ROBUST ROBOTIC VISUAL SERVOING FOR UNCERTAIN SYSTEMS

by

Akbar Assa

Master of Science in Electrical Engineering,
Amirkabir University of Technology, 2009
Bachelor of Science in Electrical Engineering,
University of Tehran, 2006

A dissertation
presented to Ryerson University
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy
in the Program of
Mechanical and Industrial Engineering

Toronto, Ontario, Canada, 2015

Author's Declaration

Abstract

ROBUST ROBOTIC VISUAL SERVOING FOR UNCERTAIN SYSTEMS

Doctor of Philosophy, 2015

Akbar Assa

Department of Mechanical and Industrial Engineering, Ryerson University

The control of robotic manipulators in unstructured environments is a challenging task. Exploiting the camera images for that purpose, known as visual servoing, offers an interesting solution to the problem. Classic visual servoing techniques were the first attempts towards this goal. However, these methods suffered from system's shortcomings such as ones imposed by the limited camera's field of view and the robot's reachability. Numerous approaches were proposed to overcome these limitations. Nevertheless, most of these techniques assumed full knowledge about the system and did not account for uncertainties.

Uncertainties in visual servoing systems are introduced by multiple sources such as camera image noise and robot parameters. The lack of knowledge about system's parameters may lead to reduced accuracy or even total failure. Adaptive techniques were introduced previously to cope with this matter. However, those techniques were usually useful for deterministic uncertainties (e.g., camera calibration errors). Alternately, robust methods were employed to improve the performance of the system under uncertainties. Yet, those methods were usually conservative and more concerned with the stability of the system, rather than its accuracy.

This work proposes three steps towards robust and accurate visual servoing. First, the pose estimation algorithm, used in many visual servoing systems, is revised by introducing novel sensor fusion techniques. Multiple fusion algorithms at different levels of estimation are introduced to enhance the accuracy and robustness of the estimation against system uncertainties. Second, a novel uncertainty estimation technique is presented to approximate the level of uncertainties induced by image noise at different levels of the system. A general approach is used for that matter which has applicability over a wide range of controllers. Finally, multiple constraint-aware and robust controllers with improved stability and numerical feasibility are proposed to enhance the performance of the visual servoing systems in presence of uncertainties. The developed uncertainty model is exploited in robust control design. The effectiveness of each proposed technique is verified through numerous simulations and experiments. As it is expected, the proposed methods are capable of handling the uncertainties and enhancing the accuracy, while accounting for system constraints.

Acknowledgements

First, I would like to thank my supervisor, Dr. Farrokh Janabi-Sharifi, who made this work possible. I am profoundly grateful to him for his support during my Ph.D. studies. His dedication to the research inspired me to work hard, while his great management skills taught me valuable lessons for life. It was through his guidance that I found my way in the research and was able to improve my skills in various areas. I am also thankful to him for financially supporting my work.

I would also like to thank my supervisory committee members, Dr. Soosan Beheshti, Dr. Guanjun Liu, Dr. Vincent Chan, and Dr. Siyuan He, who helped me improve my thesis with their constructive feedback. I am also grateful to my internal, Dr. Hekmat Alighanbari, and my external examiner, Dr. Seth Hutchinson, for their time and consideration.

I am grateful to the staff of the department of Mechanical and Industrial engineering, who helped me throughout my Ph.D. studies. I especially thank Devin Ostrom for his help and collaboration in preparation of required equipment.

I am very grateful to Dr. Aleksandar Vakanski and Troy MacAvelia for the support and friendship they offered during my Ph.D. studies. They helped me a lot to adapt to my new position and overcome problems. I am also thankful to RMAL students who welcomed me with open arms and offered me their friendship.

Last but not least, I would like to express my sincere appreciation to my family from the bottom of my heart, for their love and constant support. Without them, it would be impossible to finish this work. I am very thankful to my parents, Masoud and Fatemeh, who motivated me for this work and always supported me through all these years. My deepest gratitude goes to my wife, Negar, who offered me her utter love and always stood by my side. I would also like to thank my parents-in-law, Faramarz and Batoul, for their support and kindness. Finally, I am thankful to my sister Solmaz, my brother Manoochehr, and my brothers-in-law, Faraz and Christiaan, for their support and help.

Dedication

To my wife, *Negar*, and my parents, *Masoud* and *Fatemeh*,
for their utter love and sincere support.

Table of Contents

List of Tables

List of Figures

Nomenclature

a_i	First row of rotation matrix R_o^c
a_j	Second row of rotation matrix R_o^c
a_k	Third row of rotation matrix R_o^c
A_k	Matrix representing the dynamics of the system
c	Camera frame
c^*	Desired camera frame
c'	Known Constant
$\mathrm{cov}(x,s)$	Covariance between x and s
Cm_i	Coordinate frame of ith EIH camera
Cf_j	Coordinate frame of jth ETH camera
$C_{k\|k-1}$	Error covariance matrix prediction
$C_{y,k}$	Covariance of transferred sigma points
$C_{\omega,y,k}$	Covariance of transferred sigma points and posterior sigma points
C_k	Posterior error covariance matrix
d_i	Image information from camera i
d_h	Fused data
$\tilde{d}_j$	Equivalent data from camera j
D_k	Autocorrelation of the system measurements
e	Coordinate frame of robot's end-effector
e_s	Image features servoing error
e_p	Pose servoing error
e_h	Hybrid servoing error
e_g	Gauss-Newton error to be minimized
$e_{t,k}$	Translation servoing error

$e_{R_1,k}$	Orientation servoing error based on angle and axis representation
$e_{R_2,k}$	Orientation servoing error based on Euler angles representation
$e_{z,k}$	Depth servoing error
e_G	General servoing error
$e_{H_1,k}$	Hybrid servoing error including depth error
$e_{H_2,k}$	Hybrid servoing error including full position error
$\overline{e}_{G,k}$	General servoing error
$\exp(\cdot)$	Exponential function
$E(\cdot)$	Expectation function
E_k	Autocorrelation of the system innovation term
f_c	Error covariance function at present time
f_{c1}	Error covariance function at present time
f_{c2}	Error covariance function of state derivatives at present time
f_{c3}	Error covariance function of state and its derivatives at present time
f_{ss}	Steady state of the error covariance function f_c
$f_i(\tilde{P}_{O,i}^C)$	Function that maps the coordinates of point *i* to its image plane coordinates
$f'(\cdot)$	Function that maps the information from all cameras to the desired pose
$F(\omega)$	Nonlinear function which maps the camera pose to image feature vector
F_k	Measurement mapping matrix
$F_{cs,k}$	Matrix of error covariance of image features
g_i	Quality measure of the *i*th feature point
$g_m(\cdot)$	Image features mean
$g_c(\cdot)$	Image features covariance
$g(\cdot)$	Function which maps the system output to system input
G_P	Transfer function mapping the image error to pose error

G_t	Transfer function mapping the rotation error to position error
G_π	Transfer function mapping the camera/object rotation error to orientation error
G_v	Transfer function mapping the pose error to camera velocity error
$G_{\rho\vartheta}$	Transfer function mapping the orientation error to angle and axis error
$G_{c,k}$	Matrix of error covariance between image features and pose
G'_π	Transfer function mapping the camera/world rotation error to orientation error
$h(\cdot)$	Function which maps the system input to system output
H_o^c	Transformation matrix between the object and camera frame in camera frame
i	Arbitrary scalar
I	Identity matrix
I_6	6×6 identity matrix
j	Arbitrary scalar
$J_{s,k}$	Image Jacobian (interaction matrix)
$J_{s',v}$	Translational part of the Jacobian matrix of the vector s'
$J_{s',\varpi}$	Rotational part of the Jacobian matrix of the vector s'
J_ω	Gauss-Newton Jacobian
$J_{P,k}$	Pose Jacobian
$J_{p,i}$	Position Jacobian
$J_{\theta,k}$	Angle Jacobian
$J_{z,k}$	Depth Jacobian
$\bar{J}_{s,k}$	Offline image Jacobian
$\bar{J}_{G,k}$	General offline Jacobian
J_R	Robot Jacobian
k	Time step
k_P	Proportional control gain
k_D	Derivative control gain

K_k	Kalman filter gain
l	Total number of the cameras
L_ϖ	Camera pose minimization cost function
$L_{s,k}$	Cost function of image-based control
$L_{p,k}$	Cost function of position-based control
$L_{H,k}$	Cost function of hybrid control
$L_{H_1,k}$	Cost function of hybrid control with $e_{H_1,k}$ servoing error
$L_{H_2,k}$	Cost function of hybrid control with $e_{H_2,k}$ servoing error
m	Deterministic error caused by calibration
M	Intermediate matrix relating the image features time derivation and pose changes
$M_{c,k}$	Matrix of error covariance of pose
n	Number of selected feature points
n'	Size of the state
N	Number of samples in adaptation window
N_c	Control horizon
o	Object frame
$\mathrm{O}(x^n)$	Terms of order *n* or higher
O_T	Target object frame
O_R	Axillary object on the robot frame
$p_{O,i}^C$	Image plane projection of point $\tilde{P}_{O,i}^C$
$\tilde{P}_{O,i}^C$	Vector of homogenous coordinates of *i*th point of the object in the camera frame
$\tilde{P}_i^O$	Vector of homogenous coordinates of point *i* in the object frame
q_k	Process noise
q_r	Robot joint position
$q_{r,\min}$	Minimum allowable value for robot joints
$q_{r,\max}$	Maximum allowable value for robot joints
Q_k	Covariance of the process noise

r_k	Measurement noise
R_o^c	Rotation matrix between the object and camera frame in camera frame
s	Vector of current image features
s^*	Vector of desired image features
s'	Vector of current image features and their depths logarithm
s'^*	Vector of desired image features and their depths logarithm
$s_{\min}$	Minimum allowable value for image features
$s_{\max}$	Maximum allowable value for image features
$\bar{s}_k$	Offline image features
$S(\cdot)$	Skew symmetric matrix
t_o^c	Translation vector between the object and camera frame in camera frame
$\dot{t}_c$	Translational velocity of the camera in camera frame
$T(\cdot)$	Transformation matrix relating the Euler angles to the angular velocity
$u_{O,i}^C$	First element of the image coordinates of $p_{O,i}^C$
U	Set of optimal camera velocities
$v_{O,i}^C$	Second element of the image coordinates of $p_{O,i}^C$
V_c	Camera velocity in camera frame
$\bar{V}_{c,k}$	Offline camera velocity
w	World coordinate frame
w_k	Weighting scalar
w_t'	Wiener-Levy process
W_k	Weighting matrix
x	First element of the translation vector
X_i	Unnormalized first row of rotation matrix R_o^c
X_j	Unnormalized second row of rotation matrix R_o^c
y	Second element of the translation vector

$\hat{Y}_k^{\pm i}$	Transferred *i*th sigma points
$\hat{Y}_k$	Mean of transferred sigma points
z	Third element of the translation vector
$(\cdot)^{\dagger}$	Matrix Moore-Penrose pseudo inverse function
$[\cdot]_i$	The *i*th element of the bracketed vector
$(\cdot)^{-1}$	Matrix inverse function
$(\hat{\cdot})$	Estimated value
0_6	6×6 matrix of zeros
α	Arbitrary coordinate frame
β	Arbitrary coordinate frame
B_p	Pose error weighting matrix
B_t	Translational error weighting matrix
B_H	Hybrid error weighting matrix
B_z	Depth weighting matrix
B_{H_1}	Hybrid error including depth error weighting matrix
B_{H_2}	Hybrid error including full position error weighting matrix
B_G	General weighting matrix
B_s	Image error weighting matrix
γ	Quality threshold
Γ_k	Covariance of the measurement noise
Δt	Sampling time
Δs	The difference between current and estimated features
ε	Error caused by uncertainty
ε'	Temporary scalar used in Dementhon's pose estimation method
ε_V	Camera velocity noise
ε_s	Image noise

η_i	Innovation term
θ	Roll angle
ϑ	Angle value in angle and axis representation of the rotation matrix $R_c^{c^*}$
λ	Controller gain
λ_1	Offline controller gain
λ_2	Online controller gain
Λ	Set of acceptable camera velocities
μ	Known Constant
ξ	Vector of noisy measurements
π_1^3	Vector form of the rotation matrix, R_1^3
$\Pi(\cdot)$	Function that converts matrices to vectors
ϖ_c	Rotational velocity of the camera in camera frame
ρ	Axis vector in angle and axis representation of the rotation matrix $R_c^{c^*}$
σ	Standard deviation of error
Σ_x	Error covariance of x
$\Sigma_{s,k}$	Matrix of feature error covariance
τ_s	Image features constraint threshold
τ_V	Camera velocity constraint threshold
τ_P	Camera position constraint threshold
$\upsilon_{c,k}^T$	Translational part of camera velocity
Υ	Camera velocity weighting matrix
ϕ	Vector of Euler angles
φ	Pitch angle
χ	Vector of unknown noise parameters
ψ	Yaw angle
Ψ	Arbitrary positive definite matrix
ω	Vector of pose parameters

$\hat{\omega}_{k\|k-1}$	State prediction
$\hat{\omega}_{k\|k-1}^{\pm i}$	i^{th} prior sigma points
$\hat{\omega}_{k}^{\pm i}$	i^{th} posterior sigma points
ω_{k}^{i}	State vector after *i* iterations
Ω_{e}^{c}	Velocity transformation matrix between end-effector and the camera

Abbreviations

ARMA	Auto-Regressive Moving Average
CAD	Computer Aided Design
DOF	Degree Of Freedom
EIH	Eye-In-Hand
EKF	Extended Kalman Filter
ETH	Eye-To-Hand
FOV	Field Of View
GPC	Generalized Predictive Control
GPS	Global Positioning System
HPC	Hybrid Predictive Control
HVS	Hybrid Visual Servoing
IAEKF	Iterative Adaptive Extended Kalman Filter
IAUKF	Iterative Adaptive Unscented Kalman Filter
IBPC	Image-Based Predictive Control
IBVS	Image-Based Visual Servoing
IEKF	Iterative Extended Kalman Filter
i.i.d.	Independent and Identically Distributed
IMU	Inertial Measuring Unit
LMI	Linear Matrix Inequality
LQG	Linear Quadratic Gaussian
LQR	Linear Quadratic Regulator
LTV	Linear Time-Varying
MIMO	Multi-Input Multi-Output
MPC	Model Predictive Control
P	Proportional
PBPC	Pose-Based Predictive Control
PBVS	Pose-Based Visual Servoing
PD	Proportional-Derivative
PF	Particle Filter

PnP	Perspective n-Points
POS	Pose from Orthography and Scaling
POSIT	POS with ITerations
RCONPC	Robust Control with Online Predictive Controller
RCOFPC	Robust Control with Offline Predictive Controller
RVS	Robotic Visual Servoing
SISO	Single-Input Single-Output
SLAM	Simultaneous Localization And Mapping
UAV	Unmanned Aerial Vehicle
UKF	Unscented Kalman Filter
VVS	Virtual Visual Servoing

Chapter 1

Introduction

1.1 Overview

The fast movement towards globalization has made the competition in most markets tighter than ever. The exceeding needs for reduction in production costs and time has turned automated manufacturing into an important asset to many companies. Industrial robots and particularly manipulators play a major role in manufacturing automation. These manipulators facilitate fast and accurate production, while reducing high costs. Many applications such as welding, painting, cutting, and assembling have benefited from the exploitation of industrial manipulators and the demands for these approaches in new applications are ever-increasing.

In order to produce the required quality, manipulators usually require a highly-structured workspace. In many applications, the tool (e.g., welding gun, spray nozzle, etc.) is mounted on the robot's end-effector and is brought to the desired location with respect to the object of interest to carry out the required task. Therefore, the relative pose (i.e., position and orientation) of the object with respect to the tool should be known a priori. Structuring the workspace is usually very demanding for each task and requires a few days to several months, depending on the size of the workspace and the number of robots involved. In addition, the kinematics of the robot might not be accurate enough (due to system uncertainties and nonlinearities such as backlash and slippage) to be used for such purposes. In summary, conventional robotic manipulation is time-consuming, expensive and requires major revisions and programming efforts for each task.

Vision-guided robot control, also known as *robotic visual servoing* (RVS), offers an attractive remedy to this problem. In this technique, the images of the camera taken from the object of interest are exploited to provide the system with adequate information about the current pose of the robot with respect to the object. Contrary to previous trends in which images are used in the planning level (also known as "look-and-move"), RVS involves use of images directly in the execution control level. Two major groups of RVS schemes were introduced previously [1.2]. In the first group, the difference between the current and desired images (usually that between the image features) is directly exploited for servoing purposes. Therefore, this structure is known as image-based visual servoing (IBVS). In IBVS systems, the interest points of each image (i.e., image features) are extracted (denoted by s) and compared with those of the desired image (denoted by s^*). A controller is employed to reduce the image difference. The image space control action is then mapped to the camera velocity through the image Jacobian matrix (or interaction matrix). The robot is then moved based on this velocity. The process is repeated till the robot gets to its desired pose. Figure 1.1 demonstrates the block diagram of an IBVS system. Alternately, each image from the camera might be used to acquire the relative pose of the object with respect to the robot. The difference between desired and current poses (denoted by $H_o^{c^*}$ and H_o^c, respectively) is then passed to the control unit for robot control. As a result, this structure is called pose-based visual servoing (PBVS). The relative pose of the object with respect to camera is estimated for each image by using the image features. Various pose estimation algorithms are available for that purpose, which will be introduced in the coming chapters. Figure 1.2 exhibits the block diagram of a PBVS system.

The properties of the aforementioned RVS structures were thoroughly investigated by the previous works [1.3, 1.4]. It was shown that IBVS has a good robustness against robot or camera uncertainties. These methods are simple to implement and does not require a model of the object. However, the robot trajectory of these methods might not be feasible, as a result of controlling in 2D space. In addition, the methods suffer from possible systems local minima and singularity of image Jacobian. Moreover, the visibility of the object is not guaranteed. On the other hand, PBVS systems offer smooth robot trajectories in Cartesian space and have global stability. Nevertheless, the pose estimations used in these methods are sensitive to the uncertainties of the system and usually require a 3D model of the object. Likewise, the object is likely to leave the camera's field of view (FOV), which leads to system failure. Hybrid visual servoing (HVS) schemes were introduced to alleviate the aforementioned shortcomings of IBVS and PBVS [1.5]. By sharing the control action between the image and Cartesian space, i.e. decomposing of position and orientation control, these methods were able to successfully guide the robot towards the goal, providing global stability. However, the satisfaction of all system constraints was not guaranteed. Figure 1.3 depicts a block diagram view of this scheme.

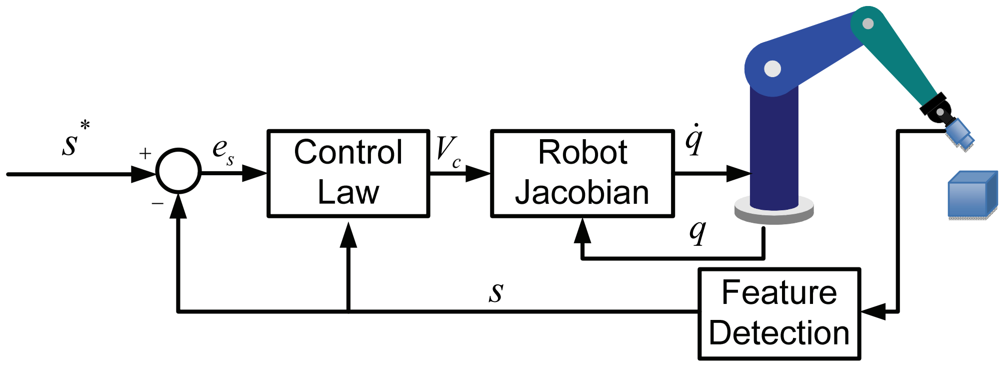


Figure 1.1 The block diagram of an IBVS system.

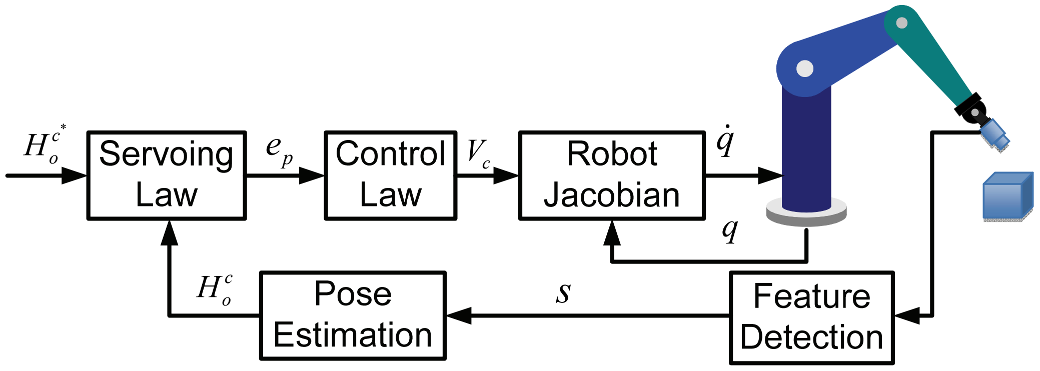


Figure 1.2 The block diagram of a PBVS system.

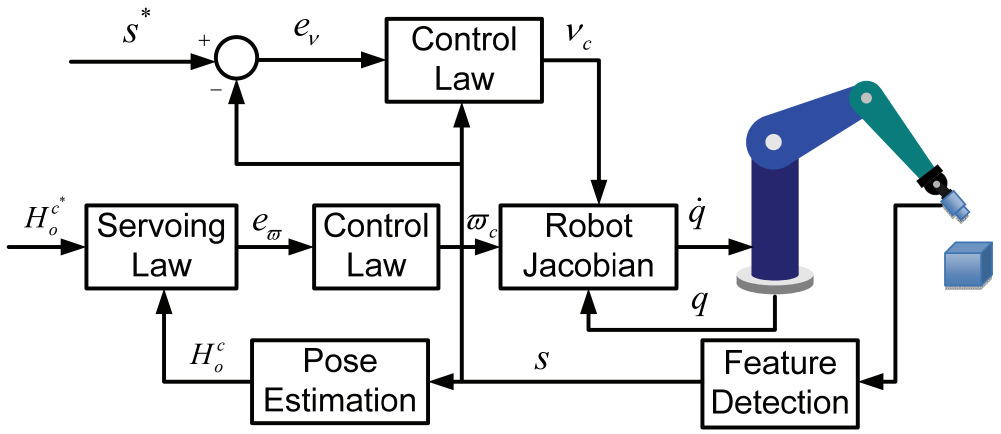


Figure 1.3 The block diagram of an HVS system.

Despite their impressive performance in theory, RVS systems may not be as successful in practical cases. As a matter of fact, RVS systems usually suffer from system's uncertainties which lead them to accuracy reduction or even total failure. Multiple factors are responsible for system's uncertainties. One source of uncertainties in the system is improper calibration of the camera or the robot. These uncertainties remain the same throughout the servoing and are therefore deterministic. Image noise and feature extraction algorithms could be counted as other sources of uncertainties. Examples include noise or uncertainty due to the the angle of the view, lighting, etc. [1.6]. These uncertainties change throughout the servoing and have a stochastic nature. Adaptive techniques were previously introduced to compensate for uncertainties of the system (e.g., [1.7]). However, these methods are more suitable for deterministic uncertainties. Alternatively, robust systems were presented to relieve the effects of uncertainties (e.g., [1.8]); yet, most of these systems targeted specific uncertainties in the system and were mainly concerned with the stability of the system, and not its accuracy. In short, robustness of visual servoing techniques is critical to the successful integration of RVS techniques into the practice, and there are still open problems before achieving a reliable RVS approach for industry. This work presents novel approaches for robust RVS to deal with the system uncertainties.

First, sensor fusion is introduced as an effective remedy to accurately estimate the pose of the object in spite of system uncertainties, since most RVS approaches require full or partial pose estimation during servoing. Multiple sensor fusion algorithms are introduced for fast and accurate pose estimation. For this matter, fusion at different levels of pose estimation is practiced and the results are compared to highlight the strengths and weaknesses of each algorithm. The wide range of provided fusion methods enables us to select the most efficient algorithm, considering the available resources and required characteristics.

Second, in order to simulate and predict the performance of RVS techniques, a good model of uncertainties and errors is needed to be developed. Such model can also serve as the basis of a robust control system design. In this thesis, a novel uncertainty modeling for closed-loop RVS is developed to accurately model the propagated noise in different levels of the system. For this matter, the effect of image noise on different parts of different visual servoing systems is investigated and a proper general model is proposed. The proposed approach considers the discrete-time and closed-loop nature of the visual servoing system and therefore leads to high accuracy noise estimation. This model is then exploited for robust control of the system.

Ultimately, several robust controllers are introduced for RVS as the main goal of this work, in further attempts of improving the system accuracy and incorporating the system constraints. For that matter, in this thesis, several controllers are developed which are capable of dealing with the system constraints and expand the stability region of the system. In addition to that, other controllers are proposed to minimize the system uncertainties and provide robust performance. Robust control is made possible by introducing a novel two-stage controller structure which increases the speed of the system and predicts the outcome of the system. It is shown that this control structure is capable of minimizing the effect of uncertainties and compensate for the remaining errors effectively.

Numerous simulations and experiments are conducted to certify the effectiveness of the proposed methods. While the simulations are important to verify the proposed systems in theory, experimentation is necessary to emphasize the applicability of these methods in real world tasks. It is shown that the proposed methods are capable of improving the accuracy of the system, while surviving undesired situations such as partial occlusion. In addition, these methods enable the system to fulfill the constraints in spite of system uncertainties.

1.2 Motivation and Approach

Pose estimation plays a crucial role in many of visual servoing systems as they require full pose (PBVS) or partial pose (IBVS and HVS) for servoing. The success of these visual servoing methods highly depends on the accuracy of the estimated pose. Most of the previously proposed pose estimation methods provide accurate estimations of the pose in the absence of system uncertainties (e.g., inaccurate object model, camera calibration error, etc.). However, the performance of these methods declines drastically once the uncertainties of the system are increased. Therefore, robust and accurate pose estimation is of great importance.

Traditionally, monocular vision is used for that purpose. The camera may be installed on the robot's end-effector and move with it, or alternatively set apart from the robot at a known pose with respect to the object. The former configuration is known as eye-in-hand (EIH), while the latter is usually referred to as the eye-to-hand (ETH) [1.9]. The EIH cameras have the advantage of higher accuracy as they can get very close to the object of interest; however, their FOV become limited for the very same reason. The reverse is true for ETH cameras. These cameras usually provide a large FOV which usually include the robot and the object of interest, while their accuracy is comparatively low. In order to benefit from both of these configurations, multi-camera systems were introduced.

Multi-camera systems were used in many of previous works; yet in most of those works the data from the cameras were complementary rather than redundant. Therefore, their information could not be used to improve the overall robustness and accuracy. Sensor fusion provides a tool to combine the data from multiple cameras synergistically to enhance the accuracy of the systems. A few sensor fusion works were available for improving the accuracy of pose estimation. However, those methods had limited accuracy and applications. A detailed survey of sensor fusion methods is provided in the coming chapter.

As a part of robust RVS, this work presents a comprehensive design tool for pose estimation through sensor fusion. Sensor fusion is carried over three different levels of pose estimation. First, fusion at *measurement level* is discussed. Novel methods are proposed for measurement fusion, which are shown to have higher accuracy and degrees of robustness compared to their previous rivals. These methods process all the data from different cameras simultaneously, which result in a high-accuracy output. As a result, they are computationally expensive. Sensor fusion at the *state level* is presented as an alternative. These methods have reduced accuracy compared to measurement fusion techniques; however, they are more efficient computationally. This difference becomes more significant as the number of cameras increases. Finally, a fusion technique at the *pre-processing stage* is introduced. Unlike the previous methods, this scheme may be practiced with any pose estimation method. Once again, the outcome of this fusion technique is shown to be more accurate than monocular systems, but not as accurate as measurement fusion techniques.

Despite their usefulness, the proposed fusion techniques can only reduce the effect of uncertainty in the system and are not able to fully resolve this issue. In addition, these techniques do not address the limitations of the system (e.g., those caused by the camera's FOV or robot's reachability). These problems can be handled at the control level but before that, an accurate model of uncertainty seems to be necessary. This model helps to estimate the impact of uncertainties at different levels of the system which in turn facilitates the treatment of errors through robust controllers. Without this model, the control actions taken against the possible uncertainties might be conservative. Current models of uncertainty in RVS are either conservative [1.10] or restricted to a particular controller [1.11].

In order to resolve this issue, a novel methodology is proposed to estimate the effect of uncertainties at different levels of RVSs. This methodology enables us to calculate the error covariance of various signals in the system, assuming known error covariance of the image features. The main advantage of this approach is its practicality over a wide range of controllers. In addition, the discrete-time and closed-loop nature of the system is taken into account which makes the method more realistic compared to the previous modeling attempts. It is noteworthy that only the effect of image noise is considered in this work and other sources of uncertainties are left for future works.

Once the uncertainty model is complete, control strategies may be taken to address the aforementioned problems of visual servoing systems. As it was discussed before, RVS systems suffer from several limitations. One important limitation of these systems is their inability to address the constraints, imposed by the structure of the system. Cameras usually have a limited FOV. This means that each image can capture only a part of the environment, usually the part that is directly in front of the camera. This limitation is a burden to RVS systems as the object of the interest may fall out of camera's FOV. Moreover, the robots used for servoing have limited reachability and motion. The length of robot's links and robot's joint limits set a boundary for the space which the robot can reach. The acceleration of the robot is usually limited by the payload and the dynamics of the robot. The velocity of robots is usually restricted for safety reasons. In summary, the system constraints are mainly imposed by the camera's limited FOV and robot's restricted reachability and motion, which are considered as the system's limitation in this work.

Another major shortcoming of RVS systems is their sensitivity to system uncertainties. In particular, image noise contributes to the errors of camera velocity, which in turn affects the accuracy of robot's trajectory. As a result, the robot may experience irradic motions or the system may fail if the constraints are violated. Therefore, it is important to minimize the effect of image noise on the system and compensate for any undesirable influence.

This work proposes a two-level controller for this purpose. In this proposed technique, an offline controller is employed to predict the trajectories of the system and its camera velocities in the absence of any uncertainties. Next, an online controller is engaged to guide the robot as close as possible to the predicted trajectories, using the pre-calculated velocities as a part. Model predictive control (MPC) strategies are then taken to guarantee the satisfaction of system constraints. Proper changes had been made to these systems to enhance their feasibility and stability. It has been shown that the proposed control technique have the capability to minimize the error covariance of the system, while maintaining the global stability by taking the constraints of the system into consideration. Moreover, it is shown that the method can be benefited from the developed error modeling technique to predict the violation of constraints at offline stage and avoid them accordingly. As a result, the system is capable of handling the uncertainties and constraints very well.

1.3 Robotic Visual Servoing Structures

In this section, the main basic RVS techniques are introduced. This introduction helps the reader to better understand how the RVS systems work. For that matter, IBVS, PBVS, and HVS systems are briefly presented as follows.

1.3.1 IBVS

In IBVS systems, a controller is applied to guide the image features towards their desired locations. For this purpose, the image features are extracted at each time step and compared with those of the desired image (taken prior to the servoing). The servoing error is formed as the difference between these features, i.e.,

$$e_s = s^* - s, \tag{1.1}$$

where s and s^* are the current and desire image feature vectors, respectively. One simple way to reduce this error exponentially to zero is to employ a proportional controller, i.e.,

$$\dot{e}_s + \lambda e_s = 0, \tag{1.2}$$

where λ is the controller gain. It is known that the image features are a nonlinear function of object pose with respect to the camera, formulated as follows,

$$s_k = F(\omega_k), \tag{1.3}$$

where ω_k is the vector of pose parameters. Having the time derivative of both sides of (1.3) yields,

$$\dot{s}_k = \frac{\partial F(\omega_k)}{\partial \omega_k} \dot{\omega}_k. \tag{1.4}$$

The time derivative of the pose is related to the camera velocity. Consequently, (1.4) may be rewritten as follow,

$$\dot{s}_k = J_k V_{c,k}, \tag{1.5}$$

where $V_{c,k}$ is the camera velocity in camera frame and $J_{s,k}$ is the image Jacobian (or interaction matrix) at time step *k*, respectively. Using (1.5) along with (1.1) and (1.2) results in,

$$J_{s,k} V_{c,k} = \dot{s}^* + \lambda\left(s^* - s\right), \tag{1.6}$$

which yields,

$$V_{c,k} = J_{s,k}^{\dagger} \dot{s}^* + \lambda J_{s,k}^{\dagger}\left(s^* - s\right). \tag{1.7}$$

Here the pseudo-inverse of the Jacobian matrix is denoted by $J_{s,k}^{\dagger}$. The camera velocity is recomputed at each time step to find the new camera velocity. The camera velocity is transferred to robot's angular joint velocities in turn to move the robot accordingly. As it can be seen from (1.7), the camera velocity will be zero if the feature error happens to be in the inverse Jacobian's null space. In addition, singularities of the image Jacobian may render the inverse of the Jacobian used in (1.7) impossible. Therefore, local minima and Jacobian singularities are some of the main shortcomings of IBVS systems. Moreover, the stability of these systems are guaranteed only locally as the Jacobian matrix is a local linearization between the Cartesian and image spaces. Furthermore, the camera trajectories created by (1.7) might not be feasible for the robot due to its limited workspace.

1.3.2 PBVS

In these systems, the image features extracted from camera images are exploited to estimate the relative pose of the object with respect to the camera. This pose is denoted by the transformation matrix between the object and camera frame,

$$H_o^c = \begin{bmatrix} R_o^c & t_o^c \\ 0 & 1 \end{bmatrix}, \tag{1.8}$$

where R_o^c and t_o^c are the rotation matrix and translation vector between the object and camera frames, respectively. Once the desired pose ($H_o^{c^*}$) is estimated, the pose error is formed as a function of the current and the desired pose. Various pose errors may be formed for this purpose. In this section, transformation between the current and desired camera poses is exploited for that purpose, i.e.,

$$H_c^{c^*} = H_o^{c^*}\left(H_o^c\right)^{-1}. \tag{1.9}$$

The pose error is then defined as a vector including the translation vector and the angle and axis representation of the rotation matrix obtained in (1.9) [1.12],

$$e_p = [t_c^{c^*} \quad \rho\vartheta]. \tag{1.10}$$

Once again, the proportional controller similar to (1.2) is employed to reduce the error to zero exponentially. It can be shown that the camera velocity is calculated as follows [1.2],

$$V_{c,k} = \begin{bmatrix} -\lambda R_{c^*}^c t_c^{c^*} \\ -\lambda\rho\vartheta \end{bmatrix}. \tag{1.11}$$

Such controller guides the camera to its desire pose through a straight line in Cartesian space, which is desirable. This method offers global stability assuming perfect pose calculation [1.2]. On the other hand, the features may leave the camera FOV at any time. Moreover, the pose information used for this control scheme is only an estimate of the real value and sensitive to uncertainties.

1.3.3 HVS

In order to combine the advantages of IBVS and PBVS systems, HVS systems were introduced which relayed the control partially to image space and partially to Cartesian space. For that matter, the image feature used by IBVS were extended to include the logarithm of depth for each feature and the angle and axis of the rotation between the current and desired camera frames is added to the error, similar to PBVS,

$$e_h = \begin{bmatrix} s'^* - s' & \rho\vartheta \end{bmatrix}, \tag{1.12}$$

where s' and s'^* are the vectors of current and desire image feature plus their depth logarithm. Using the aforementioned proportional controller to reduce this error to zero yields [1.5],

$$V_{c,k} = \begin{bmatrix} -\lambda J_{s',v}^{\dagger}(s'^* - s' - J_{s',\varpi}\rho\vartheta) \\ -\lambda\rho\vartheta \end{bmatrix}, \tag{1.13}$$

where $J_{s',v}$ and $J_{s',\varpi}$ are the translational and rotational parts of the Jacobian matrix which relates the time derivative of the extended features to the camera velocity,

$$\dot{s}'_k = \begin{bmatrix} J_{s',v} & J_{s',\varpi} \end{bmatrix} V_{c,k}. \tag{1.14}$$

Exploiting the camera velocity introduced in (1.13) improves the stability of the system and smoothness of the system trajectories, both in image and Cartesian spaces. However, none of the constraints of the system is yet guaranteed.

1.4 Contributions

The contributions of this work are listed as follows,

- Introduction and development of iterative adaptive extended Kalman filter (IAEKF) for centralized sensor fusion.
- Introduction and development of virtual visual servoing (VVS), its relation with Gauss-Newton method, and the proof of its superiority.
- Development of decentralized fusion techniques such as extended Kalman filter (EKF) and virtual visual servoing (VVS) for pose estimation.
- Formulation of pose estimation error for ETH cameras.
- Introduction and development of a pre-processing fusion technique for pose estimation.
- Establishment of a closed-loop error modeling methodology for visual servoing systems.
- Introduction and development of the two-stage control structure for robust RVS.
- Development of novel model predictive controllers for constrained RVS.
- Integration of control and uncertainty estimation for robust constraint handling.

1.5 Thesis Structure

The steps towards robust visual servoing are presented in consecutive chapters as follows.

