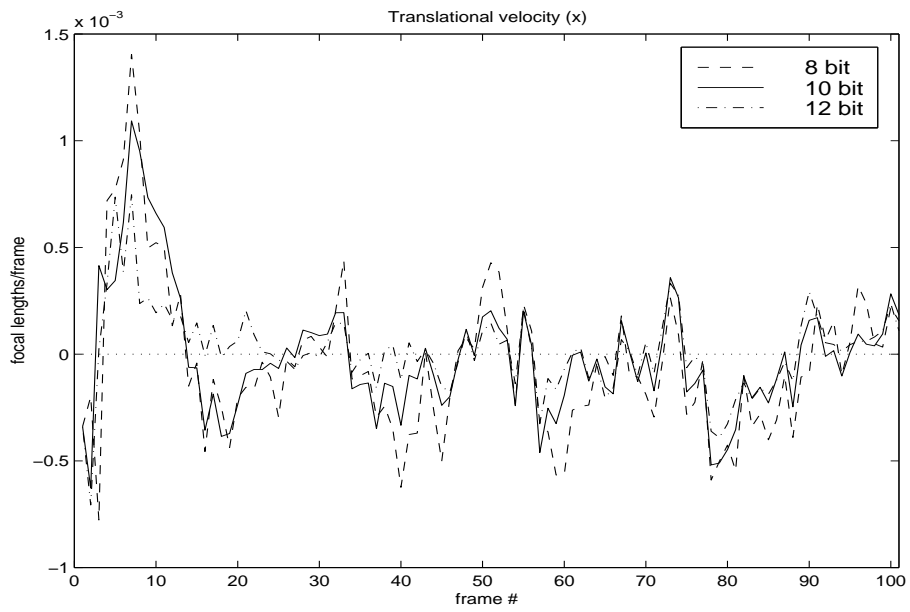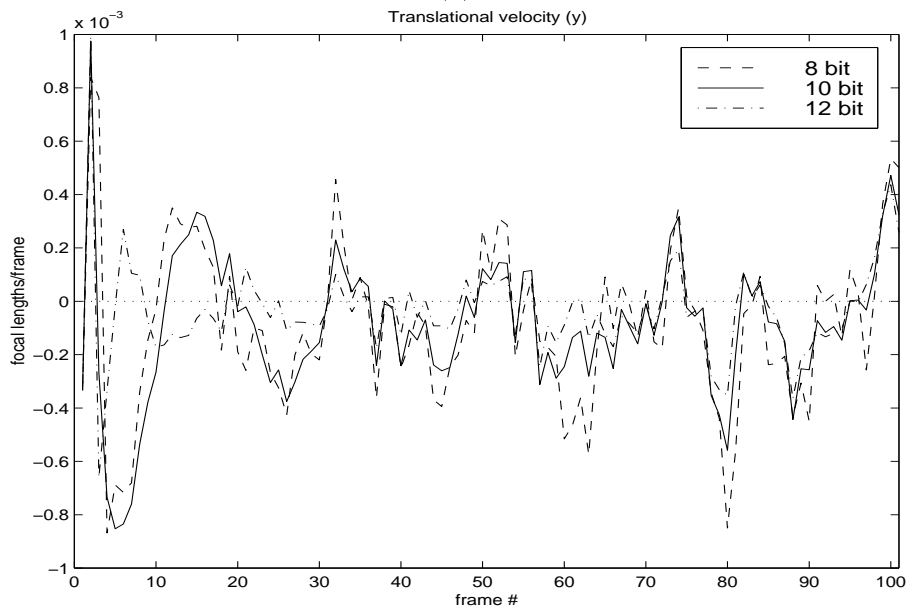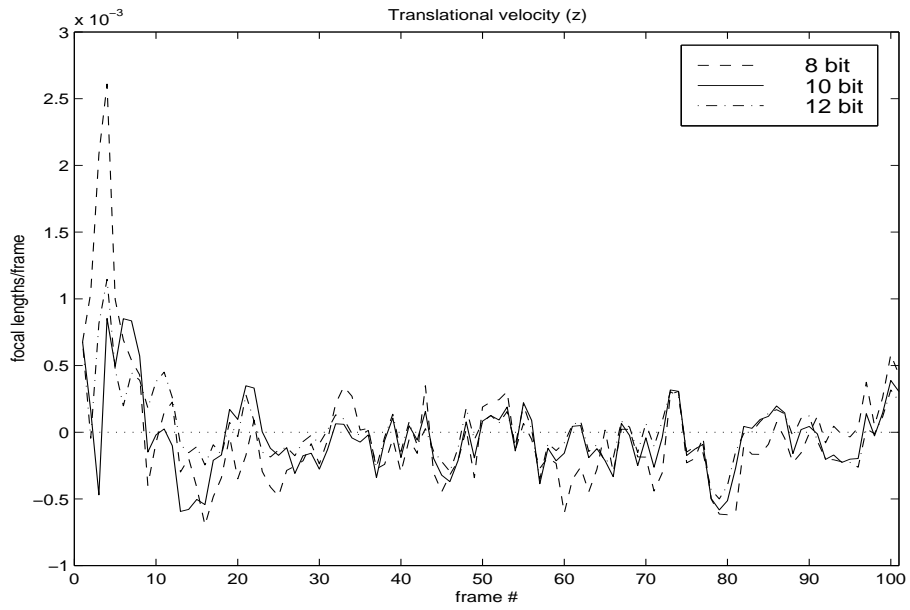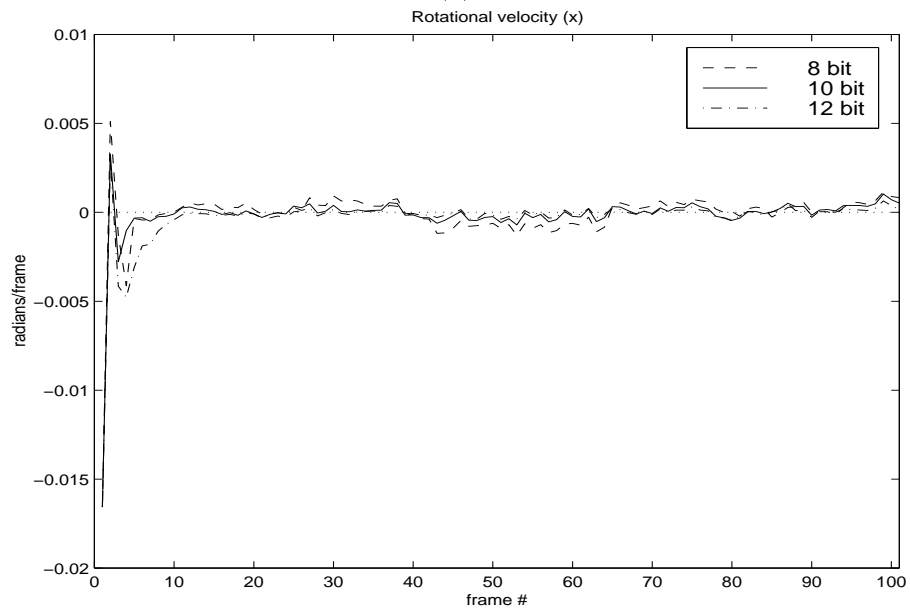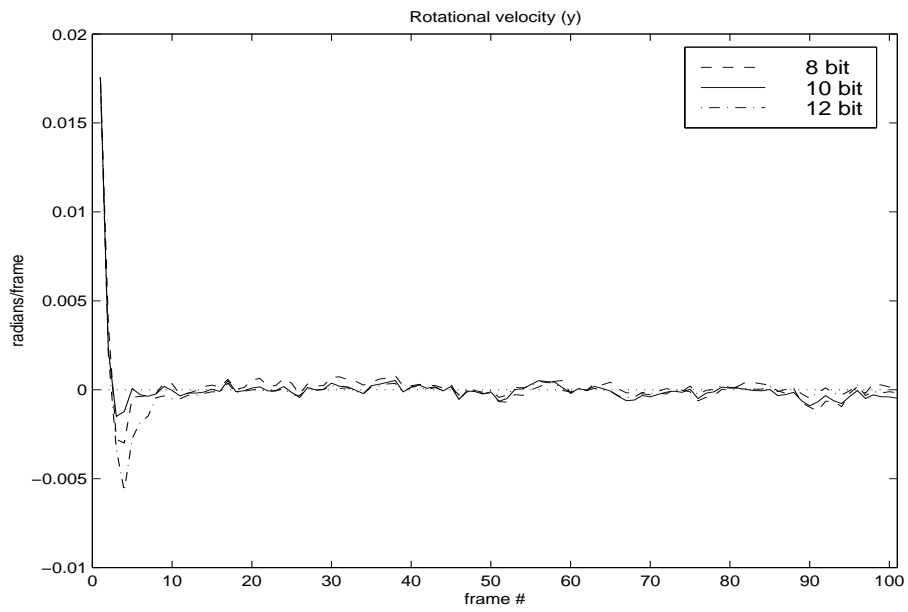Figure 4.1: Experiment 1: Results of Monte Carlo simulations (bias)
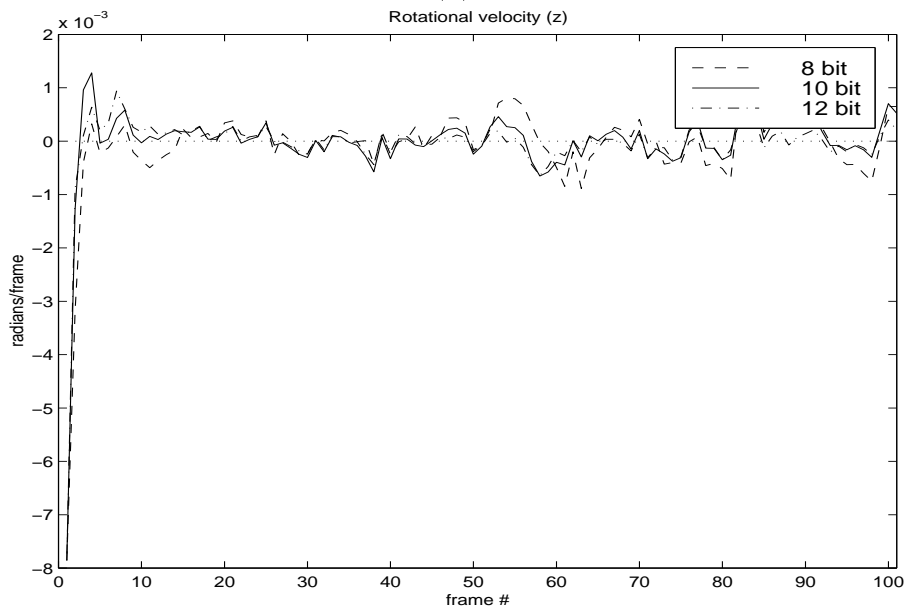
44
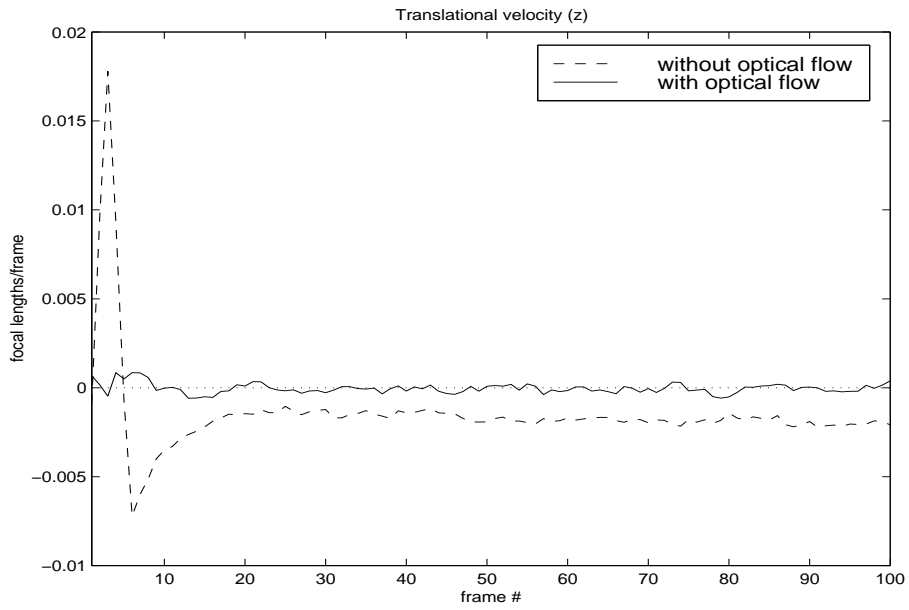
(e)
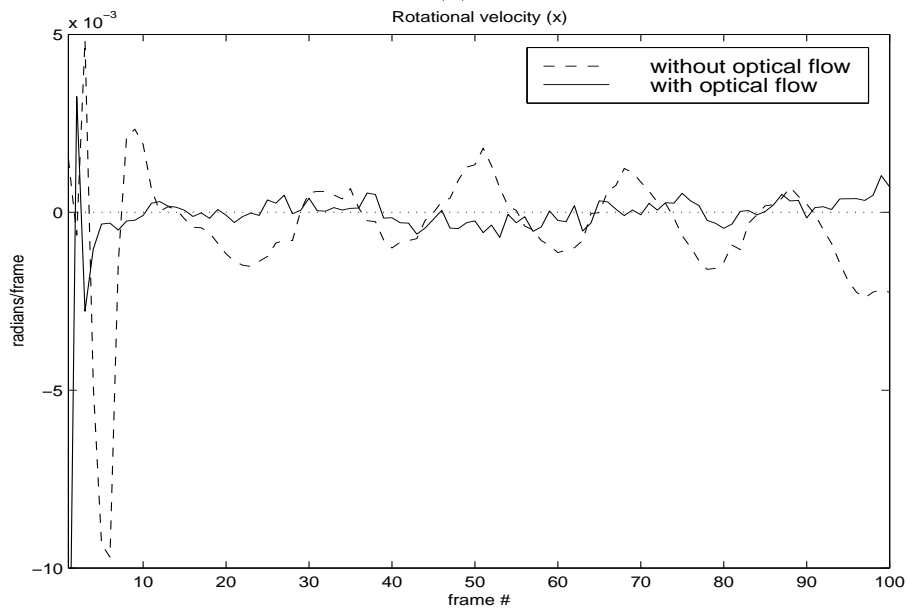
(f)