- The pose estimation techniques are the focus of Chapter 2. First, Kalman filter is introduced as one of the most important pose estimation techniques which can be easiliy modified for sensor fusion. The tuning problems of this method are investigated and proper solutions such as adaptive methods, iterative techniques, and iterative adaptive schemes are discussed in details. In addition, Virtual Visual Servoing (VVS) and Gauss-Newton methods are introduced as alternative optimization-based solutions. The close relation of these methods is investigated towards the end of this chapter.
- Chapter 3 is dedicated to sensor fusion applications for robust and accurate pose estimation which is used in RVSs. The proposed sensor fusion techniques are discussed in three groups as centralized, decentralized, and pre-processing fusion schemes. It is shown that the centralized fusion techniques have the highest accuracy, while they are computationally expensive. Two novel schemes are introduced for that purpose. Next, decentralized fusion techniques are proposed as alternatives to centralized methods. The accuracy of these systems is lower than the centralized algorithms; however, they bring the advantages of reduced computational cost and straight-forward fault detection and isolation. Two methods are proposed for that reason. Finally, a pre-processing fusion technique is presented as an alternative method which could be used with any pose estimation technique. The effectiveness of each proposed techniques is demonstrated through numerous simulations and experiments.
- The focus of Chapter 4 is on error modeling in RVS. A novel closed-loop error modeling strategy is proposed. Generality and simplicity are the main advantages of this method. Through such technique, the error covariance of different signals in visual servoing, including the camera velocities and robot pose, is estimated. The accuracy of the predictions is then validated through Monte Carlo simulations. These methods are then used for robust control design, introduced in the subsequent chapter.
- The robust control design is discussed in Chapter 5. The goal of this chapter is to introduce novel controllers to improve the performance of RVS. First, model predictive

controllers with global stability are introduced for constrained visual servoing. Then, a novel two-stage controller is proposed for accurate and robust control in RVS. The first stage of this controller allows predictions to be made for the system without the uncertainties. Next, the prepared reference trajectories are exploited to reduce the effect of uncertainties in action. Model predictive controllers are exploited to protect the system against possible constraint violations. In addition, the error model of the system is employed to predict possible violations of constraints and avoid them accordingly. It is shown that the methods are efficient in handling system's constraints and uncertainties, while providing global stability.

- The summary of the thesis is presented in Chapter 6. The contributions of the works are highlighted in this chapter, and the directions for possible future works are presented.

Chapter 2

Pose Estimation

2.1 Introduction

Acquiring the relative position and orientation (pose) of an object with respect to a camera by means of the camera images is known as 3D object pose estimation which plays an important role in many applications including visual servoing. For this purpose, the interest regions of the object (i.e., image features) from camera images are extracted and used along with the 3D model of the object and camera calibration information. Since the projection on the camera image plane includes only 2D data from the 3D scene, the relative pose of the object could be estimated only up to certain accuracy.

The accuracy of estimations depends on many parameters such as camera calibration and image noise. Lack of information about the system may result in inferior pose estimations. Traditionally, monocular (i.e., single camera) vision was exploited for this purpose which was prone to camera calibration errors, image noise, and partial occlusion. As a remedy, multi-camera systems with sensor fusion techniques were introduced. Since sensor fusion techniques used for pose estimation are built up on monocular pose estimation methods, the structure of these methods need to be investigated first. The sensor fusion technique will be discussed afterwards.

This chapter reviews the pose estimation methods that may be used for sensor fusion which will be discussed later in Chapter 3. Three major optimal techniques are introduced for this purpose. First, the Kalman filter-based methods are introduced as the most important group of pose estimation algorithms. The advantages of these methods include temporal noise filtering, recursive implementation, and expedition of dynamic windowing for feature selection, which have made these techniques very popular. Novel iterative and adaptive techniques are introduced to improve the robustness of these methods against the uncertainties of the system. Next, the

Gauss-Newton method is introduced as one of the most practiced optimization-based pose estimation techniques which can provide the RVS system with accurate estimations of the pose. This method is closely related to Levenberg-Marquardt and bundle adjustment techniques. In addition to that, VVS is introduced as an accurate method for pose estimation. In this technique, the pose of the object is acquired through virtually servoing the camera to a known pose. Next, the close relation between the VVS and Gauss-Newton method is proven and the superiority of VVS over the Gauss-Newton method in terms of accuracy and speed are discussed in details. The introduced methods will be used in sensor fusion which will be discussed in the coming chapter.

This chapter is organized as follows. First a survey over the previous literature on pose estimation is presented in Sec. 2.2. Next, the configuration of the system is discussed in Sec. 2.3. Kalman filter-based pose estimation methods are the focus of Sec. 2.4. The Gauss-Newton and VVS methods are introduced in Sec. 2.5 and 2.6, respectively. The relation between these two is demonstrated in Sec. 2.7. Finally, the chapter is concluded in Sec. 2.8.

2.2 Literature Survey

The problem of 3D object pose estimation through camera images, also known as external camera calibration, is an old though fundamental problem in many vision related fields such as computer vision and photogrammetry [2.1]. Pose estimation was applied to numerous domains, such as virtual reality [2.2], 3D reconstruction [2.3], object recognition [2.4], 3D tracking [2.5], visual servoing [2.6] and simultaneous localization and mapping (SLAM) [2.7]. The pose of an object may be acquired through other sensors such as Inertial Measuring Units (IMUs) and Global Positioning System (GPS) sensors as well. However, the former is only accurate for short periods of time [2.8] and the latter is less accurate and reliable [2.9]. Vision is usually a better option since it can provide rich information, which if processed properly, can lead to accurate and dependable results. Moreover, vision sensor is able to provide the relative pose of the object with respect to the sensor, which is beneficial for many applications such as visual servoing.

Vast variety of methods has been introduced for pose estimation through camera images. Linear methods based on perspective n-points (PnP) were early responses to the problem [2.10, 2.11]. A thorough review of these methods can be found in [2.12]. More recently, linear solutions based on polynomial constraints [2.1] and linear equations solved through singular value decomposition [2.13] have been proposed. These methods are fast and do not need initialization; however they usually suffer from image noise. Closed-form solution such as those in [2.14] and [2.15] were introduced for simple but fast estimations. In addition, numerical and iterative solutions were also introduced for simple pose estimation [2.16]; nevertheless, they were also shown to be sensitive to image noise. Nonlinear optimization-based pose estimation algorithms based on Gauss-Newton or Levenberg-Marquardt [2.17] and least-square estimators [2.18, 2.19] have shown to be relatively accurate in presence of image noise. These algorithms iteratively approach the correct pose from an initial guess using the task-function Jacobian. Recently, a new nonlinear scheme based on visual servoing control systems was introduced to solve the same problem [2.20, 2.21]. In this method, a virtual camera is servoed from an initial known pose to the current (i.e., desired) pose using the image features. Later, the method was extended to be robust against outliers [2.22]. The main drawback of nonlinear optimization-based algorithms was their stability. It was shown that these methods were only locally stable and therefore the final estimation was guaranteed to be accurate only if the initial pose guess was sufficiently close to the current one [2.20]. However, this problem is not very concerning for visual servoing systems, since consequent poses of the robot are close to each other due to high sampling rates of the camera(s). Kalman filter-based pose estimation offered another solution to the problem. Since the problem of pose estimation is nonlinear, extended Kalman filter (EKF) should be exploited. Early attempts on this scheme can be found in [2.23] and were later applied to robotic field for grasping [2.24] and visual servoing [2.25]. One big disadvantage of these approaches was their challenging tuning and initialization problems. The effectiveness of these methods dramatically decreased, once the filter was out of tune. Adaptive techniques were introduced as a remedy to filter tuning problem [2.26-2.29]; yet, these methods were prone to system initialization. Alternately, iterative methods were proposed to reduce the error caused by system linearization. It was shown that these methods reduced the sensitivity of the system to tuning and initialization problems [2.30]. Lately, a cooperative iterative and adaptive scheme was introduced which had the capacity to adapt to the changes of the system while following the system's fast dynamics [2.31].

This chapter reviews the advancements on pose estimation techniques and propose the iterative adaptive unscented Kalman filter as a powerful tool for robust and accurate pose estimation. The introduced pose estimation techniques are then exploited in the coming chapter.

2.3 System Configuration

A multi-camera robotic cell is considered in this work. Two types of camera configurations are employed for this purpose. Much of the previous work on multi-camera pose estimation assumes the relative pose between the cameras to be known a priori (e.g., [2.35-2.37]). Though this is a good assumption in many practical cases, it may not be viable in all scenarios. For instance, the relative pose of a camera mounted on an Unmanned Aerial Vehicle (UAV) may not be available. Moreover, the calibration between different cameras is a time-consuming procedure (especially for a large number of cameras) and is prone to errors. In this work, the EIH cameras have a direct look at the target object (O_T) and the ETH cameras have the target object and the object on the robot end-effector (O_R) in their FOV. Figure 2.1 demonstrates this configuration. Relevant coordinate frames are considered for the cameras, objects, and robot end-effector.

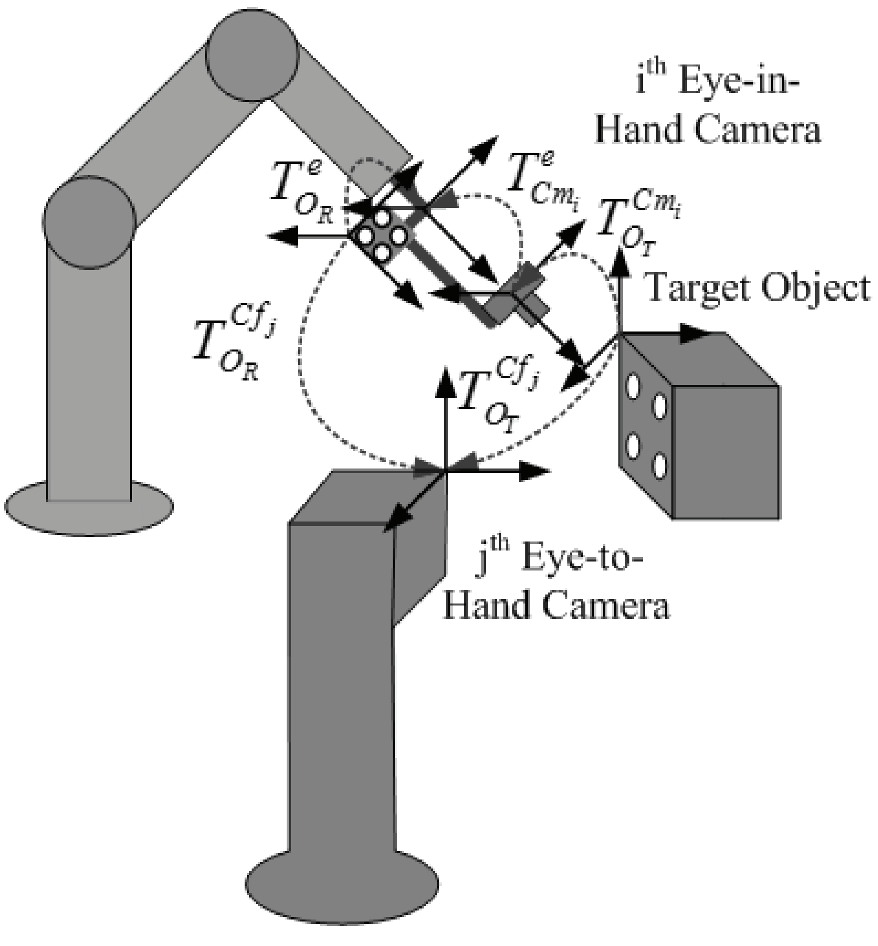


Figure 2.1 System configuration: The eye-in-hand (mobile) cameras have a direct view of the target object, while the eye-to-hand (fixed) cameras have the auxiliary object in their FOV.

The relative pose between α and β coordinate frames is expressed by the transformation matrix, H_{α}^{β}. The transformation matrix is composed of rotation matrix $R_{\alpha}^{\beta}(\phi_{\alpha}^{\beta})$ (where $\phi_{\alpha}^{\beta}=\begin{bmatrix}\varphi_{\alpha}^{\beta} & \theta_{\alpha}^{\beta} & \psi_{\alpha}^{\beta}\end{bmatrix}^{T}$ is the vector of Euler angles with roll, pitch, and yaw elements) and the translation vector ($t_{\alpha}^{\beta}=\begin{bmatrix}x_{\alpha}^{\beta} & y_{\alpha}^{\beta} & z_{\alpha}^{\beta}\end{bmatrix}^{T}$) as follows,

$$H_{\alpha}^{\beta}=\begin{bmatrix} R_{\alpha}^{\beta} & t_{\alpha}^{\beta} \\ 0_{1\times 3} & 1 \end{bmatrix}, \tag{2.1}$$

where $0_{1\times 3}$ is a row vector with 3 zero elements. The relative pose of the target object with respect to the robot end-effector is desirable and is obtained separately for EIH cameras,

$$H_{O_T}^{e} = H_{Cm_i}^{e} H_{O_T}^{Cm_i}, \tag{2.2}$$

and ETH cameras,

$$H_{O_T}^{e} = H_{O_R}^{e} \left(H_{O_R}^{Cf_j} \right)^{-1} H_{O_T}^{Cf_j}. \tag{2.3}$$

Here Cm_i, Cf_j, O_T, O_R, and e denote the coordinate frames of ith EIH camera, jth ETH camera, target object, end-effector object and end-effector, respectively. A few of these transformation matrices are obtained through calibration procedures (e.g., $H_{Cm_i}^{e}$ and $H_{O_R}^{e}$), while others are calculated directly or indirectly through camera images.

The coordinates of a point of interest of the object in a camera frame is calculated using the transformation matrix,

$$\tilde{P}_{O,i}^{C} = H_{O}^{C} \tilde{P}_{i}^{O}, \tag{2.4}$$

where $\tilde{P}_{i}^{O} = \begin{bmatrix} x_i^O & y_i^O & z_i^O & 1 \end{bmatrix}^T$ is the vector of homogenous coordinates of point i in the object frame, which is known a priori from the Computer Aided Design (CAD) model of the object, and $\tilde{P}_{O,i}^{C} = \begin{bmatrix} x_{O,i}^C & y_{O,i}^C & z_{O,i}^C & 1 \end{bmatrix}^T$ is the vector of homogenous coordinates of the same point in the camera frame.

The pinhole camera model is considered for the image projection. Point features are exploited since they are simple to extract. A point in camera frame $\tilde{P}_{O,i}^{C}$ is projected onto the camera image plane, i.e.,

$$p_{O,i}^{C} = \begin{pmatrix} u_{O,i}^{C} & v_{O,i}^{C} \end{pmatrix} = f_i(\tilde{P}_{O,i}^{C}) = \begin{pmatrix} \dfrac{x_{O,i}^{C}}{z_{O,i}^{C}} & \dfrac{y_{O,i}^{C}}{z_{O,i}^{C}} \end{pmatrix}, \tag{2.5}$$

where $p_{O,i}^{C}$ is the image plane projection of point $\tilde{P}_{O,i}^{C}$, $u_{O,i}^{C}$ and $v_{O,i}^{C}$ are its image coordinates. Without the loss of generality, the focal length in (2.5) is assumed to be unity. Here $f_i(\tilde{P}_{O,i}^{C})$ is a nonlinear function that maps the homogeneous coordinates of point i to its image plane coordinates. The image plane projection of each point is obtained from image feature points using camera intrinsic parameters, which are provided by the camera calibration procedure. The image plane coordinates of the feature points are exploited to estimate the object pose afterward. It is worth mentioning that the extracted image features are noisy. In this work, the image noise is assumed to be a zero-mean Gaussian variable.

In this chapter, pose estimation methods used for sensor fusion are discussed. The main focus of the chapter is on Kalman filter-based methods, which have several advantages over other pose estimation methods such as recursive implementation, capability to be used for sensor fusion and image feature windowing, and temporal filtering [2.64]. In addition, virtual visual servoing and Gauss-Newton methods are presented as alternatives. The latter methods are optimisation-based methods and are closely related as shown in the sequel.

2.4 Kalman Filter-Based Methods

Kalman filter-based techniques have been commonly used for pose estimation applications. As the pose estimation problem is nonlinear, extensions of Kalman filters for nonlinear systems, namely EKF and UKF are employed. This section briefly introduces the linear Kalman filter structure, followed by pose estimation methods through employing EKF and UKF. These methods are then robustified against uncertainties in estimation via novel iterative and adaptive techniques as follows.

2.4.1 Linear Kalman Filter

Kalman filter in general is a sequential procedure to optimally estimate the state of the system, ω, through noisy measurements, ξ, in a linear system. For this purpose, a linear model of system dynamics and measurements is exploited. At each time step, the current state is assumed to be related to the previous state with some uncertainties. That is,

$$\omega_{k+1} = A_k \omega_k + q_k \, , \tag{2.6}$$

where A_k represents the dynamics of the system and q_k is a zero mean Gaussian noise with the covariance of Q_k. In addition to that, the state of the system is related to system measurements through the measurement mapping matrix, F_k, written as,

$$Y_k = F_k \omega_k + r_k \,, \tag{2.7}$$

where r_k is a zero mean Gaussian noise with the covariance of Γ_k. Here $\hat{\omega}_k$ denotes the state estimation. Equations (2.6) and (2.7) form the foundation of the optimal estimation through Kalman filtering, which is outlined as follows. First the initial value of the system state is estimated (usually through other methods, or directly through measurements). Moreover, the error covariance of this initial estimation is approximated,

$$\hat{\omega}_0 = \omega(0) \,, \tag{2.8}$$

$$C_0 = C(0) \,, \tag{2.9}$$

where C_k denotes the error covariance matrix at time step *k*. Next, at each prediction step, the state and error covariance of the state are predicted based on the dynamics of the system, described in (2.5).

$$\hat{\omega}_{k|k-1} = A_k \hat{\omega}_{k-1} \,, \tag{2.10}$$

$$C_{k|k-1} = A_k C_{k-1} A_k^T + Q_k \,. \tag{2.11}$$

Here, $\hat{\omega}_{k|k-1}$ and $C_{k|k-1}$ are the predictions of the state estimation and error covariance matrix, respectively. Once the measurements are available, the state of the system and the associated error covariance are adjusted as follows,

$$\hat{\omega}_k = \hat{\omega}_{k|k-1} + K_k(\xi_k - F_k \hat{\omega}_{k|k-1}) \,, \tag{2.12}$$

$$C_k = (I - K_k F_k) C_{k|k-1} (I - K_k F_k)^T + K_k \Gamma_k K_k^T \,, \tag{2.13}$$

where,

$$K_k = C_{k|k-1} F_k^T (F_k C_{k|k-1} F_k^T + \Gamma_k)^{-1}, \tag{2.14}$$

and I is the identity matrix. It is worth mentioning that if the measurements are accurate (i.e., $\Gamma_k \approx 0$), (2.12) and (2.14) yield,

$$\hat{\omega}_k \approx F_k^{-1} \xi_k . \tag{2.15}$$

On the other hand, noisy measurements cause the magnitude of the Kalman gain to be small and therefore,

$$\hat{\omega}_k \approx \hat{\omega}_{k|k-1} . \tag{2.16}$$

It can readily be noted that (2.6) and (2.7) model a system with linear dynamics and linear measurement functions. Since the measurement function in the pose estimation problem is nonlinear, the formulation of the filter requires proper modifications, which is discussed in the sequel.

2.4.2 Nonlinear Kalman Filter

The application of Kalman filters can be extended to systems with nonlinear dynamics and/or measurement functions. Two types of filters were introduced for this purpose. In EKF, the state of the system is estimated by linearizing the system around the current state, while sigma points are exploited in UKF to serve this purpose. These methods are explained for pose estimation as follows.

Extended Kalman Filter

Since the camera projection model is nonlinear, the extended Kalman filter (EKF) is considered as a base for pose estimation. The system state vector entails parameters of the relative pose of the target object with respect to the robot's end-effector and its velocity,

$$\omega_k = \begin{bmatrix} t_{O_T}^e & \phi_{O_T}^e & \dot{t}_{O_T}^e & \dot{\phi}_{O_T}^e \end{bmatrix}^T, \tag{2.17}$$

where ω_k is the state vector at time step k. The system motion and measurement models are as follows,

$$\omega_k = A\omega_{k-1} + q_k , \tag{2.18}$$

$$\xi_k = F(\omega_k) + r_k , \tag{2.19}$$

where A is the system dynamics matrix, ξ_k and F denote the measurement vector and modeling function, r_k is the measurement noise, and q_k represents the process noise. Assuming a constant velocity model, the matrix A is defined as follows,

$$A = \begin{bmatrix} I_6 & \Delta t I_6 \\ 0_6 & I_6 \end{bmatrix}, \tag{2.20}$$

where I_6 is a 6×6 identity matrix, 0_6 is a 6×6 matrix of zeros , and Δt is the sampling time of the system. The camera measurement vector is defined as,

$$\xi_k = \begin{bmatrix} u_{O,1}^C & v_{O,1}^C & . & . & u_{O,n}^C & v_{O,n}^C \end{bmatrix}, \tag{2.21}$$

where n is the number of selected feature points for pose estimation. The measurement modeling function is described as,

$$F(\omega_k) = \begin{bmatrix} f_1(\omega_k) & . & . & f_n(\omega_k) \end{bmatrix}, \tag{2.22}$$

where,

$$f_i(\omega_k) = f_i(\hat{P}_{O,i}^C) = \begin{pmatrix} \dfrac{\hat{x}_{O,i}^C}{\hat{z}_{O,i}^C} & \dfrac{\hat{y}_{O,i}^C}{\hat{z}_{O,i}^C} \end{pmatrix}. \tag{2.23}$$

Here $\hat{P}_{O,i}^C$ is the estimation of point i coordinates from the object (i.e., estimation of $\tilde{P}_{O,i}^C$). The EKF is formulated as follows. First in the prediction phase the system state and its error covariance are predicted based on the dynamics of the system,

$$\hat{\omega}_{k|k-1} = A\hat{\omega}_{k-1}, \tag{2.24}$$

$$C_{k|k-1} = AC_{k-1}A^T + Q, \tag{2.25}$$

where $\hat{\omega}_{k|k-1}$ and $\hat{\omega}_{k-1}$ are a priori and a posteriori state estimation vectors respectively. Moreover, C_{k-1} and $C_{k|k-1}$ denote a priori and a posteriori estimation-error covariance matrices, respectively. The process noise matrix is denoted by Q and is defined as follows,

$$Q = E\left\{q_k {q_k}^T\right\}, \tag{2.26}$$

where $E\{\cdot\}$ is the expectation function. Next, the state vector and error covariance are updated, based on the new measurements, in the estimation phase.

$$F_k = \left.\frac{\partial F(\omega)}{\partial \omega}\right|_{\omega=\hat{\omega}_{k|k-1}} = \left[\frac{\partial f_1(\omega)}{\partial \omega} \quad . \quad . \quad \frac{\partial f_n(\omega)}{\partial \omega}\right]_{\omega=\hat{\omega}_{k|k-1}}, \tag{2.27}$$

$$K_k = C_{k|k-1}F_k^T\left(\Gamma_k + F_kC_{k|k-1}F_k^T\right)^{-1}, \tag{2.28}$$

$$\hat{\omega}_k = \hat{\omega}_{k|k-1} + K_k\left(\xi_k - F(\hat{\omega}_{k|k-1})\right), \tag{2.29}$$

$$C_k = \left(I_{12} - K_kF_k\right)C_{k|k-1}\left(I_{12} - K_kF_k\right)^T + K_k\Gamma_kK_k^T. \tag{2.30}$$

Here I_{12} is a 12×12 identity matrix, F_k is the Jacobian matrix, K_k is the Kalman gain, and Γ_k is the measurement noise covariance. It should be noted that the differentiability of the image mapping function (f) certifies the differentiability of the measurement modeling function (F). Each element of the Jacobian matrix is calculated as follows,

$$\frac{\partial f_i(\omega)}{\partial \omega} = \frac{\partial f_i(\tilde{P}^C_{O,i})}{\partial \tilde{P}^C_{O,i}}\frac{\partial \tilde{P}^C_{O,i}}{\partial \omega}. \tag{2.31}$$

The first term of the right hand side of (2.31) is derived as follows,

$$\frac{\partial f_i(\tilde{P}_{O,i}^C)}{\partial \tilde{P}_{O,i}^C} = \begin{bmatrix} \frac{1}{z_{O,i}^C} & 0 & \frac{-x_{O,i}^C}{z_{O,i}^C} & 0 \\ 0 & \frac{1}{z_{O,i}^C} & \frac{-y_{O,i}^C}{z_{O,i}^C} & 0 \end{bmatrix}, \tag{2.32}$$

whereas the calculation of the second term of (2.31) is followed separately for EIH and ETH cameras. Recalling from the previous section, the 3D Cartesian parameters of the *j*th point on target object in the *i*th EIH camera frame is calculated as follows,

$$\tilde{P}_{O_T,j}^{Cm_i} = H_e^{Cm_i} H_{O_T}^e \tilde{P}_j^{O_T} . \tag{2.33}$$

Therefore, in case of EIH cameras,

$$\frac{\partial \tilde{P}_{O_T,i}^{Cm_j}}{\partial \omega} = H_e^{Cm_i} \frac{\partial H_{O_T}^e \tilde{P}_i^{O_T}}{\partial \omega}, \tag{2.34}$$

and for ETH cameras,

$$\tilde{P}_{O_R,j}^{Cf_j} = H_{O_T}^{Cf_j} \left(H_{O_T}^e \right)^{-1} H_{O_R}^e \tilde{P}_j^{O_R}, \tag{2.35}$$

$$\frac{\partial \tilde{P}_{O_R,i}^{Cf_j}}{\partial \omega} = H_{O_T}^{Cf_i} \frac{\partial \left(H_{O_T}^{e\ -1} H_{O_R}^e \tilde{P}_i^{O_R} \right)}{\partial \omega}. \tag{2.36}$$

Derivation of (2.36) is straightforward since $H_{O_R}^e$ and $\tilde{P}_i^{O_R}$ parameters are known a priori and the matrix $H_{O_T}^e$ is a function of ω.

Unscented Kalman Filter

The adaptation to nonlinearities of the system may also be performed through the unscented transformations [2.48]. UKF estimates the system's state and its error covariance by means of data sampling. First the sampled data, known as sigma points, are selected,

$$\hat{\omega}_{k|k-1}^{\pm i} = \hat{\omega}_{k-1} \pm \left[\sqrt{n' C_{k-1}} \right]_i, \tag{2.37}$$

where n' is the size of the state, $\left(\sqrt{n'C_{k-1}}\right)^T\left(\sqrt{n'C_{k-1}}\right)=n'C_{k-1}$, and $\hat{\omega}_{k|k-1}^{\pm i}$ are the sigma points. Next, the sigma points are passed through the system's dynamics to form the state and its error covariance predictions,

$$\hat{\omega}_{k|k-1}=\frac{1}{2n}\sum_{i=1}^{2n'}\hat{\omega}_{k|k-1}^{\pm i}\ , \tag{2.38}$$

$$C_{k|k-1}=\frac{1}{2n'}\sum_{i=1}^{2n'}\{(\hat{\omega}_{k|k-1}^{\pm i}-\hat{\omega}_{k|k-1})(\hat{\omega}_{k|k-1}^{\pm i}-\hat{\omega}_{k|k-1})^T\}+Q_k. \tag{2.39}$$

Similar procedure is taken to calculate the state and it error covariance estimations,

$$\hat{\omega}_k^{\pm i}=\hat{\omega}_{k|k-1}\pm\left[\sqrt{nC_{k|k-1}}\right]_i, \tag{2.40}$$

$$\hat{Y}_k^{\pm i}=f_i\left(\hat{\omega}_k^{\pm i}\right), \tag{2.41}$$

$$\hat{Y}_k=\frac{1}{2n}\sum_{i=1}^{2n'}\hat{Y}_k^{\pm i}\ , \tag{2.42}$$

$$C_{y,k}=\frac{1}{2n'}\sum_{i=1}^{2n'}(\hat{Y}_k^{\pm i}-\hat{Y}_k)(\hat{Y}_k^{\pm i}-\hat{Y}_k)^T+\Gamma_k\ , \tag{2.43}$$

$$C_{\omega,y,k}=\frac{1}{2n'}\sum_{i=1}^{2n'}(\hat{\omega}_k^{\pm i}-\hat{\omega}_{k|k-1})(\hat{Y}_k^{\pm i}-\hat{Y}_k)^T\ , \tag{2.44}$$

which result in,

$$K_k=C_{x,y,k}C_{y,k}^{-1}, \tag{2.45}$$

$$\hat{\omega}_k=\hat{\omega}_{k|k-1}+K_k(\xi_k-F(\hat{\omega}_{k|k-1}))\ , \tag{2.46}$$

$$C_k=C_{k|k-1}-K_kC_{y,k}K_k^T\ . \tag{2.47}$$

It was shown that UKF results in more accurate estimation for nonlinear systems when compared to EKF [2.69]. Yet, this algorithm also approximates the nonlinear system and it provides sub-optimal estimates of the real system. It is worth mentioning that Kalman filters were designed under system's zero-mean Gaussian noise assumption. If the noise of the system is non-Guassian, Kalman filter remains the optimal linear estimator. However, particle filters (PFs) may work better in this case. Yet, these filters impose high computational burdens on the system and may not be desirable, especially for real-time applications. For such cases, UKF offers a balance between low computational costs of Kalman filters and high performance of particle filters. In this chapter, the accuracy of UKF is enhanced even further through the proposed IAUKF, which is explained in the sequel.

The optimality (sub-optimality) of Kalman filters (EKF and UKF) is directly related to proper estimations of noise covariance matrices, Γ_k and Q_k, tuning of which is non-trivial. Moreover, these parameters usually vary during the estimation procedure, making the initial adjustments less effective. Adaptive schemes were proposed as a remedy to these problems, which is explained in the sequel.

2.4.3 Adaptive Schemes

Different schemes were presented to deal with the lack of knowledge about system noise statistics. These methods were mainly categorized into multiple model systems and innovation-based adaptive techniques [2.70]. The former was based on operation of multiple Kalman filters with different parameters in parallel. The closest filter output to true value was chosen as the state estimation. Since these techniques usually need a huge bank of filters and yet the optimality of the system is not guaranteed, they are not discussed in this chapter. Alternatively different adaptive schemes are introduced as follows.

Correlation-based Algorithms

In these methods, the unknown noise parameters are computed based on autocorrelation of the system measurements,

$$D_k = E\left\{\xi_i \xi_{i-k}{}^T\right\}, \tag{2.48}$$

or autocorrelation of the system innovation term,

$$E_k = E\left\{\eta_i \eta_{i-k}{}^T\right\}, \tag{2.49}$$

where $\eta_i = \xi_i - F(\omega_i)$ is the innovation term of the system. The aforementioned autocorrelations are estimated over a number of sample set,

$$\hat{D}_k = \frac{1}{N}\sum_{i=k+1}^{N} \xi_i \xi_{i-k}{}^T , \tag{2.50}$$

$$\hat{E}_k = \frac{1}{N}\sum_{i=k+1}^{N} v_i v_{i-k}{}^T , \tag{2.51}$$

where N is the number of samples. The noise matrices are then calculated based on the relationship of the autocorrelation and noise matrices,

$$\hat{\Gamma} = \hat{D}_0 - FCF^T , \tag{2.52}$$

$$\hat{Q} = C - ACA^T , \tag{2.53}$$

where C is the steady-state error covariance and is computed based on the estimates in (2.50) or (2.51). As it can be inferred, the relationship is usually based on the steady-state Kalman filter, which requires the system to be linear and stationary [2.71]. Also these methods usually perform less efficiently compared to maximum likelihood schemes [2.71], which are explained next.

Maximum Likelihood Algorithms

In these algorithms, the unknown parameters of the system are found by maximizing their likelihood,

$$\chi = \arg\max_{\chi} L(\chi), \tag{2.54}$$

where χ is the vector of unknown noise parameters and $L(\chi) = \log p\left(\chi \middle| \xi_i, .., \xi_1\right)$ is its likelihood function. The maximization procedure often involves iterative schemes, increasing the computational expense of the system. This problem can be solved by introducing the covariance matching algorithms.

Covariance Matching Algorithms

These methods compute the noise matrices based on system residuals [2.72]. The differences between state prediction and estimation, and also actual measurements and predicted measurements, are recorded as an estimation of process and measurement noise respectively,

$$\hat{q}_k = \omega_k - \omega_{k|k-1} \ , \tag{2.55}$$

$$\hat{r}_k = \xi_k - F(\omega_k) \ . \tag{2.56}$$

On the other hand, it can be shown that,

$$\operatorname{cov}(\hat{q}_k, \hat{q}_k) \approx Q_k + C_k - A_k C_{k-1} A_k^T \ , \tag{2.57}$$

$$\operatorname{cov}(\hat{r}_k, \hat{r}_k) \approx \Gamma_k + F_k C_k F_k^T \ . \tag{2.58}$$

Calculating the noise residues from (2.55), (2.56) and employing them in (2.57), (2.58) yields,

$$\hat{\Gamma}_N = \frac{1}{N-1} \sum_{i=1}^{N} \left(\hat{r}_i - \overline{r}\right)\left(\hat{r}_i - \overline{r}\right)^T - \frac{1}{N} \sum_{i=1}^{N} F_i C_i F_i^T \ , \tag{2.59}$$

$$\hat{Q}_N = \frac{1}{N-1} \sum_{i=1}^{N} \left(\hat{q}_i - \overline{q}\right)\left(\hat{q}_i - \overline{q}\right)^T - \frac{1}{N} \sum_{i=1}^{N} A C_{i-1} A^T - C_i \ , \tag{2.60}$$

where,

$$\overline{r} = \frac{1}{N} \sum_{i=1}^{N} \hat{r}_i \ , \tag{2.61}$$

$$\bar{q} = \frac{1}{N}\sum_{i=1}^{N}\hat{q}_i \,. \tag{2.62}$$

Here $\hat{\Gamma}_N$ and $\hat{Q}_N$ are the estimates of measurement and process noise covariance, respectively. One main advantage of this form of estimation is its computation that can be performed sequentially as follows,

$$\hat{\Gamma}_N = \hat{\Gamma}_{N-1} + \frac{(\hat{r}_N - \bar{r})(\hat{r}_N - \bar{r})^T}{N-1} - \frac{F_N C_N F_N{}^T}{N}, \tag{2.63}$$

$$\hat{Q}_N = \hat{Q}_{N-1} + \frac{(\hat{q}_N - \bar{q})(\hat{q}_N - \bar{q})^T}{N-1} - \frac{A C_{N-1} A^T - C_N}{N}. \tag{2.64}$$

This form of adaptation reduces the computational cost by almost *N* times (for this part) as iterating for *N* loops, shown in (2.59) and (2.60), is no longer required. It is worth mentioning that since the linearized modeling matrices (i.e., A_k and F_k) are not calculated explicitly in UKF, (2.57) and (2.58) cannot be employed directly to estimate the noise parameters. It can be shown that [2.69],

$$C_{k|k-1} - Q_K \approx A_k C_{k-1} A_K{}^T \,, \tag{2.65}$$

$$C_{y,k} \approx F_k C_k F_k^T \,, \tag{2.66}$$

which are used in conjunction with (2.57) and (2.58) to approximate the noise covariance matrices as follows,

$$\hat{\Gamma}_N = \frac{1}{N-1}\sum_{i=1}^{N}(\hat{r}_i - \bar{r})(\hat{r}_i - \bar{r})^T - \frac{1}{N}\sum_{i=1}^{N}C_{y,i} \,, \tag{2.67}$$

$$\hat{Q}_N = \frac{1}{N-1}\sum_{i=1}^{N}(\hat{q}_i - \bar{q})(\hat{q}_i - \bar{q})^T - \frac{1}{N}\sum_{i=1}^{N}C_{i|i-1} - \hat{Q}_i - C_i \,. \tag{2.68}$$

In adaptive methods, the covariance of residues is approximated over a window of past measurements. The size of this window (*N*) is usually chosen empirically [2.71]. Large window size offers a more accurate approximation at the cost of reduced flexibility to system dynamicity, while small window size may lead to system instability.

It should be mentioned that despite their good performance, the adaptive methods are sensitive to initialization and camera calibration. Moreover, these schemes could not adapt to very fast changes in the system as the linearization made by EKF or UKF become less valid. To make the system more robust against fast dynamics of the system and its parameter errors, iterative schemes are introduced.

2.4.4 Iterative Schemes

The linearization method, used in EKF and UKF, is based on local system approximation. As a result, the estimations may deviate from the original states if changes in the system state are considerable. To alleviate this limitation, iterative schemes are applied through which, the state prediction and estimation (i.e., $\hat{\omega}_{k|k-1}$ and $\hat{\omega}_k$) are recalculated iteratively, using the states from the last iteration in linearization. With enough number of iterations, the output of the system becomes steady, i.e.,

$$\omega_k^i \approx \omega_k^{i-1}, \tag{2.69}$$

where ω_k^i is the state vector at time step k after i iterations. Using (2.69) in (2.29) and (2.46) yields,

$$F\left(\omega_k^i\right) \approx \xi_k. \tag{2.70}$$

Similar results are achieved by increasing the process noise (Q) or reducing the measurement noise (Γ) in linear KFs, but not EKFs (nor UKFs). The benefit of iterative schemes is two-fold. First, these methods approximate the nonlinear functions closely, depending on the number of iterations. This property allows the system to follow highly dynamic systems. More importantly, these schemes alleviate the sensitivity of the system on the error covariance matrices (as was reported in [2.58]). Therefore, the system follows the measurements closely and does not diverge, in case of high error covariance mismatches. Higher dependency of system estimations on the measurements is rewarding when an accurate model of system's dynamics is not available (e.g., sensor data estimation). However, this benefit comes at the price of higher sensitivity to measurement noise. Moreover, iterations usually lead to high computational costs, which is not desirable for real-time systems. In such systems, the number of iterations should be selected as a compromise between the accuracy and time allowance.

Equation (2.70) may be interpreted as a nonlinear optimization solution to find the pose of the system. As an example, Gauss-Newton and VVS methods may result in similar outcomes as are explained in the sequel.

2.4.5 Iterative and Adaptive Schemes

Previously, adaptive and iterative techniques were discussed separately. Despite their advantages, each of these approaches has their shortcomings. As was mentioned before, adaptive methods are vulnerable to system uncertainties such as initialization, while the iterative methods suffer from the lack of information about the system noise. This section reviews the previously introduced IAEKF and proposes IAUKF as a novel method under the same methodology.

Iterative Adaptive Extended Kalman Filter (IAEKF)

Iterative and adaptive schemes are employed in synergy for robust and accurate pose estimation [2.64]. The adaptive algorithm adjusts the noise parameters in the filter, while the iterative scheme reduces the estimation error caused by any maladaptation. Though various adaptive schemes might be used for filter tuning, this chapter focuses on the covariance matching method described before.