Figure 4.1: Experiment 1: Results of Monte Carlo simulation (bias)
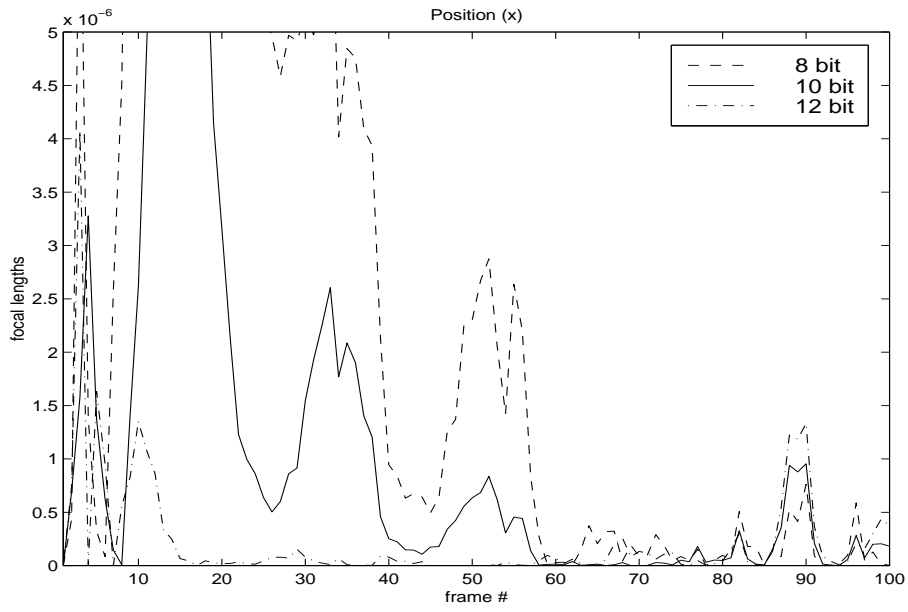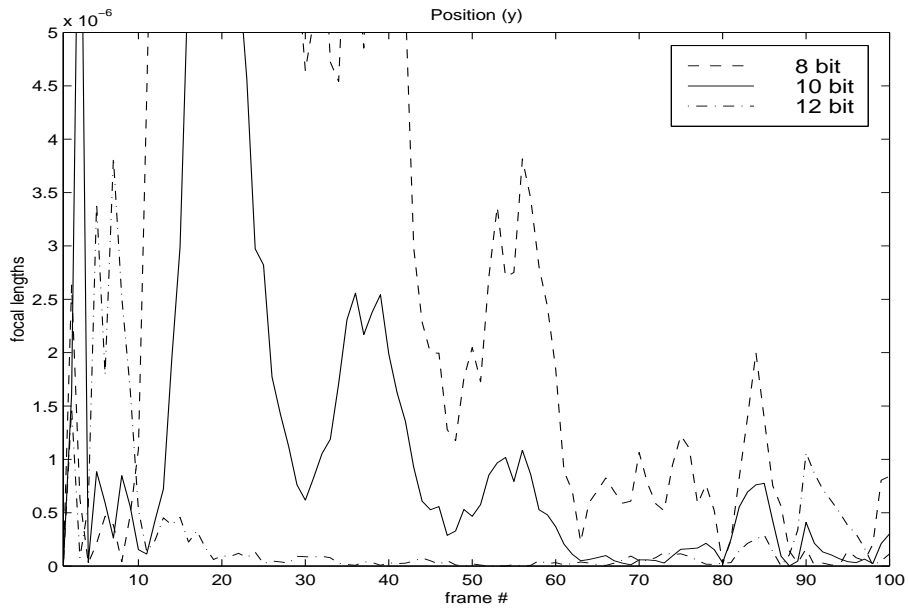
(g)
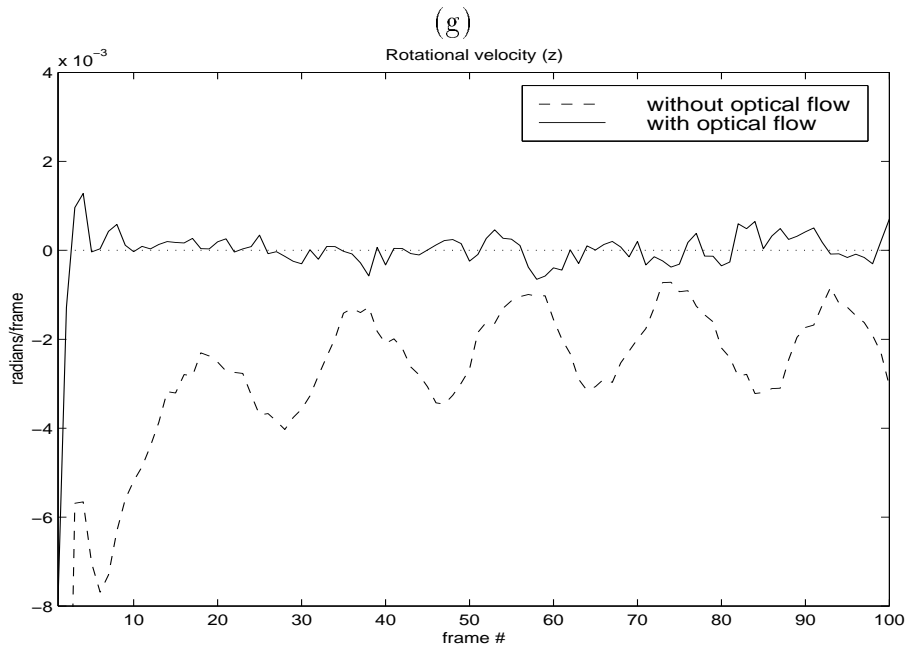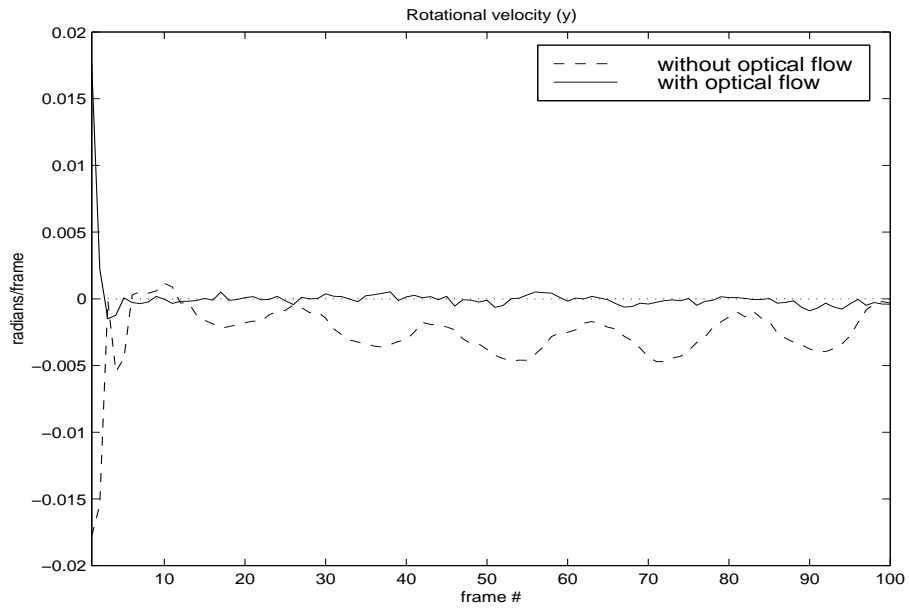
(h)