It is noteworthy that the combination of these two techniques has to be conveyed carefully, not to make the system slow or unstable. For this reason, the adaptation of noise covariance matrices is performed out of the iteration loop. For a faster adaptation, only measurements of N previous samples are considered. Moreover, a fading-memory approach is taken to give higher weights (w_k) to more recent readings and lower weights ($1-w_k$) to initial samples. Finally, the covariance matrices are checked for positive definiteness.

The number of iterations is an open parameter in the system, which augments the system with additional flexibility. This parameter may be regulated by the dynamicity of the system or the convergence of the states. Alternately, the number of iterations may also be selected subject to system's computational resources since extra iterations are performed at the cost of increased computational time. Yet, thanks to recent advancements in computing systems, such problem is deemed as insignificant. The pseudocode of IAEKF with fixed number of iterations is summarized as follows:

```
Pseudocode 1: IAEKF Pseudocode

Initialize $\omega_0$ and $C_0$.

FOR $k$ =1:$M$

        $\omega_{k|k-1} = A_k \omega_{k-1}$

        $C_{k|k-1} = A_k C_{k-1} A_k^T + Q_k$

        $\omega_k^1 = \omega_{k|k-1}$

        FOR $j$ =1: $\Upsilon$

                $F_k^j = \left. \frac{\partial F}{\partial \omega} \right|_{\omega = \omega_k^j}$

                $K_k^j = C_{k|k-1} F_k^{jT} (F_k^j C_{k|k-1} F_k^{jT} + \Gamma_k)^{-1}$

                $\omega_k^{j+1} = \omega_k^j + K_k^j (\xi_k - F(\omega_k^j))$

        END FOR

        $C_k = (I - K_k^\Upsilon F_k^\Upsilon) C_{k,k-1} (I - K_k^\Upsilon F_k^\Upsilon)^T + K_k^\Upsilon \Gamma_k K_k^{\Upsilon T}$

        $q_k = \omega_k^{\Upsilon+1} - \omega_{k|k-1}^{\tau+1}$
```

$$\bar{q}_k = \frac{1}{N}\sum_{l=k-N+1}^{N} q_l$$

$$r_k = \xi_k - F(\omega_k^{\Upsilon+1})$$

$$\bar{r}_k = \frac{1}{N}\sum_{l=k-N+1}^{N} r_l$$

$$\begin{aligned} Q_{k+1} = Q_k + \frac{w_k}{N-1}\Big\{ & (q_k-\bar{q}_k)(q_k-\bar{q}_k)^T - (q_{k-N}-\bar{q}_k)(q_{k-N}-\bar{q}_k)^T + (q_k-q_{k-N})(q_k-q_{k-N})^T \\ & + \frac{N-1}{N}\left(C_k - A_k^{\tau} C_{k-1} A_k^{\tau T} + A_{k-N}^{\tau} C_{k-N-1} A_{k-N}^{\tau}{}^{T} - C_{k-N}\right)\Big\} \end{aligned}$$

$$\begin{aligned} \Gamma_{k+1} = \Gamma_k + \frac{w_k}{N-1}\Big\{ & (r_k-\bar{r}_k)(r_k-\bar{r}_k)^T - (r_{k-N}-\bar{r}_k)(r_{k-N}-\bar{r}_k)^T + (r_k-r_{k-N})(r_k-r_{k-N})^T \\ & - \frac{N-1}{N}\left(F_k^{\tau} C_k F_k^{\tau T} - F_{k-N}^{\tau} C_{k-N} F_{k-N}^{\tau}{}^{T}\right)\Big\} \end{aligned}$$

END FOR

Iterative Adaptive Unscented Kalman Filter (IAUKF)

Iterative UKFs were previously proposed to improve nonlinear estimation; however, IAUKF is proposed *for the first time* in this work. As in IAEKF, the state prediction and estimation are done through an iterative scheme, while the noise parameters are tuned based on most recent measurements. The iterations minimize the effect of system nonlinearities on the filter, while the noise adaptation adjusts the parameters of the system exploiting a fading memory scheme. The cooperative iterative and adaptive schemes work similar to IAEKF, benefiting the system with extra accuracy and robustness to system uncertainties.

The pseudocode of IAUKF with fixed number of iterations is summarized below.

Pseudocode 2: IAUKF Pseudocode

```
Initialize $\omega_0$ and $C_0$.

FOR $k = 1{:}M$

    $\omega_{k|k-1} = \frac{1}{2n}\sum_{l=1}^{n} A_k(\omega_{k-1} \pm \left[\sqrt{nC_{k-1}}\right]_l)$

    $C_{k|k-1} = \frac{1}{2n}\sum_{l=1}^{n}\left\{(A_k(\omega_{k|k-1} \pm \left[\sqrt{nC_{k-1}}\right]_l) - \omega_{k|k-1})(A_k(\omega_{k|k-1} \pm \left[\sqrt{nC_{k-1}}\right]_l) - \omega_{k|k-1})^T\right\} + Q$

    $\omega_k^1 = \omega_{k|k+1}$

    FOR $j = 1{:}\Upsilon$

        $y_k^j = \frac{1}{2n}\sum_{l=1}^{n} F(\omega_k^j \pm \left[\sqrt{nC_{k,k-1}}\right]_l)$

        $C_{y,k}^j = \frac{1}{2n}\sum_{l=1}^{n}\left\{(F(\omega_k^j \pm \left[\sqrt{nC_{k|k-1}}\right]_l) - y_k^j)(F(\omega_k^j \pm \left[\sqrt{nC_{k|k-1}}\right]_l) - y_k^j)^T\right\} + \Gamma_k$

        $\omega_{x,y,k}^j = \frac{1}{2n}\sum_{l=1}^{n}\left\{(A_k(\omega_{k|k-1} \pm \left[\sqrt{nC_{k-1}}\right]_l) - \omega_{k|k-1})(F(\omega_k^j \pm \left[\sqrt{nC_{k|k-1}}\right]_l) - y_k^j)^T\right\}$

        $K_k^j = C_{x,y,k}^j C_{y,k}^{j\ -1}$

        $\omega_k^{j+1} = \omega_k^j + K_k^j(\xi_k - F(\omega_k^j))$

    END FOR

    $C_k = C_{k,k-1} - K_k^\Upsilon C_{y,k}^\Upsilon K_k^{\Upsilon T}$

    $q_k = \omega_k^{\Upsilon+1} - \omega_{k|k-1}$

    $\overline{q}_k = \frac{1}{N}\sum_{l=k-N+1}^{N} q_l$
```

$$r_k = \xi_k - F(\omega_k^{\Upsilon+1})$$

$$\overline{r}_k = \frac{1}{N}\sum_{l=k-N+1}^{N} r_l$$

$$Q_{k+1} = Q_k + \frac{w_k}{N-1}\Big\{(q_k - \overline{q}_k)(q_k - \overline{q}_k)^T - (q_{k-N} - \overline{q}_k)(q_{k-N} - \overline{q}_k)^T + (q_k - q_{k-N})(q_k - q_{k-N})^T$$
$$+ \frac{N-1}{N}\left(C_k - C_{k,k-1} + Q_k + C_{k-N,k-N-1} - Q_{k-N} - C_{k-N}\right)\Big\}$$

$$\Gamma_{k+1} = \Gamma_k + \frac{w_k}{N-1}\Big\{(r_k - \overline{r}_k)(r_k - \overline{r}_k)^T - (r_{k-N} - \overline{r}_k)(r_{k-N} - \overline{r}_k)^T + (r_k - r_{k-N})(r_k - r_{k-N})^T$$
$$- \frac{N-1}{N}\left(C_{y,k}^{\Upsilon} - C_{y,k-N}^{\Upsilon}{}^T\right)\Big\}$$

END FOR

2.5 Gauss-Newton Method

Another approach to solve the pose estimation problem is to find a solution that totally relies on the system measurements. The goal is then to find the state vector ω_k such that (2.70) holds. In this method, pose estimation problem is treated as a non-linear least-square minimization problem. All features from the cameras are gathered in a single vector such as *s*,

$$s = \begin{bmatrix} u_1^c & v_1^c & u_2^c & u_2^c & . & . & u_j^c & u_j^c \end{bmatrix}. \tag{2.71}$$

In addition, a set of features is calculated based on pose estimation,

$$\hat{s} = \begin{bmatrix} \frac{\hat{x}_1^c}{\hat{z}_1^c} & \frac{\hat{y}_1^c}{\hat{z}_1^c} & \frac{\hat{x}_2^c}{\hat{z}_2^c} & \frac{\hat{y}_2^c}{\hat{z}_2^c} & . & . & \frac{\hat{x}_j^c}{\hat{z}_j^c} & \frac{\hat{y}_j^c}{\hat{z}_j^c} \end{bmatrix}, \tag{2.72}$$

where $\hat{P}_j^c = \begin{bmatrix} \hat{x}_j^c & \hat{y}_j^c & \hat{z}_j^c \end{bmatrix}$ is calculated as follows,

$$\hat{P}_j^c = \hat{R}_o^{c_i} P_j^o + \hat{t}_o^{c_i} . \tag{2.73}$$

Here, the estimated values are denoted by a “hat”. The pose parameters used in (2.73) are calculated based on estimated pose of the target object with respect to the end-effector and known transformation which were mentioned in (2.2) and (2.3). The cost function is defined as the difference between current image features and estimated image features,

$$e_g = \left\| \Delta s \right\|^2 , \tag{2.74}$$

where,

$$\Delta s = \hat{s} - s . \tag{2.75}$$

One way to minimize this cost function is to employ a simple proportional control law,

$$\Delta \dot{s} = -\lambda \Delta s . \tag{2.76}$$

where λ is the control gain. This control law leads to an exponential decrease in the cost function. The changes in the cost function are related to the changes of the pose,

$$\Delta \dot{s} = \dot{\hat{s}} = J_\omega \dot{\hat{\omega}}_{o_T}^e , \tag{2.77}$$

where $\omega_{o_T}^e = \begin{bmatrix} t_{o_T}^e & \phi_{o_T}^e \end{bmatrix}$, $J_\omega = \dfrac{\partial \hat{s}}{\partial \hat{\omega}_{o_T}^e}$ and is calculated as follows for a set of feature coordinates (i.e.,$\begin{bmatrix} u_j^{c_i} & v_j^{c_i} \end{bmatrix}$),

$$J_\omega = \begin{bmatrix} \dfrac{1}{\hat{z}_j^c} & 0 & \dfrac{-\hat{x}_j^c}{\left(\hat{z}_j^c\right)^2} \\ 0 & \dfrac{1}{\hat{z}_j^c} & \dfrac{-\hat{y}_j^c}{\left(\hat{z}_j^c\right)^2} \end{bmatrix} D(\hat{\omega}_{o_T}^e) . \tag{2.78}$$

Here, $D(\omega_{o_T}^e)$ is a pose dependent matrix which is computed differently for eye-in-hand cameras,

$$D(\hat{\omega}_{O_T}^{e}) = \left[\hat{R}_{cm_i}^{e}{}^{T} \quad -\hat{R}_{cm_i}^{e}{}^{T} S\left(\hat{R}_{O_T}^{e} P_j^{O_T}\right) T\left(\hat{\varphi}_{O_T}^{e}\right) \right], \tag{2.79}$$

and eye-to-hand cameras,

$$D(\hat{\omega}_{O_T}^{e}) = \left[-\hat{R}_{cf_i}^{e}{}^{T} \quad \hat{R}_{cf_i}^{e}{}^{T} S\left(\hat{R}_{O_R}^{e} P_j^{O_R} + \hat{t}_{O_R}^{e} - \hat{t}_{O_T}^{e}\right) T\left(\hat{\varphi}_{O_T}^{e}\right) \right]. \tag{2.80}$$

Using (2.76) and (2.77), the time update of pose is calculated,

$$\dot{\hat{\omega}}_{O_T}^{e} = -\lambda J_{\omega}^{\dagger} \Delta s \,. \tag{2.81}$$

This time update is used to estimate the pose through integration from an initial guess,

$$\hat{\omega}_{O_T,k+1}^{e} = \hat{\omega}_{O_T,k}^{e} + \dot{\hat{\omega}}_{O_T}^{e} \,, \tag{2.82}$$

which leads to the same estimation of the pose as calculate by Lowe in [2.17]. The inverse of the *J* matrix may be calculated through Moore-Penrose pseudo-inverse (which leads to the Gauss-Newton solution) or damped least-square inverse (which is equivalent to Levenberg-Marquardt solution). Generally, the latter is preferred since it was proven to have stronger stability; however, assuming sufficiently closed initial guess and avoidance of singularity of the image Jacobian matrix (due to high number of features from multiple cameras), the Gauss-Newton provides the system with high accuracy estimation of the pose.

It is also worth mentioning that the measurement error Δs and also the update equation (2.82) are very similar to innovation term and update equation in EKF-based pose estimation methods. In fact, it can be shown that if the measurements of the system are fully trusted (i.e. the measurement error covariance is equal to an all zero matrix), the Iterative EKF (IEKF) approach is equivalent to Gauss-Newton solution to the problem. However, the Gauss-Newton approach has the advantage of lower computation time compared to IEKF, while preserving the same or better accuracy. The major drawback of this method is sensitivity to measurement noise, which is not accounted for.

2.6 Virtual Visual Servoing

Image-based visual servoing was employed in previous works to obtain the object relative pose [2.20]. A virtual camera was servoed through an initial known pose to a pose that minimizes the image space error, Δs. The reached pose was considered as an estimation of the current pose. The same idea is proposed in this work for fast and accurate fusion of the cameras.

Given an initial pose for the virtual manipulator, the measurements of each camera from object of interest are simulated using,

$$\hat{p}_j^c = \begin{bmatrix} \frac{\hat{x}_j^c}{\hat{z}_j^c} & \frac{\hat{y}_j^c}{\hat{z}_j^c} \end{bmatrix}, \tag{2.83}$$

where virtual point $\hat{P}_j^c = \begin{bmatrix} \hat{x}_j^c & \hat{y}_j^c & \hat{z}_j^c \end{bmatrix}$ is calculated through known transformations between objects and cameras,

$$\hat{P}_j^{cm_i} = \hat{R}_{o_T}^{cm_i} P_j^{o_T} + \hat{t}_{o_T}^{cm_i}, \tag{2.84}$$

$$\hat{P}_j^{cf_i} = \hat{R}_{o_R}^{cf_i} P_j^{o_R} + \hat{t}_{o_R}^{cf_i}. \tag{2.85}$$

It worth mentioning that, since the relative pose between object and the end-effector ($H_{o_T}^e$) is assumed to be known during virtual servoing, an estimation of all pose parameters used in (2.2) and (2.3) is virtually available at each time step. The velocity of the virtual camera is related to the changes in its image features,

$$\dot{s}_i = J_{s,i} V_{c_i}, \tag{2.86}$$

where $V_{c_i} = \begin{bmatrix} \dot{t}_{c_i} \\ \varpi_{c_i} \end{bmatrix}$ is the velocity of ith camera in its own frame, $\dot{s}_i$ is the time derivative of the feature vector of ith camera (s_i), and $J_{s,i} = \begin{bmatrix} J_{s,i,1}{}^T & . & . & J_{s,i,n}{}^T \end{bmatrix}^T$, where n is the number of features detected in the ith camera and,

$$J_{s,i.j} = \begin{bmatrix} \frac{-1}{\hat{z}_j^{c_i}} & 0 & \frac{\hat{u}_j^{c_i}}{\hat{z}_j^{c_i}} & \hat{u}_j^{c_i}\hat{v}_j^{c_i} & -1-\left(\hat{u}_j^{c_i}\right)^2 & \hat{v}_j^{c_i} \\ 0 & \frac{-1}{\hat{z}_j^{c_i}} & \frac{\hat{v}_j^{c_i}}{\hat{z}_j^{c_i}} & 1+\left(\hat{v}_j^{c_i}\right)^2 & -\hat{u}_j^{c_i}\hat{v}_j^{c_i} & -\hat{u}_j^{c_i} \end{bmatrix}. \tag{2.87}$$

Here $\dot{t}_{c_i}$ and ϖ_{c_i} are the translational and angular velocity of the i^{th} camera in its own frame, respectively. Now the virtual end-effector should be moved, exploiting the calculated velocities of all cameras. The velocity of each camera is transferable to the equivalent end-effector velocity, i.e.,

$$V_{c_i} = \Omega_e^{c_i} V_{e_i}, \tag{2.88}$$

where $V_{e_i} = \begin{bmatrix} \dot{t}_{e_i} \\ \varpi_{e_i} \end{bmatrix}$ is the velocity of the virtual end-effector in its own frame coordinates, based on the velocity of the i^{th} camera. It is straight-forward to realize this velocity transformation for the eye-in-hand virtual cameras, since they are assumed to be fixed to the robot,

$$\Omega_e^{cm_i} = \begin{bmatrix} \hat{R}_e^{cm_i} & S\left(\hat{t}_e^{cm_i}\right)\hat{R}_e^{cm_i} \\ 0 & \hat{R}_e^{cm_i} \end{bmatrix}, \tag{2.89}$$

where $S\left(\hat{t}_e^{cm_i}\right)$ is the skew-symmetric matrix of the vector $\hat{t}_e^{cm_i}$, defined as follows,

$$S\left(\begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}\right) = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix}. \tag{2.90}$$

On the other hand, the equivalent velocity of the end-effector should be computed in case of eye-to-hand virtual cameras. Similar velocity transformation matrix has been derived, with a sign difference,

$$\Omega_e^{cf_i} = \begin{bmatrix} -\hat{R}_e^{cf_i} & -S\left(\hat{t}_e^{cf_i}\right)\hat{R}_e^{cf_i} \\ 0 & -\hat{R}_e^{cf_i} \end{bmatrix}. \tag{2.91}$$

The details of the calculations are forwarded to the Appendix. The velocity of the virtual end-effector is calculated based on IBVS control law as follows,

$$V_{e_i} = -\lambda \left(J_{s,i} \Omega_e^{c_i} \right)^{\dagger} \Delta s_i \,. \tag{2.92}$$

Once the velocity of the virtual end-effector is achieved, the pose parameters updates are calculated as follows,

$$\dot{R}_o^e = S(\varpi_e) R_o^e \,, \tag{2.93}$$

$$\dot{t}_o^e = \dot{t}_e + S(\varpi_e) t_o^e \,, \tag{2.94}$$

and the pose parameters are updated through integration.

In order to speed up the algorithm even further, the use of Jacobian matrix calculated at the desired pose is proposed for calculation of (2.87). For this matter, the desired feature points are available (i.e., currant feature measurements) and the initial depth is acceptable as estimation of depth at desired location since the initial guess is sufficiently close to the desired pose.

In the next step, the velocity of different cameras should be fused into a single velocity, guiding the virtual robot towards the current pose accurately. Two different approaches are discussed for fusion of different virtual camera velocities, which are explained in the sequel.

2.7 Comparison of Optimization-Based Algorithms

In this part, it will be shown that the pose estimation through VVS leads to the same solution as Gauss-Newton. The velocity of the virtual end-effector is connected to the time derivation of the pose,

$$V_e = G \dot{\omega}_{o_T}^e \,, \tag{2.95}$$

where,

$$G = -\begin{bmatrix} I & S(\hat{t}^e_{o_T}) \\ 0 & I \end{bmatrix}\begin{bmatrix} I & 0 \\ 0 & T(\hat{\phi}^e_{o_T}) \end{bmatrix} = \begin{bmatrix} -I & -S(\hat{t}^e_{o_T})T(\hat{\phi}^e_{o_T}) \\ 0 & -T(\hat{\phi}^e_{o_T}) \end{bmatrix}. \tag{2.96}$$

Here $T(\hat{\phi}^e_{o_T})$ is the transformation matrix that relates the time derivation of the Euler angles to the angular velocity,

$$\varpi^e_{o_T} = T(\phi^e_{o_T})\dot{\phi}^e_{o_T} . \tag{2.97}$$

It is worth mentioning that while (2.95) relates the velocity of the virtual end-effector, V_e, to the time derivation of the pose, $\dot{\hat{\omega}}^e_o$, this conversion does not happen in VVS algorithm. Instead, (2.93, 2.94) are exploited to calculate the pose update parameters. By comparing (2.95) with (2.93, 2.94), one can notice that the position is updated similarly, while the orientation update is different.

Using (2.86), (2.88) and (2.89) the relation between image features time derivation and pose changes is calculated as follows,

$$\dot{s} = M\dot{\hat{\omega}}^e_{o_T} , \tag{2.98}$$

where,

$$M = J_{s,i}\Omega^{c_i}_e G . \tag{2.99}$$

Image Jacobian $J_{s,i}$, used in (2.87), can be rewritten as,

$$J_{s,i,j} = \begin{bmatrix} \frac{-1}{\hat{z}^{c_i}_j} & 0 & \frac{\hat{x}^{c_i}_j}{\left(\hat{z}^{c_i}_j\right)^2} \\ 0 & \frac{-1}{\hat{z}^{c_i}_j} & \frac{\hat{y}^{c_i}_j}{\left(\hat{z}^{c_i}_j\right)^2} \end{bmatrix}\left[I \quad -S\left(\hat{P}^{c_i}_j\right)\right]. \tag{2.100}$$

Using (2.89), (2.95), and (2.99) it can be shown that for eye-in-hand cameras,

$$M = \begin{bmatrix} \frac{1}{\hat{z}_j^{c_i}} & 0 & \frac{-\hat{x}_j^{c_i}}{\left(z_j^{c_i}\right)^2} \\ 0 & \frac{1}{\hat{z}_j^{c_i}} & \frac{-\hat{y}_j^{c_i}}{\left(\hat{z}_j^{c_i}\right)^2} \end{bmatrix} \left[\hat{R}_e^{c_i} \quad -\hat{R}_e^{c_i} S(\hat{R}_{c_i}^e P_j^{c_i} + \hat{t}_{c_i}^e - \hat{t}_{o_T}^e) T(\hat{\phi}_{o_T}^e) \right]. \tag{2.101}$$

The skew-symmetric matrix in (2.101) can be calculated as,

$$\begin{aligned} S\left(\hat{R}_{c_i}^e P_j^{c_i} + \hat{t}_{c_i}^e - \hat{t}_{o1}^e\right) &= S\left(\hat{R}_{c_i}^e \left(\hat{R}_{o1}^{c_i} P_j^{o1} + \hat{t}_{o1}^{c_i}\right) + \hat{t}_{c_i}^e - \left(\hat{R}_{c_i}^e \hat{t}_{o1}^{c_i} + \hat{t}_{c_i}^e\right)\right) \\ &= S\left(\hat{R}_{c_i}^e \hat{R}_{o1}^{c_i} P_j^{o1} + \hat{R}_{c_i}^e \hat{t}_{o1}^{c_i} - \hat{R}_{c_i}^e \hat{t}_{o1}^{c_i}\right) = S\left(\hat{R}_{o1}^e P_j^{o1}\right). \end{aligned} \tag{2.102}$$

Replacing (2.102) in (2.101) will give the same result as was calculated by (2.78) and (2.79). In case of eye-to-hand cameras,

$$M = \begin{bmatrix} \frac{1}{\hat{z}_j^{c_i}} & 0 & \frac{-\hat{x}_j^{c_i}}{\left(\hat{z}_j^{c_i}\right)^2} \\ 0 & \frac{1}{\hat{z}_j^{c_i}} & \frac{-\hat{y}_j^{c_i}}{\left(\hat{z}_j^{c_i}\right)^2} \end{bmatrix} \left[-\hat{R}_e^{c_i} \quad \hat{R}_e^{c_i} S(\hat{R}_{c_i}^e P_j^{c_i} + \hat{t}_{c_i}^e - \hat{t}_{o_T}^e) T(\hat{\phi}_{o_T}^e) \right]. \tag{2.103}$$

The skew-symmetric matrix in (2.103) is rewritten as,

$$\begin{aligned} S(\hat{R}_{c_i}^e P_j^{c_i} + \hat{t}_{c_i}^e - \hat{t}_{o_T}^e) &= S\left(\hat{R}_{c_i}^e \left(\hat{R}_{o_R}^{c_i} P_j^{o_R} + \hat{t}_{o_R}^{c_i}\right) + \hat{t}_{c_i}^e - \hat{t}_{o_T}^e\right) \\ &= S\left(\hat{R}_{c_i}^e \hat{R}_{o_R}^{c_i} P_j^{o_R} + \left(\hat{R}_{c_i}^e \hat{t}_{o_R}^{c_i} + \hat{t}_{c_i}^e\right) - \hat{t}_{o_T}^e\right) = S\left(\hat{R}_{o_R}^e P_j^{o_R} + \hat{t}_{o_R}^e - \hat{t}_{o_T}^e\right), \end{aligned} \tag{2.104}$$

which makes the matrix M similar to its counterpart in Gauss-Newton method, described in (2.78) and (2.80).

It is very important to know that these two approaches are basically equivalent; however, their minor differences separate them from each other in terms of accuracy and speed. The first major difference between the two algorithms comes from the variation in the formation of the Jacobian matrix in these two methods. The Jacobian matrix in VVS is composed of two parts, i.e., image Jacobian L_i which is dependent of virtual features, and the velocity transformation matrix $\Omega_e^{c_i}$

which is depending only on virtual pose parameters. Therefore, the Jacobian matrix is calculated quickly via a single multiplication of the image Jacobian of all features by the velocity transformation matrix. On the other hand, the Jacobian matrix used in the Gauss-Newton method is composed of two matrices that are both dependent on the feature, as was shown in (2.78-2.80). As a result, the Jacobian is formed by concatenating the multiplications shown in (2.78) for each set of features. In programs that are optimized for matrix operations (such as MATLAB®), the former implementation is performed faster, despite their similarity in the hypothetical computational costs.

The second major deviation occurs in the orientation updates of these methods. The VVS algorithm exploits the virtual end-effector velocity, V_e, to update the rotation matrix as was described in (2.93). On the other hand, the Gauss-Newton algorithm calculates the Euler angles at each iteration using (2.95), and then revert them back to update the rotation matrix, used in (2.79-2.80). The transformation of end-effector angular velocity to Euler angles and then back to rotation matrix causes an additional operation which may increase the system uncertainty. In order to follow this matter more closely, the error propagation from the end-effector angular velocity to the rotation matrix is examined for each of these two methods. The methodology of error propagation presented in [2.73] is exploited for this purpose. By employing (2.95), the noise of the rotation matrix update ($\dot{R}^e_{o_T}$) in the VVS method is found as,

$$\varepsilon_{\dot{R}^e_{o_T}} = S(\varepsilon_{\varpi_e})\dot{R}^e_{o_T}, \tag{2.105}$$

where $\varepsilon_{\dot{R}^e_{o_T}}$ and ε_{ϖ_e} are the noises of $\dot{R}^e_o$ and ϖ_e, respectively. After the integration, the resultant noise of the rotation matrix in the VVS fusion method is calculated as,

$$\varepsilon_{R^e_{o_T}} = S(\varepsilon_{\varpi_e})R^e_{o_T}\Delta t. \tag{2.106}$$

In case of the Gauss-Newton method, the noise of the end-effector angular velocity is initially transferred to the Euler angles and is calculated using (2.97), i.e.,

$$n_{\phi^e_{o_T}} = T(\phi^e_{o_T})^{-1} n_{\varpi_e}\Delta t. \tag{2.107}$$

The noise of the rotation matrix is found as,

$$\varepsilon_{R_{O_T}^e} = \dot{R}_z R_y R_x \varepsilon_{\varphi_{O_T}^e} + R_z \dot{R}_y R_x \varepsilon_{\theta_{O_T}^e} + R_z R_y \dot{R}_x \varepsilon_{\psi_{O_T}^e}\ , \tag{2.108}$$

where R_z , R_y , R_x are the rotation matrix components related to Euler angles (i.e., $R_{O_T}^e = R_z R_y R_x$), and $\varepsilon_{\varphi_{O_T}^e}$, $\varepsilon_{\theta_{O_T}^e}$, $\varepsilon_{\psi_{O_T}^e}$ are the noise of the Euler angles $\varphi_{O_T}^e$, $\theta_{O_T}^e$, and $\psi_{O_T}^e$, respectively. Reformulating (2.108) results in,

$$\varepsilon_{R_{O_T}^e} = S\left(\begin{bmatrix} \varpi_\varphi & R_z \varpi_\theta & R_z R_y \varpi_\psi \end{bmatrix} \varepsilon_{\phi_{O_T}^e}\right) R_{O_T}^e\ , \tag{2.109}$$

where,

$$\varpi_\varphi = \begin{bmatrix} 0 & 0 & \dot{\varphi}_{O_T}^e \end{bmatrix}, \varpi_\theta = \begin{bmatrix} 0 & \dot{\theta}_{O_T}^e & 0 \end{bmatrix}, \varpi_\psi = \begin{bmatrix} \dot{\psi}_{O_T}^e & 0 & 0 \end{bmatrix}. \tag{2.110}$$

It can be shown that,

$$\begin{bmatrix} \varpi_\varphi & R_z \varpi_\theta & R_z R_y \varpi_\psi \end{bmatrix} = T(\phi_{O_T}^e) \begin{bmatrix} \dot{\varphi}_{O_T}^e & 0 & 0 \\ 0 & \dot{\theta}_{O_T}^e & 0 \\ 0 & 0 & \dot{\psi}_{O_T}^e \end{bmatrix}. \tag{2.111}$$

Using (2.107-2.111), the rotation matrix noise may be formulated as,

$$\varepsilon_{R_{O_T}^e} = S(T(\phi_{O_T}^e) \begin{bmatrix} \dot{\varphi}_{O_T}^e & 0 & 0 \\ 0 & \dot{\theta}_{O_T}^e & 0 \\ 0 & 0 & \dot{\psi}_{O_T}^e \end{bmatrix} T(\phi_{O_T}^e)^{-1} \varepsilon_{\varpi_e}) R_{O_T}^e \Delta t\ . \tag{2.112}$$

Comparing (2.112) with (2.106), it is obvious that noise of the rotation matrix in Gauss-Newton method is a function of changes in orientation, which is a direct function of the convergence speed of the algorithm. High rates of change in orientation may lead to high noise projection in the estimated pose. It is worth mentioning that the noise in the orientation affects the translation as well, through matrix *G*, described in (2.96). Therefore, it may be concluded that the VVS method is more robust to noise, compared to the Gauss-Newton method.

2.8 Summary

Accurate pose estimation of the object is a crucial part of many RVS systems. Various techniques were introduced previously to address this concern. However, only a few of these methods are suitable for sensor fusion. In addition, the accuracy and robustness of these methods vary from one to another. This chapter was devoted to robust and accurate pose estimation techniques which had application in sensor fusion. For this purpose, Kalman filter-based methods were introduced as the most important pose estimation technique with several advantages. The shortcomings of these methods were alleviated through the introduction of iterative and adaptive techniques. In particular, IAEKF and IAUKF were proposed for robust and accurate pose estimation. In addition, well-known Gauss-Newton method for pose estimation was developed. Moreover, VVS for pose estimation was reformulated as an accurate optimization-based technique. It was shown that this method outperforms Gauss-Newton approach in terms of accuracy and speed. The developed pose estimation techniques are to be used for sensor fusion in the coming chapter. The performance assessment of these techniques is therefore referred to the next chapter.

Chapter 3

Sensor Fusion

3.1 Introduction

As it was mentioned ealier, robust and accurate pose estimation is required by many RVS systems. The performance of these RVS systems is dependent on the accuracy of the estimated pose. Usually, single camera systems are exploited for this purpose; however, the performance of monocular vision is limited. Multi-camera systems are proposed as a remedy to this problem. Then, sensor fusion techniques are employed to combine multiple sensor data synergistically, enhancing the richness of the outcome data. Therefore, the goal is to use the data from multiple cameras to reach a highly accurate and robust estimation of the object pose. Previously EKF-based central fusion and bundle adjustment were introduced as effective methods for sensor fusion to obtain accurate pose estimations. However, these methods were prone to system parameter uncertainty and measurement noise. Moreover, the data from cameras were fused without discrimination, which could lead to inferior results.

This chapter proposes three fusion structures for pose estimation, namely centralized, decentralized, and pre-processing fusions. Centralized fusion offers high accuracy at the price of increased computation time. Two novel centralized fusion techniques, namely Iterative Adaptive Extended Kalman Filter (IAEKF) and VVS are introduced for this matter and their performance was shown to be superior compared to their previously introduced competitors. On the other hand, decentralized fusion provides the system with a faster estimation by sacrificing the accuracy partially. This work proposes two decentralized fusion methods based on extended Kalman filter (EKF) and VVS for pose estimation. Finally, a pre-processing fusion technique is discussed which enables the system to fuse the image information from multiple cameras prior to their processing. The main advantage of this method is its independency from the pose estimation technique involved, which makes it suitable for any available pose estimation method.

3.2 Literature Survey

Multi-camera configurations were previously proposed to overcome the deficiencies of a single camera. These systems were capable of enhancing the overall accuracy and robustness of the estimation and also increasing the total FOV. Multiple cameras could benefit different situations. In most scenarios, all of the cameras had the target object in their field-of-view. The transformations between the individual cameras were known through pre-calibration of the system (for fixed cameras) or forward kinematics of calibrated robots (for end-mounted cameras). The visibility of the object (in at least one of the cameras) was crucial for such schemes. In addition, the common presence of errors in object modeling (e.g., object CAD modeling errors) impaired the overall estimation significantly. Only few works have addressed this problem so far. For instance, multiple eye-in-hand cameras with different FOVs were employed for visual servoing in [3.1]. In addition, a recent work has benefited from using several eye-in-hand/eye-to-hand cameras to control the robot more accurately [3.2]. Moreover, multiple fixed cameras were used in [3.3] for accurate visual servoing. A combination of eye-in-hand and eye-to-hand cameras was exploited to overcome occlusion using epipolar geometry-based visual servoing [3.4]. A multi camera system was utilized to servo a robot and a stereo rig simultaneously [3.5]. Several eye-in-hand cameras were used for accurate visual servoing in [3.1]. Also control level fusion of multiple eye-in-hand/eye to-hand cameras was implemented for robust servoing [3.6]. In addition, multi-camera configurations were previously employed for 3D reconstruction [3.7] and object recognition [3.8]. However, none of these works focused on the pose estimation problem. Multiple camera pose estimation schemes were traditionally based on stereo vision technique [3.9], thus facing several issues such as system calibration and camera correspondence problems. In order to fully utilize the data from each camera and improve the estimation robustness, data fusion techniques were introduced.

A full realisation of potential capacity given by multiple sensors was achieved by techniques known as sensor fusion methods. These methods were generally categorized into centralized and decentralized fusion algorithms. Centralized fusion methods, also known as measurement fusion, took all measurement data into a central unit and estimated the pose based on all available measurements. A well-known example of this method (i.e., bundle adjustment) was proposed in [3.10], where a weighted least square approach was taken to acquire the pose estimation. A similar method based on Extended Kalman Filter (EKF) was also introduced in [3.11, 3.12] for the same purpose. The advantage of centralized fusion methods was the minimal loss of information, whereas their computational cost grew drastically as the number of sensors increases. Moreover, they were prone to faulty measurements or outliers since they combine all data from every sensor without discrimination.

Decentralized fusion methods, on the other hand, relied on local estimators that each processed the data from a single sensor. The outcome of all estimators were then fused together to form the optimal/suboptimal estimation. These methods were generally less accurate than centralized methods, yet the fused information was more accurate than any local estimation [3.13]. In return, they were computationally lighter and could survive false measurements by isolating the corresponding estimator [3.14]. Aside from the aforementioned fusion structures, the fusion techniques also played a major role in the success of the overall fusion. Various strategies were employed for sensor fusion, among which Kalman filtering remained one of the most popular methods [3.14-3.17].

Kalman filter was a well-known sequential estimation method which could provide optimal results under certain conditions. The main advantages of Kalman filtering were its optimality and simplicity, which made it appealing for real-time applications. Due to the nonlinearities of many measurement systems, extended Kalman filter (EKF) was developed and exploited for fusion in many applications such as odometry [3.18], tracking [3.19], navigation [3.20, 3.21] and pose estimation [3.22]. However, EKF performance was degraded drastically when the system noise conditions were unknown. Moreover, EKF was sensitive to high dynamics of the system, since it assumed a linearized model of the system which was valid only locally.

Several remedies were proposed to address the shortcomings of EKF-based estimators. Unscented Kalman filter (UKF) was introduced to increase the level of accuracy in linearization [3.23] and was exploited for sensor fusion in many applications such as navigation [3.24] and cooperative driving [3.25]. As was discussed by [3.26] and [3.27], EKF had the advantage of lower computational complexity, whereas UKF was more accurate and did not require an explicit Jacobian calculation. Yet, UKF could not fully address the aforementioned limitations. Adaptive techniques were introduced to tune the filter noise parameters and were applied both to EKF

[3.28-3.31] and UKF [3.32, 3.33]. The adaptations were mainly performed based on previous measurements of the system, which led to low convergence rates. Besides, these methods were unable to cope with the initialization errors. Iterative schemes were proposed to compensate the linearization error in EKF [3.34-3.36] and UKF [3.37, 3.38] estimators. However, these methods placed overconfidence on the measurements. In addition, an adaptive iterated Kalman filter was briefly introduced in [3.39]; however, the method did not find wide applications. Recently, a novel technique based on iterative and adaptive schemes was adopted for pose estimation [3.40]. The method was shown to have superior performance, compared to other Kalman filter-based estimators. Nonetheless, the method was applied only for monocular pose estimation.

In spite of the aforementioned advances in adaptive and nonlinear Kalman filtering, only few of these techniques were applied to sensor fusion. A number of works proposed adaptive EKF [3.41, 3.42] and UKF [3.43] for specific sensor fusion applications. However, these methods suffered from the aforementioned shortcomings. In addition, a self-tuning approach based on the Auto-Regressive Moving Average (ARMA) innovation model was introduced for linear sensor fusion [3.44]; however the method was not extendable to nonlinear systems. Furthermore, similar to other adaptive techniques, the convergence rate of this method was low; thus the approach was more suitable for linear systems with constant error covariance.

In this chapter, sensor fusion techniques for the proposed pose estimation methods are discussed. Three possible fusion structures, namely centralized, decentralized, and pre-processing schemes are introduced. The centralized methods fuse the measurements together and use the fused data as the input to the pose estimating unit, while the decentralized methods estimate the pose parameters separately and then fuse their estimated states. As a result, they are also known as measurement and state fusion algorithms, respectively. Pre-processing techniques combine data from all cameras before processing and send it to the pose estimation unit as a single unified source of data. Each of these schemes is discussed separately as follows.