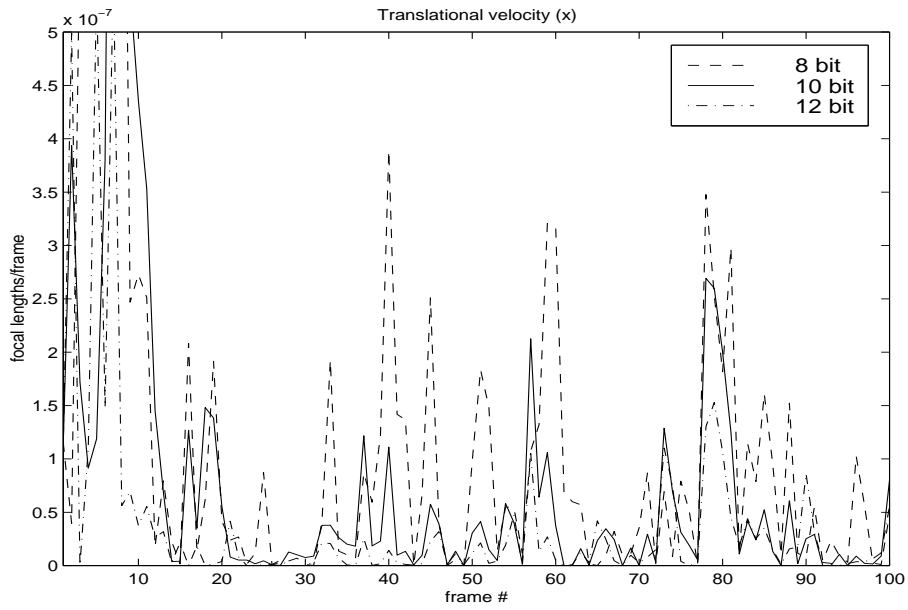Figure 4.1: Experiment 1: Results of Monte Carlo simulation (bias)
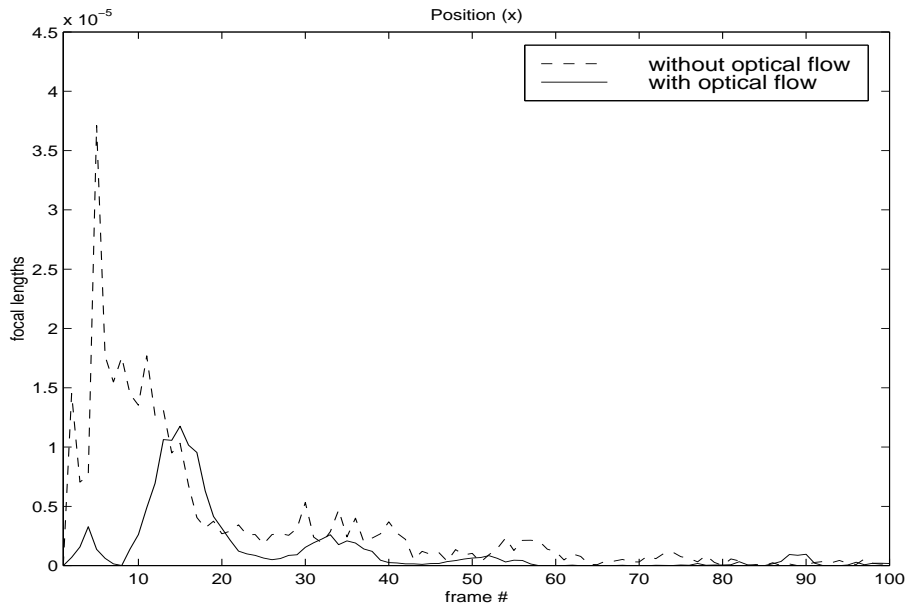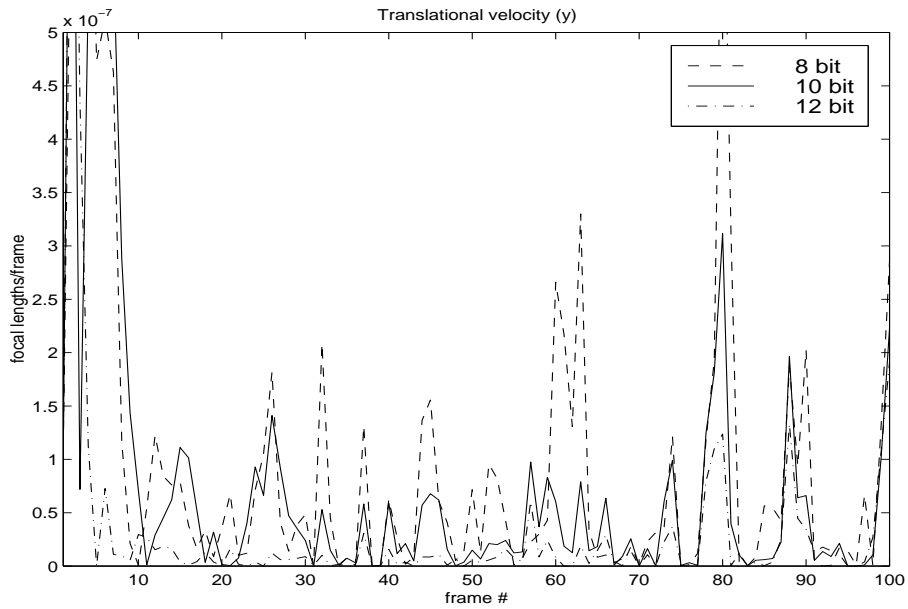
(a)
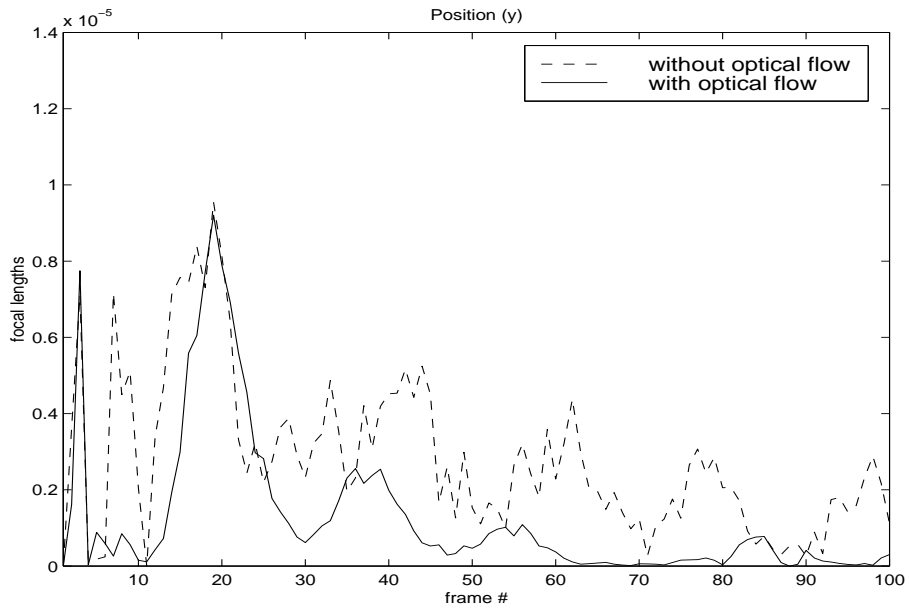
(b)

Figure 4.2: Experiment 1: Results of Monte Carlo simulations (variance)
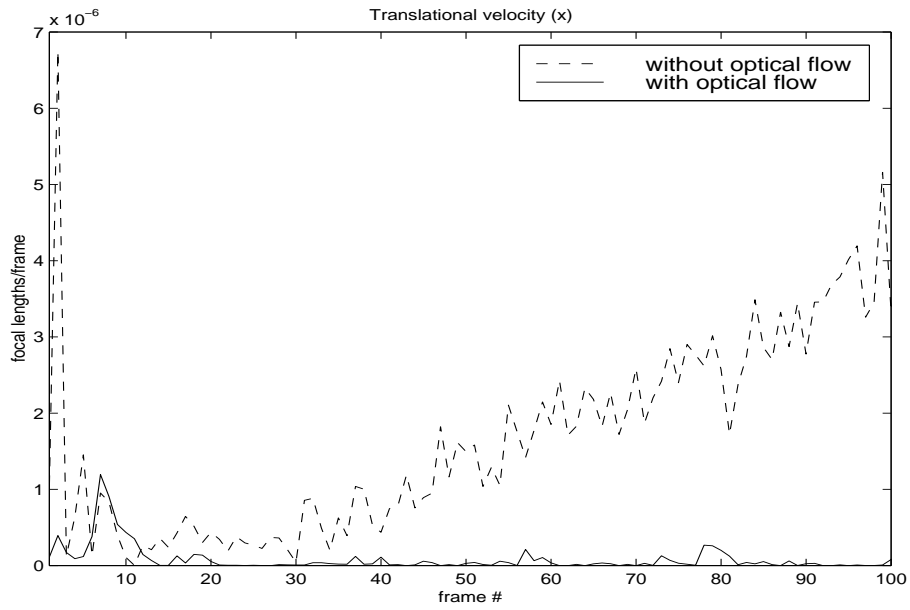
(c)
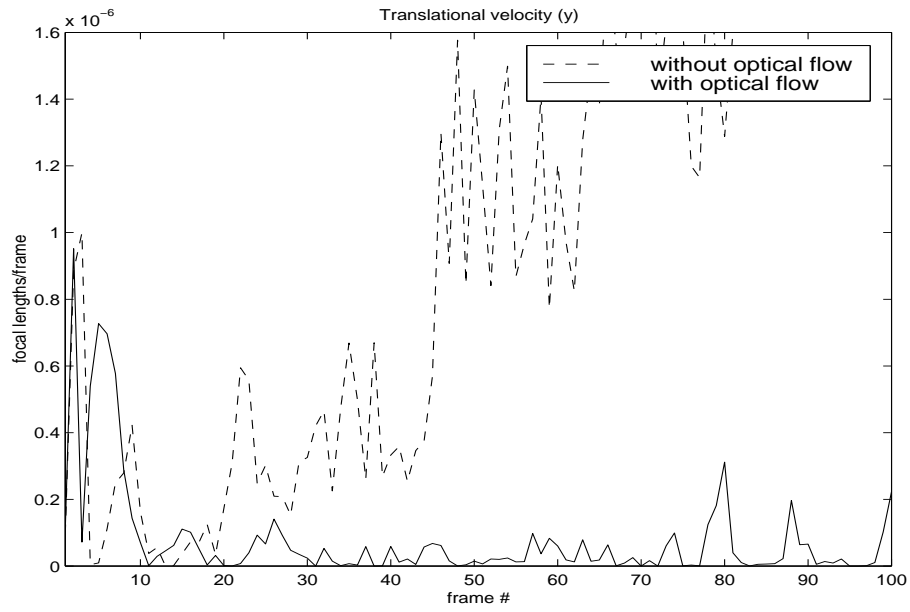
(d)

Figure 4.2: Experiment 1: Results of Monte Carlo simulations (variance)

48
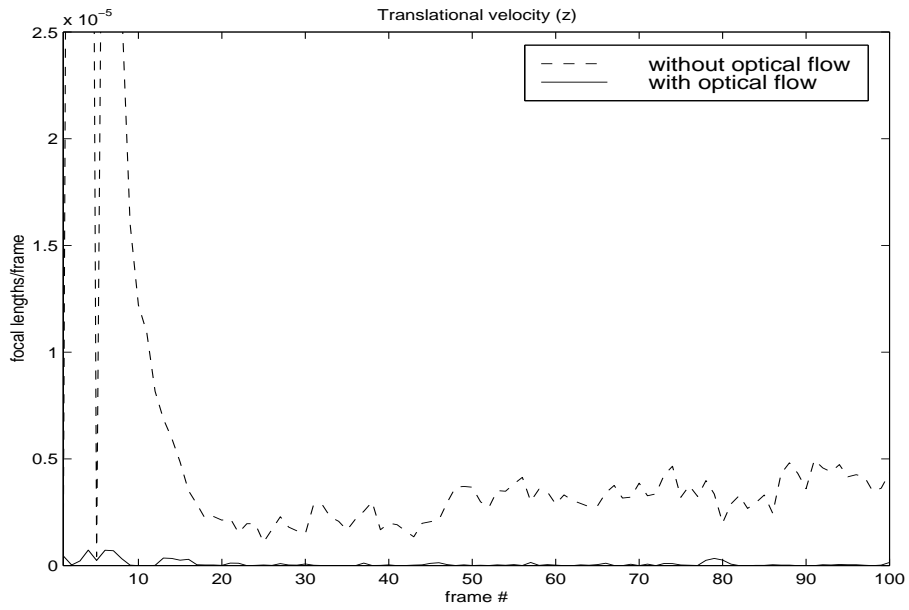
(e)
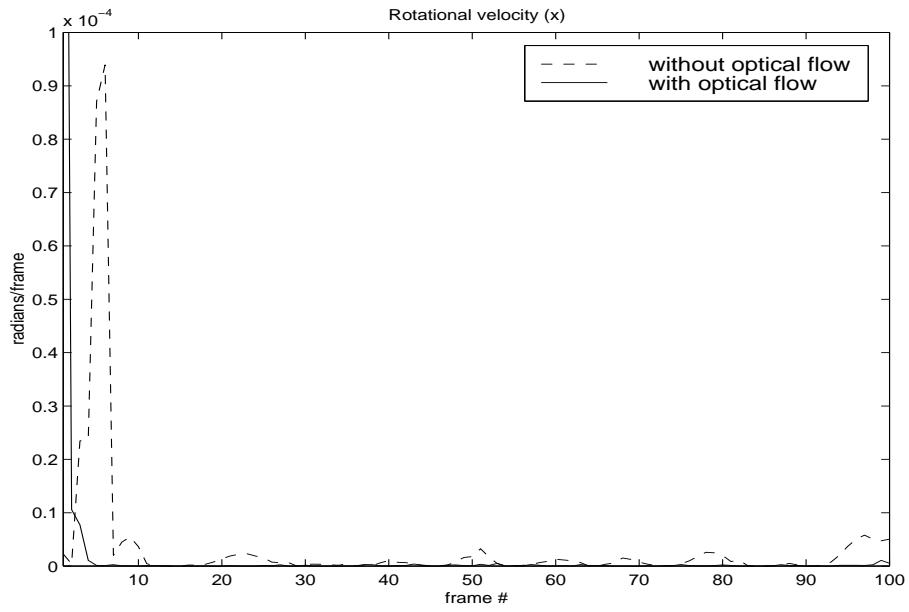
(f)

Figure 4.2: Experiment 1: Results of Monte Carlo simulation (variance)

(g)
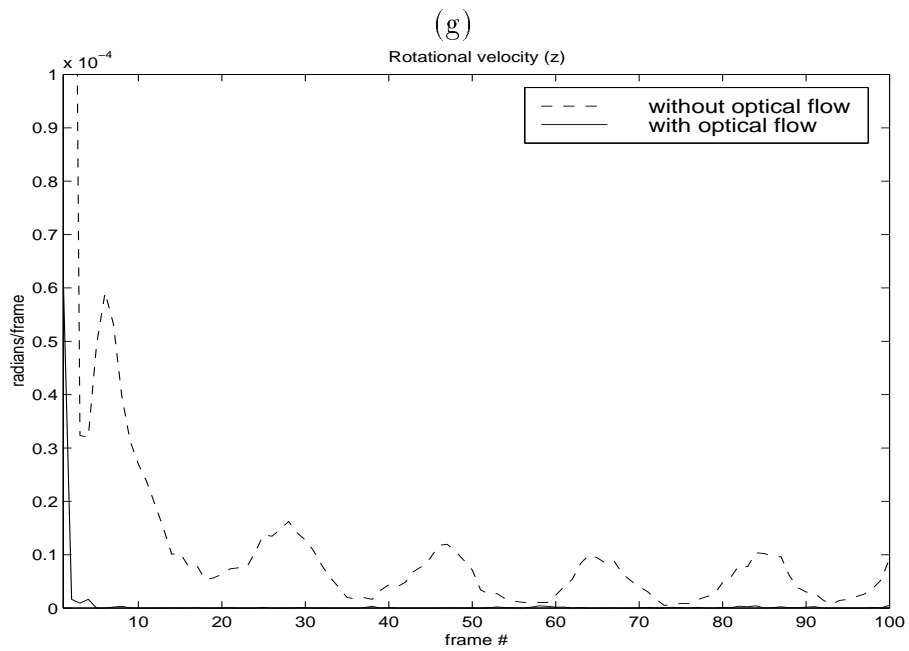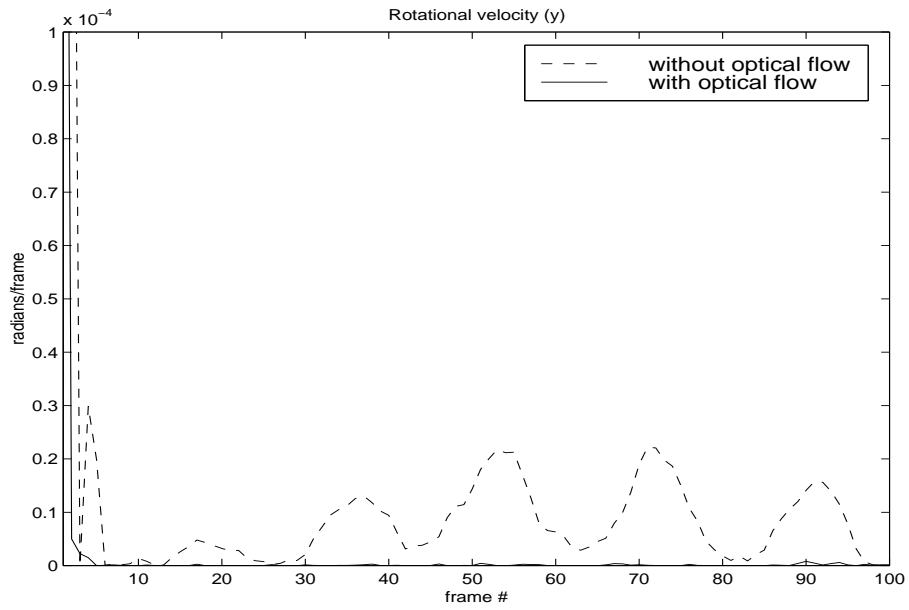
(h)

Figure 4.2: Experiment 1: Results of Monte Carlo simulation (variance)

three different spatial resolutions are too close to be visually distinguishable, the mean squared errors are listed in Table 4.4.

Spatial Resolution (pixels)	Position (x)	Position (y)	Trans. Vel. (x)	Trans. Vel. (y)
256×256	1.25×10^{-6}	1.26×10^{-6}	4.03×10^{-8}	4.20×10^{-8}
512×512	1.24×10^{-6}	1.24×10^{-6}	4.03×10^{-8}	4.18×10^{-8}
1024×1024	1.21×10^{-6}	1.21×10^{-6}	4.01×10^{-8}	4.07×10^{-8}
Spatial Resolution (pixels)	Trans. Vel. (z)	Rot. Vel. (x)	Rot. Vel. (y)	Rot. Vel. (z)
256×256	5.53×10^{-8}	1.06×10^{-7}	1.02×10^{-7}	7.95×10^{-8}
512×512	5.53×10^{-8}	1.05×10^{-7}	1.01×10^{-7}	7.90×10^{-8}
1024×1024	5.51×10^{-8}	1.05×10^{-7}	1.01×10^{-7}	7.89×10^{-8}

Table 4.4: Mean squared error with gray value resolution of 10 bit

4.6.4   Experiment 3: Translational Acceleration

Experiment 2 simulates the motion model of the object at constant translational and rotational velocity. In this experiment, we simulate a motion with translational acceleration with this constant model, which means that the motion model does not accurately describe the motion of the object. The purpose is to observe the performance of the hybrid feature/flow-based algorithm in this kind of non-ideal situation. The position of the rotation centre, at any time, is given by (4.20) where (a) denotes the translational acceleration. Table 4.5 shows the motion parameters used in this experiment.

Initial position $S_R(t_0)(FL)$	$\begin{bmatrix} -5 & 5 & 20 \end{bmatrix}^T$
Translational velocity T (FL/frame)	$\begin{bmatrix} 0.05 & -0.1 & -0.2 \end{bmatrix}^T$
Rotational velocity ω(radians/frames)	$\begin{bmatrix} 0.03 & 0.04 & 0.05 \end{bmatrix}^T$
Translational acceleration a(FL/frame)	$\begin{bmatrix} 0.0005 & -0.001 & 0.002 \end{bmatrix}^T$