3.3 Centralized Sensor Fusion

3.3.1 Centralized Kalman Filter-Based Fusion

To fuse the measurement data in Kalman filter-based methods (e.g. IAEKF), the data from each source is merged with others into a large measurement vector. The measurement data from different cameras are first concatenated into a single measurement vector,

$$\xi_k = \begin{bmatrix} \xi_{1,k} & . & . & \xi_{l,k} \end{bmatrix}, \tag{3.1}$$

where ξ_k is the overall measurement vector, $\xi_{i,k}$ is the measurement from the i^{th} camera (either EIH or ETH), and l is the total number of the cameras. The measurement modeling function is then modified as,

$$F(\omega_k) = [F_1(\omega_k) \quad . \quad . \quad F_l(\omega_k)], \tag{3.2}$$

where $F_i(\omega_k)$ is the measurement modeling function, described in (2.22), for the i^{th} camera. Moreover, the measurement noise covariance is altered as follows,

$$\Gamma_k = diag\left\{\Gamma_{1,k} \quad . \quad . \quad \Gamma_{l,k}\right\}, \tag{3.3}$$

where $diag\{\cdot\}$ is the diagonal matrix with the elements of the contained matrices as diagonal elements, and $\Gamma_{i,k}$ is the measurement noise covariance matrix of the i^{th} camera.

In addition, a selective fusion scheme is engaged to enhance the accuracy and robustness of the system towards feature's impairments such as occlusion, illumination and point of view changes. A quality measure is exploited to evaluate the reliability of the extracted features [3.45]. If the quality of an extracted feature drops below a threshold value (e.g., due to partial occlusion), the feature will be eliminated from the list of features employed in fusion algorithm. To serve this purpose, a weighting matrix is introduced,

$$W_k = diag\left\{w_1 \quad . \quad . \quad w_{n\times l}\right\}, \tag{3.4}$$

$$w_i = \begin{cases} [0 \quad 0] & g_i < \gamma \\ [1 \quad 1] & g_i \geq \gamma \end{cases} , \tag{3.5}$$

where W_k is the weighting matrix, w_i is the weight associated to the i^{th} feature point, g_i is the quality measure of the i^{th} feature point, and γ is the predefined quality threshold. The gain matrix in (2.28) and (2.45) are then modified to exclude the undesirable features from the pose estimation method, i.e.,

$$K_k = C_{k|k-1} F_k^T \left(\Gamma_k + F_k C_{k|k-1} F_k^T \right)^{-1} W_k , \tag{3.6}$$

$$K_k = C_{x,y,k} C_{y,k}{}^{-1} W_k . \tag{3.7}$$

3.3.2 Centralized VVS Fusion

In this method, VVS is selected as the central pose estimation unit, where the virtual eye-in-hand cameras move with a virtual manipulator, while the virtual eye-to-hand cameras observe the motion of the virtual auxiliary object attached to the virtual manipulator. The velocity of the virtual manipulator is calculated at each time step based on equivalent velocity of all virtual cameras. For this purpose, all features are gathered in one vector and use the IBVS control law to calculate overall velocity of the virtual end-effector through centralized fusion. The vector of all features is related to the velocity of the virtual end-effector as follows,

$$\begin{bmatrix} \dot{s}_1 & . & . & \dot{s}_l \end{bmatrix}^T = \begin{bmatrix} J_{s,1} \Omega_e^{c_1} & . & . & J_{s,l} \Omega_e^{c_l} \end{bmatrix}^T V_e , \tag{3.8}$$

which yields,

$$V_e = -\lambda \begin{bmatrix} J_{s,1} \Omega_e^{c_1} & . & . & J_{s,l} \Omega_e^{c_l} \end{bmatrix}^\dagger \begin{bmatrix} \Delta_{s_1} & . & . & \Delta_{s_n} \end{bmatrix}^T . \tag{3.9}$$

Here $V_e = \begin{bmatrix} \dot{t}_e \\ \varpi_e \end{bmatrix}$ is the overall velocity of the virtual end-effector in its own coordinate frame and $(\cdot)^\dagger$ is the pseudo-inverse function. Once the overall velocity is calculated, the pose is updated as before through (2.93) and (2.94). As was proven earlier, Gauss-Newton method is very similar to

VVS method. As a result, their fusion techniques are also similar and therefore are not discussed here. It should be mentioned that the size of the overall Jacobian matrix in (3.9) increases with the number of cameras, which in turn can prolong the inversion calculation. Decentralized VVS fusion is a way to alleviate this difficulty, which is explained in the sequel.

3.4 Decentralized Sensor Fusion

In this section, decentralized fusion schemes are introduced. In these schemes, data from EIH and ETH cameras is exploited to estimate the pose for each camera separately. The estimated poses are then fused based on their pose error covariance. From this point of view, using EKF for pose estimation is advantageous, since it provides the estimated pose along with its error covariance. To accommodate visual servoing requirements, the relative pose of the object with respect to the robot's end-effector is estimated. This choice of pose allows some of the needed calibrations in previous works (i.e., robot and ETH camera external calibration) to be relaxed. The object's pose (and its error covariance) can directly be obtained for EIH cameras, whereas in case of the ETH cameras the relative pose of the object should be combined with the relative pose of the end-effector with respect to the camera.

3.4.1 Decentralized Kalman Filter-Based Fusion

A decentralized sensor fusion scheme is selected for accurate and robust pose estimation, while preserving the computational cost. The pose of the target object with respect to robot's end-effector is estimated separately using ETH and EIH camera information. These estimations are next checked for fault detection. If any of the EKF methods are detected as faulty, the estimation involving the erroneous pose will be eliminated from fusion step. A fault detection based on EKF innovation term is used for this purpose [3.46]. A decentralized fusion layer is exploited to fuse the pose estimation into a more accurate output. Figure 3.1 demonstrates the block diagram of this fusion scheme. The fusion is based on maximum likelihood criterion developed by [3.13]. The fused estimation is calculated as follows,

$$\hat{\omega}_{k|k} = \sum_{i=1}^{l} W_i \hat{\omega}_{k|k,i} \ , \tag{3.10}$$

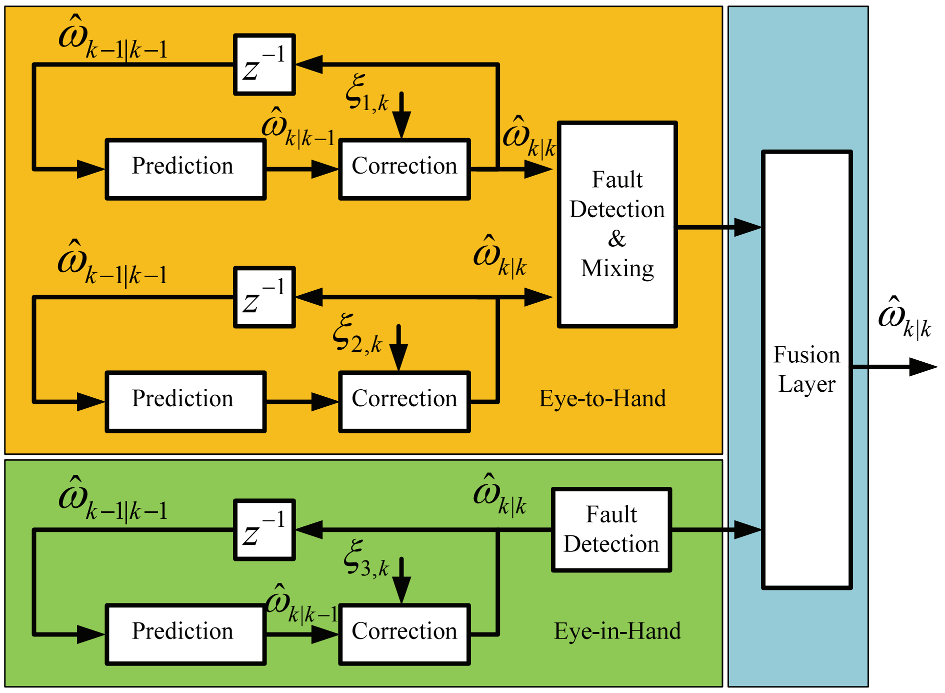


Figure 3.1 Decentralized sensor fusion block diagram.

where l is the number of local estimators, $\hat{\omega}_{k|k}$ is the fused estimation, $\hat{\omega}_{k|k,i}$ is i^{th} local estimation and W_i is its fusion weighting matrix and is defined as,

$$W_i = C_i^{-1}\left(\sum_{j=1}^{l} C_j^{-1}\right)^{-1}, \tag{3.11}$$

where C_i is the error covariance of i^{th} estimation, and the superscript "-1" indicates the inverse of the matrix. In fusion algorithm that exploits (3.10) and (3.11), the two fusing estimations are assumed to be independent. The error covariance of the final fused pose is equal to,

$$C_f = \left(\sum_{j=1}^{l} C_j^{-1}\right)^{-1}, \tag{3.12}$$

which is less than any of the error covariance matrices [3.13].

The error covariance of the estimation error using EIH camera data is directly obtained from EKF; however the error covariance of the pose estimated using ETH camera information is not available and should be estimated separately. The estimation of error covariance using the EKF pose estimators is discussed next.

3.4.2 Error Covariance Computation

As it can be seen from (2.3), the estimated pose through ETH requires three poses to be combined. The error covariance of the estimated pose is impacted by each of the comprising pose estimations and their error. This section illustrates how to estimate the error covariance of a pose, composed of two dependent pose estimations.

The error covariance is discussed in two separated parts, the angular error covariance and the translational error covariance. Each of these covariance matrices are estimated for a combined transformation matrix,

$$H_1^3 = H_2^3 H_1^2 , \tag{3.13}$$

where H_1^2 and H_2^3 represent two known pose estimations with known error covariance. Finally the error covariance of an inverse transformation is estimated, since (2.3) necessitates this computation. It should be noted that the coordinate frames 1-3, used in this section, are symbolic and can be replaced with any coordinate frames in this work. The angular error of the pose H_1^3 is defined as,

$$\varepsilon_{\phi_1^3} = \phi_1^3 - \hat{\phi}_1^3 , \tag{3.14}$$

where ε represents the estimation error, ϕ_1^3 and $\hat{\phi}_1^3$ are true and estimated Euler angles related to R_1^3. This error is related to the rotation matrix error,

$$\varepsilon_{\pi_1^3} \approx J_1^3 \Delta\phi_1^3 , \tag{3.15}$$

where π_1^3 is the vector form of the rotation matrix, R_1^3,

$$\pi_1^3 = \Pi(R_1^3) . \tag{3.16}$$

Here operator $\Pi(\cdot)$ converts matrices to vectors. The Jacobian matrix J_1^3 relates the error of the rotation matrix ($\varepsilon_{r_1^3}$) and Euler angles' error ($\Delta\phi_1^3$) and is calculated as follows,

$$J_1^3 = \left[\Pi(\frac{\partial R_1^3}{\partial \varphi_1^3}) \quad \Pi(\frac{\partial R_1^3}{\partial \theta_1^3}) \quad \Pi(\frac{\partial R_1^3}{\partial \psi_1^3}) \right]. \tag{3.17}$$

The rotation error is calculable from the two estimations' errors,

$$\varepsilon_{R_1^3} \approx \varepsilon_{R_2^3} R_1^2 + R_2^3 \varepsilon_{R_1^2} . \tag{3.18}$$

The error of rotations is then related using (3.15),

$$\varepsilon_{\pi_1^3} = K_1 \varepsilon_{\varphi_2^3} + K_2 \varepsilon_{\varphi_1^2} , \tag{3.19}$$

where,

$$K_1 = \left[\Pi(\frac{\partial R_2^3}{\partial \varphi_2^3} R_1^2) \quad \Pi(\frac{\partial R_2^3}{\partial \theta_2^3} R_1^2) \quad \Pi(\frac{\partial R_2^3}{\partial \psi_2^3} R_1^2) \right], \tag{3.20}$$

$$K_2 = \left[\Pi(R_2^3 \frac{\partial R_1^2}{\partial \varphi_1^2}) \quad \Pi(R_2^3 \frac{\partial R_1^2}{\partial \theta_1^2}) \quad \Pi(R_2^3 \frac{\partial R_1^2}{\partial \psi_1^2}) \right]. \tag{3.21}$$

The angular error covariance is calculated using (3.15) and (3.19),

$$E\left\{\varepsilon_{\phi_1^3} \varepsilon_{\phi_1^{3T}}\right\} = J_1^{3\dagger} K_1 E\left\{\varepsilon_{\varphi_2^3} \varepsilon_{\varphi_2^3}{}^T\right\} K_1{}^T J_1^{3\dagger T} + J_1^{3\dagger} K_2 E\left\{\varepsilon_{\varphi_1^2} \varepsilon_{\varphi_1^2}{}^T\right\} K_2{}^T J_1^{3\dagger T} . \tag{3.22}$$

The translational error is defined as follows,

$$\varepsilon_{t_1^3} = t_1^3 - \hat{t}_1^3 , \tag{3.23}$$

where t_1^3 and $\hat{t}_1^3$ are the true and estimated translation. This error is calculated as follows,

$$\varepsilon_{t_1^3} = \varepsilon_{R_2^3} t_1^2 + R_2^3 \varepsilon_{t_1^2} + \varepsilon_{t_2^3} . \tag{3.24}$$

The equation can be rewritten as follows,

$$\varepsilon_{t_1^3} = K_3 \varepsilon_{\phi_2^3} + R_2^3 \varepsilon_{t_1^2} + \varepsilon_{t_2^3}, \tag{3.25}$$

where,

$$K_3 = \begin{bmatrix} \frac{\partial R_2^3}{\partial \varphi_2^3} t_1^2 & \frac{\partial R_2^3}{\partial \theta_2^3} t_1^2 & \frac{\partial R_2^3}{\partial \psi_2^3} t_1^2 \end{bmatrix}. \tag{3.26}$$

Therefore the covariance of the translational error is computed as:

$$\begin{aligned} E\left\{\varepsilon_{t_1^3} \varepsilon_{t_1^3}{}^T\right\} &= K_3 E\left\{\varepsilon_{\phi_2^3} \varepsilon_{\phi_2^3}{}^T\right\} K_3{}^T + R_2^3 E\left\{\varepsilon_{t_1^2} \varepsilon_{t_1^2}{}^T\right\} R_2^{3T} + E\left\{\varepsilon_{t_2^3} \varepsilon_{t_2^3}{}^T\right\} \\ &+ K_3 E\left\{\varepsilon_{\phi_2^3} \varepsilon_{t_2^3}{}^T\right\} + E\left\{\varepsilon_{t_2^3} \varepsilon_{\phi_2^3}{}^T\right\} K_3{}^T. \end{aligned} \tag{3.27}$$

Computation of the desire pose using ETH camera involves the inverse relative pose of the end-effector with respect to the ETH camera. Therefore, the error covariance of this inverse pose should be calculated. The angular error of the inverse pose $H_2^1 = \left(H_1^2\right)^{-1}$ estimation is defined as follows,

$$J_2'^1 \varepsilon_{\phi_2^1} = J_1^2 \varepsilon_{\phi_1^2}, \tag{3.28}$$

where,

$$J_2'^1 = \begin{bmatrix} \Pi(\frac{\partial R_2^{1T}}{\partial \varphi_2^1}) & \Pi(\frac{\partial R_2^{1T}}{\partial \theta_2^1}) & \Pi(\frac{\partial R_2^{1T}}{\partial \psi_2^1}) \end{bmatrix}. \tag{3.29}$$

Hence, the angular error covariance of the inverse pose is calculated as follows,

$$E\left\{\varepsilon_{\phi_2^1} \varepsilon_{\phi_2^1}{}^T\right\} = J_2'^{1\dagger} J_1^2 E\left\{\varepsilon_{\phi_1^2} \varepsilon_{\phi_1^2}{}^T\right\} J_1^{2T} J_2'^{1\dagger T}. \tag{3.30}$$

The translational error of the inverse pose is defined as follows,

$$\varepsilon_{t_2^1} = -\varepsilon_{R_1^2}{}^T t_1^2 - R_1^{2T} \varepsilon_{t_1^2} = -K_4 \varepsilon_{\phi_1^2} - R_1^{2T} \varepsilon_{t_1^2} \quad , \tag{3.31}$$

where,

$$K_4 = \left[\frac{\partial R_1^{2T}}{\partial \varphi_1^2} t_1^2 \quad \frac{\partial R_1^{2T}}{\partial \theta_1^2} t_1^2 \quad \frac{\partial R_1^{2T}}{\partial \psi_1^2} t_1^2 \right]. \tag{3.32}$$

So the translational error covariance is computed as follows,

$$\begin{aligned} &E\left\{\varepsilon_{t_2^1}\varepsilon_{t_2^1}\right\} = K_4 E\left\{\varepsilon_{\phi_1^2}\varepsilon_{\phi_1^2}{}^T\right\} K_4{}^T + R_1^{2T} E\left\{\varepsilon_{t_1^2}\varepsilon_{t_1^2}{}^T\right\} R_1^2 + K_4 E\left\{\varepsilon_{\phi_1^2}\varepsilon_{t_1^2}{}^T\right\} R_1^2 \\ &+R_1^{2T} E\left\{\varepsilon_{t_1^2}\varepsilon_{\phi_1^2}{}^T\right\} K_4{}^T \end{aligned} . \tag{3.33}$$

3.4.3 Decentralized VVS Fusion

Another method for decentralized fusion is through VVS. For this matter, the equivalent end-effector velocities are calculated separately for each camera using (2.86) and (2.88),

$$V_{e_i} = -\lambda \left(J_{s,i} \Omega_e^{c_i} \right)^\dagger \Delta s_i , \tag{3.34}$$

and use a decentralized fusion technique to reconstruct the overall velocity of the virtual camera. Here λ is a gain factor. For this purpose, a weighted averaging operator is employed,

$$V_e = \sum_{i=1}^{m} W_i V_{e_i} , \tag{3.35}$$

where W_i is the associated weight for the i^{th} camera. The weights may be adjusted to give more weights to more accurate sensors (e.g., eye-in-hand cameras), or to isolate possible faulty sensors (e.g., those that have encountered occlusion), similar to Kalman-based methods. If no prior knowledge about the sensors is available, the weight might be selected to be equal. The accuracy of decentralized fusion techniques are generally lower than centralized ones, however they are faster and they also ease the sensor fault detection and separation [3.14], which makes this technique appealing.

3.5 Pre-Processing Fusion

A common way for sensor fusion is to centrally fuse the data by concatenating all data into a single vector and feed the new data into the processing (e.g., pose estimation) unit. Since the characteristics of the new data have changed, the processing unit should be modified to deal with the newly formed data. This modification may be demanding and different for every individual processing method. In addition, the size of the data increases with the number of sensors, which usually result in exponential rise in computational costs. Another solution is to fuse all the data into a single element and pass the fused data to the processing unit. This is especially beneficial since the size of the data is preserved and the processing unit is needless of any modifications; however this method is usually practical for data of the same type. A brief discussion on this matter can be found in [3.15].

In this section, a novel framework for pre-processing sensor fusion is proposed. For this matter, the observations from different cameras are transformed into equivalent data, seen by a virtual camera which is located at the desired place (i.e., robot end-effector). A weighted averaging operator is used as the fusion unit. The fused data is then processed by a pose estimation algorithm. Since an estimation of the object depth is required for data transfer, an iterative algorithm (i.e., Dementhon algorithm proposed in [2.16]) is considered for pose estimation. The main advantage of the proposed method is its applicability to a large group of pose estimation methods, since it is almost independent of the estimation algorithm.

Let $d_i = \left[p_{O,1}^{C_i} \quad . \quad . \quad p_{O,l}^{C_i} \right]$ be the image information from camera i, and $\omega_{o_T,i}^{e} = f_i'(d_i)$ be the desired pose calculated based on camera i information. Here $f_i'(\cdot)$ is the pose estimating function. Then the goal is to find

$$\omega_{o_T}^{e} = f'\left(d_1,..,d_n\right), \tag{3.36}$$

where $f'\left(\cdot\right)$ is the fusing/estimating function that maps the information from all cameras to the desired pose. Fusion of multiple data sources might be done at different stages of a system. In this section, the focus is on the fusion of the data before getting processed for pose estimation (i.e. pre-processing fusion). For this purpose, a virtual camera with same frame coordinates as robot end-effector is assumed. The fusion of mapped data (to this virtual camera) is considered for pose estimation,

$$f\left(d_1,..,d_n\right)=f_h\left(d_h\right), \tag{3.37}$$

where,

$$d_h = h\left(d_1,..,d_n\right). \tag{3.38}$$

One should realize that fused data, d_h has the same form as any d_i, therefore it neither increases the computation, nor changes the structure of the pose estimator. To form the fused data, each set of information from a camera is first mapped to equivalent data from virtual camera *h*, and is fused next by means of a weighted averaging operator,

$$d_h = \sum_{j=1}^{n} W_j \tilde{d}_j , \tag{3.39}$$

where W_j is the weight matrix associated to equivalent data from camera *j*, denoted by $\tilde{d}_j$. If the error covariance of the transferred data from each camera, denoted by C_i, is known, the weights are calculated as follows,

$$W_i = C_i^{-1}\left(\sum_{j=1}^{n} C_j^{-1}\right)^{-1}. \tag{3.40}$$

It can be shown that the error covariance of such fused data is equal to,

$$C_h = \left(\sum_{j=1}^{n} C_j^{-1}\right)^{-1}, \tag{3.41}$$

which is smaller than any other error covariance,

$$C_h \le C_i \quad \forall i \in \left[1,n\right]. \tag{3.42}$$

The proof of this fact is available in [3.13]. If C_i is not available, the weights will be assumed equal. It also worth mentioning that the weights are normalized,

$$\sum_{j=1}^{n} W_j = 1 . \tag{3.43}$$

The weights might also be used to exclude the corrupted/noisy data from the fusion process.

Now to calculate the equivalent data for each camera, the relation between the presentations of a single feature point in two different frames, namely the camera i and the virtual camera, is discussed. From (2.4) one can write,

$$P^e_{o,j} = R^e_{c_i} P^{c_i}_{o,j} + t^e_{c_i} . \tag{3.44}$$

Assuming $\tilde{d}_i = \begin{bmatrix} \tilde{p}^e_{o,i,1} & . & . & \tilde{p}^e_{o,i,l} \end{bmatrix}$, the equivalent data for each camera is calculated as follows using (2.4),

$$\begin{bmatrix} \tilde{p}^e_{o,i,j} & 1 \end{bmatrix} = \frac{z^{c_i}_j}{z^e_j} R^e_{c_i} \begin{bmatrix} p^{c_i}_{o,j} & 1 \end{bmatrix} + \frac{t^e_{c_i}}{z^e_j} , \tag{3.45}$$

where $R^e_{c_i}$ and $t^e_{c_i}$ are assumed to be known through calibration, and $p^{c_i}_{o,j}$ is available through feature detection on image from camera j; however $z^{c_i}_j$ and z^e_j are not available and needed to be estimated. Since (3.45) provides the system with 3 linear equations for 2 unknown parameters of the equivalent data, one of the unknown parameters (i.e., $z^{c_i}_j$ or z^e_j) can be calculated from the other. The depth of features in each camera frame is generally unknown, but the depth of features in end-effector frame can be approximated by the previous calculated pose.

$$\hat{z}^e_j = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \left(\hat{R}^e_o P^o_{o,j} + \hat{t}^e_o \right) , \tag{3.46}$$

where $\hat{R}^e_o$ and $\hat{t}^e_o$ are the approximations of the current pose, obtained from the previous estimated pose. This approximation is valid in visual servoing, since the consequent poses are close due to camera's high frame rate. Having z^e_j estimated, (3.45) yields,

$$z_j^{ck} = \frac{\hat{z}_j^{ci} - \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} t_{ck}^{ci}}{\begin{bmatrix} 0 & 0 & 1 \end{bmatrix} R_{ck}^{ci} p_j^{ck}} . \tag{3.47}$$

The approximation used in (3.47) can be improved if an iterative technique is considered for pose estimation. Then, the pose obtained at each iteration is used as a pose approximation in next iteration calculation of (3.47).

3.6 Simulation and Experimental Results

In order to verify the effectiveness of the proposed sensor fusion methods for pose estimation, numerous simulations and experiments are conducted. The accuracy of each method is the main concern of the tests. The simulations are performed in MATLAB® 2011b from Mathworks (Natick, MA, USA). The system configuration for the simulations and experiments are similar to the system which was explained in Sec. 2.2.

3.6.1 Centralized Fusion

First, the performance of the centralized IAEKF and IAUKF fusion is put into test and compared with that of the EKF-based fusion, UKF-based fusion and monocular IAEKF pose estimators. For this matter, an experimental setup composed of a 5-DOF CRS robot from CRSRobotics (Burlington, ON, Canada), an EIH Firefly Point Grey camera (Richmond, BC, Canada), a similar camera as an ETH camera, an accurate optical tracker from NDI (Waterloo, ON, Canada), the target and auxiliary objects are exploited. The image size of each camera is 640 by 480 pixels. The cameras work at 60 frames per second and are calibrated prior to the experiments. The location of the EIH camera coincides with that of the robot's end-effector. The target object and robot's end-effector object each has four circular features (black dots) with known distance between the features. The exploited optical tracker is very accurate and capable of locating a set of infrared markers with accuracy of 0.1 mm with a frequency of up to 4500 targets per second. Since the accuracy of this measurement device is higher than the camera, its measurements have been selected as the ground truth. The infrared markers are installed on the robot and the target object and their positions are known with respect to the target object and robot's end-effector. The robot is a 5 DOF anthropomorphic arm. Each joint of the robot can move as fast as 210 degrees per second, with a maximum acceleration of 498 degrees per second squared. The experimental setup is shown in Figure 3.2. The setup and the frame coordinates are the same as the setup shown in Figure 2.1.

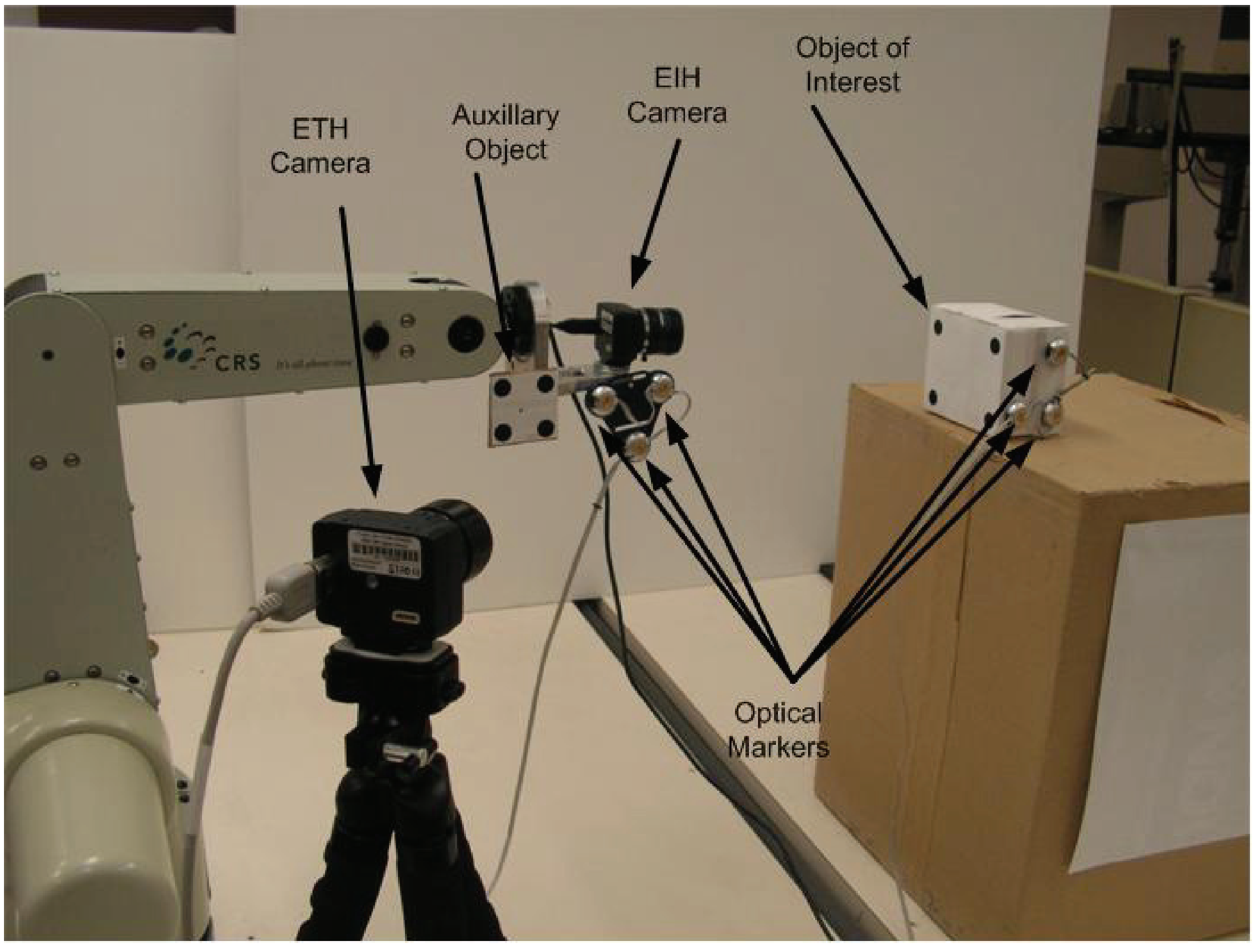


Figure 3.2 Experimental setup used for sensor fusion.

In the first experiment, the similarity of the proposed iterative methods is put into test. The performance of IEKF is compared with those of the EKF that have zero measurement noise and the Gauss-Newton algorithm. The chosen number of iterations suffices for (2.69) to hold. The same number of iterations is considered in the inverse-Jacobian algorithm. Figure 3.3 shows the error of these methods and Table 3.1 summarizes the results. The first column of Figure 3.3 entails the translational errors, while the second column demonstrates the errors of orientation (represented in Euler angles). Apparently all of the methods result in very similar estimation error, as was predicted. In fact, the results from IEKF and Gauss-Newton are almost identical, while EKF with zero measurement noise is slightly different (since it represents the Gauss-Newton method with no iterations). The results suggest the applicability of alternative methods in case of highly dynamic systems.

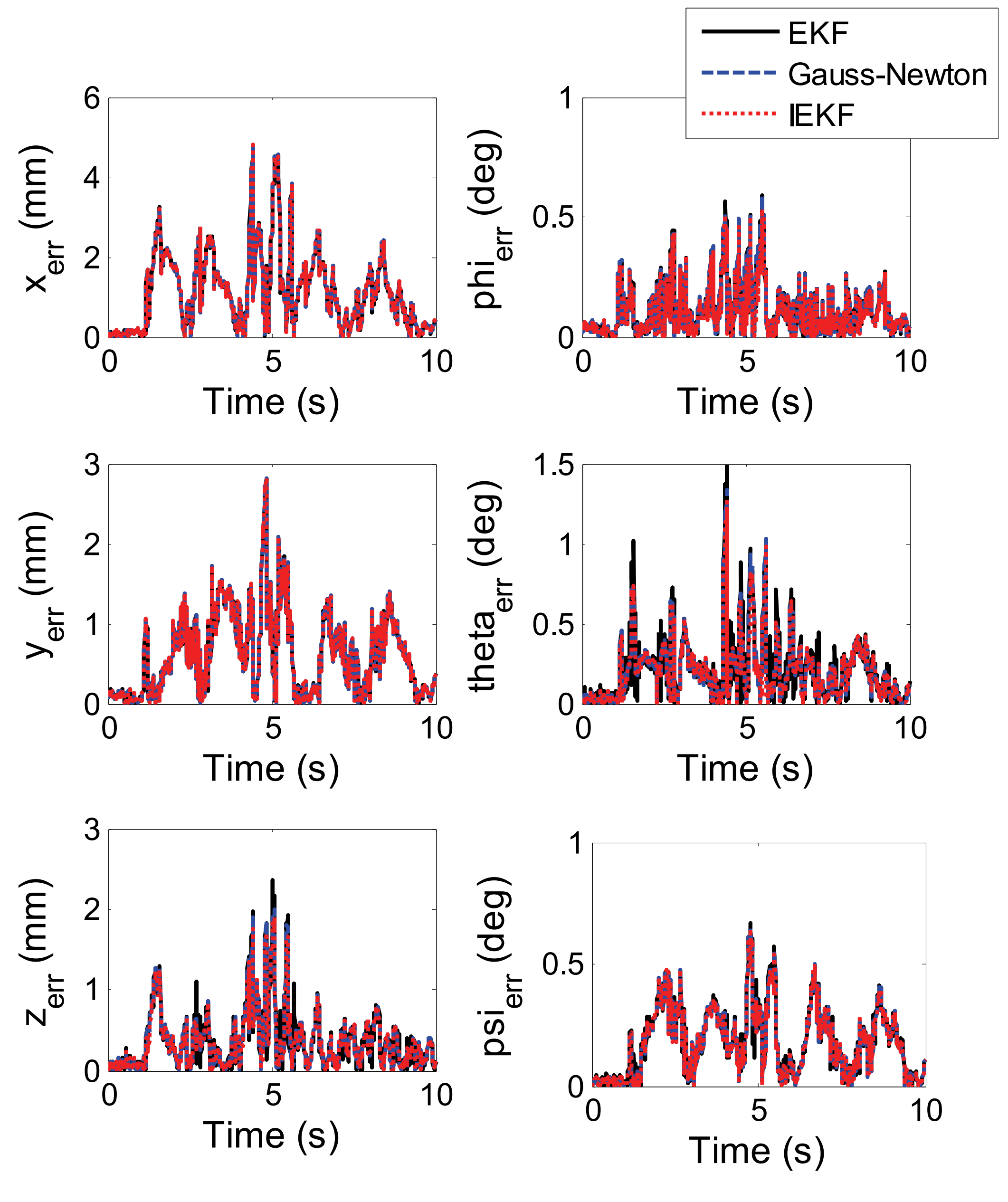


Figure 3.3 First experiment: Pose estimation error of EKF with zero measurement noise, Gauss-Newton, and IEKF.

		x (mm)	y (mm)	z (mm)	ϕ (deg)	θ (deg)	Ψ (deg)
Max Error	EKF	4.8	2.8	2	0.6	1.4	0.7
	Gauss-Newton	4.8	2.8	2	0.6	1.3	0.7
	IEKF	4.8	2.8	2	0.6	1.3	0.7
Mean Error	EKF	1.1	0.6	0.4	0.1	0.2	0.2
	Gauss-Newton	1.1	0.6	0.4	0.1	0.2	0.2
	IEKF	1.1	0.6	0.4	0.1	0.2	0.2
Std	EKF	1	0.5	0.4	0.1	0.2	0.2
	Gauss-Newton	1	0.5	0.4	0.1	0.2	0.1
	IEKF	1	0.5	0.4	0.1	0.2	0.1

Table 3.1 First experiment: Iterative schemes in comparison with zero noise EKF.

In the second experiment, the accuracy of the proposed IAEKF fusion technique is measured and compared to similar methods (namely EKF fusion and monocular IAEKF) and the reference pose. For this purpose, the noise covariance matrices are tuned initially based on offline measurements. Figure 3.4 demonstrates the output of pose estimation methods in comparison to ground truth. Figure 3.5 magnifies the error of each estimation technique. The statistics of this error is summarized in Table 3.2. Since the trajectories of the pose are dynamic, EKF fusion cannot estimate the pose accurately, even when the filter is tuned initially. On the contrary, IAEKF methods compensate for changes in the system noise variations. The mean of estimation error of the proposed fusion method is generally lower than the other two methods. This is because IAEKF fusion tunes the noise parameters automatically, is more robust to changes in system dynamics, and also benefits the data from different sources.

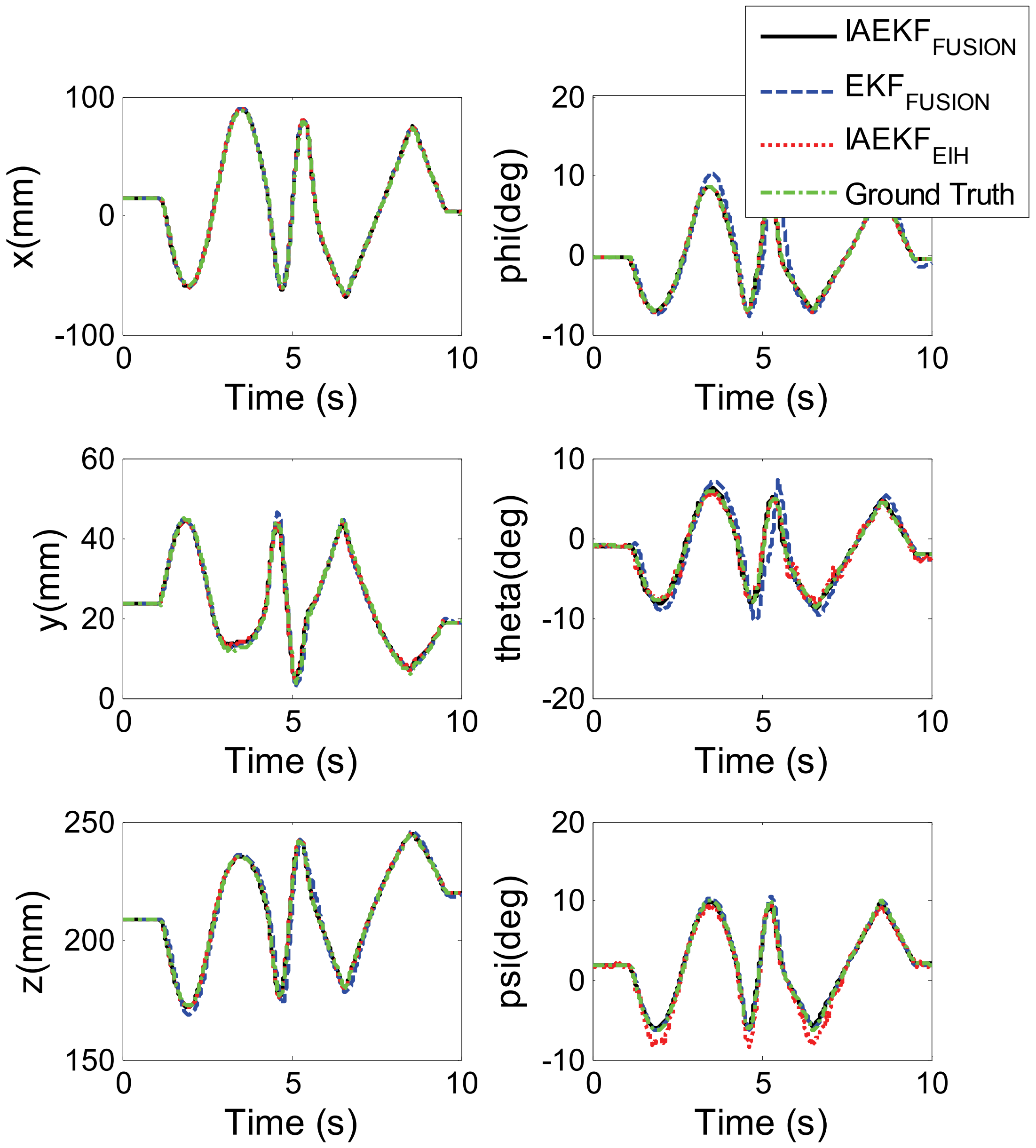


Figure 3.4 Second experiment: Pose estimation output of IAEKF fusion in comparison with monocular IAEKF (IAEKF$_{EIH}$), EKF fusion and ground truth, in case of tuned filters.