Table 4.5: Motion parameters for Experiment 3

All pairs spatial resolution and gray-level resolution were tried. Although the

51
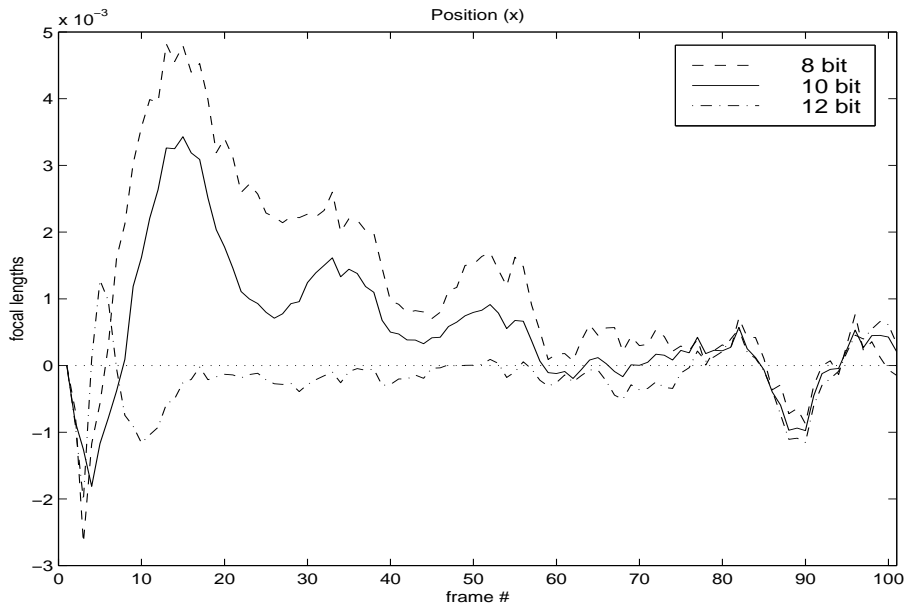
(a)
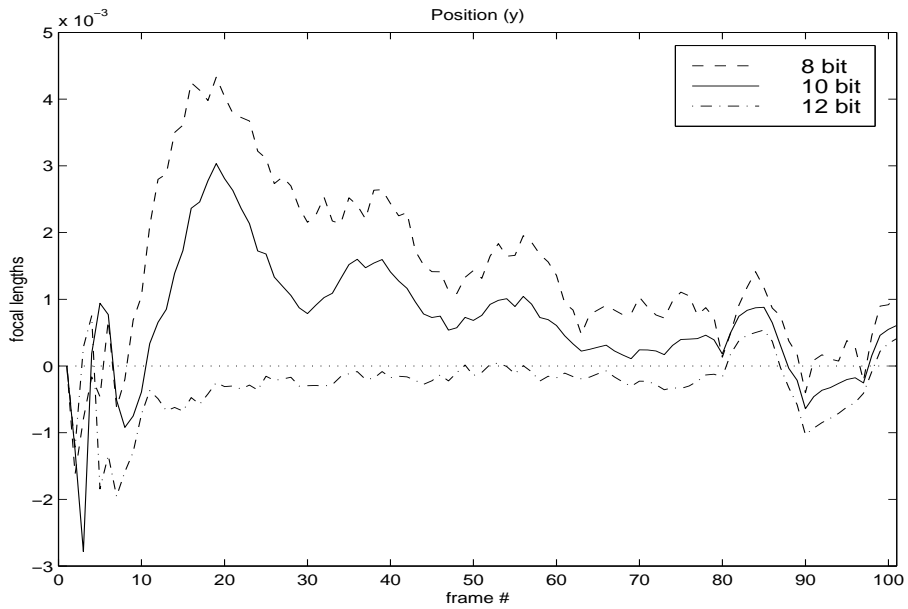
(b)

Figure 4.3: Experiment 2: Results of Monte Carlo simulations (bias)

(c)

(d)