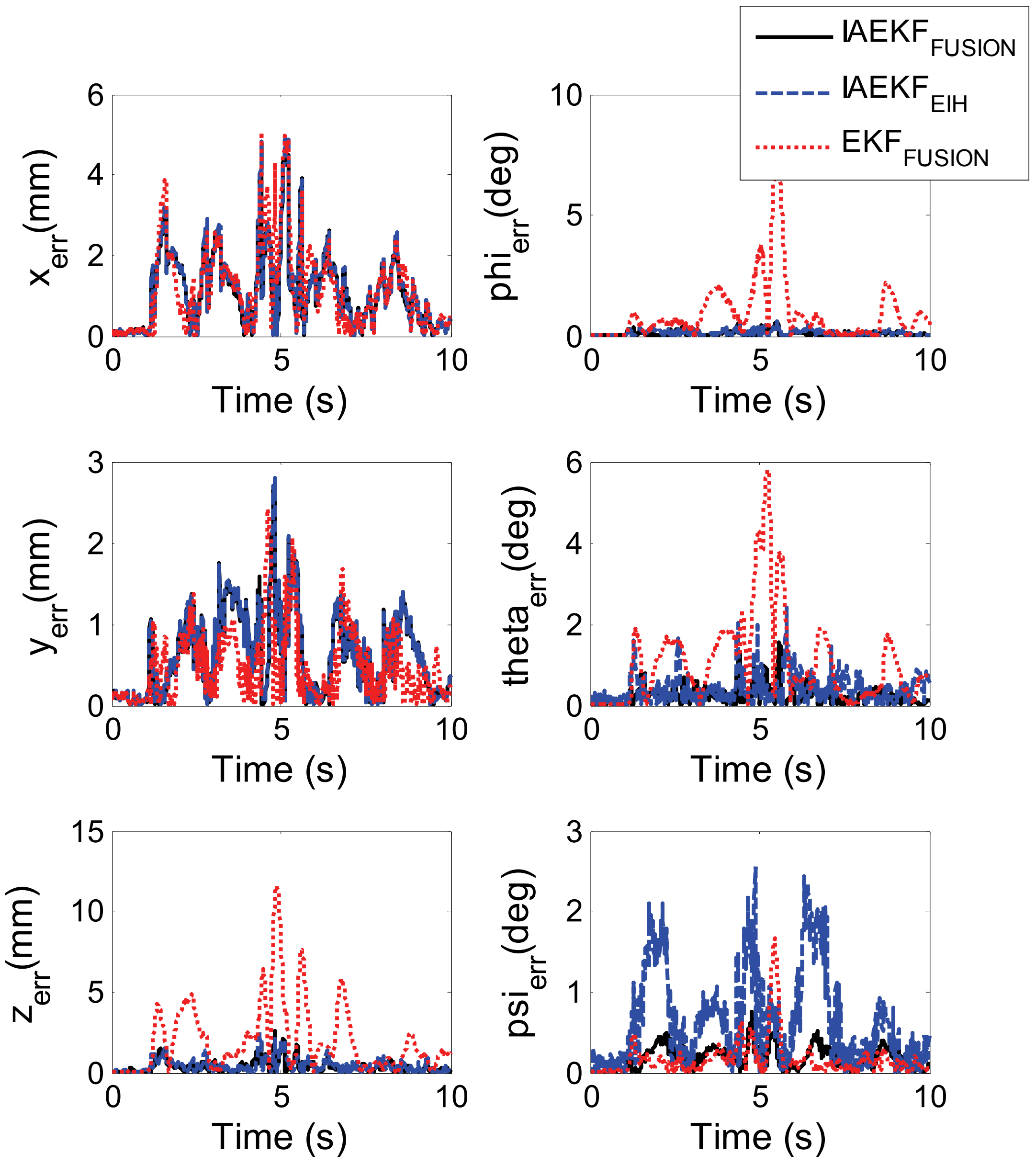


Figure 3.5 Second experiment: Pose estimation error of IAEKF fusion in comparison with monocular IAEKF (IAEKF$_{\mathrm{EIH}}$), EKF fusion, in case of finely tuned filters.

		x (mm)	y (mm)	z (mm)	ϕ (deg)	θ (deg)	Ψ (deg)
Max Error	IAEKF Fusion	4.8	2.8	2.6	0.6	1.6	0.8
	IAEKF EIH	4.9	2.8	2.5	0.6	2.4	2.5
	EKF Fusion	5.1	2.4	11.6	7.7	5.8	1.7
Mean Error	IAEKF Fusion	1.1	0.6	0.4	0.1	0.2	0.2
	IAEKF EIH	1.2	0.6	0.4	0.1	0.4	0.6
	EKF Fusion	1.2	0.5	2	0.9	1.1	0.2
Std	IAEKF Fusion	0.9	0.5	0.4	0.1	0.2	0.1
	IAEKF EIH	1	0.5	0.4	0.1	0.3	0.6
	EKF Fusion	1	0.4	2.2	1.3	1.1	0.2

Table 3.2 Second experiment: IAEKF fusion in comparison with EKF fusion and monocular IAEKF, in case of fine tuning.

In the third experiment, the accuracy of the IAUKF fusion is tested and compared to that of UKF fusion. Similar to previous experiment, the both filters are initially tuned and initialized properly. The same trajectories are taken to test both methods. The results of this comparison are shown in Figure 3.6 and the result summary is depicted in Table 3.3. As it can be seen, the proposed IAUKF performs better that UKF in terms of accuracy. This fact is also reflected in the mean of error in Table 3.3.

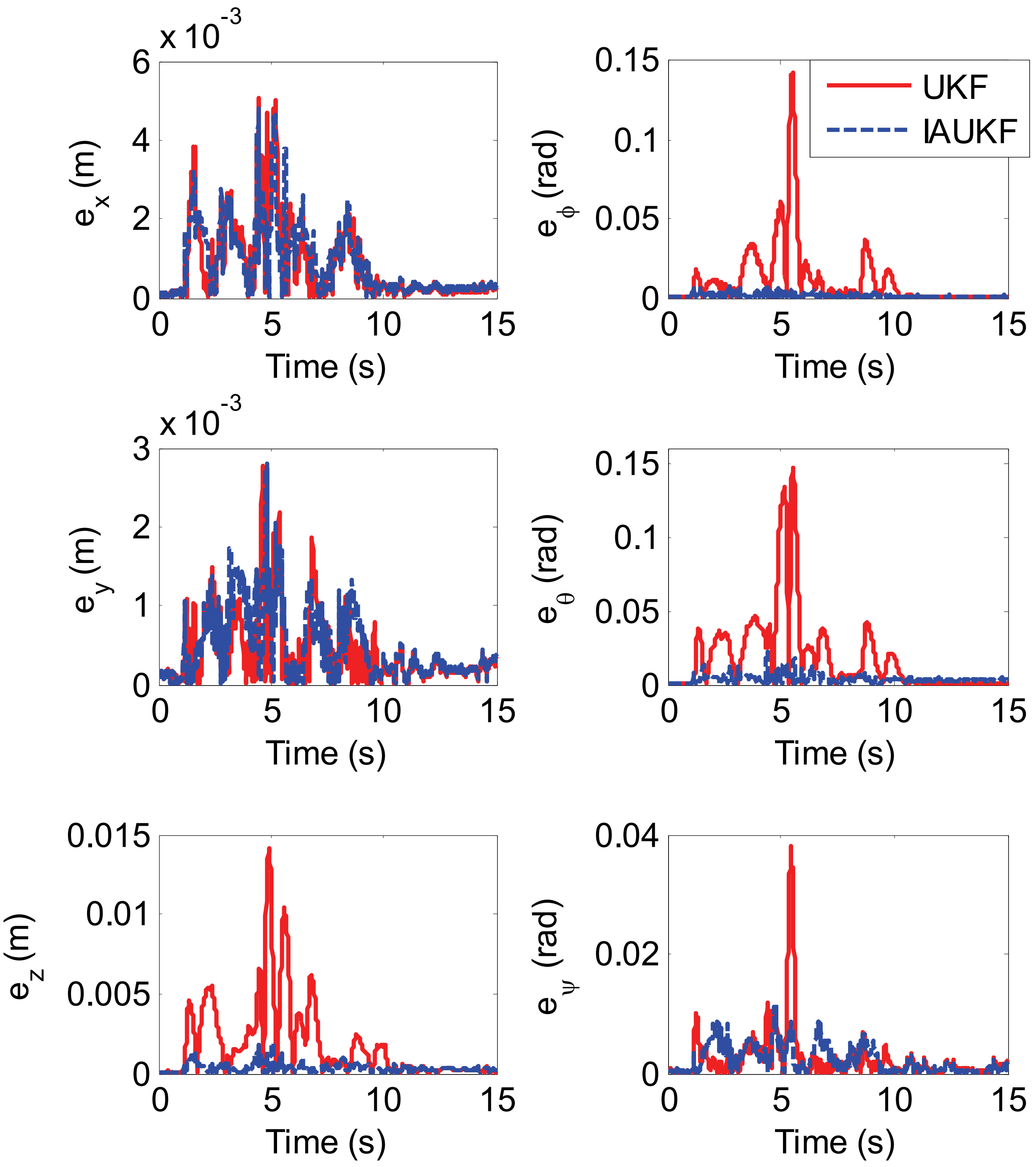


Figure 3.6 Third experiment: Pose estimation error of IAUKF fusion in comparison with UKF fusion, in case of finely tuned filters.

		x (mm)	y (mm)	z (mm)	ϕ (rad)	θ (rad)	Ψ (rad)
Max Error	IAUKF Fusion	4.8	2.8	1.9	0.008	0.022	0.011
	UKF Fusion	5.0	2.8	14.2	0.142	0.146	0.038
Mean Error	IAUKF Fusion	0.8	0.5	0.3	0.001	0.004	0.002
	UKF Fusion	0.9	0.4	1.8	0.011	0.018	0.003
Std	IAUKF Fusion	0.9	0.4	0.3	0.001	0.003	0.002
	UKF Fusion	1.0	0.4	2.5	0.020	0.027	0.004

Table 3.3 Third experiment: IAUKF fusion in comparison with UKF fusion, in case of fine tuning.

In the fourth experiment, the process noise covariance of the filters is reduced 100 times to verify the effect of filter tuning in the proposed method and its counterparts. The error of IAEKF fusion pose estimation is compared with EKF fusion and IAEKF based on EIH camera in Figure 3.7. The statistics of these errors are summarized in Table 3.4. As it was expected, the performance of EKF fusion algorithm decreases significantly, while the IAEKF-based methods remain almost without change. This fact can be inferred from large errors of the EKF fusion method. This experiment magnifies the role of filter tuning in Kalman-filter based pose estimation methods. As mentioned before, adaptive and robust schemes limit the undesirable effects of any mistuning.

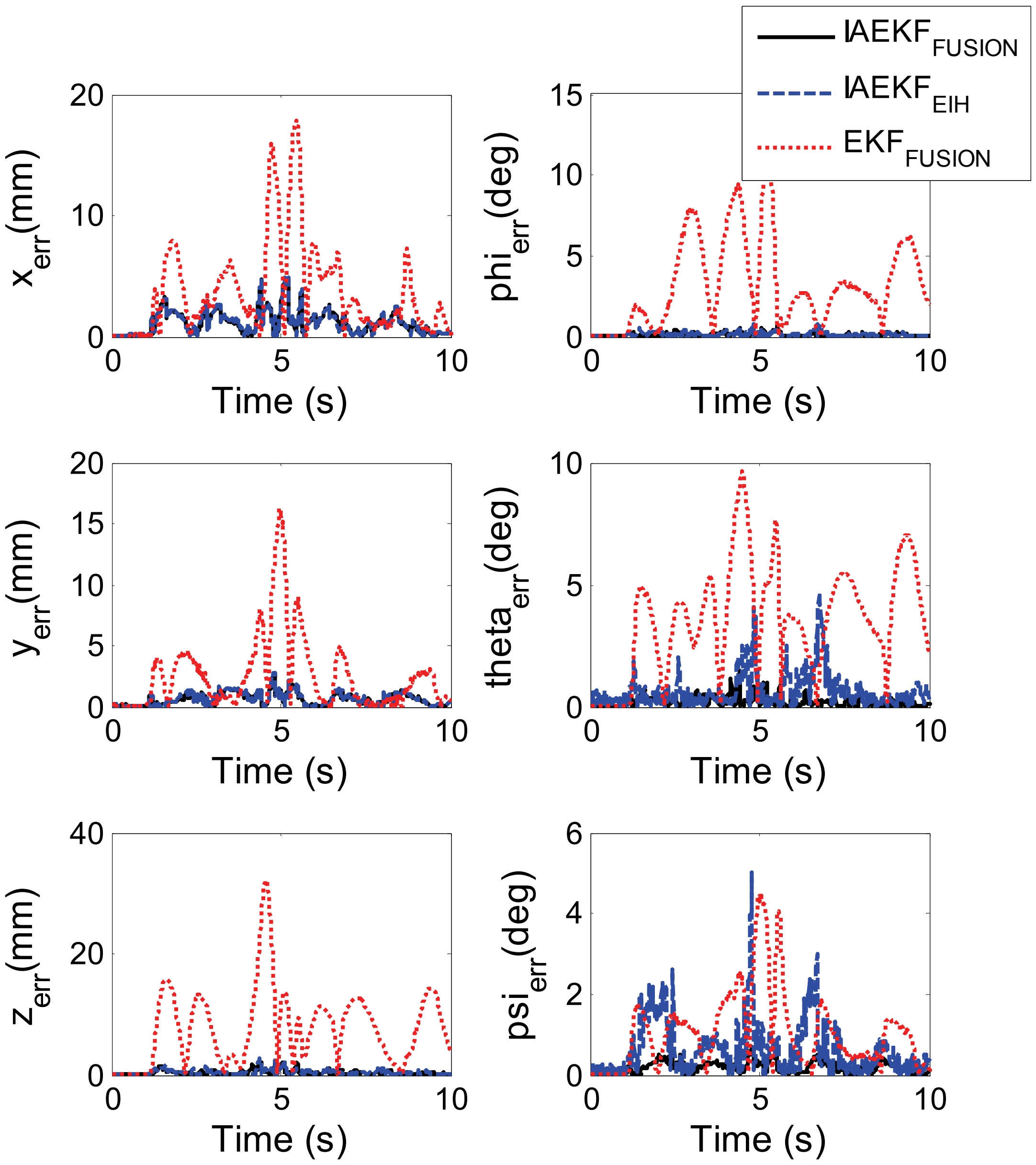


Figure 3.7 Fourth experiment: Pose estimation error of IAEKF fusion in comparison with monocular IAEKF ($IAEKF_{EIH}$), EKF fusion, in case of noise mismatch.

		x (mm)	y (mm)	z (mm)	ϕ (deg)	θ (deg)	Ψ (deg)
Max Error	IAEKF Fusion	4.8	2.7	2.2	0.6	1.5	0.7
	IAEKF EIH	4.9	2.8	2.6	0.8	4.7	5
	EKF Fusion	17.8	16.3	32.2	12.6	9.6	4.4
Mean Error	IAEKF Fusion	1.1	0.6	0.4	0.1	0.2	0.2
	IAEKF EIH	1.1	0.6	0.4	0.1	0.6	0.6
	EKF Fusion	3.7	2.5	7.8	3.2	3.6	1.1
Std	IAEKF Fusion	1	0.5	0.4	0.1	0.2	0.1
	IAEKF EIH	1	0.5	0.4	0.1	0.7	0.7
	EKF Fusion	3.9	3.1	6.4	2.8	2.2	1

Table 3.4 Fourth experiment: IAEKF fusion in comparison with EKF fusion and monocular IAEKF, in case of noise mismatch.

In experiment 5, the sensitivity of the proposed method to initial state adjustment is compared to two previously discussed pose estimation methods. For this matter, the initial position states are misadjusted by 0.2 meters in each direction and the initial orientation states are misadjusted by almost 28 degrees (0.5 rad) for each Euler angle. The results are magnified in Figure 3.8. As it can be seen from the results, the IAEKF fusion algorithm converges towards the true value faster than the two others, though with an overshoot. This property assures the system to be minimally affected by erroneous initial estimations.

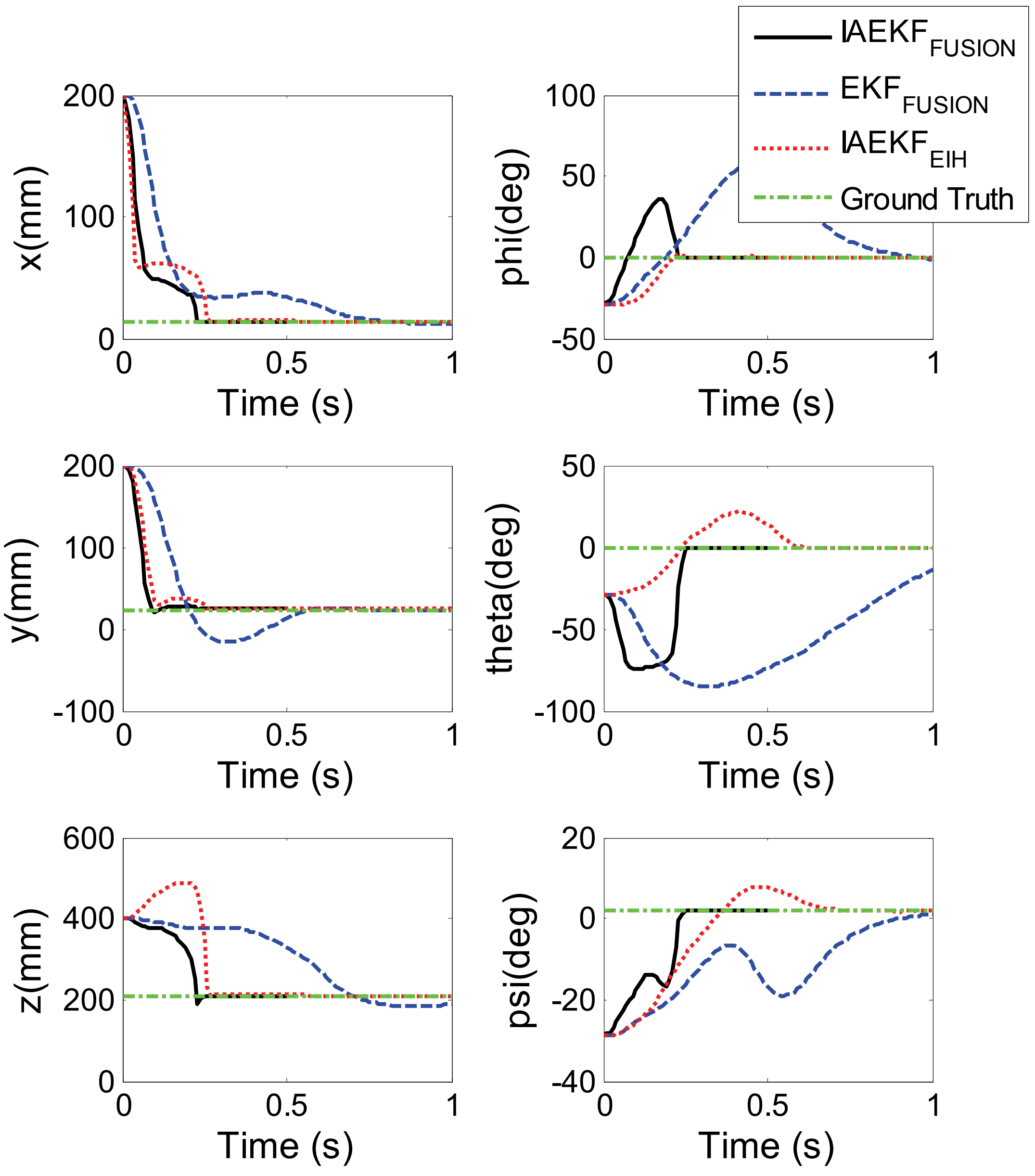


Figure 3.8 Fifth experiment: Pose estimation output of IAEKF fusion in comparison with monocular IAEKF (IAEKF$_{EIH}$), EKF fusion, in case of initialization maladjustment.

In experiment 6, the pose estimation algorithms are tested under a different sampling rate for the same motion as previous experiments to test the effect of the sampling rate change on the estimations. Various factors such as change of the camera might cause variations in the sampling rate of the system. The sampling frequency is selected 5 times lower than original sampling frequency (T_s=50 ms). The proposed IAEKF fusion algorithm is compared in performance to

IAEKF based on EIH camera and EKF fusion algorithm under the new sampling time condition. Figure 3.9 shows the error results for these methods. The performance of the IAEKF fusion is superior to that of the two other methods in this case as the errors of this method is much less than the others. Sampling time changes particularly affect the EKF method, which is sensitive to changes of system parameters.

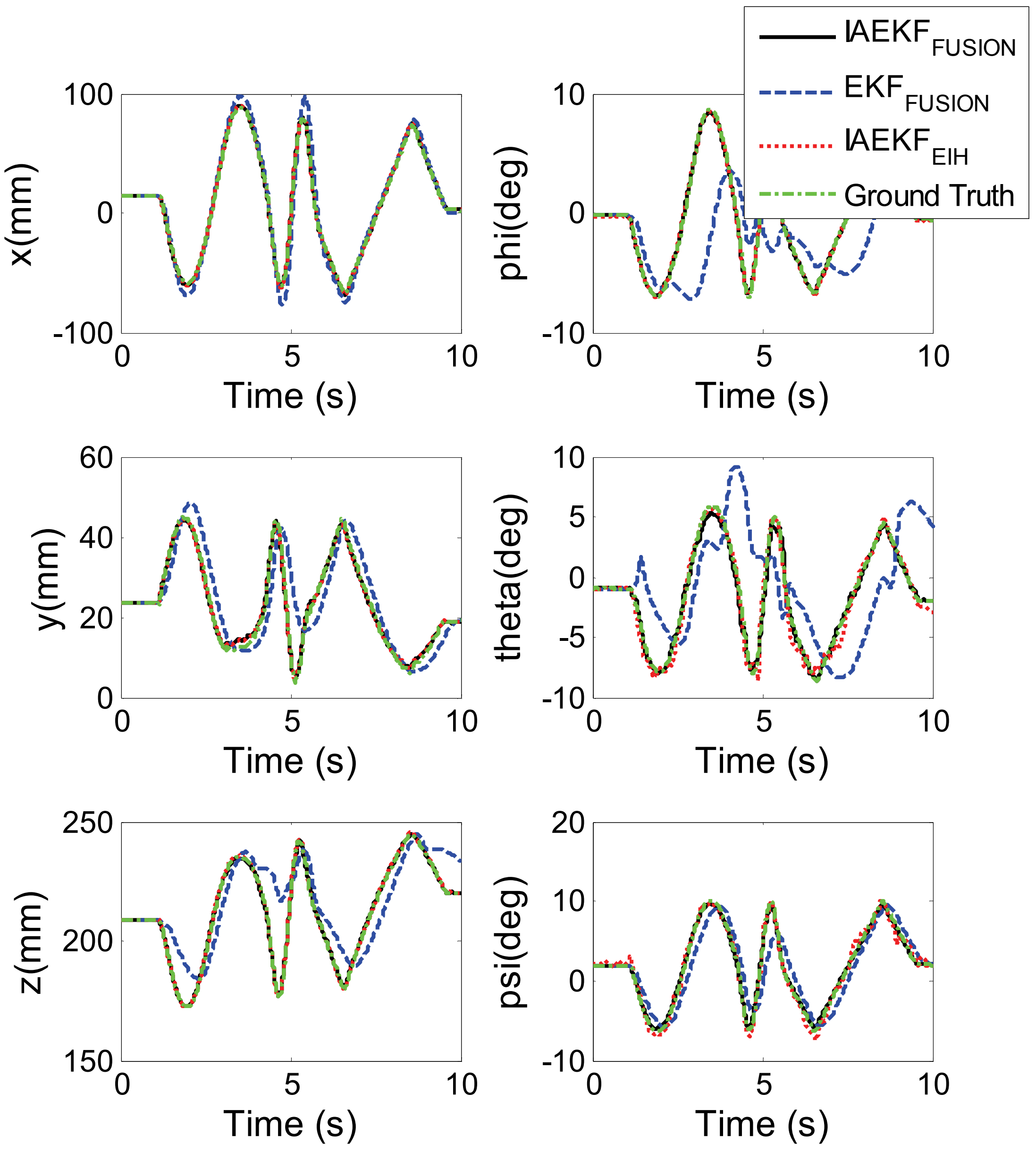


Figure 3.9 Sixth experiment: Pose estimation output of IAEKF fusion in comparison with monocular IAEKF (IAEKF$_{EIH}$), EKF fusion, in case of sampling mismatch.

In experiment 7, the robustness of the proposed estimation method is investigated for feature occlusion. To serve this purpose, two of the features in EIH camera are occluded for 1 second. The IAEKF fusion outcome of feature loss is shown in Figure 3.10 in comparison to two previously mentioned estimation methods. As it was expected, the IAEKF, based on EIH, diverges, since there are not enough feature points to follow the pose changes. Also the EKF fusion method is lost during the occlusion, however both return to the correct values once the features are visible again. The best performance belongs to IAEKF fusion which remains almost unchanged, thanks to the weighting system which isolates the faulty features from the measurement vector.

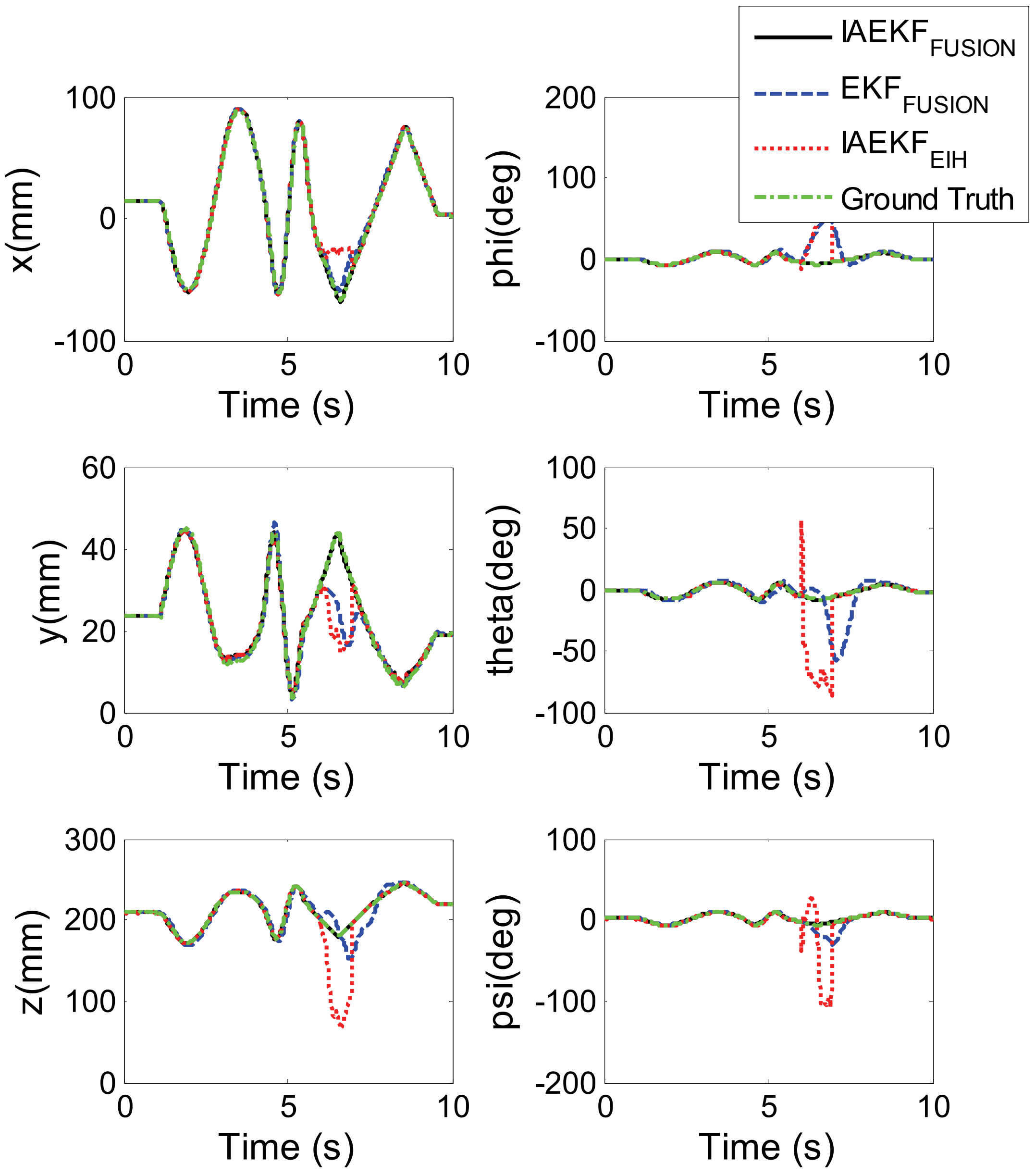


Figure 3.10 Seventh experiment: Pose estimation output of IAEKF fusion in comparison with monocular IAEKF ($IAEKF_{EIH}$), EKF fusion, in case of occlusion.

In order to test the superiority of data fusion over estimations using single data sources, the focal length of EIH camera is misadjusted by %2 in experiment 8. The error statistical parameters of this experiment are summarized in Table 3.5. As it can be concluded, fusion methods have superior performance especially in the estimated depth as its errors are less than the other methods. The focal length error of EIH camera is compensated by ETH camera data. Moreover, IAEKF fusion shows better performance than EKF fusion.

		x (mm)	y (mm)	z (mm)	ϕ (deg)	θ (deg)	Ψ (deg)
Max Error	IAEKF Fusion	6.2	2.5	3.2	1.3	2	0.5
	IAEKF EIH	4.9	2.8	6.7	0.6	2.3	2.7
	EKF Fusion	6.5	2.7	14.1	7.8	5.3	1.8
Mean Error	IAEKF Fusion	1.8	0.6	0.7	0.2	0.6	0.1
	IAEKF EIH	1.2	0.6	4.5	0.1	0.4	0.7
	EKF Fusion	1.8	0.5	2.8	0.9	1.6	0.2
Std	IAEKF Fusion	1.2	0.5	0.7	0.2	0.4	0.1
	IAEKF EIH	1	0.5	0.6	0.1	0.3	0.7
	EKF Fusion	1.4	0.4	2.8	1.3	1.3	0.3

Table 3.5 Eighth experiment: IAEKF fusion in comparison with EKF fusion and monocular IAEKF, in case of EIH calibration error (%2 focal length error).

The speed of the proposed IAEKF and IAUKF central fusion algorithm is compared with its counterparts in the 9th experiment. The estimation time is calculated for each method using a Core i3 2.2 GHz laptop with 4GB RAM. The results are shown in Table 3.6. As it was expected, iterative methods (i.e., IAEKF-based estimations) are more time consuming. Moreover, the amount of data processed for fusion techniques makes these methods slower compared to monocular camera-based algorithms. It is also noteworthy that IAUKF is considerably slower than the other methods. However, the imposed computational cost is usually manageable in today's fast systems and is not considered as a major problem.

Method	CPU Time per estimation (ms)
IAUKF Fusion	131.9
IAEKF Fusion	33
IAEKF EIH	15.9
UKF Fusion	3.5
EKF Fusion	1.8

Table 3.6 Ninth experiment: Pose estimation time of the different algorithms.

In experiment 10, the efficiency of the proposed method is verified under an altered camera configuration. For this purpose, experiment 2 is repeated, having the cameras relocated in the workspace. The location of cameras and the robot are shown in Figure 3.11. Figure 3.12 shows the result of the proposed pose estimation versus its competitors. The errors of these methods are demonstrated in Figure 3.13. As it can be inferred, the proposed method provides a more stable and accurate pose estimation compared to the two others, regardless of the pose of the cameras.

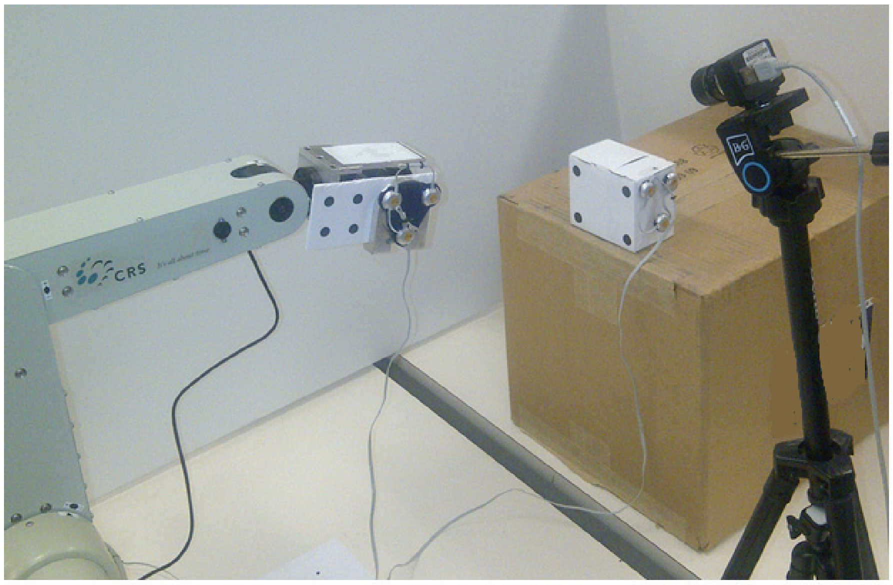


Figure 3.11 The experiment configuration of experiment 10.

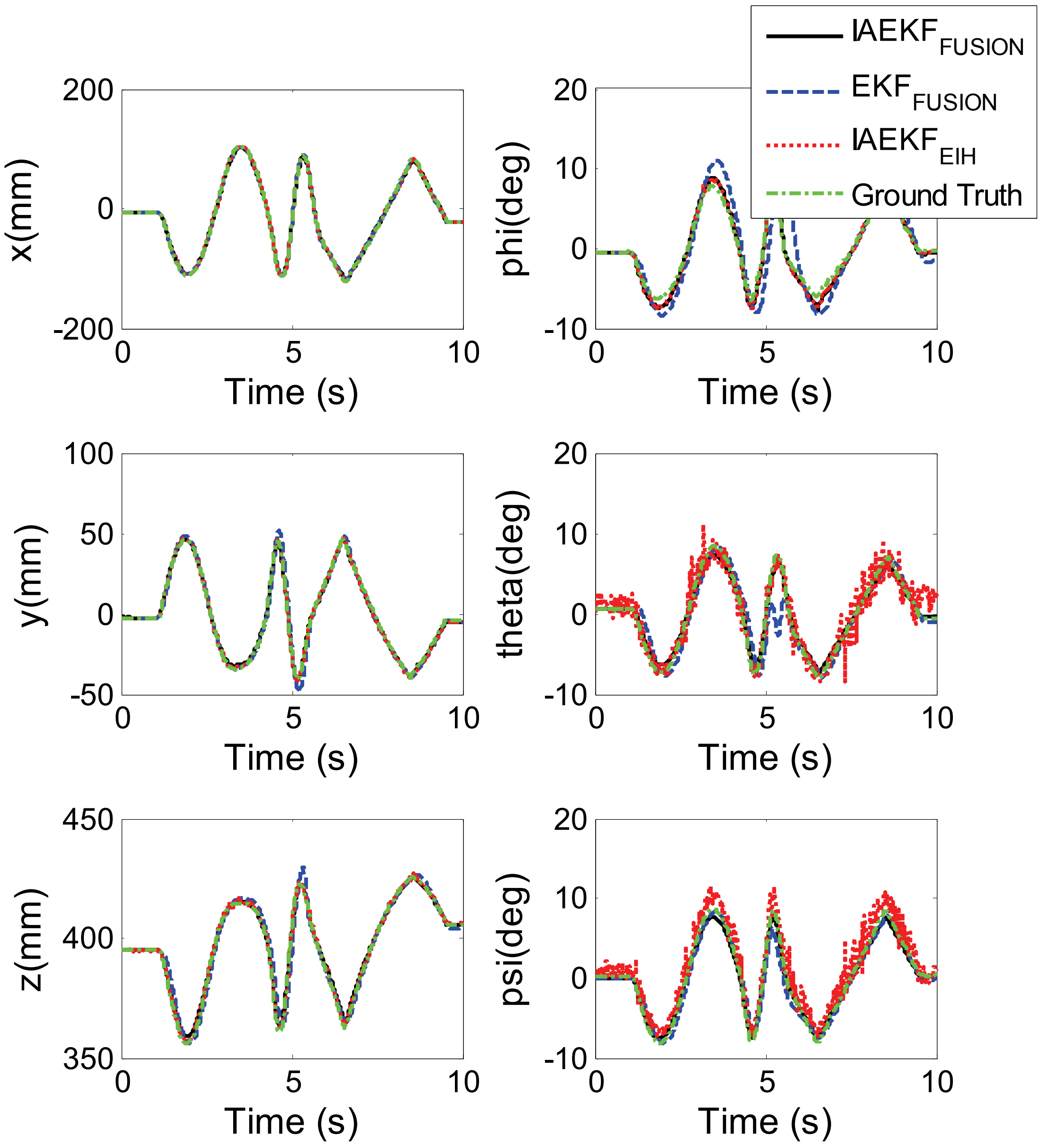


Figure 3.12 Tenth experiment: Pose estimation output of IAEKF fusion in comparison with monocular IAEKF (IAEKF$_{EIH}$), EKF fusion and ground truth, in case of altered camera configuration.

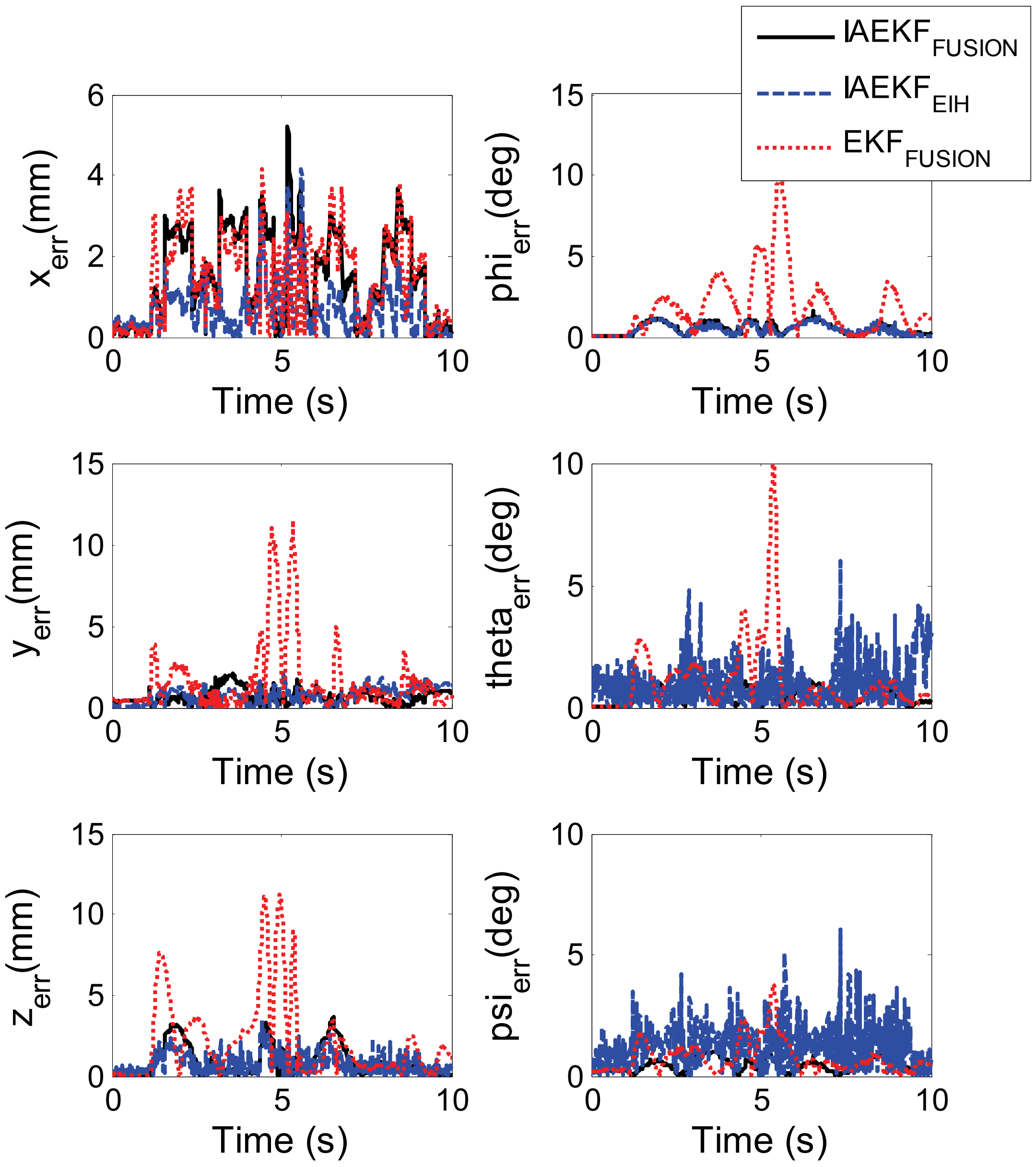


Figure 3.13 Tenth experiment: Pose estimation error of IAEKF fusion in comparison with monocular IAEKF ($IAEKF_{EIH}$), EKF fusion, in case of altered camera configuration.

Next, the proposed VVS fusion method is compared to its rival methods, namely Gauss-Newton optimization-based and EKF algorithms through an experiment. All algorithms are initialized through POSIT algorithm proposed by [2.16]. The POSIT method approximates the pose initially through geometric projection of the features on the camera plane, and then makes the estimation accurate through iterations. At each step, the estimated pose of previous step is utilized as the initial pose. Since the initial poses are close to final estimations, a total number of three iterations are considered for each of the algorithms. Accuracy and time efficiency are the criteria of this comparison.

In the 11th experiment, the accuracy of centrally fused virtual visual servoing is compared with that of Gauss-Newton optimization and EKF methods. A trajectory composed of three different movements is considered as a test bench. The estimation error of each algorithm during robot operation is highlighted in Figure 3.14. As it was expected, the estimations resulted from VVS and Gauss-Newton optimizations are very close. However, VVS shows to be more robust to image noise, which makes it more attractive. The same property was reported by [2.20] in the case of a single camera. Both of these methods outperform EKF in terms of accuracy, as it can be inferred from Figure 3.14. The error statistics are briefed in Table 3.6. As it can been seen from the table, VVS performs slightly better than Gauss-Newton optimization, and much better than EKF method. The difference between VVS and Gauss-Newton optimization is more significant in case of translational parameters (first three columns), while EKF performance is comparable only in case of *x* and *y* directions.

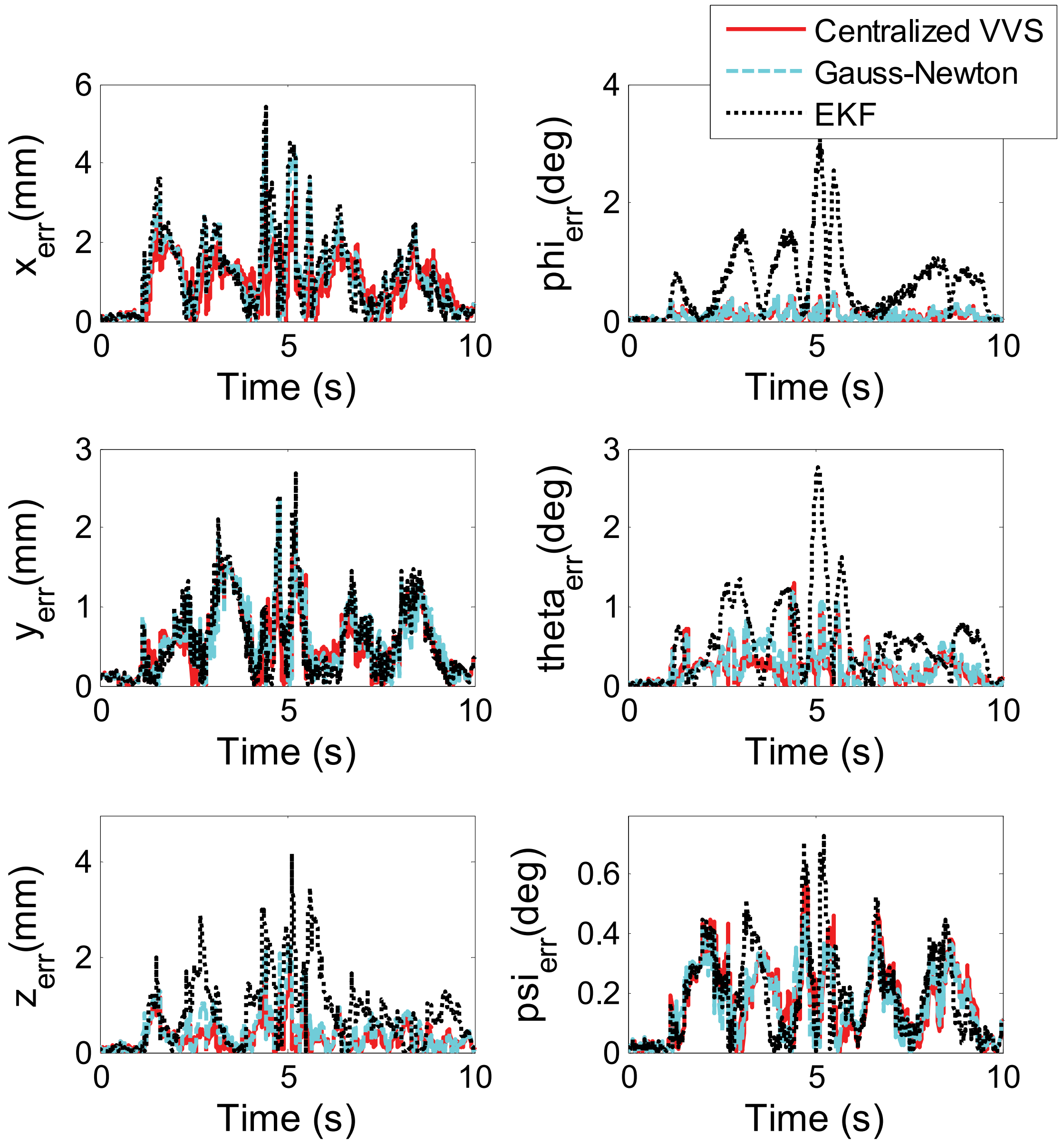


Figure 3.14 Eleventh experiment: Centralized VVS pose estimation errors versus Gauss-Newton and EKF pose estimation errors.

		x (mm)	y (mm)	z (mm)	ϕ (deg)	θ (deg)	Ψ (deg)
Max Error	Centralized VVS	3.6	2.1	1.7	0.5	1.3	0.6
	Gauss-Newton	4.6	2.3	2.3	0.7	1.4	1.2
	EKF	5.4	2.7	4.2	3.1	2.7	0.7
Mean Error	Centralized VVS	1	0.5	0.3	0.1	0.2	0.2
	Gauss-Newton	1.1	0.6	0.4	0.1	0.3	0.2
	EKF	1.2	0.6	0.9	0.6	0.6	0.2
Std	Centralized VVS	0.7	0.4	0.3	0.1	0.2	0.1
	Gauss-Newton	0.9	0.4	0.4	0.1	0.2	0.1
	EKF	1	0.5	0.8	0.6	0.5	0.2

Table 3.7 Eleventh experiment: The results of the proposed VVS fusion in comparison with those of Gauss-Newton and EKF.

3.6.2 Decentralized Fusion

In order to verify the effectiveness of the proposed decentralized EKF-based method, three different simulations and one experiment have been performed. The first simulation puts the idea of error covariance calculation into test. For that matter, the estimations of relative pose of the end-effector and the relative pose of the target object with respect to ETH camera are combined to acquire the relative pose of the object with respect to the end-effector. The error covariance of the combined estimation is calculated during a visual servoing robot maneuver by Monte Carlo simulation, using 100 of different samples and is compared with the estimated error covariance computed previously. The random Gaussian noise with zero mean and 0.001 pixel standard deviation is used for this purpose. Figure 3.15 shows the response of this two error covariance estimations. As it can be seen, the estimated error covariance is matching very closely the true error covariance. Slight mismatches are present as a result of approximations made by this work.

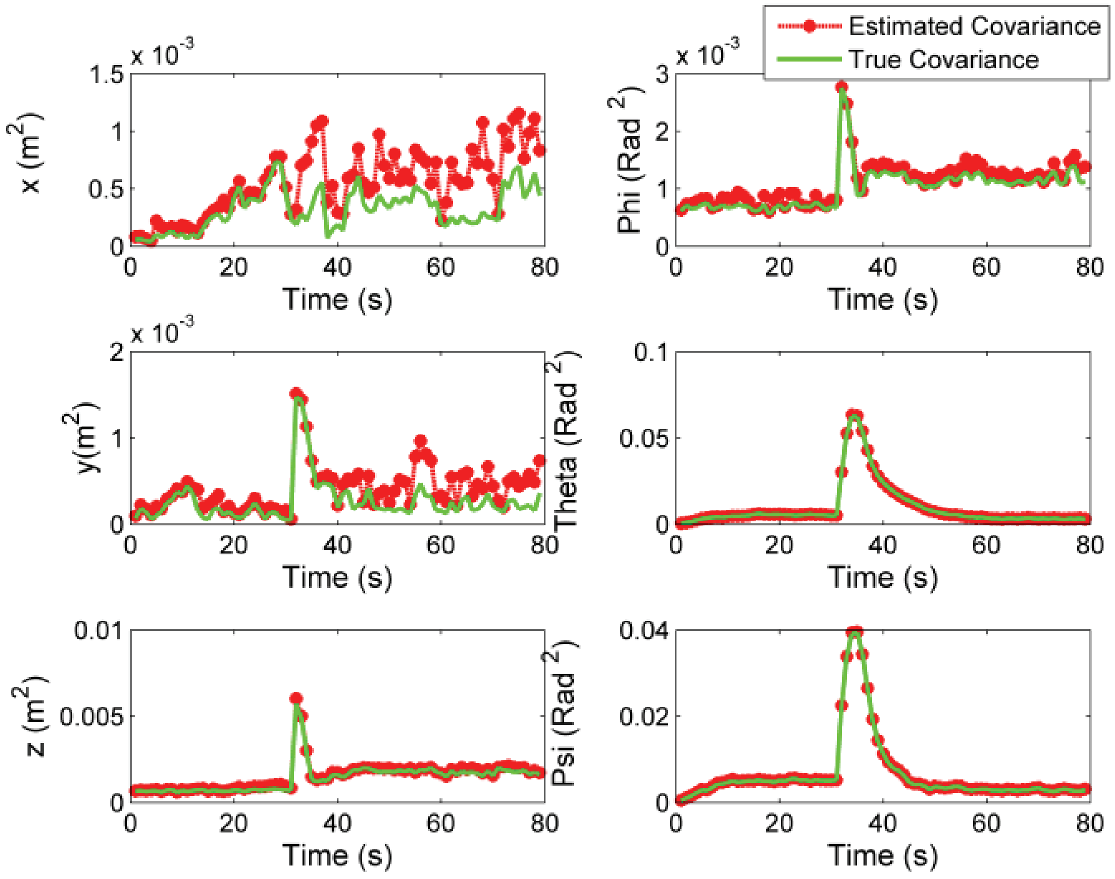


Figure 3.15 First simulation: Error covariance estimation.

The second simulation verifies the effectiveness of the decentralized fusion and analyses the accuracy of this method. For this purpose, the results of a decentralized fusion that exploits the pose estimations from ETH and EIH cameras are verified. The results of this method are brought in Figure 3.16, comparing them with the poses estimated by each camera. The estimation made by data from ETH is generally noisier, since two estimations are used to form this pose. Though, in some of the estimations (e.g., ψ) EIH provides less accurate estimation and the fusion inclines to the data provided by ETH.

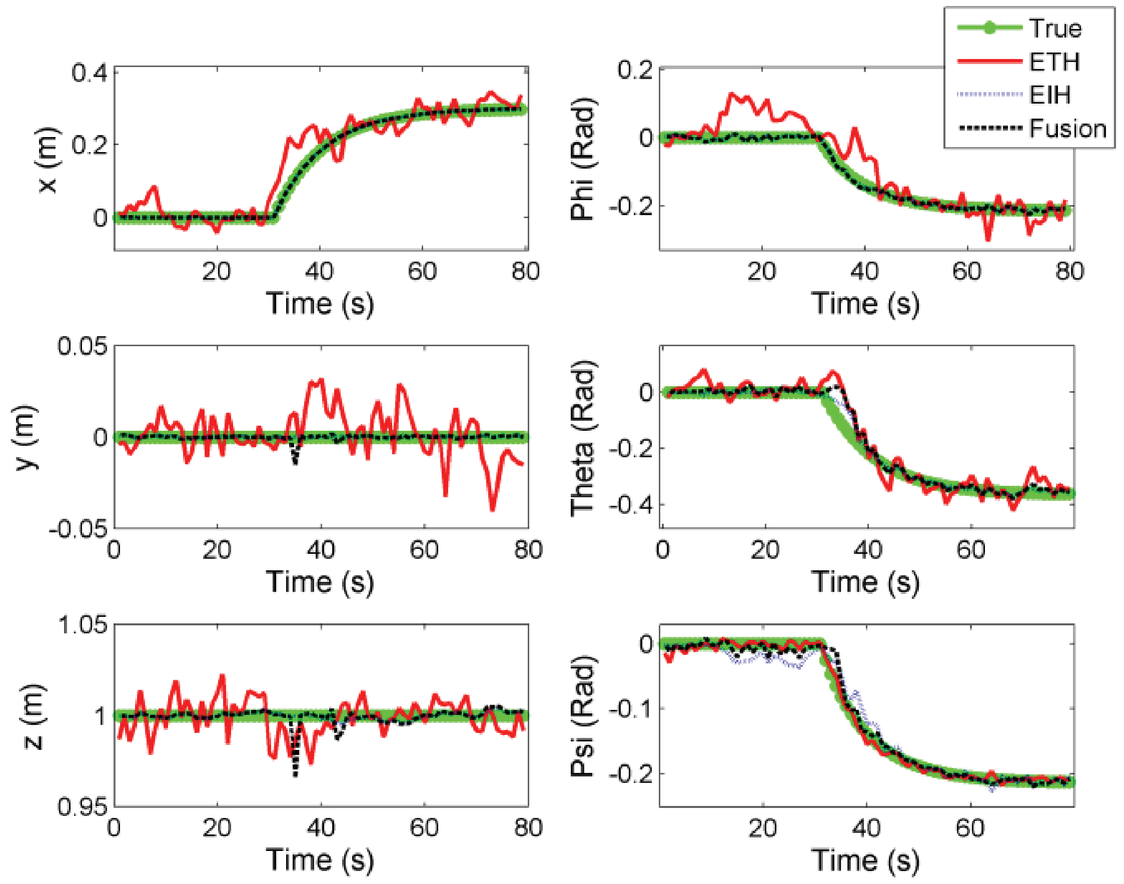


Figure 3.16 Second simulation: Decentralized EKF-based fusion compared with monocular pose estimators.

The third simulation demonstrates the power of decentralized fusion method in the case of fault occurrence. In this simulation, two feature points of the object is occluded from second 40 to second 50. Figure 3.17 shows how the fused pose survives this problem by cutting the faulty data off the fusion layer. The estimation based on EIH camera become unreliable after the occlusion, while the fusion estimation is the same as estimation based on ETH camera, since only two cameras has been used.

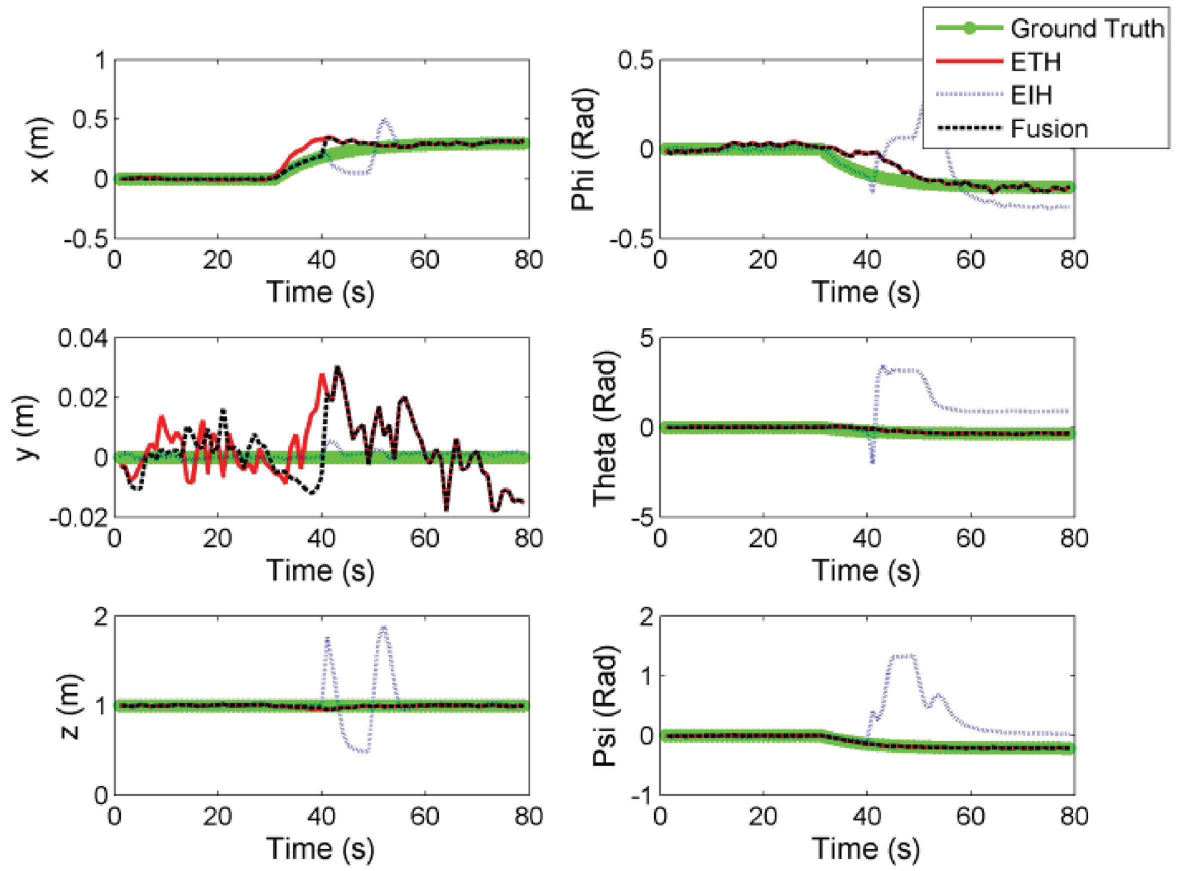


Figure 3.17 Third simulation: Decentralized EKF-based fusion compared with monocular pose estimators in case of faulty local estimation.

Finally, an experiment has been conducted to demonstrate the effectiveness of the proposed method in practice. The experimental setup used for this matter is the same as before. The pose of the object is estimated from each camera separately and is compared to the results from the proposed fusion method. Each Kalman filter is tuned through several experiments. Figure 3.18 shows the results of this comparison. It can be inferred that the result from the fusion technique follows the reality closely, while the other estimations have been deviated slightly in some cases. However, the fusion algorithm is twice slower than the single-camera estimators.

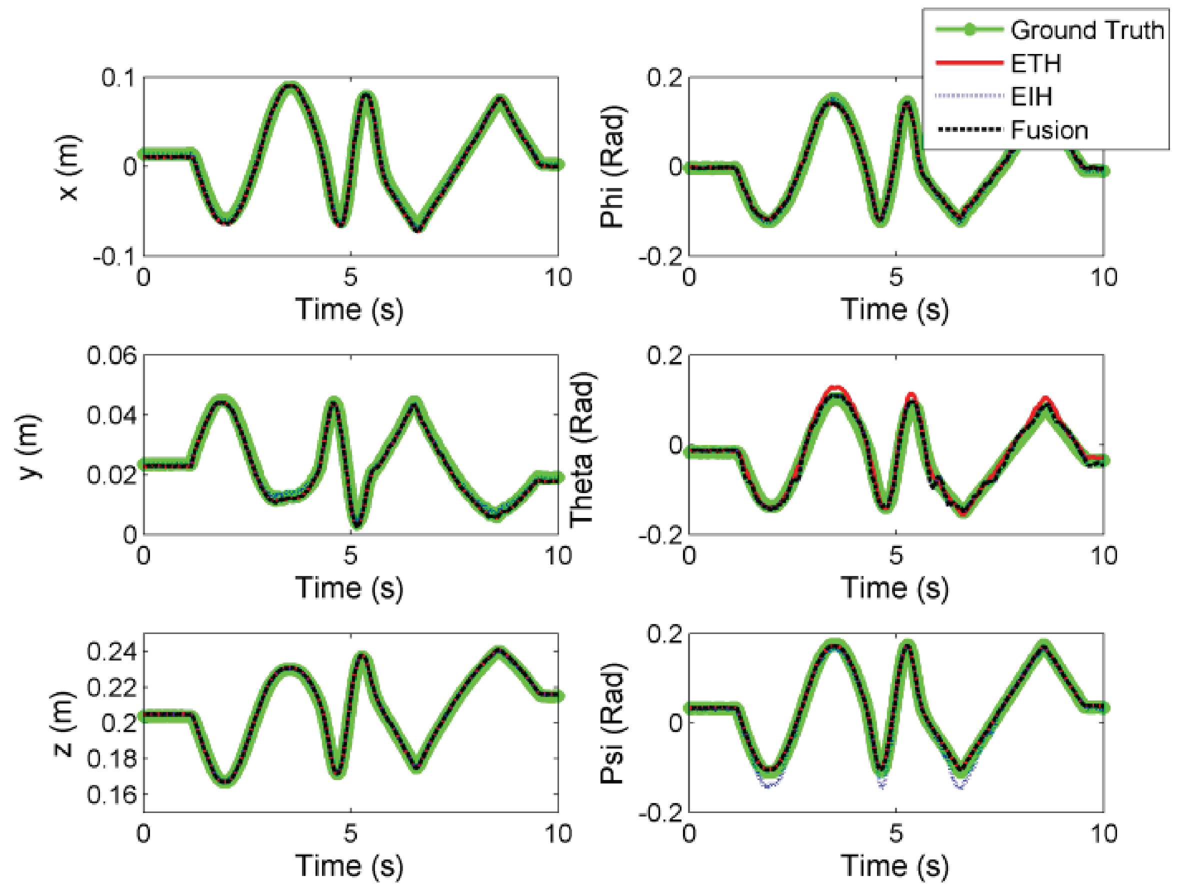


Figure 3.18 Twelfth experiment: Decentralized EKF-based fusion in practice compared with monocular pose estimators.

Next, the accuracy of the proposed decentralized VVS fusion algorithms is put into test through an experiment. Similar trajectories as the experiment for the centralized fusion are used for this purpose. Pre-computed Jacobian, calculated based on measured features, is used for the decentralized fusion. The estimation error of each method is shown in Figure 3.19. The error of the decentralized fusion method seems to be larger, as it was expected. A summary of the important statistical points of the estimation errors can be found in Table 3.8. Judging by the mean value of the estimation errors, the decentralized fusion algorithm is almost as accurate as the centralized method. In some cases it shows equal or even better estimation compared to Gauss-Newton method.

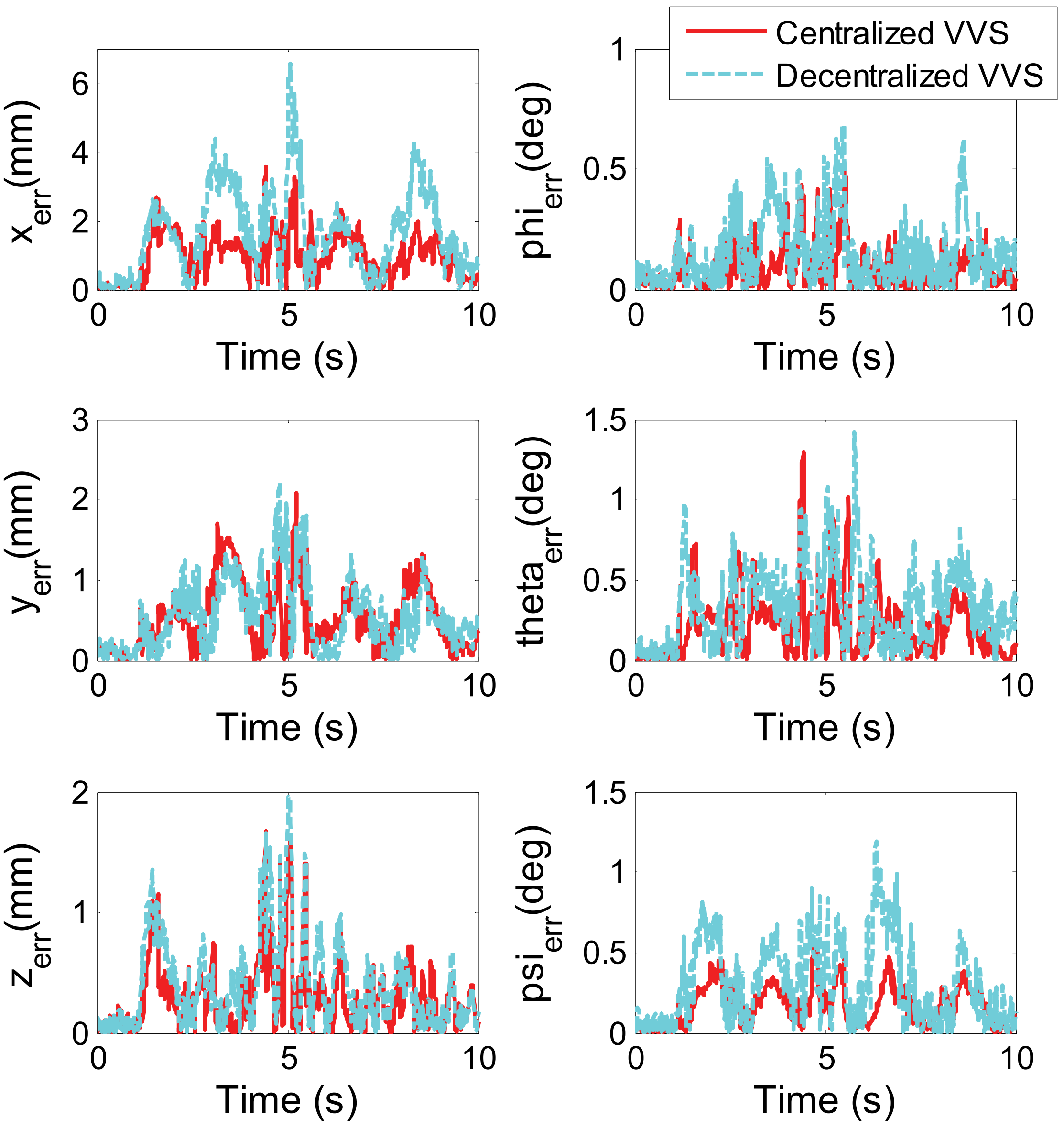


Figure 3.19 Thirteenth experiment: Decentralized VVS pose estimation errors versus centralized VVS pose estimation error.

		x (mm)	y (mm)	z (mm)	ϕ (deg)	θ (deg)	Ψ (deg)
Max Error	Centralized VVS	3.6	2.1	1.7	0.5	1.3	0.6
	Gauss-Newton	4.6	2.3	2.3	0.7	1.4	1.2
	EKF	5.4	2.7	4.2	3.1	2.7	0.7
	Decentralized VVS	6.6	2.2	2	0.7	1.4	1.2
Mean Error	Centralized VVS	1	0.5	0.3	0.1	0.2	0.2
	Gauss-Newton	1.1	0.6	0.4	0.1	0.3	0.2
	EKF	1.2	0.6	0.9	0.6	0.6	0.2
	Decentralized VVS	1.6	0.5	0.4	0.1	0.3	0.3
Std	Centralized VVS	0.7	0.4	0.3	0.1	0.2	0.1
	Gauss-Newton	0.9	0.4	0.4	0.1	0.2	0.1
	EKF	1	0.5	0.8	0.6	0.5	0.2
	Decentralized VVS	1.2	0.4	0.4	0.1	0.2	0.3

Table 3.8 Thirteenth experiment: Decentralized fusion accuracy compared to that of the centralized, EKF-based, and Gauss-Newton-based fusion.

In an experiment, the time efficiency of VVS fusion methods is measured. A laptop with Core i3 2.2 GHz CPU and 4GB of RAM is used for these time measurements. The time of estimation for a single posture is found as an average of pose estimation time over the whole trajectory. The experiment was conducted for a setup with 2 and 4 cameras. The results of these measurements are shown in Table 3.9. The VVS estimation methods are shown to be faster than their counterpart (i.e., the Gauss-Newton). The reason lies within the efficient formulation of the Jacobian matrix in VVS algorithms. Moreover, the decentralized method has shown to be slightly faster than the centralized method. However, the speed improvement of this method, compared to the centralized VVS, is not significant. In fact, the calculation of Jacobian matrix accounts for most of the computational time and the reversion of the Jacobian matrix plays a less significant role. Yet, the time increase in the VVS methods is reported to be small compared to other methods. As it was expected, EKF performs fastest of all, since it is not an iterative algorithm.

Method	Estimation Time (ms)	
	2 Cameras	4 Cameras
Centralized VVS	1.36	1.67
Gauss-Newton	1.76	2.58
EKF	0.96	1.24
Decentralized VVS	1.29	1.6

Table 3.9 Estimation times of VVS fusion methods per estimation compared with that of EKF-based and Gauss-Newton-based fusion for 2 and 4 cameras.

3.6.3 Pre-Processing Fusion

A set of simulations are conducted to justify the proposed pre-processing fusion method. An eye-in-hand camera and an eye-to-hand camera are considered in the simulations. The focal lengths of the cameras are assumed to be 1000 pixels. The end-effector coordinate frame coincides with those of eye-in-hand camera. The eye-in-hand robot moves with a known trajectory, while the eye-to-hand camera and the object are assumed to be stationary in the simulation environment. Initially, the object of interest is located in front of the eye-in-hand camera within distance of 1 meter, parallel to the camera plane. The object consists of four non-coplanar points. The eye-to-hand camera is located a meter away from the eye-in-hand camera, having the object in its FOV. After 3 seconds, the eye-in-hand camera is moved to a new pose and becomes stable after three seconds. An inverse Jacobian scheme is exploited to move the camera to its new location. The iterative pose estimation method is implemented. The number of iterations is set to 10. The fusion is not performed in the first iteration of the first estimate, since the depth information is not available.

In the fourth simulation, the pose is estimated counting only on the eye-in-hand camera. Additive Gaussian noise with standard deviation of 1 pixel is considered for features of this camera. The noise covariance is counted as the noise level of the camera data. Figure 3.20 shows the estimated pose compared to ground truth. As it was expected, this method shows high sensitivity to the measurement noise. This is especially noticeable in case of orientation angles and depth estimation. Another reason behind high errors is the long distance of the camera from the object. The estimation algorithm will perform better once the camera is closer to the object.

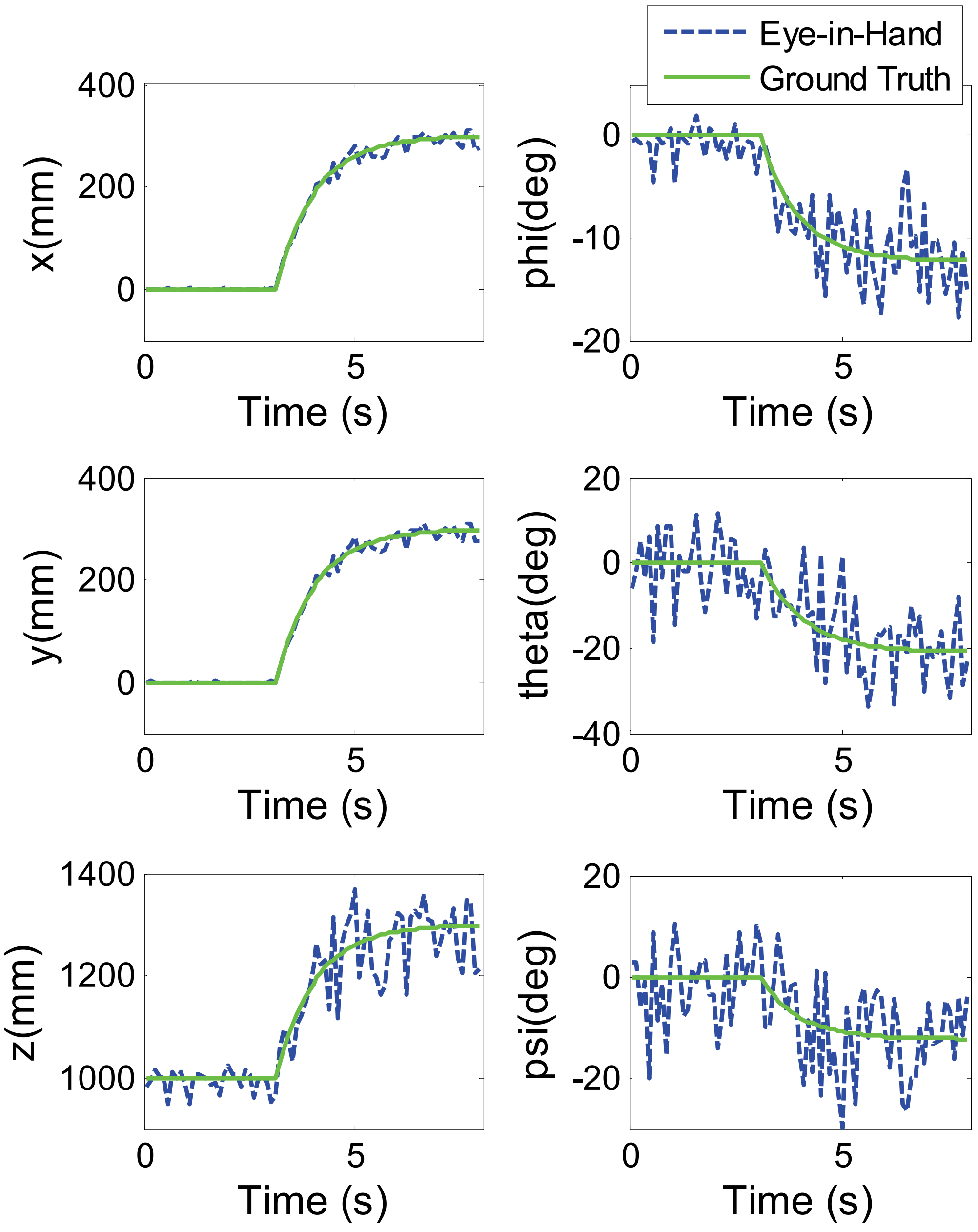


Figure 3.20 Fourth simulation: Monocular pose estimation with an eye-in-hand camera.