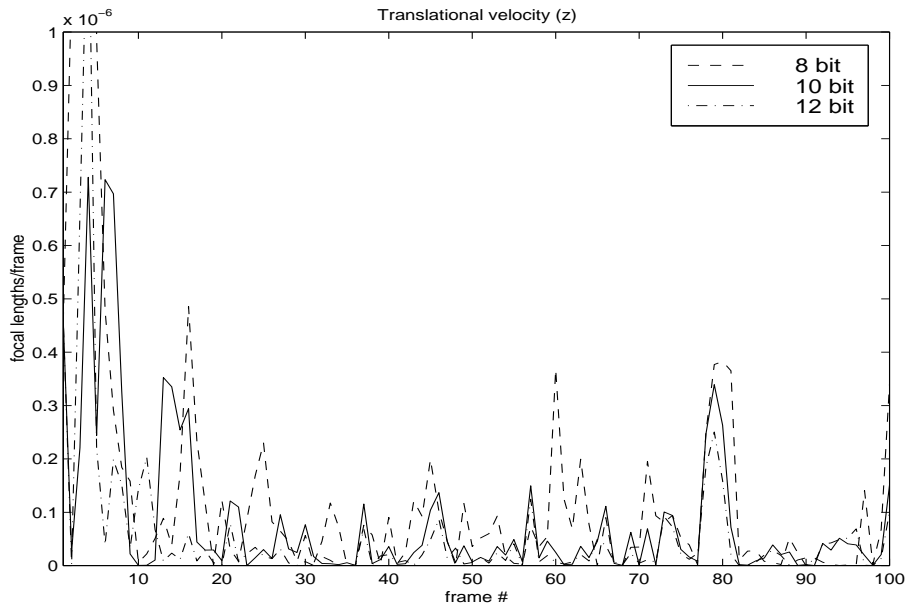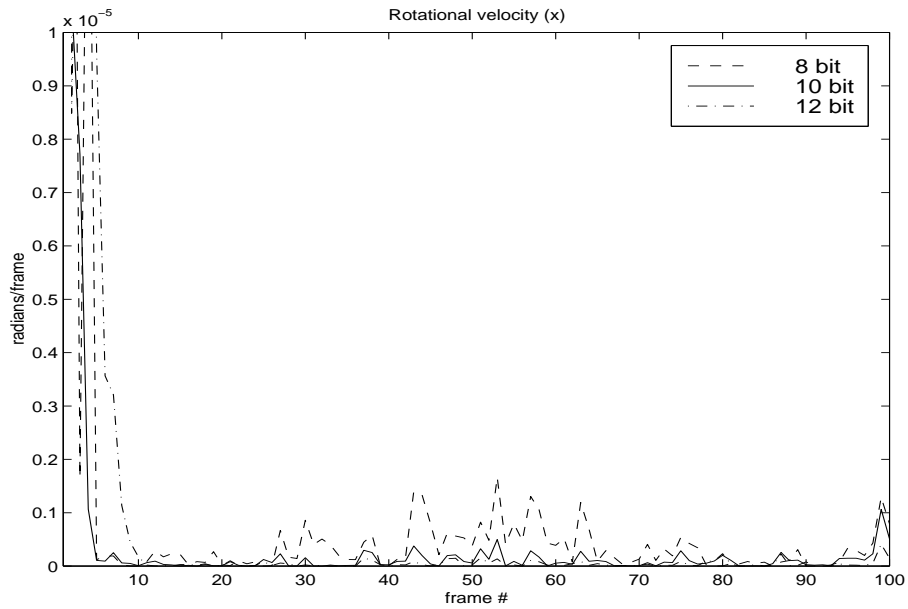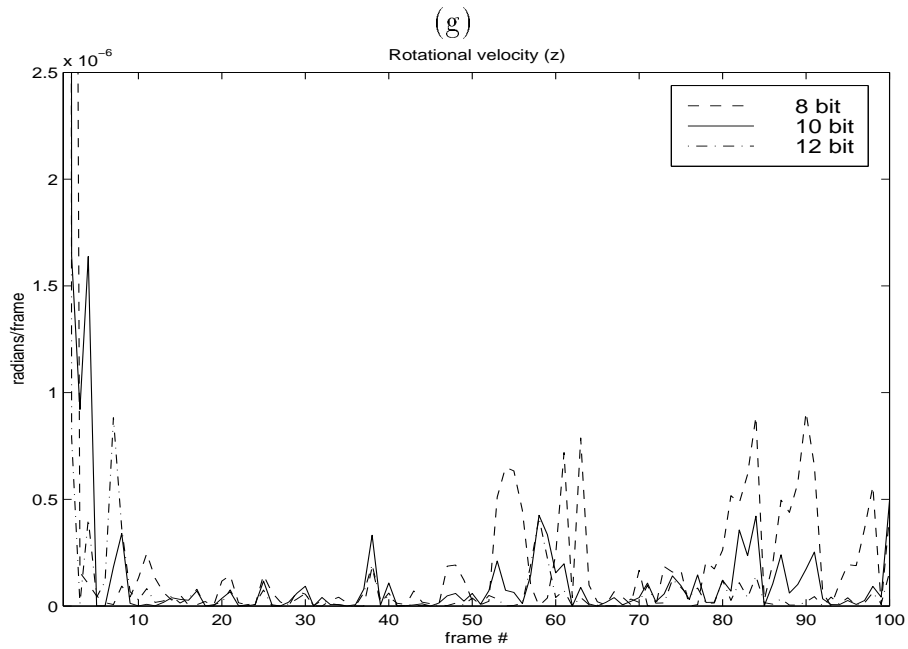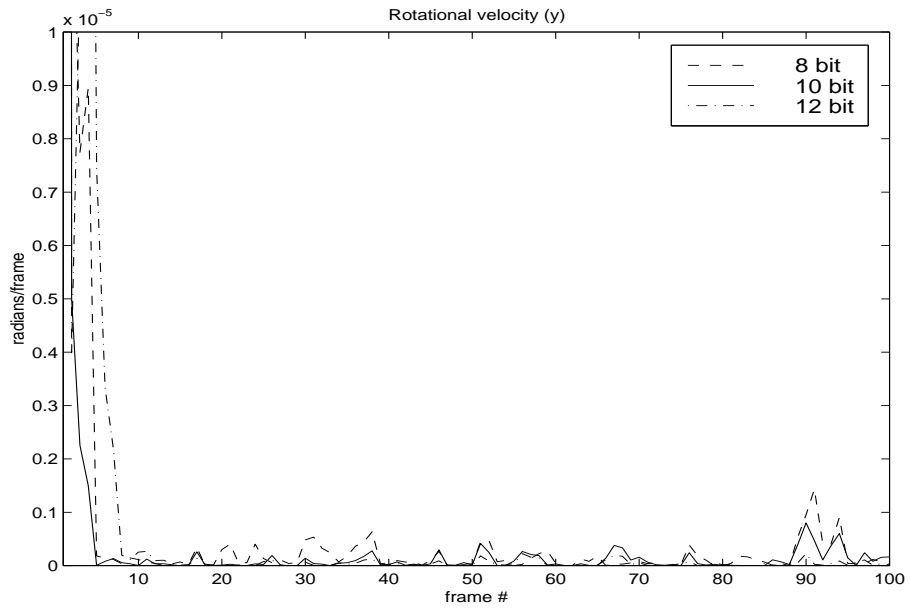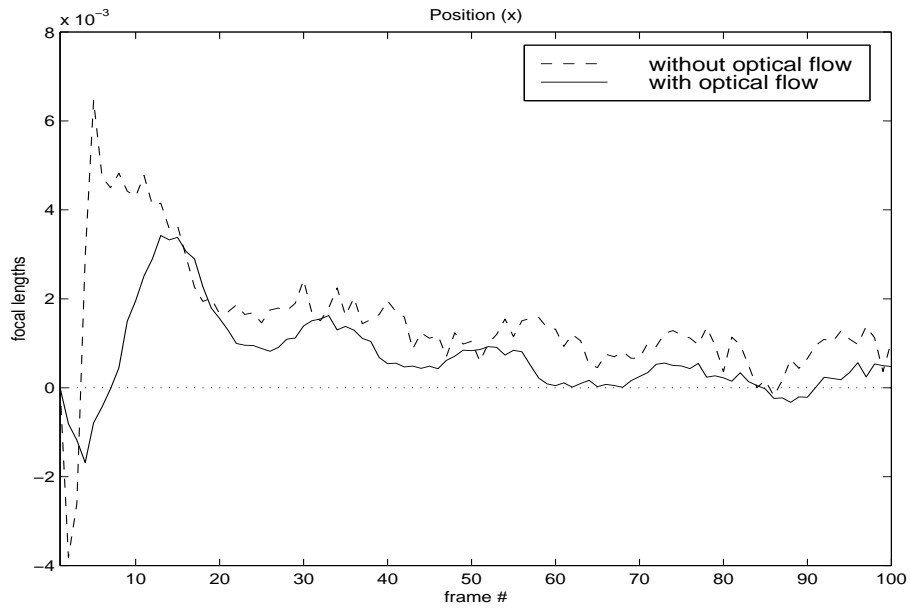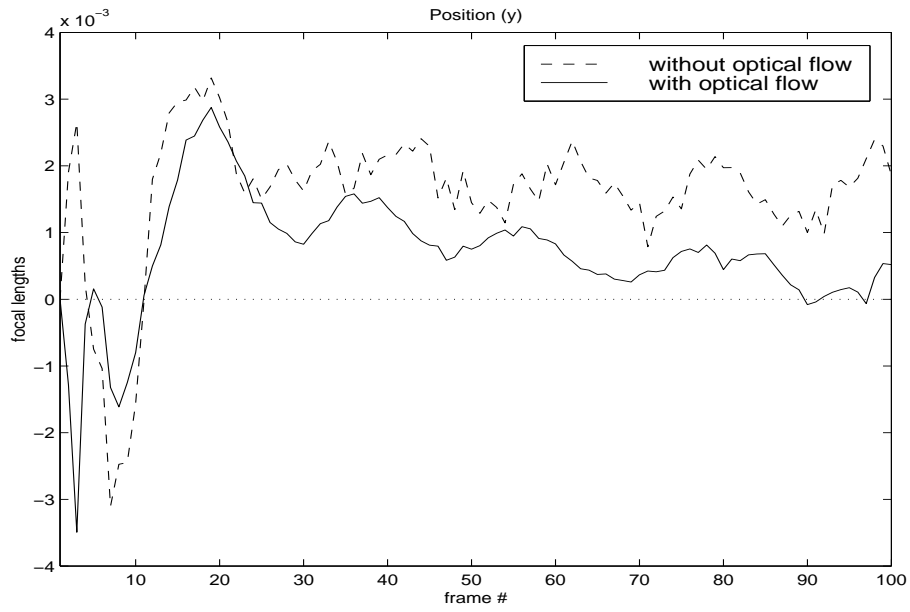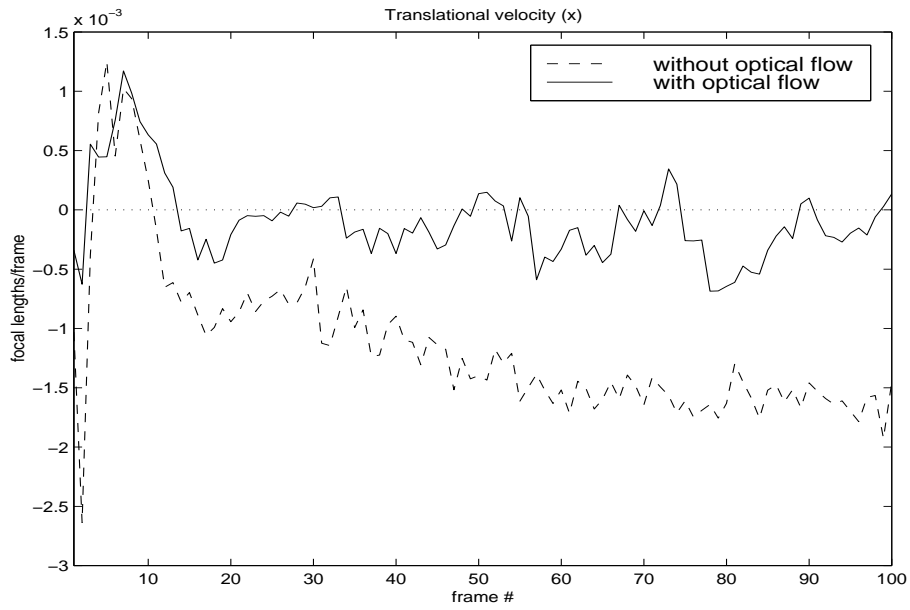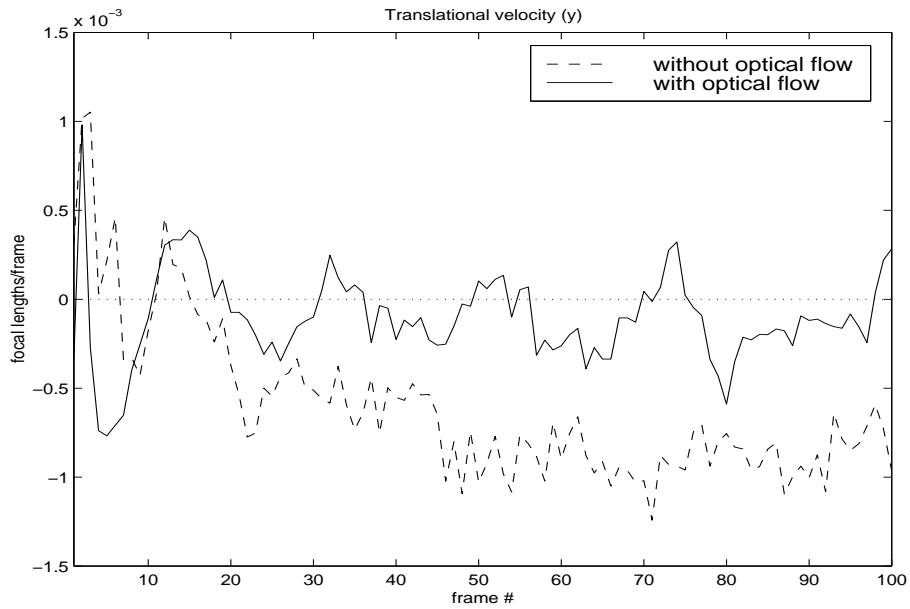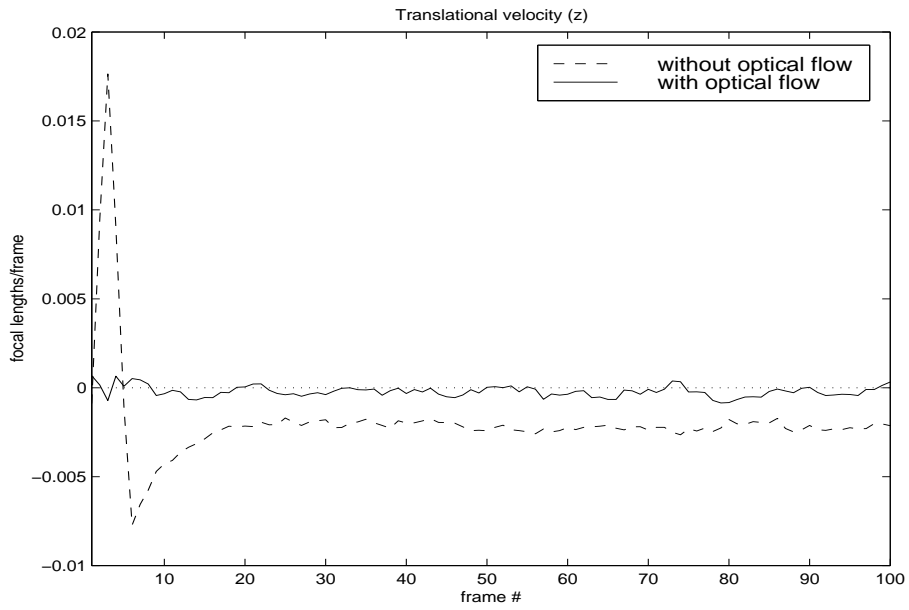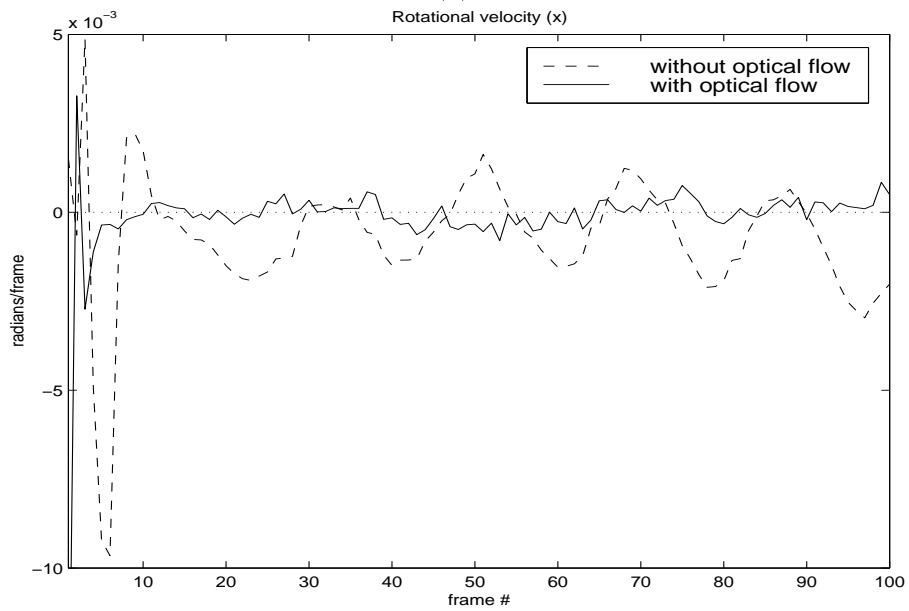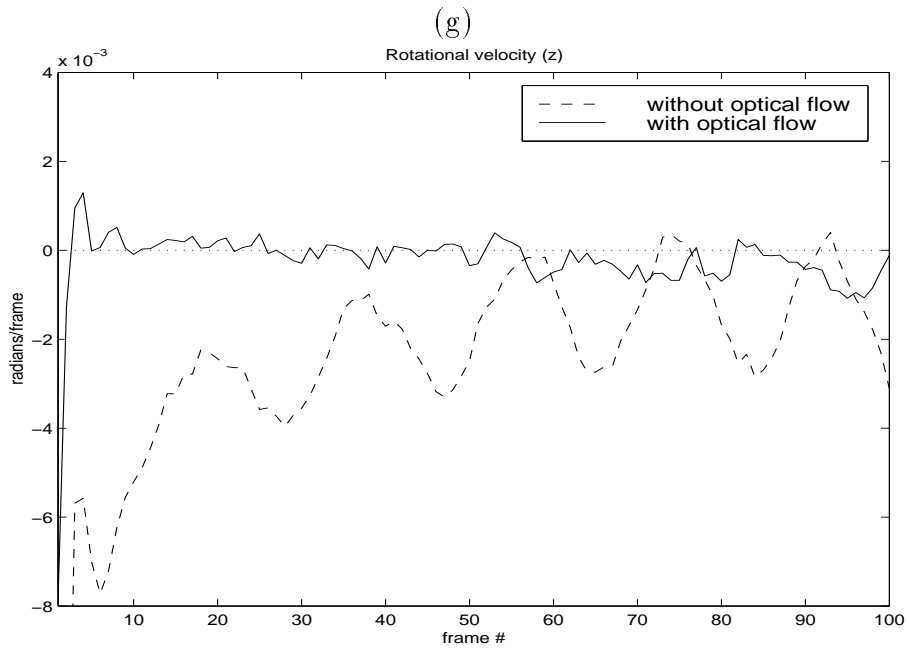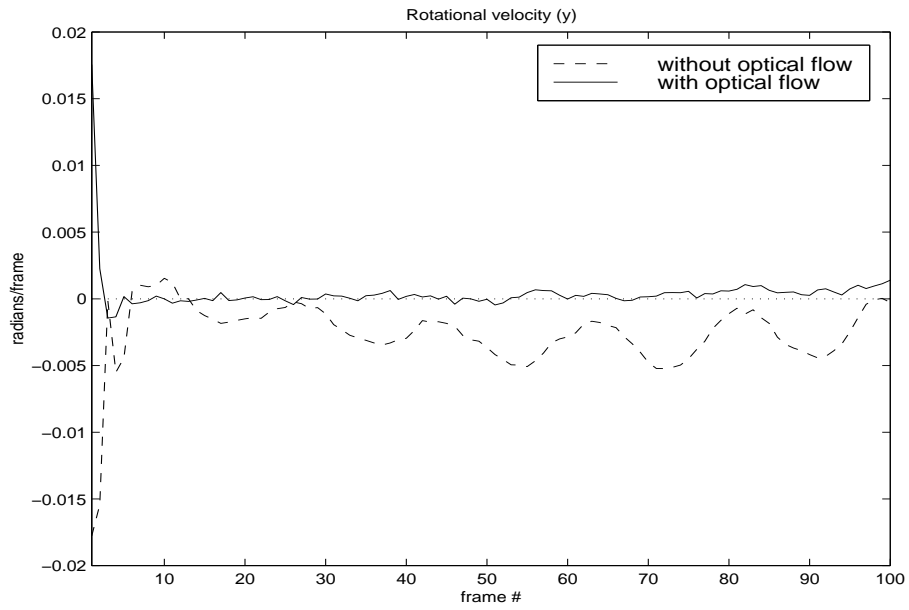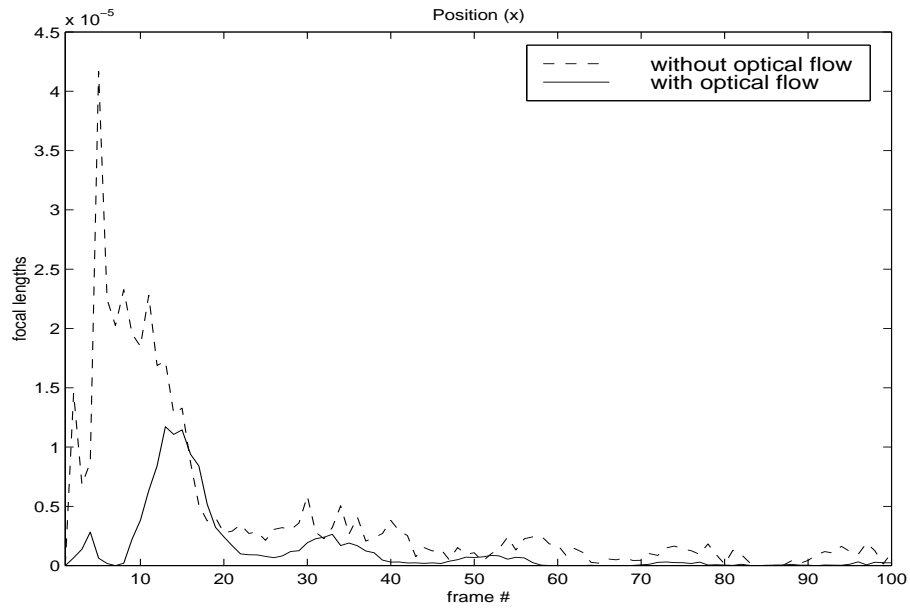Figure 4.3: Experiment 2: Results of Monte Carlo simulations (bias)