In order to alleviate the noise sensitivity, the proposed fusion scheme is employed in fifth simlation. The data from ETH camera is fused with data from EIH camera. Three different noise values are considered for the eye-to-hand camera data. This noise is assumed to be additive Gaussian and has the standard deviation of 0.5, 1, and 2 pixels for different simulations. The weights of fusion are equal, based on the assumption of no prior knowledge about the noise values of the camera. Figures 3.21-3.23 show the results of these fusions. As it can be seen from the figures, the higher the noise level gets, the worse the estimation result will become. This is mainly due to the maladjustment of the weights in the fusion process. By having the same weights for all sensors, the noisy sensors might dominate the less noisy sensors. As it was mentioned before, one solution to that problem is to detect the highly noisy data and exclude them from the fusion data (robust to outliers). The other way is to adjust the fusion weights properly. Table 3.10 briefs the important statistics of the fusion estimation error for different levels of noise, in addition to monocular estimation error. As it was expected, data fusion of cameras with lower and even equal noise level results in improved accuracy of pose estimation. One should realize that once the noise levels are equal, the weight tuning results in equal weights, which is the default value for the weights. If the weights are not tuned, the noisy sensor may corrupt the overall fusion data, which is the case for fusion of eye-to-hand data with noise level twice as the eye-in-hand camera data.

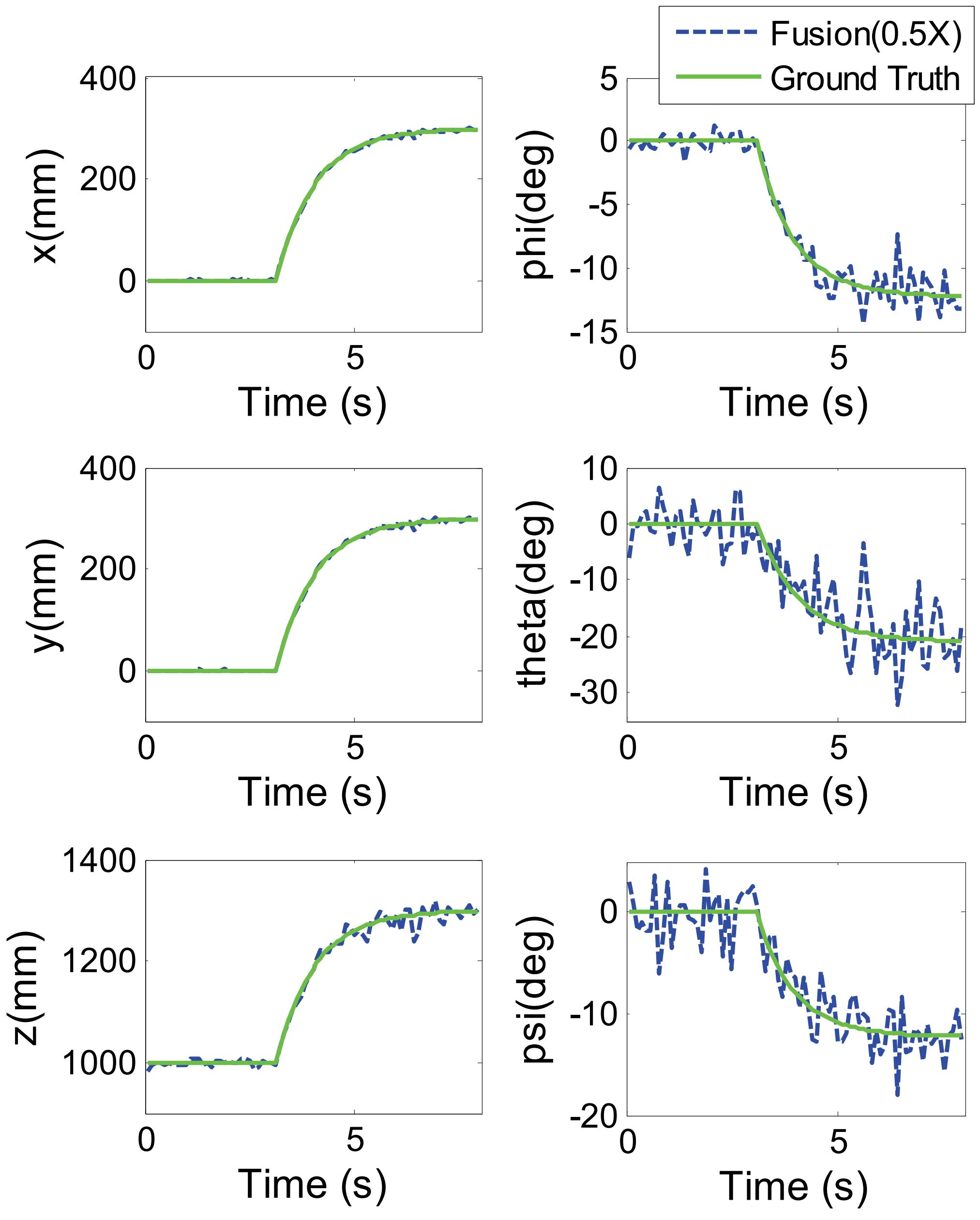


Figure 3.21 Fifth simulation: Decentralized EKF-based fusion for pose estimation with half level noise for ETH camera.

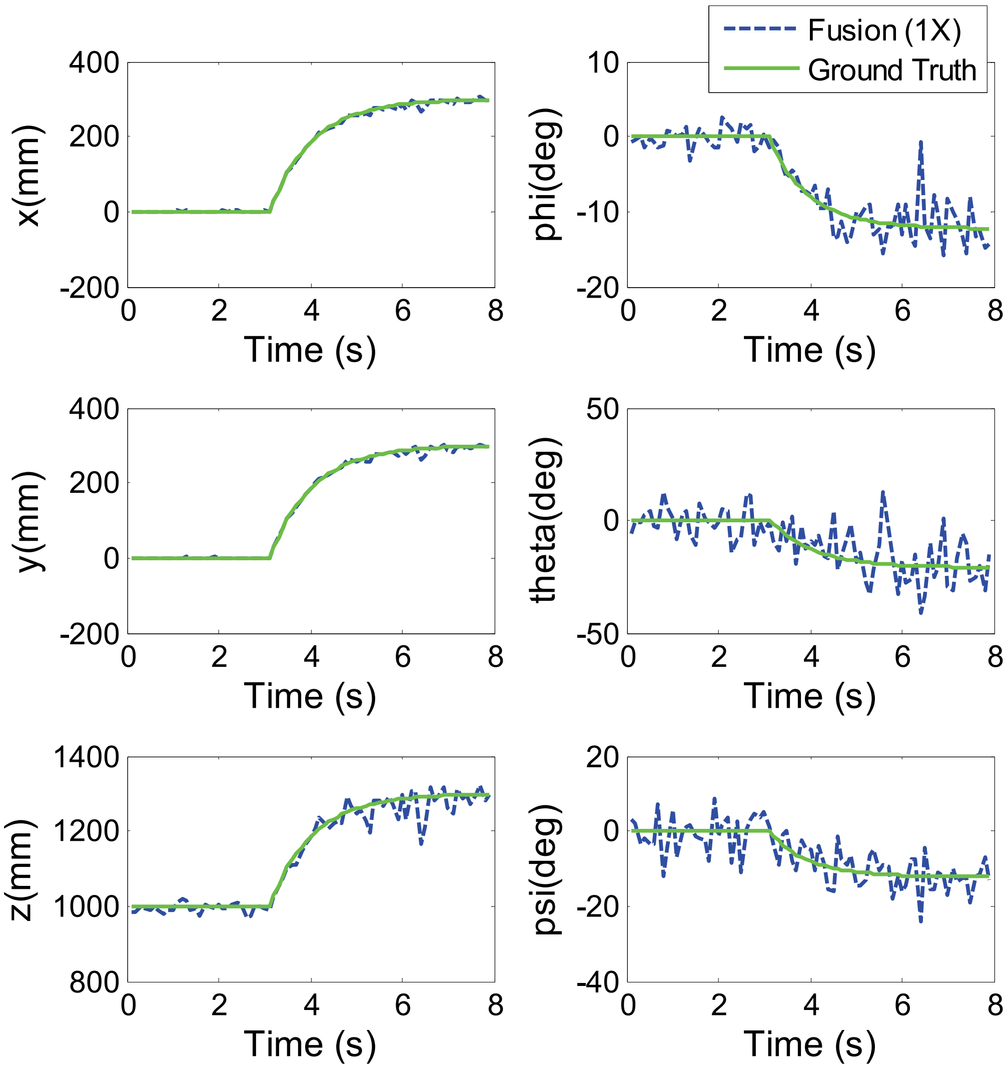


Figure 3.22 Fifth simulation: Decentralized EKF-based fusion for pose estimation with equal camera noise levels.

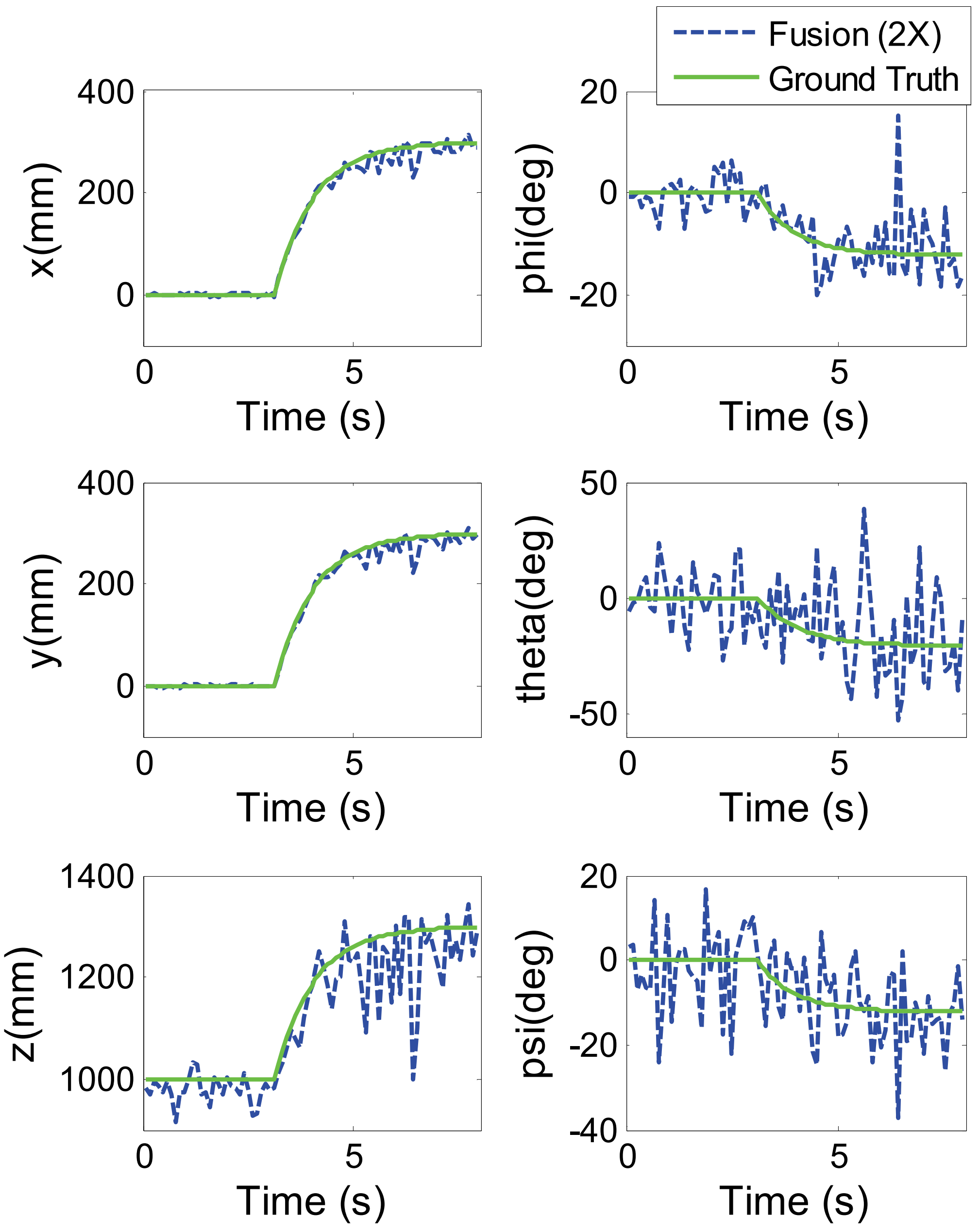


Figure 3.23 Fifth simulation: Decentralized EKF-based fusion for pose estimation with double level noise for ETH camera.

		x (mm)	y (mm)	z (mm)	ϕ (deg)	θ (deg)	Ψ (deg)
Max Error	Eye-in-hand	26.7	27.2	125.4	8.7	19.7	19.9
	Fusion(0.5x)	7.3	7.6	32.7	1.8	4.6	4.6
	Fusion(1x)	24.9	30.8	126.2	11.2	31.9	12.3
	Fusion(2x)	60.1	70.3	293.9	27	57.5	25.2
Mean Error	Eye-in-hand	6.1	6.3	35.5	2.1	6.4	6.4
	Fusion(0.5)	1.7	1.7	8.3	0.8	3.6	1.9
	Fusion(1)	3.5	3.2	16.8	1.6	7.1	3.7
	Fusion(2)	8.4	7.4	42.5	3.5	13.4	7.3
Std	Eye-in-hand	7.3	7.6	32.7	1.8	4.6	4.6
	Fusion(0.5)	1.8	2.2	9.1	0.7	2.9	1.4
	Fusion(1)	4	4.8	20.2	1.6	5.8	2.8
	Fusion(2)	10.5	11.6	49	3.6	10.7	5.6

Table 3.10 Decentralized EKF-based fusion results compared to those of monocular EIH camera.

In the next simulation, the importance of weight tuning for fusion is highlighted. For this purpose, the case of fusion of two cameras with different noise levels is considered again; however the weights are adjusted based on the noise covariance as was described in (3.40). The results of this fusion compared to equally weighted fusion are shown in Figure 3.24. As it was expected, tuning the weights increased the performance of the system significantly and prevented the noisy data from dominating the overall output of the fusion scheme.

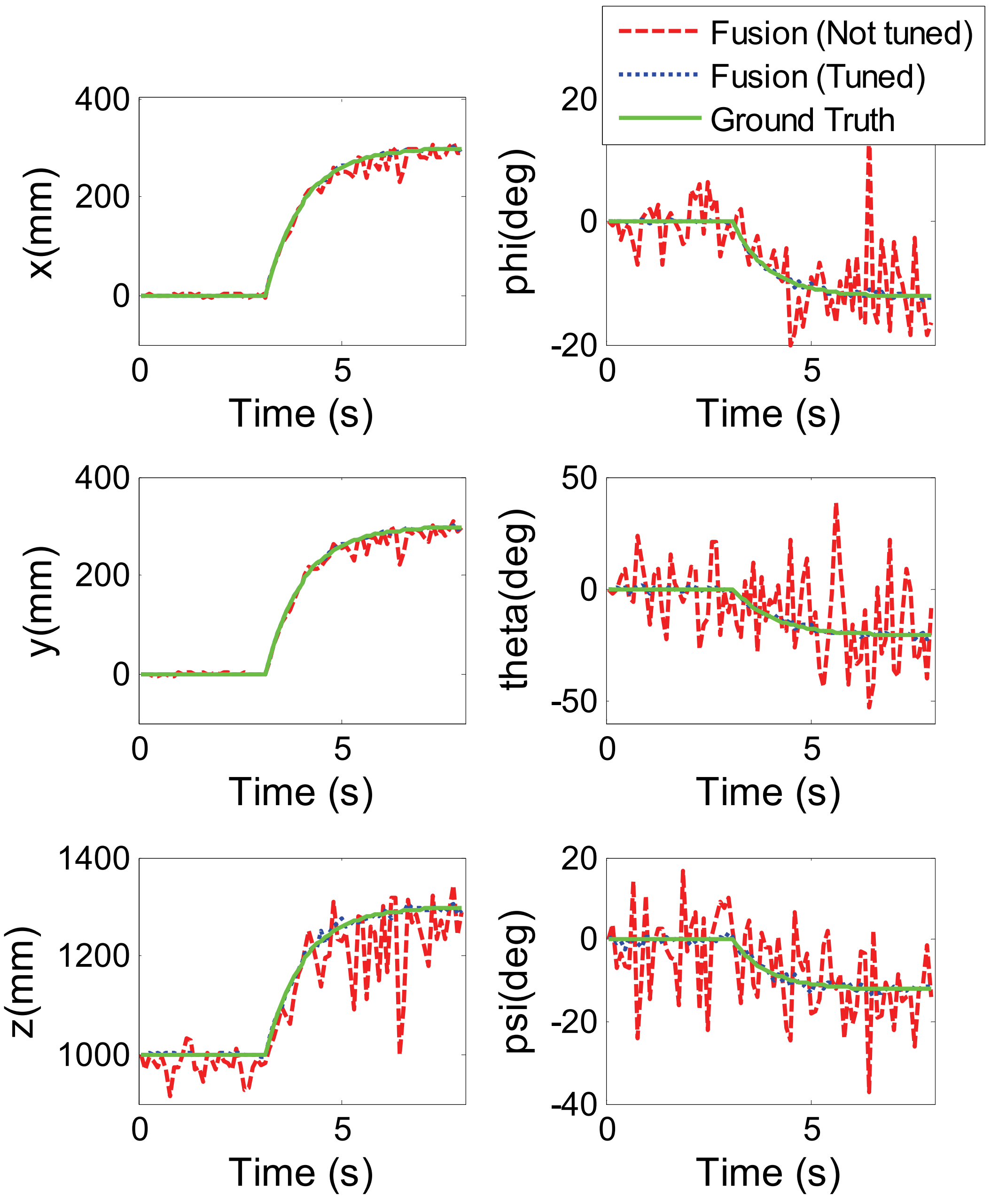


Figure 3.24 Fusion with tuned weights pose estimation versus fusion with equal weights pose estimation results.

3.7 Summary

Accurate and robust pose estimation plays a key role in visual servoing systems. Traditionally, monocular vision was exploited for this purpose; however, the accuracy of a single camera is limited and its measurements are prone to outliers and occlusions. Sensor fusion of a multi-camera system was proposed to enhance both the accuracy and robustness of the pose estimation. Three fusion structures were introduced for this matter. Centralized fusion algorithms enabled the system to enhance the accuracy of the system. Two centralized methods, namely IAEKF and VVS were introduced, which had the capacity to increase the robustness of the system to parameter uncertainties and occlusion in addition to the accuracy. The major drawback of these systems was their increasing computation time with the number of cameras. Decentralized fusion algorithms were proposed to alleviate this problem. These methods were relatively less accurate compared to their centralized fusion counterparts; however, they performed comparatively faster. This speed difference was more tangible once the number of sensors is significant. In addition, these schemes were capable of fault isolation, which was difficult in centralized fusion algorithms. Finally, a pre-processing fusion scheme was introduced which was independent of the pose estimation algorithm, unlike the centralized fusion techniques. This fusion method was shown to be effective once the noise level of each camera is known in advance.

Despite the advantages of multi-camera systems and fusion techniques, these methods impose higher costs to the system. Moreover, the overall processing time of these systems increases (almost linearly) with the number of sensors. However, the price to be paid for the achievements in increased accuracy and robustness seems to be minimal, since the efficiency of the visual servoing systems directly depends on the performance of pose estimation system. In addition, multi-camera systems add to the flexibility of the system by adding new degrees of freedom. The system designer may decide on the number of sensors and fusion algorithms based on the application, requirements, and available system hardware.

Chapter 4

Uncertainty Modeling in Visual Servoing Systems

4.1 Introduction

Integration of visual servoing systems into real-life applications mainly depends on their robustness to system uncertainties. While the robustness in the sense of system's stability has been well studied, the accuracy of system during the servoing (i.e., system trajectories) has not been thoroughly investigated. In order to enhance the accuracy of the system, its behavior under uncertain conditions must be investigated. Therefore, uncertainty modeling remains a key step towards the development of accurate systems. This chapter briefly reviews previous uncertainty modeling methods for visual servoing and their shortcomings, which led to the introduction of a novel closed-loop error modeling approach developed by following a probabilistic methodology. The modeling strategy is established based on a single-input single-output (SISO) system which accounts for discrete-time nature of the systems. Subsequently, the derived method is applied to visual servoing systems. Unlike the previous works, the proposed method is expandable to various types of controllers used for servoing purposes. In addition, it basically presents a straight-forward way to calculate the error covariance function over time, which is used in control loops. Thus, simplicity and generality of the proposed method are its main advantages over the previously proposed methods. Moreover, this method models the system uncertainties more accurately by considering the discrete-time characteristic of the system. The proposed uncertainty modeling methods are verified through Monte Carlo simulations, since the assumptions of this work (e.g., noise distribution) are hard to be realized in experimental setups. The proposed modeling method is exploited for control purposes, in the following chapter.

This chapter is organized as follows. In Sec. 4.2 a brief review of previous works on uncertainty modeling in visual servoing systems is presented. The open-loop and closed-loop uncertainty modeling methods based on probabilistic error propagation are explained in Sec. 4.3. The application of uncertainty models in visual servoing systems are brought in Sec. 4.4. The simulation results and discussion are provided in Sec. 4.5. A summary of this chapter in Sec. 4.6 concludes the chapter.

4.2 Literature Survey

In visual servoing scenarios, image data is usually assumed to be accurate, which might not hold in practice. As a matter of fact, the uncertainties in intensity function, digitalization, and image processing levels [4.1] all contribute to image uncertainties. Moreover, the system uncertainties and modeling errors affect the robot's end-effector pose, which would lead to the inaccurate positioning and even task failure. The focus of this chapter is on uncertainties introduced by image features. Two different methodologies have been developed in previous works to model the uncertainties in visual servoing systems, namely the boundary and the probabilistic methods.

On one hand, the boundary method [4.2, 4.3] offered the upper and lower boundaries of camera displacement error using an eye-in-hand camera, assuming the level of image noise boundaries to be known. This method was later employed for feature selection [4.4] and global path planning [4.5]. While these works have proven their usefulness, they do not usually consider the servoing method used for visual servoing, i.e., only the final pose error is discussed. They assume that the exact pose of the object is known, which could only be estimated using visual data. Besides, this approach only provides conservative bounds of the pose error given a known image error bound [4.6]. Moreover, this method is time consuming and computationally expensive.

On the other hand, the approach of probabilistic error analysis was proposed based on error covariance propagation through different components of a visual servoing system. This method was first employed in [4.7, 4.8] to analyze the error in PBVS and hybrid visual servoing (HVS) [4.9], in an open-loop fashion. The method divided these visual servoing systems into three components, i.e., pose estimation, servoing, and control. Subsequently, the image noise covariance was propagated through the linearized model of each part, similar to [4.10]. This work was recently extended to entail IBVS method and the closed-loop nature of visual servoing systems [4.11], since the previous works failed to address this important characteristic. The continuous-time Ornstein-Uhlenbeck process was exploited to model a proportional feedback controller of a simple linear system. The characteristics of this process were explored in IBVS and PBVS systems, and simulations were run for verification. It was shown that only steady-state error covariance is obtainable for PBVS systems through this method. Moreover, the simulation results had some discrepancies between the theory and simulations, which was left for future research [4.11]. In this work, the closed-loop visual servoing method was considered as a time continuous system, which does not hold for visual servoing systems as the vision sensor data is acquired in discrete-time intervals. Moreover, the chosen stochastic process, (i.e., Ornstein-Uhlenbeck process) is limited to the proportional control law only and does not apply to other controllers. This chapter presents a novel uncertainty modeling method for visual servoing systems that overcomes the shortcomings of the previous methods. In the sequel, the modeling methods are discussed in open-loop and closed-loop systems.

4.3 Uncertainty Modeling in SISO Systems

Probabilistic modeling of uncertainties propagated through a system is the topic of this section. The previous works on open-loop error modeling for a SISO system are briefly reviewed and a superior closed-loop model is proposed to address the shortcomings of these works as follows.

4.3.1 Open-Loop Uncertainty Propagation

This approach is based on first-order linear approximation of the system. A SISO system expressed by a function which relates its input and output is assumed,

$$y = h(x), \tag{4.1}$$

where *x, y,* and *h* represent the system's input, output, and the mapping function, respectively. The first-order approximation based on Taylor series suggests,

$$y - y_0 \approx \left.\frac{dh(x)}{dx}\right|_{x=x_0} .(x - x_0)\,. \tag{4.2}$$

Assuming the system error as a deviation from true value, one can write,

$$\hat{y} = y + \varepsilon_y, \quad \hat{x} = x + \varepsilon_x, \tag{4.3}$$

where the noisy signals are denoted by "hat", the noise values are denoted by ε in x and y directions and are assumed to be zero-mean random variables. Then (4.3) could be rewritten as,

$$y - \hat{y} \approx \left. \frac{dh(x)}{dx} \right|_{x=\hat{x}} .(x - \hat{x}), \tag{4.4}$$

which is equal to,

$$\varepsilon_y \approx \varepsilon_x \left. \frac{dh(x)}{dx} \right|_{x=\hat{x}} . \tag{4.5}$$

If the covariance of the input error (ε_x) is known, the covariance of the output error could be approximated as follows,

$$\sum\nolimits_y = E\left(\varepsilon_y{}^2\right) = E\left(\varepsilon_x{}^2 \left(\left. \frac{dh(x)}{dx} \right|_{x=\hat{x}} \right)^2 \right) = E\left(\varepsilon_x{}^2\right) \left(\left. \frac{dh(x)}{dx} \right|_{x=\hat{x}} \right)^2 = \sum\nolimits_x \left(\left. \frac{dh(x)}{dx} \right|_{x=\hat{x}} \right)^2 , \tag{4.6}$$

where $\sum_x$ is the input error covariance, $\sum_y$ is the output error covariance and $E(\cdot)$ is the expectation function. Equation (4.6) relates the covariance of the input and output of a system and is the basis of error propagation in the probabilistic approach.

While the open-loop modeling discussed above provides a useful tool for estimating the uncertainties of the system, it may not be useful for closed-loop systems since the feedback loop of the system has not been accounted for. A novel closed-loop uncertainty modeling is proposed to address this shortcoming as follows.

4.3.2 Closed-Loop Uncertainty Propagation

In this subsection, the uncertainty propagation in a closed-loop SISO control system is considered. The developed method is extendable to multi-input multi-output (MIMO) systems such as visual servoing. Figure 4.1 shows such a control system, where $g(\cdot)$ is the sensor transfer function and $h(\cdot)$ represents the system's controller and plant transfer functions. As it can be seen, the following relations are considered for input and output of each subsystem,

$$e_x = x^* - \hat{x}, \tag{4.7}$$

$$y = h(e_x), \tag{4.8}$$

$$x = g(y), \tag{4.9}$$

$$\hat{x} = x + \varepsilon_x. \tag{4.10}$$

where e_x is the closed-loop error, x^* is the system reference input, y is the system output, and $\hat{x}$ is the feedback signal with uncertainty, x is the feedback's real value and ε_x is the feedback noise, which is assumed to have zero mean and known covariance. The noise sequence is assumed to be independent and identically distributed (i.i.d.). The goal of the system is to minimize the closed-loop control error, e_x. Different controllers are usually employed for this purpose. This work investigates the uncertainty propagation in case of two most well-known controllers, namely proportional (P) and proportional-derivative (PD), which also have become very popular in visual servoing systems.

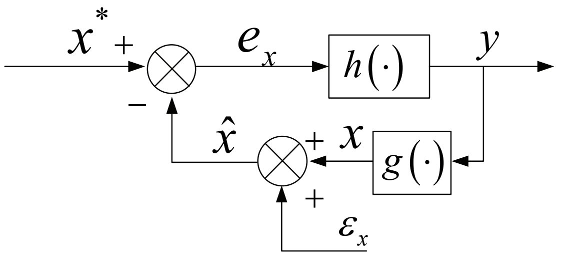


Figure 4.1 A general SISO control system.

Proportional Controller

The closed-loop uncertainty propagation in a system with a proportional controller was previously followed by exploiting the solution of a stochastic process [4.8]. This process, known as Ornstein-Uhlenbeck process, is closely related to closed-loop control using simple proportional control law. The process is defined as follows,

$$dx_t = -\lambda(x_t - \mu)dt + \sigma dw'_t, \tag{4.11}$$

where w'_t is a Wiener-Levy process with unity variance, λ, μ, and σ are known constants. If initial point is known ($x_0 = c'$), the mean and covariance of this process is solved as follows,

$$E\left[x_t \middle| x_0 = c'\right] = \mu + (c' - \mu)e^{-\lambda t}, \tag{4.12}$$

$$Cov(x_s, x_t | x_0 = c') = \frac{\sigma^2}{2\lambda}\left(e^{-\lambda|s-t|} - e^{-\lambda(s+t)}\right), \tag{4.13}$$

and the variance is obtained by coinciding the times,

$$Cov(x_t, x_t | x_0 = c') = \frac{\sigma^2}{2\lambda}\left(1 - e^{-2\lambda t}\right). \tag{4.14}$$

Despite the usefulness of this approach for uncertainty modeling in systems with proportional controllers, it cannot be extended to other types of controllers. Moreover, the discrete-time nature of the system is not considered. Therefore, a new uncertainty modeling approach is proposed to overcome these shortcomings in the sequel.

In a system with proportional controller, the error is decreased to zero exponentially using a proportional control law,

$$\dot{e}_{x,k} + \lambda e_{x,k} = 0, \tag{4.15}$$

where λ is the controller gain. Using (4.7) and (4.10), it can be shown that,

$$\dot{x} = -\lambda(x_k - x^*) - \lambda\varepsilon_x + \dot{x}^*. \tag{4.16}$$

In this work, the system reference input was assumed to be constant ($\dot{x}^* = 0$). The process x could be approximated by integration over time,

$$x_{k+1} = x_k + \Delta t\dot{x}, \tag{4.17}$$

where x_k is the values of process x at time step k and Δt is the time increment. Injecting (4.16) into (4.17) yields,

$$x_{k+1} = x_k + \Delta t\left(-\lambda(x_k - x^* + \varepsilon_x)\right) = \left(1-\lambda\Delta t\right)x_k + \lambda\Delta t x^* - \lambda\Delta t\varepsilon_x . \tag{4.18}$$

The covariance of the process is defined as,

$$\operatorname{cov}(x_{k+1}, x_{k+1}) \equiv E\left\{\left(x_{k+1} - E(x_{k+1})\right)^2\right\}. \tag{4.19}$$

By assuming the reference input to be error free, the covariance of the process is calculated,

$$\begin{aligned}
\operatorname{cov}(x_{k+1}, x_{k+1}) &= E\left\{\left(\left(1-\lambda\Delta t\right)x_k + \lambda\Delta t x^* - \lambda\Delta t\varepsilon_x - E\left\{\left(1-\lambda\Delta t\right)x_k + \lambda\Delta t x^* - \lambda\Delta t\varepsilon_x\right\}\right)^2\right\} \\
&= E\left\{\left(\left(1-\lambda\Delta t\right)x_k + \lambda\Delta t x^* - \lambda\Delta t\varepsilon_x - \left(1-\lambda\Delta t\right)E\left\{x_k\right\} - \lambda\Delta t x^*\right)^2\right\} \\
&= E\left\{\left(\left(1-\lambda\Delta t\right)\left(x_k - E\left\{x_k\right\}\right) - \lambda\Delta t\varepsilon_x\right)^2\right\} \\
&= \left(1-\lambda\Delta t\right)^2 E\left\{\left(x_k - E\left\{x_k\right\}\right)^2\right\} - \left(\lambda\Delta t\right)^2 E\left\{\left(\varepsilon_x\right)^2\right\} \\
&= \left(1-\lambda\Delta t\right)^2 \operatorname{cov}(x_k, x_k) + \left(\lambda\Delta t\right)^2 \sigma^2.
\end{aligned} \tag{4.20}$$

Here σ^2 is the covariance of the estimation error. It is worth mentioning that the current noise is independent of x_k. Now if the covariance of process x is modeled with a time varying function, it can be shown that,

$$\operatorname{cov}(x_k, x_k) = f_c(t). \tag{4.21}$$

The time derivation of (4.21) results in,

$$\begin{aligned}
\frac{\partial f_c}{\partial t} &= \frac{\partial\left(\operatorname{cov}(x_k, x_k)\right)}{\partial t} = \frac{\partial\left(E\left\{\left(x_k - E\left\{x_k\right\}\right)^2\right\}\right)}{\partial t} \\
&= \frac{E\left\{\partial\left(\left(x_k - E\left\{x_k\right\}\right)^2\right)\right\}}{\partial t} = E\left\{2\left(x_k - E\left\{x_k\right\}\right)\frac{\partial\left(x_k - E\left\{x_k\right\}\right)}{\partial t}\right\} \\
&= 2E\left[\left(x_k - E\left\{x_k\right\}\right)\left(\dot{x}_k - E\left\{\dot{x}_k\right\}\right)\right] = 2\operatorname{cov}(x_k, \dot{x}_k).
\end{aligned} \tag{4.22}$$

The covariance of next time step is calculated as follows using (4.17),

$$\begin{aligned}&\operatorname{cov}(x_{k+1}, x_{k+1}) = E\left\{\left(x_k + \Delta t\dot{x} - E\left\{x_k + \Delta t\dot{x}\right\}\right)^2\right\}\\&= E\left\{\left(x_k - E\left(x_k\right)\right)^2\right\} + 2\Delta t E\left\{\left(x_k - E\left(x_k\right)\right)\left(\dot{x} - E\left(\dot{x}\right)\right)\right\},\\&+\Delta t^2 E\left\{\left(\dot{x} - E\left(\dot{x}\right)\right)^2\right\} = f_c(t) + \frac{\partial f_c}{\partial t}\Delta t + \mathrm{O}(\Delta t^2)\end{aligned} \tag{4.23}$$

which is in agreement with Tylor expansion of function *f*. Here $\mathrm{O}(\Delta t^2)$ denotes terms of order Δt^2 or higher and are assumed to be negligible. Then (4.15) may be rewritten as,

$$f_c + \frac{\partial f_c}{\partial t}\Delta t \approx \left(1 - \lambda\Delta t\right)^2 f(t) + \lambda^2 \Delta t^2 \sigma^2 \tag{4.24}$$

which results in,

$$\frac{\partial f_c}{\partial t} \approx \left(-2\lambda + \lambda^2 \Delta t\right) f_c(t) + \lambda^2 \Delta t \sigma^2 . \tag{4.25}$$

The solution to this differential equation is as follows,

$$f_c(t) = \frac{\lambda \Delta t \sigma^2}{(2 - \lambda\Delta t)}\left(1 - e^{\left(\lambda^2 \Delta t - 2\lambda\right)t}\right). \tag{4.26}$$

It is interesting to note that this result is close to the covariance expected by using Ornstein-Uhlenbeck process (as was discussed in [4.11]) when time increment is small,

$$f_c(t) \approx \Delta t \frac{\lambda \sigma^2}{2}\left(1 - e^{-2\lambda t}\right). \tag{4.27}$$

However, the model shown in (4.26) is more accurate than the previous model of [4.11] in discrete-time systems as it accounts for the sampling time of the system. The steady-state error covariance could also be calculated by assuming,

$$\operatorname{cov}(X^+, X^+) = \operatorname{cov}(X^-, X^-) = f_{ss} . \tag{4.28}$$

Using this assumption with (4.25) will result in,

$$f_{ss} = \frac{\lambda \Delta t \sigma^2}{(2 - \lambda \Delta t)}, \tag{4.29}$$

which can also be obtained from (4.26).