(e)

(f)

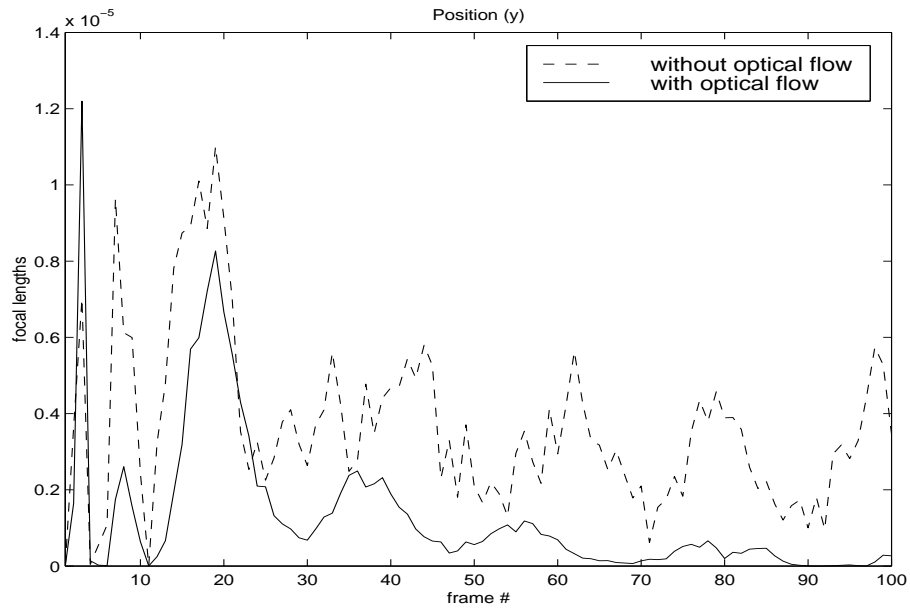Figure 4.3: Experiment 2: Results of Monte Carlo simulation (bias)

(g)

(h)

Figure 4.3: Experiment 2: Results of Monte Carlo simulation (bias)
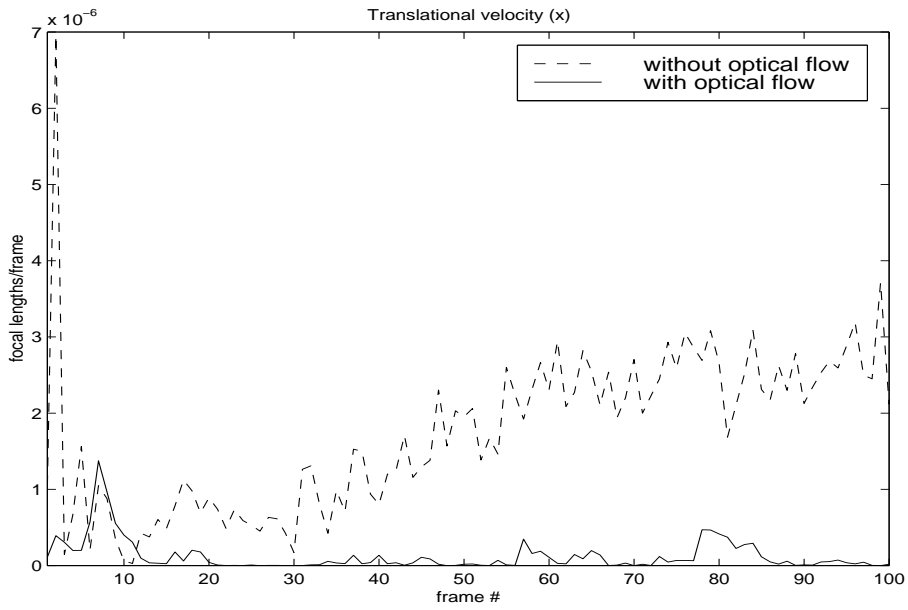
55

(a)

(b)

Figure 4.4: Experiment 2: Results of Monte Carlo simulations (variance)

(c)

(d)

Figure 4.4: Experiment 2: Results of Monte Carlo simulations (variance)

(e)

(f)

Figure 4.4: Experiment 2: Results of Monte Carlo simulation (variance)

(g)

(h)

Figure 4.4: Experiment 2: Results of Monte Carlo simulation (variance)

Figure 4.5: Experiment 3: Results of Monte Carlo simulations (bias)

(c)

(d)

Figure 4.5: Experiment 3: Results of Monte Carlo simulations (bias)

(e)

(f)

Figure 4.5: Experiment 3: Results of Monte Carlo simulation (bias)

(g)

(h)

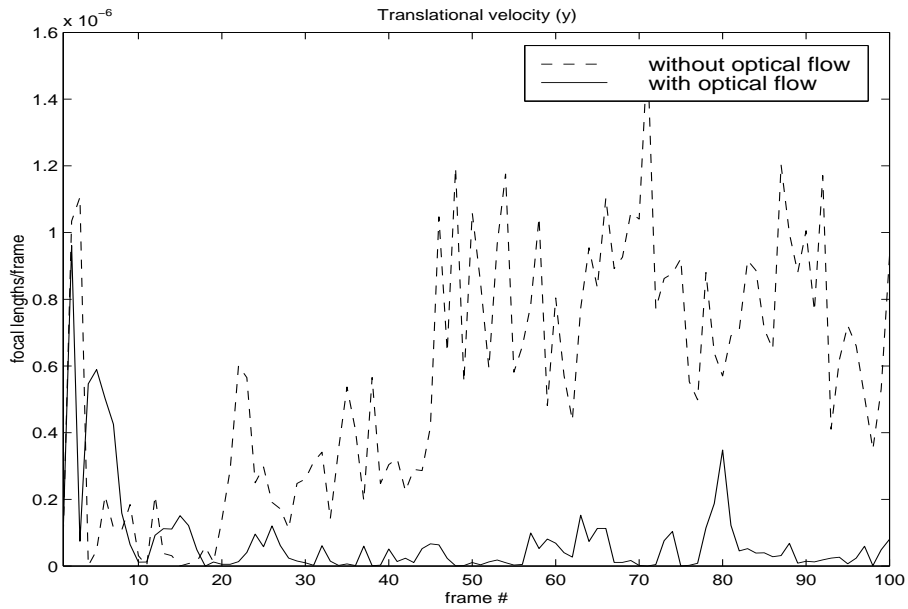Figure 4.5: Experiment 3: Results of Monte Carlo simulation (bias)

(a)

(b)

Figure 4.6: Experiment 3: Results of Monte Carlo simulations (variance)
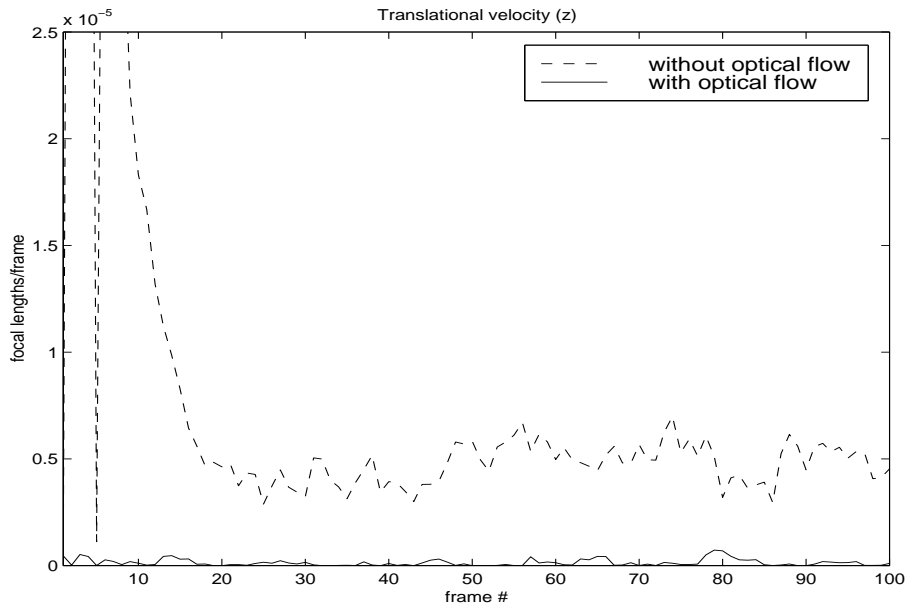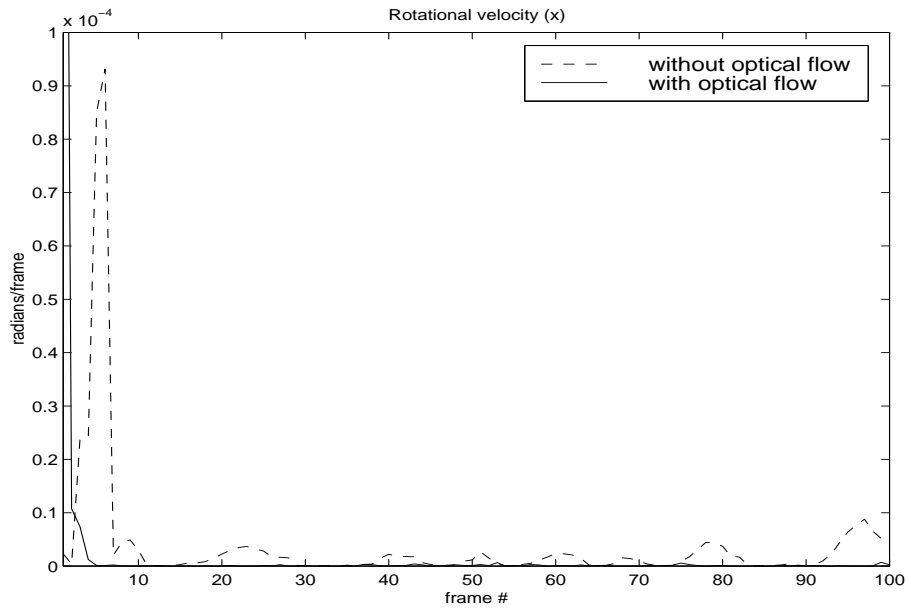
(c)

(d)

Figure 4.6: Experiment 3: Results of Monte Carlo simulations (variance)
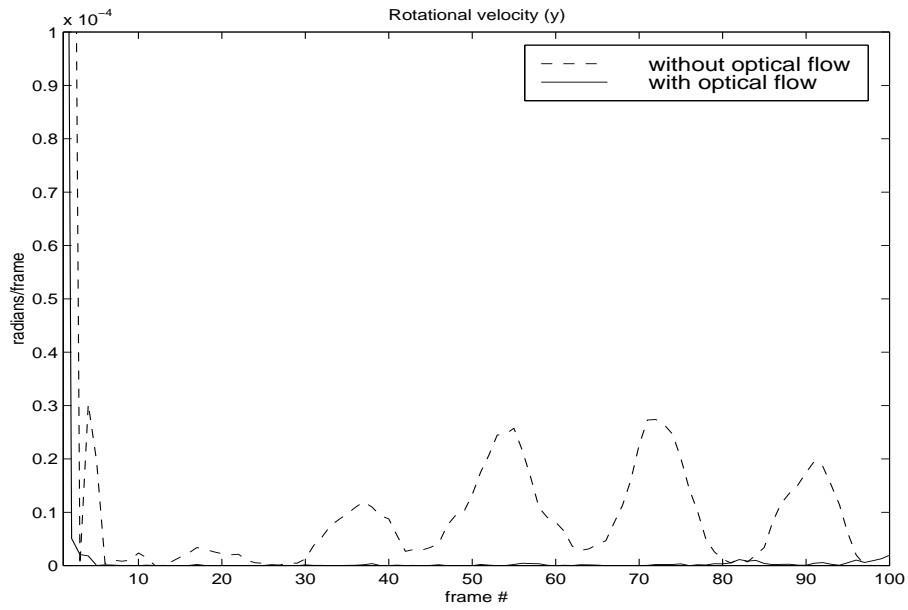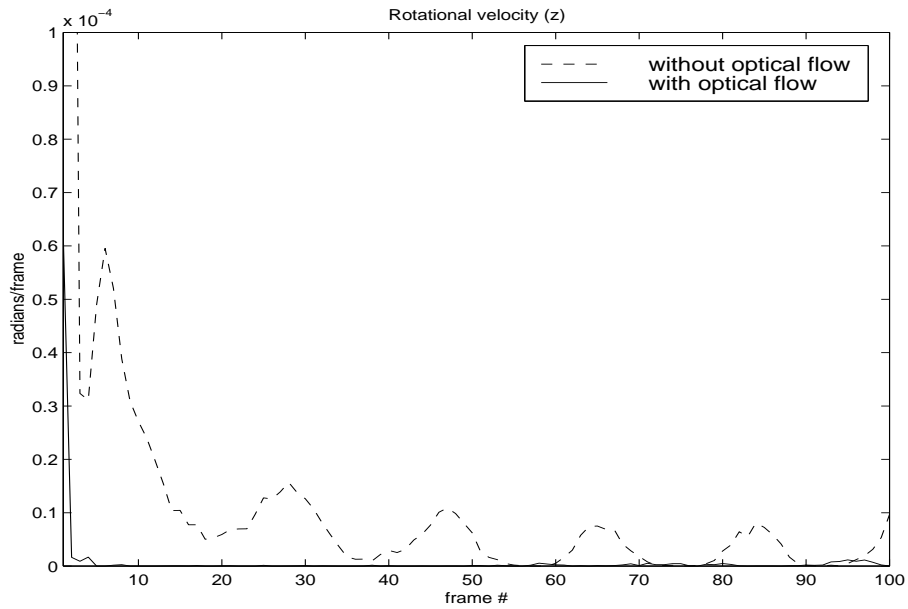
(e)

(f)

Figure 4.6: Experiment 3: Results of Monte Carlo simulation (variance)

(g)

(h)

Figure 4.6: Experiment 3: Results of Monte Carlo simulation (variance)

description of the motion model (process equation) is not accurate, the results are all still stable and convergent. This is very important in practice. One could, of cause, build a system model as accurate as possible. However, this would cause an increase in the size of state vector, state propagation matrix, and the computational load. It would increase the difficulty for the real-time application because of its decreasing computational efficiency. Our simulation results have shown that we can obtain good state estimates based on our relatively accurate measurements with a non-ideal process model. In some practical problems, computational efficiency might be more important than accuracy improvement of system model The simulation results are shown in Figure 4.5 and 4.6, in which the spatial resolution 512×512 pixels, and gray-value resolution of 10-bit have been used.

4.7   Summary

In this Chapter, we have examined the performance of this hybrid algorithm with our new measurement equation with measurement error. We have experimented with different pairs of spatial and gray-value resolution. Experiment results showed stable and convergent performance, which are much better than those described in [4].

Chapter 5

Summary and Conclusion

We began this project report with the background to 3-D motion estimation. Two main categories of approaches, feature-based and optical-flow-based, were discussed with their fundamental differences: feature-based techniques recover 3-D motion information by extracting a number of 2-D features and then establishing inter-frame correspondence, while optical-flow-based methods compute velocity fields using a gray value spatiotempory model. The most popular recursive estimation algorithm, Kalman filtering, is briefly introduced with time update equations, measurement update equations, and filtering procedures. The details of the hybrid feature/ flow-based recursive 3-D motion estimation algorithm were completely presented.

The major contribution is the development of a new approach that can compute both optical flow and position on the image plane given an image sequence. The derivation of this measurement equation is based on Nagel's approach that computes optical flow from second-order intensity derivatives. In his approach, first, a second-order Taylor expansion is used in modelling the measured gray-value surface within small window of interested. The parameters estimated on this surface is obtained by applying least squares estimation. Optical flow can then be computed in terms of these estimated parameters by minimizing the mean squared difference between two frames of a very small time interval and simplifying it under the assumption of "gray value corner" with special constraints. We also found that in a special case of the Gaussian surface we could obtain the optical flow expression in terms of the

parameters. We have derived the equations of computing the positions in these two cases, which are also in terms of the parameters. By using Taylor expansion about the parameter estimate, the new measurement vector is structured with both optical flow and position.

Another contribution is that we have combined the hybrid feature/flow-based recursive 3-D trajectory estimation algorithm with this new measurement equation. Our simulation results showed the performance of this hybrid algorithm combined with our new measurement equation achieved significant improvement in estimating all important trajectory states under a wide range of gray-value and spatial resolutions. The hybrid algorithm provided reliable state estimates. Even when there existed slight deviations which were not described in the system motion model, the performance was still stable and convergent with fairly good state estimates based on the measurements.

Bibliography

[1] S. D. Blostein, R. M. Chann, *Hybrid Feature/flow-based 3-D Trajectory Estimation*, Vision Interface, pp.125–130, 1993

[2] S. D. Blostein, R. M. Chann, *The Use of Image Plane Velocity Measurements in Recursive 3-D Motion Estimation from a Monocular Image Sequence*, Proc. Canadian Conf. Elec. and Comp. Eng., 1994

[3] R. M. Chann, *Recursive Estimation of 3-D Motion and Structure in Image Sequence Based on Measurement Transformations*, Master's thesis, Queen's University, 1994

[4] T. J. Broida, S. Chandrashekhar, R. Chellappa, *Recursive 3-D Motion Estimation from a Monocular Image Sequence*, IEEE Trans. AES, vol. 26, pp. 639–656, 1990

[5] M. S. Santina, A. R. Stubberud, G. H. Hostetter, *Digital Control System Design*, 2nd Ed., Saunders College Publishing, 1994

[6] H. H. Nagel, *Displacement Vector Derived from Second-order Intensity Variations in Image Sequences*, CVGIP, vol. 21, pp.55–117, 1983

[7] L. L. Scharf, *Statistical Signal Processing*, Addison-Wesley, 1991

[8] J. L. Barron, D. J. Fleet, S. S. Beauchemin, *Systems and Experiment Performance of Optical Flow Techniques*, Inter. Journal of Computer Vision, vol. 12, pp. 43–77, 1994

[9] S. Uras, F. Girosi, A. Verri, V. Torre, *A Computational Approach to Motion Perception*, Biological Cybermetics, 1988

[10] A. K. Jain, *Fundamentals of Digital Image Processing*, Prentice Hall, 1989

[11] R. M. Haralick, L. G. Shapiro, *Computer and Robot Vision*, vol. I, vol. II, Addison-Wesley, 1992

[12] R. J. Schalkoff, *Digital Image Processing and Computer Vision*, John Wiley & Sons, 1989

[13] *MATLAB Reference Guide*, the Math Works, Inc.