Second-order controller

In a system with second-order controller, also known as PD controller, the error is reduced to zero through the following control law,

$$\ddot{e}_{x,k} + k_D \dot{e}_{x,k} + k_P e_{x,k} = 0, \tag{4.30}$$

where k_P and k_D are controller gains. Then using (4.7) and (4.10), the following equation is found,

$$\ddot{x} = -k_D(\dot{x}_k - \dot{x}^*) - k_P(x_k - x^*) - k_p \varepsilon_x + \ddot{x}^*. \tag{4.31}$$

Similar to before, the processes X and $\dot{X}$ are calculated as follows,

$$\dot{x}_{k+1} = \dot{x}_k + \Delta t \ddot{x}, \tag{4.32}$$

$$x_{k+1} = x_k + \Delta t \dot{x}_{k+1}. \tag{4.33}$$

Applying the same methodology as before and taking (4.31-4.33) into consideration yields,

$$x_{k+1} = \left(1 - k_P \Delta t^2\right) x_k + k_P \Delta t^2 x^* + k_D \Delta t^2 \dot{x}^* + \Delta t^2 \ddot{x}^* + \Delta t\left(1 - k_D \Delta t\right) \dot{x}_k - k_p \Delta t^2 \varepsilon_x, \tag{4.34}$$

$$\dot{x}_{k+1} = \left(1 - k_D \Delta t\right) \dot{x}_k - k_P \Delta t x_k + k_P \Delta t x^* + k_D \Delta t \dot{x}^* + \Delta t \ddot{x}^* - k_p \Delta t \varepsilon_x, \tag{4.35}$$

which yield (assuming the reference input to be constant and error-free),

$$\begin{aligned}\operatorname{cov}(x_{k+1},x_{k+1}) &= E\left\{\begin{pmatrix}\left(1-k_P\Delta t^2\right)x_k + k_P\Delta t^2 x^* + \Delta t\left(1-k_D\Delta t\right)\dot{x}_k - k_p\Delta t^2\varepsilon_x \\ -E\left\{\left(1-k_P\Delta t^2\right)x_k + k_P\Delta t^2 x^* + \Delta t\left(1-k_D\Delta t\right)\dot{x}_k - k_p\Delta t^2\varepsilon_x\right\}\end{pmatrix}^2\right\} \\ &= E\left\{\left(\left(1-k_P\Delta t^2\right)\left(x_k - E\{x_k\}\right) + \Delta t\left(1-k_D\Delta t\right)\left(\dot{x}_k - E\{\dot{x}_k\}\right) - k_p\Delta t^2\varepsilon_x\right)^2\right\} \\ &= \left(1-k_P\Delta t^2\right)^2\operatorname{cov}(x_k,x_k) + \Delta t^2(1-k_D\Delta t)^2\operatorname{cov}(\dot{x}_k,\dot{x}_k) \\ &\quad + 2\Delta t(1-k_D\Delta t)\left(1-k_P\Delta t^2\right)\operatorname{cov}(x_k,\dot{x}_k) + k_P{}^2\Delta t^4\sigma^2,\end{aligned} \tag{4.36}$$

$$\begin{aligned}\operatorname{cov}(\dot{x}_{k+1},\dot{x}_{k+1}) &= E\left\{\begin{pmatrix}\left(1-k_D\Delta t\right)\dot{x}_k - k_P\Delta t x_k + k_P\Delta t x^* - k_p\Delta t\varepsilon_x \\ -E\left\{\left(1-k_D\Delta t\right)\dot{x}_k - k_P\Delta t x_k + k_P\Delta t x^* - k_p\Delta t\varepsilon_x\right\}\end{pmatrix}^2\right\} \\ &= E\left\{\left(\left(1-k_D\Delta t\right)\left(\dot{x}_k - E\{\dot{x}_k\}\right) - k_P\Delta t\left(x_k - E\{x_k\}\right) - k_p\Delta t\varepsilon_x\right)^2\right\} \\ &= (1-k_D\Delta t)^2\operatorname{cov}(\dot{x}_k,\dot{x}_k) + \left(k_P\Delta t\right)^2\operatorname{cov}(x_k,x_k) \\ &\quad - 2k_P\Delta t(1-k_D\Delta t)\operatorname{cov}(x_k,\dot{x}_k) + k_P{}^2\Delta t^2\sigma^2.\end{aligned} \tag{4.37}$$

$$\begin{aligned}\operatorname{cov}(x_{k+1},\dot{x}_{k+1}) &= E\left\{\begin{pmatrix}\left(1-k_D\Delta t\right)\dot{x}_k - k_P\Delta t x_k + k_P\Delta t x^* - k_p\Delta t\varepsilon_x \\ -E\left\{\left(1-k_D\Delta t\right)\dot{x}_k^- - k_P\Delta t x_k + k_P\Delta t x^* - k_p\Delta t\varepsilon_x\right\}\end{pmatrix}^2\right\} \\ &= E\left\{\begin{matrix}\left(\left(1-k_P\Delta t^2\right)\left(x_k - E\{x_k\}\right) + \Delta t\left(1-k_D\Delta t\right)\left(\dot{x}_k - E\{\dot{x}_k\}\right) - k_p\Delta t^2\varepsilon_x\right) \\ \left(\left(1-k_D\Delta t\right)\left(\dot{x}_k - E\{\dot{x}_k\}\right) - k_P\Delta t\left(x_k - E\{x_k\}\right) - k_p\Delta t\varepsilon_x\right)\end{matrix}\right\} \\ &= -k_P\Delta t\left(1-k_P\Delta t^2\right)\operatorname{cov}(x_k,x_k) + \Delta t\left(1-k_D\Delta t\right)^2\operatorname{cov}(\dot{x}_k,\dot{x}_k) \\ &\quad + \left(1-k_D\Delta t\right)\left(1-2k_P\Delta t^2\right)\operatorname{cov}(x_k,\dot{x}_k) + k_P{}^2\Delta t^3\sigma^2.\end{aligned} \tag{4.38}$$

Now assuming,

$$f_{c1}(t) = \operatorname{cov}(x_k,x_k)\,, \tag{4.39}$$

$$f_{c2}(t) = \operatorname{cov}(\dot{x}_k,\dot{x}_k)\,, \tag{4.40}$$

$$f_{c3}(t) = \operatorname{cov}(x_k,\dot{x}_k), \tag{4.41}$$

and using (4.22), (4.36)-(4.38) are rewritten as follows,

$$
\begin{aligned}
f_{c1} + \frac{\partial f_{c1}}{\partial t}\Delta t = \left(1 - k_P \Delta t^2\right)^2 f_{c1} + \Delta t^2 (1 - k_D \Delta t)^2 f_{c2} \\
+ 2\Delta t (1 - k_D \Delta t)\left(1 - k_P \Delta t^2\right) f_{c3} + k_P{}^2 \Delta t^4 \sigma^2,
\end{aligned} \tag{4.42}
$$

$$
f_{c2} + \frac{\partial f_{c2}}{\partial t}\Delta t = (1 - k_D \Delta t)^2 f_{c2} + \left(k_P \Delta t\right)^2 f_{c1} - 2k_P \Delta t (1 - k_D \Delta t) f_{c3} + k_P{}^2 \Delta t^2 \sigma^2. \tag{4.43}
$$

$$
\begin{aligned}
f_{c3} + \frac{\partial f_{c3}}{\partial t}\Delta t = -k_P \Delta t\left(1 - k_P \Delta t^2\right) f_{c1} + \Delta t\left(1 - k_D \Delta t\right)^2 f_2 \\
+ \left(1 - k_D \Delta t\right)\left(1 - 2k_P \Delta t^2\right) f_{c3} + k_P{}^2 \Delta t^3 \sigma^2.
\end{aligned} \tag{4.44}
$$

The covariance of process X is obtained through solving (4.42)-(4.44) as follows,

$$
f_{c1}(t) = \frac{\left(2 - k_D \Delta t\right) k_P \Delta t \sigma^2}{4k_D}\left(1 - e^{-\frac{2k_P k_D}{2k_D + 1}t}\right). \tag{4.45}
$$

It is worth mentioning that such function cannot be calculated through the Ornstein-Uhlenbeck process, discussed in [4.11]. The steady-state error covariance is calculated through (4.42)-(4.44) by assuming,

$$
\frac{\partial f_{c1}}{\partial t} = \frac{\partial f_{c2}}{\partial t} = \frac{\partial f_{c3}}{\partial t} = 0, \tag{4.46}
$$

which yields,

$$
\operatorname{cov}(X, X)_{ss} = \frac{\left(2 - k_D \Delta t\right) k_P \Delta t \sigma^2}{4k_D}. \tag{4.47}
$$

The error covariance of closed-loop systems with other controllers is calculated similarly. In the next section the application of the proposed error modeling in visual servoing systems is presented.

4.4 Error Modeling in Visual Servoing

The focus of this section is the uncertainty propagation analysis in visual servoing applications. Similar to previous section, the open-loop error modeling is discussed first, followed by the novel closed-loop error modeling developed for classic visual servoing systems, namely IBVS and PBVS.

4.4.1 Open-Loop Approach

This subsection follows the open-loop error propagation in visual servoing systems. First, the image error propagation to camera velocities in an IBVS system is discussed. Next, the propagation of image noise through three different parts of the PBVS system, namely pose estimation, servoing, and proportional control, is investigated. The HVS analysis is similar and could be found in [4.7].

Image-Based Visual Servoing

In image-based visual servoing (IBVS) systems, the image error is directly propagated to the camera velocity. The camera velocity is directly calculated as follows,

$$V_c = -\lambda \hat{J}_s^{\dagger}(s - s^*)\,. \tag{4.48}$$

Here, $\hat{J}^{\dagger}$ is the approximate pseudo-inverse of the image Jacobian matrix, J, s and s^* are the current and desired image features vectors, respectively. Assuming the final feature points to be free of noise, one can define camera velocity noise as follows,

$$\varepsilon_{V_c} = \frac{\partial V_c}{\partial s}\varepsilon_s\,, \tag{4.49}$$

where ε_V and ε_s are the camera velocity noise and image noise respectively and,

$$\frac{\partial V_c}{\partial s} = -\lambda \frac{\partial \hat{J}_s^{\dagger}}{\partial s}(s - s^*) - \lambda \hat{J}_s^{\dagger}\,, \tag{4.50}$$

In calculation of (4.50), the derivation of inverse Jacobian with respect to image features is calculated as follows,

$$\frac{\partial J_s^{\dagger}}{\partial s} = -J_s^{\dagger}\frac{\partial J_s}{\partial s}J_s^{\dagger}. \tag{4.51}$$

It is interesting to note that when the camera is reaching its desired location, the first term in (4.50) will be negligible, simplifying the error as follows,

$$\varepsilon_V = -\lambda \hat{J}_s^{\dagger}\varepsilon_s. \tag{4.52}$$

It is noteworthy that the image noise is assumed to be a zero-mean Gaussian random variable.

Position-Based Visual Servoing

Pose estimation is the first part of PBVS to be investigated, which takes image feature points as input and gives the object pose as its output. The optimization-based pose estimation methods are based on minimizing the image plane error,

$$L_\omega = \sum_{i=1}^{n}\left[\left(u_i - \frac{\left[R_o^c P_i^o + t_o^c\right]_1}{\left[R_o^c P_i^o + t_o^c\right]_3}\right)^2 + \left(v_i - \frac{\left[R_o^c P_i^o + t_o^c\right]_2}{\left[R_o^c P_i^o + t_o^c\right]_3}\right)^2\right], \tag{4.53}$$

where u_i and v_i are the image plane coordinates of ith feature point, P_i^o is the ith point in object frame, R_o^c is the rotation matrix between the object and camera frames, t_o^c is the translation from the object frame to the camera frame, n is the number of feature points, and $[\cdot]_i$ is the ith element of the bracketed vector. The error is shown as a function of minimal pose representation (translation plus minimal orientation representation such as Euler angles),

$$L_\omega = L_\omega(s, \omega_o^c), \tag{4.54}$$

where s is the vector of image features, and ω_o^c is the minimal pose representation. Then s is the input of the pose estimation and ω_o^c is its output. The error function is minimized by putting the gradient equal to zero. The input and output are assumed to have additional error as before, i.e.,

$$\hat{s} = s + \varepsilon_s, . \tag{4.55}$$

$$\hat{\omega}_o^c = \omega_o^c + \varepsilon_{\omega_o^c} . \tag{4.56}$$

Then, the gradient of error is written as,

$$\frac{\partial L_\omega(s,\omega_o^c)}{\partial \omega_o^c} = \frac{\partial L_\omega(\hat{s},\hat{\omega}_o^c)}{\partial \hat{\omega}_o^c} - \frac{\partial^2 L_\omega(\hat{s},\hat{\omega}_o^c)}{\partial \omega_o^c \partial \hat{s}} \varepsilon_s - \frac{\partial^2 L_\omega(\hat{s},\hat{\omega}_o^c)}{\partial^2 \hat{\omega}_o^c} \varepsilon_{\omega_o^c} . \tag{4.57}$$

Since the gradient should be equal to zero both for real and estimated gradients, (4.57) is simplified to,

$$\frac{\partial^2 L_\omega(\hat{s},\hat{\omega}_o^c)}{\partial \hat{\omega}_o^c \partial \hat{s}} \varepsilon_s + \frac{\partial^2 L_\omega(\hat{s},\hat{\omega}_o^c)}{\partial^2 \hat{\omega}_o^c} \varepsilon_{\omega_o^c} = 0 , \tag{4.58}$$

which yields,

$$\varepsilon_{\omega_o^c} = -\left(\frac{\partial^2 L_\omega(\hat{s},\hat{\omega}_o^c)}{\partial^2 \hat{\omega}_o^c} \right)^{-1} \frac{\partial^2 L_\omega(\hat{s},\hat{\omega}_o^c)}{\partial \hat{\omega}_o^c \partial \hat{s}} \varepsilon_s \equiv G_P \varepsilon_s . \tag{4.59}$$

Then using (4.53), the covariance of the estimated pose is obtainable.

The goal in PBVS algorithms is to minimize the pose error between current and desired camera poses. This error is defined by forming homogenous transformation between the current and desired camera poses as follows,

$$H_{c^*}^c = \begin{bmatrix} R_{c^*}^c & t_{c^*}^c \\ 0 & 1 \end{bmatrix}, \tag{4.60}$$

where $H_{c^*}^c$ is the homogenous transformation between the desired camera frame (c^*) and the current camera frame (c), $R_{c^*}^c$ and $t_{c^*}^c$ are relevant rotation matrix and translation vector. In (4.60), $H_{c^*}^c$ is calculated as follows,

$$H_{c^*}^c = H_o^c H_{c^*}^o = \begin{bmatrix} R_o^c & t_o^c \\ 0 & 1 \end{bmatrix} \begin{bmatrix} R_{c^*}^o & t_{c^*}^o \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} R_o^c R_{c^*}^o & R_o^c t_{c^*}^o + t_o^c \\ 0 & 1 \end{bmatrix}. \tag{4.61}$$

The object frame is denoted by "o". A proportional control law is assumed to be used,

$$V_c = -\lambda \begin{pmatrix} t_{c^*}^c \\ \rho\vartheta \end{pmatrix}. \tag{4.62}$$

Here, $\rho\vartheta$ is the angle and axis representation of $R_{c^*}^c$ and V_c is the camera velocity. It is assumed that the desired pose is error free. Next, the error propagations for rotation and translation are followed separately.

$$\varepsilon_{t_{c^*}^c} = \varepsilon_{R_o^c} t_{c^*}^o + \varepsilon_{t_o^c} = G_t \varepsilon_{\pi_o^c} + \varepsilon_{t_o^c}. \tag{4.63}$$

The error of $\varepsilon_{t_o^c}$ is directly obtained from the pose estimation error propagation, while the error signal $\varepsilon_{R_o^c}$ can be calculated from the error of minimum representation of orientation (Euler angles) as follows,

$$\varepsilon_{\pi_o^c} \approx \frac{\partial r_o^c}{\partial \varphi_o^c} \varepsilon_{\varphi_o^c} + \frac{\partial r_o^c}{\partial \theta_o^c} \varepsilon_{\theta_o^c} + \frac{\partial r_o^c}{\partial \psi_o^c} \varepsilon_{\psi_o^c} = G_\phi \varepsilon_{\phi_o^c}, \tag{4.64}$$

where $\varepsilon_{\pi_o^c}$ is the error of vector representation of rotation matrix, denoted by π_o^c . The Euler angles (roll, pitch, and yaw) are denoted by φ, θ, and ψ, respectively. The error of orientation is calculated as follows,

$$\varepsilon_{\pi_{c^*}^c} = G_\pi \varepsilon_{\pi_o^c}, \tag{4.65}$$

and then propagated to the angle and axis form [4.7],

$$\varepsilon_{\rho\vartheta} = G_{\rho\vartheta} \varepsilon_{\pi_{c^*}^c}. \tag{4.66}$$

Ultimately, the error propagated to camera velocity is obtained as follows,

$$\varepsilon_v = -\lambda \begin{pmatrix} \varepsilon_{t^c_{c^*}} \\ \varepsilon_{\rho\vartheta} \end{pmatrix} = -\lambda \begin{pmatrix} G_t \varepsilon_{\pi^c_o} + \varepsilon_{t^c_o} \\ G_{\rho\vartheta} G_\pi \varepsilon_{\pi^c_o} \end{pmatrix} = -\lambda \begin{pmatrix} I_3 & G_t G_\phi \\ 0 & G_{\rho\vartheta} G_\pi G_\phi \end{pmatrix} \begin{pmatrix} \varepsilon_{t^c_o} \\ \varepsilon_{\phi^c_o} \end{pmatrix} = -\lambda G_v \varepsilon_{\omega^c_o} . \tag{4.67}$$

4.4.2 Closed Loop Approach

Image-based Visual Servoing

In IBVS systems, the error is formed based on current and desired image feature locations. The error is then reduced to zero using a suitable control law, such as a proportional control law. For this matter, at each time step the camera velocity is chosen as,

$$V_{c,k} = -\lambda J_{s,k}{}^{\dagger} (\hat{s}_k - s^*) , \tag{4.68}$$

where s^* is the vector of desired image features and,

$$\hat{s}_k = s_k + \varepsilon_{s,k} , \tag{4.69}$$

is the vector of image features at time step *k*. The image features at the next time step are updated based on the given camera velocity,

$$s_{k+1} = s_k + \Delta t J_{s,k} V_{c,k} . \tag{4.70}$$

Then the covariance of image features in the next time step is calculated as,

$$\begin{aligned}
\operatorname{cov}(s_{k+1},s_{k+1}) &= E\left\{\left(s_{k+1}-E\left\{s_{k+1}\right\}\right)\left(s_{k+1}-E\left\{s_{k+1}\right\}\right)^T\right\} \\
&= E\left\{\begin{array}{l}\left(\begin{array}{l}\left(I_n-\lambda\Delta t J_{s,k}J_{s,k}{}^{\dagger}\right)s_k+\lambda\Delta t J_{s,k}J_{s,k}{}^{\dagger}s^*-\lambda\Delta t J_{s,k}J_{s,k}{}^{\dagger}\varepsilon_{s,k} \\ -E\left\{\left(I_n-\lambda\Delta t J_{s,k}J_k{}^{\dagger}\right)s_k+\lambda\Delta t J_{s,k}J_{s,k}{}^{\dagger}s^*-\lambda\Delta t J_{s,k}J_{s,k}{}^{\dagger}\varepsilon_{s,k}\right\}\end{array}\right) \\ \left(\begin{array}{l}\left(I_n-\lambda\Delta t J_{s,k}J_{s,k}{}^{\dagger}\right)s_k+\lambda\Delta t J_{s,k}J_{s,k}{}^{\dagger}s^*-\lambda\Delta t J_{s,k}J_{s,k}{}^{\dagger}\varepsilon_{s,k} \\ -E\left\{\left(I_n-\lambda\Delta t J_{s,k}J_{s,k}{}^{\dagger}\right)s_k+\lambda\Delta t J_{s,k}J_{s,k}{}^{\dagger}s^*-\lambda\Delta t J_{s,k}J_{s,k}{}^{\dagger}\varepsilon_{s,k}\right\}\end{array}\right)^T\end{array}\right\} \\
&= E\left\{\begin{array}{l}\left(\left(I_n-\lambda\Delta t J_{s,k}J_{s,k}{}^{\dagger}\right)\left(s_k-E\left\{s_k\right\}\right)-\lambda\Delta t J_{s,k}J_{s,k}{}^{\dagger}\varepsilon_{s,k}\right) \\ \left(\left(I_n-\lambda\Delta t J_{s,k}J_{s,k}{}^{\dagger}\right)\left(s_k-E\left\{s_k\right\}\right)-\lambda\Delta t J_{s,k}J_{s,k}{}^{\dagger}\varepsilon_{s,k}\right)^T\end{array}\right\} \\
&= \left(I_n-\lambda\Delta t J_{s,k}J_{s,k}{}^{\dagger}\right)\operatorname{cov}(s_k,s_k)\left(I_n-\lambda\Delta t J_{s,k}J_{s,k}{}^{\dagger}\right)^T \\
&\quad +\lambda^2\Delta t^2 J_{s,k}J_{s,k}{}^{\dagger}\Sigma_{s,k}J_{s,k}J_{s,k}{}^{\dagger},
\end{aligned} \tag{4.71}$$

where,

$$\Sigma_{s,k}=\operatorname{diag}\left\{\sigma_{s_1}{}^2 \quad . \quad . \quad \sigma_{s_n}{}^2\right\}, \tag{4.72}$$

is the matrix of feature error covariance. Now assuming,

$$\operatorname{cov}(s_k,s_k)=J_{s,k}J_{s,k}{}^{\dagger}F_{cs,k}J_{s,k}J_{s,k}{}^{\dagger}, \tag{4.73}$$

$$\operatorname{cov}(s_{k+1},s_{k+1})=J_{s,k}J_{s,k}{}^{\dagger}\left(F_{cs,k}+\frac{\partial F_{cs,k}}{\partial t}\Delta t\right)J_{s,k}J_{s,k}{}^{\dagger}. \tag{4.74}$$

One can rewrite (4.71) as follows,

$$\begin{aligned}
J_{s,k}J_{s,k}{}^{\dagger}\left(F_{cs,k}+\frac{\partial F_{cs,k}}{\partial t}\Delta t\right)J_{s,k}J_{s,k}{}^{\dagger} &= \lambda^2\Delta t^2 J_{s,k}J_{s,k}{}^{\dagger}\Sigma_{s,k}J_{s,k}J_{s,k}{}^{\dagger} \\
&\quad +\left(1-2\lambda\Delta t+\lambda^2\Delta t^2\right)J_{s,k}J_{s,k}{}^{\dagger}F_{cs,k}J_{s,k}J_{s,k}{}^{\dagger},
\end{aligned} \tag{4.75}$$

which yields,

$$\frac{\partial F_{cs,k}}{\partial t}=\lambda^2\Delta t\Sigma_{s,k}+\left(\lambda^2\Delta t-2\lambda\right)F_{cs,k}. \tag{4.76}$$

Similar to (4.25), (4.76) is solved as follows,

$$F_{cs,k} = \frac{\lambda \Delta t}{(2-\lambda \Delta t)}\left(1 - e^{\left(\lambda^2 \Delta t - 2\lambda\right)t}\right)\Sigma_{s,k} \,. \tag{4.77}$$

Replacing (4.77) in (4.73) results in,

$$\operatorname{cov}(s_k, s_k) = \frac{\lambda \Delta t}{(2-\lambda \Delta t)}\left(1 - e^{\left(\lambda^2 \Delta t - 2\lambda\right)t}\right) J_{s,k} J_{s,k}{}^{\dagger} \Sigma_{s,k} J_{s,k} J_{s,k}{}^{\dagger} \,. \tag{4.78}$$

The pose of the camera is updates as follows,

$$\omega_{c,k+1}^{w} = \omega_{c,k}^{w} + \Delta t J_{P,k} V_{c,k} \,, \tag{4.79}$$

where,

$$J_{p,k} = \begin{bmatrix} R_c^w & 0 \\ 0 & T^{-1} R_c^w \end{bmatrix}, \tag{4.80}$$

and $T(\phi)$ is the transformation between the Euler angles and angular velocity (i.e., $\omega = T(\phi)\dot{\phi}$). Replacing (4.68) in (4.79) yields,

$$\omega_{c,k+1}^{w} = \omega_{c,k}^{w} - \lambda \Delta t J_{P,k} J_{s,k}{}^{\dagger} (\hat{s}_k - s^*) \,. \tag{4.81}$$

The error covariance of pose in next time step is calculated as follows,

$$\begin{aligned}
\operatorname{cov}(\omega_{c,k+1}^{w},\omega_{c,k+1}^{w}) &= E\left\{\left(\omega_{c,k+1}^{w}-E\left\{\omega_{c,k+1}^{w}\right\}\right)\left(\omega_{c,k+1}^{w}-E\left\{\omega_{c,k+1}^{w}\right\}\right)^{T}\right\} \\
&= E\left\{\begin{matrix}\left(\omega_{c,k}^{w}-\lambda\Delta t J_{P,k}J_{s,k}{}^{\dagger}(\hat{s}_k-s^{*})-E\left\{\omega_{c,k}^{w}-\lambda\Delta t J_{P,k}J_{s,k}{}^{\dagger}(\hat{s}_k-s^{*})\right\}\right) \\ \left(\omega_{c,k}^{w}-\lambda\Delta t J_{P,k}J_{s,k}{}^{\dagger}(\hat{s}_k-s^{*})-E\left\{\omega_{c,k}^{w}-\lambda\Delta t J_{P,k}J_{s,k}{}^{\dagger}(\hat{s}_k-s^{*})\right\}\right)^{T}\end{matrix}\right\} \\
&= E\left\{\begin{matrix}\left(\omega_{c,k}^{w}-E\left\{\omega_{c,k}^{w}\right\}-\lambda\Delta t J_{P,k}J_{s,k}{}^{\dagger}\left(s_k-E\left\{s_k\right\}\right)-\lambda\Delta t J_{P,k}J_{s,k}{}^{\dagger}\varepsilon_s\right) \\ \left(\omega_{c,k}^{w}-E\left\{\omega_{c,k}^{w}\right\}-\lambda\Delta t J_{P,k}J_{s,k}{}^{\dagger}\left(s_k-E\left\{s_k\right\}\right)-\lambda\Delta t J_{P,k}J_{s,k}{}^{\dagger}\varepsilon_s\right)^{T}\end{matrix}\right\} \\
&= \operatorname{cov}(\omega_{c,k}^{w},\omega_{c,k}^{w})+\lambda^2\Delta t^2 J_{P,k}J_{s,k}{}^{\dagger}\operatorname{cov}(s_k,s_k)J_{s,k}{}^{\dagger T}J_{P,k}{}^{T} \\
&\quad +\lambda^2\Delta t^2 J_{P,k}J_{s,k}{}^{\dagger}\Sigma_{s,k}J_{s,k}{}^{\dagger T}J_{P,k}{}^{T}-\lambda\Delta t J_{P,k}J_{s,k}{}^{\dagger}\operatorname{cov}(s,\omega_{c,k}^{w}) \\
&\quad -\lambda\Delta t\operatorname{cov}(\omega_{c,k}^{w},s)J_{s,k}{}^{\dagger T}J_{P,k}{}^{T}.
\end{aligned} \tag{4.82}$$

The last two terms in (4.82) are obtained as follows,

$$\begin{aligned}
\operatorname{cov}(s_{k+1},\omega_{c,k+1}^{w}) &= E\left\{\left(s_{k+1}-E\left\{s_{k+1}\right\}\right)\left(\omega_{c,k+1}^{w}-E\left\{\omega_{c,k+1}^{w}\right\}\right)^{T}\right\} \\
&= E\left\{\begin{matrix}\left(\begin{matrix}\left(I_n-\lambda\Delta t J_{s,k}J_{s,k}{}^{\dagger}\right)s_k+\lambda\Delta t J_{s,k}J_{s,k}{}^{\dagger}s^{*}-\lambda\Delta t J_{s,k}J_{s,k}{}^{\dagger}\varepsilon_{s,k} \\ -E\left\{\left(I_n-\lambda\Delta t J_{s,k}J_{s,k}{}^{\dagger}\right)s_k+\lambda\Delta t J_{s,k}J_{s,k}{}^{\dagger}s^{*}-\lambda\Delta t J_{s,k}J_{s,k}{}^{\dagger}\varepsilon_{s,k}\right\}\end{matrix}\right) \\ \left(\omega_{c,k}^{w}-\lambda\Delta t J_{P,k}J_{s,k}{}^{\dagger}(\hat{s}_k-s^{*})-E\left\{\omega_{c,k}^{w}-\lambda\Delta t J_{P,k}J_{s,k}{}^{\dagger}(\hat{s}_k-s^{*})\right\}\right)^{T}\end{matrix}\right\} \\
&= E\left\{\begin{matrix}\left(\left(I_n-\lambda\Delta t J_{s,k}J_{s,k}{}^{\dagger}\right)\left(s_k-E\left\{s_k\right\}\right)-\lambda\Delta t J_{s,k}J_{s,k}{}^{\dagger}\varepsilon_{s,k}\right) \\ \left(\omega_{c,k}^{w}-E\left\{\omega_{c,k}^{w}\right\}-\lambda\Delta t J_{P,k}J_{s,k}{}^{\dagger}\left(s_k-E\left\{s_k\right\}\right)-\lambda\Delta t J_{P,k}J_{s,k}{}^{\dagger}\varepsilon_s\right)^{T}\end{matrix}\right\} \\
&= \left(I_n-\lambda\Delta t J_{s,k}J_{s,k}{}^{\dagger}\right)\operatorname{cov}(s_k,\omega_{c,k}^{w}) \\
&\quad -\lambda\Delta t\left(I_n-\lambda\Delta t J_{s,k}J_{s,k}{}^{\dagger}\right)\operatorname{cov}(s_k,s_k)J_{s,k}{}^{\dagger T}J_{P,k}{}^{T} \\
&\quad +\lambda^2\Delta t^2 J_{s,k}J_{s,k}{}^{\dagger}\Sigma_{s,k}J_{s,k}{}^{\dagger T}J_{P,k}{}^{T}.
\end{aligned} \tag{4.83}$$

Now assuming,

$$\operatorname{cov}(s_k,\omega_{c,k}^{w})=J_{s,k}J_{s,k}{}^{\dagger}G_{c,k}J_{s,k}{}^{\dagger T}J_{P,k}{}^{T}, \tag{4.84}$$

the covariance of pose and image features is calculated as follows,

$$J_{s,k}J_{s,k}{}^{\dagger}\left(G_{c,k}+\frac{\partial G_{c,k}}{\partial t}\Delta t\right)J_{s,k}{}^{\dagger T}J_{P,k}{}^{T}=\left(I_n-\lambda\Delta t J_{s,k}J_{s,k}{}^{\dagger}\right)J_{s,k}J_{s,k}{}^{\dagger}G_{c,k}J_{s,k}{}^{\dagger T}J_{P,k}{}^{T}$$
$$-\frac{\lambda^2\Delta t^2\left(1-\lambda\Delta t\right)}{(2-\lambda\Delta t)}\left(1-e^{\left(\lambda^2\Delta t-2\lambda\right)t}\right)J_{s,k}J_{s,k}{}^{\dagger}\Sigma_{s,k}J_{s,k}{}^{\dagger T}J_{P,k}{}^{T}$$
$$+\lambda^2\Delta t^2 J_{s,k}J_{s,k}{}^{\dagger}\Sigma_{s,k}J_{s,k}{}^{\dagger T}J_{P,k}{}^{T}, \tag{4.85}$$

which yields,

$$\lambda G_{c,k}+\frac{\partial G_{c,k}}{\partial t}=\frac{\lambda^3\Delta t^2-\lambda^2\Delta t}{(2-\lambda\Delta t)}\left(1-e^{\left(\lambda^2\Delta t-2\lambda\right)t}\right)\Sigma_{s,k}+\lambda^2\Delta t\Sigma_{s,k}. \tag{4.86}$$

Then the sought covariance is found by solving (4.86),

$$\operatorname{cov}(s_k,\omega^w_{c,k})=\frac{\lambda\Delta t}{(2-\lambda\Delta t)}\left(1-e^{\left(\lambda^2\Delta t-2\lambda\right)t}\right)J_{s,k}J_{s,k}{}^{\dagger}\Sigma_{s,k}J_{s,k}{}^{\dagger T}J_{P,k}{}^{T}. \tag{4.87}$$

It is easy to show that,

$$\operatorname{cov}(\omega^w_{c,k},s_k)=\operatorname{cov}(s_k,\omega^w_{c,k})^T=\frac{\lambda\Delta t}{(2-\lambda\Delta t)}\left(1-e^{\left(\lambda^2\Delta t-2\lambda\right)t}\right)J_{P,k}J_{s,k}{}^{\dagger}\Sigma_{s,k}J_{s,k}{}^{\dagger T}J_{s,k}{}^{T} \tag{4.88}$$

Now replacing (4.78), (4.87), and (4.88) in (4.82) results in,

$$\begin{aligned}
&\operatorname{cov}(\omega^w_{c,k+1},\omega^w_{c,k+1})= \\
&\quad\operatorname{cov}(\omega^w_{c,k},\omega^w_{c,k})+\frac{\lambda^3\Delta t^3}{(2-\lambda\Delta t)}\left(1-e^{\left(\lambda^2\Delta t-2\lambda\right)t}\right)J_{P,k}J_{s,k}{}^{\dagger}\Sigma_{s,k}J_{s,k}{}^{\dagger T}J_{P,k}{}^{T} \\
&\quad+\lambda^2\Delta t^2 J_{P,k}J_k{}^{\dagger}\Sigma_{s,k}J_k{}^{\dagger T}J_{P,k}{}^{T}-\frac{\lambda^2\Delta t^2}{(2-\lambda\Delta t)}\left(1-e^{\left(\lambda^2\Delta t-2\lambda\right)t}\right)J_{P,k}J_k{}^{\dagger}\Sigma_{s,k}J_k{}^{\dagger T}J_{P,k}{}^{T} \\
&\quad-\frac{\lambda^2\Delta t^2}{(2-\lambda\Delta t)}\left(1-e^{\left(\lambda^2\Delta t-2\lambda\right)t}\right)J_{P,k}J_{s,k}{}^{\dagger}\Sigma_{s,k}J_{s,k}{}^{\dagger T}J_{P,k}{}^{T} \\
&\qquad=\operatorname{cov}(\omega^w_{c,k},\omega^w_{c,k})+\lambda^2\Delta t^2 e^{\left(\lambda^2\Delta t-2\lambda\right)t}J_{P,k}J_{s,k}{}^{\dagger}\Sigma_{s,k}J_{s,k}{}^{\dagger T}J_{P,k}{}^{T}.
\end{aligned} \tag{4.89}$$

Similar to previous cases, (4.89) is solved by assuming,

$$\mathrm{cov}(\omega^{w}_{c,k},\omega^{w}_{c,k}) = J_{P,k}J_{s,k}{}^{\dagger}M_{c,k}J_{s,k}{}^{\dagger T}J_{P,k}{}^{T}, \tag{4.90}$$

as follows,

$$\frac{\partial M_{c,k}}{\partial t} = \lambda^{2}\Delta t e^{\left(\lambda^{2}\Delta t-2\lambda\right)t}\Sigma_{s,k}, \tag{4.91}$$

which is solved through integration, knowing that $M_{c,0} = 0$,

$$M_{c,k} = \frac{\lambda\Delta t}{2-\lambda\Delta t}(1-e^{\left(\lambda^{2}\Delta t-2\lambda\right)t})\Sigma_{s,k}. \tag{4.92}$$

Therefore, the pose error covariance is formulated as follows,

$$\mathrm{cov}(\omega^{w}_{c,k},\omega^{w}_{c,k}) = \frac{\lambda\Delta t}{2-\lambda\Delta t}(1-e^{\left(\lambda^{2}\Delta t-2\lambda\right)t})J_{P,k}J_{s,k}{}^{\dagger}\Sigma_{s,k}J_{s,k}{}^{\dagger T}J_{P,k}{}^{T}. \tag{4.93}$$

The velocity error covariance is found through (4.68), i.e.,

$$\begin{aligned}
\mathrm{cov}(V_{c,k},V_{c,k}) &= E\left\{\begin{matrix}\left(-\lambda J_{s,k}{}^{\dagger}(\hat{s}_k - s^{*}) - E\left\{-\lambda J_{s,k}{}^{\dagger}(\hat{s}_k - s^{*})\right\}\right) \\ \left(-\lambda J_{s,k}{}^{\dagger}(\hat{s}_k - s^{*}) - E\left\{-\lambda J_{s,k}{}^{\dagger}(\hat{s}_k - s^{*})\right\}\right)^{T}\end{matrix}\right\} \\
&= E\left\{\left(-\lambda J_{s,k}{}^{\dagger}\left(s_k - E\left\{s_k\right\}\right) - \lambda J_{s,k}{}^{\dagger}\varepsilon_{s,k}\right)\left(-\lambda J_{s,k}{}^{\dagger}\left(s_k - E\left\{s_k\right\}\right) - \lambda J_{s,k}{}^{\dagger}\varepsilon_{s,k}\right)^{T}\right\} \\
&= \lambda^{2}J_{s,k}{}^{\dagger}\,\mathrm{cov}(s_k,s_k)J_{s,k}{}^{\dagger T} + \lambda^{2}J_{s,k}{}^{\dagger}\Sigma_{s,k}J_{s,k}{}^{\dagger T},
\end{aligned} \tag{4.94}$$

and replacing (4.78) in (4.94),

$$\mathrm{cov}(V_{c,k},V_{c,k}) = \left(\frac{\lambda^{3}\Delta t}{(2-\lambda\Delta t)}\left(1-e^{\left(\lambda^{2}\Delta t-2\lambda\right)t}\right)+\lambda^{2}\right)J_{s,k}{}^{\dagger}\Sigma_{s,k}J_{s,k}{}^{\dagger T}. \tag{4.95}$$

Position-based Visual Servoing

In PBVS, the camera velocity is obtained based on pose difference between current and desire pose, as was stated in (4.62). The pose is then updated based on the obtained camera velocity, similar to (4.79). If the pose is defined as follows,

$$\omega = \begin{bmatrix} t_c^{c^*} & \rho\vartheta \end{bmatrix}, \qquad (4.96)$$

then the desired pose is equal to zero (i.e., $\omega^* = 0$). This pose is updated with the camera velocity,

$$\omega_{k+1} = \omega_k - \lambda \Delta t \hat{\omega}_k = (1 - \lambda \Delta t)\omega_k - \lambda \Delta t \varepsilon_{\omega_k},$$

where,

$$\varepsilon_{\omega_k} = G_v \varepsilon_{\omega_o^c} = G_v G_P \varepsilon_s, \qquad (4.97)$$

similar to (4.67). The covariance of the pose is calculated as follows,

$$\begin{aligned}
\operatorname{cov}(\omega_{k+1}, \omega_{k+1}) &= E\left\{\left(\omega_{k+1} - E\{\omega_{k+1}\}\right)\left(\omega_{k+1} - E\{\omega_{k+1}\}\right)^T\right\} \\
&= E\left\{\begin{array}{l} \left((1-\lambda\Delta t)\omega_k - \lambda\Delta t \varepsilon_{\omega_k} - E\left\{(1-\lambda\Delta t)\omega_k - \lambda\Delta t \varepsilon_{\omega_k}\right\}\right) \\ \left((1-\lambda\Delta t)\omega_k - \lambda\Delta t \varepsilon_{\omega_k} - E\left\{(1-\lambda\Delta t)\omega_k - \lambda\Delta t \varepsilon_{\omega_k}\right\}\right)^T \end{array}\right\} \\
&= E\left\{\left((1-\lambda\Delta t)\left(\omega_k - E\{\omega_k\}\right) - \lambda\Delta t \varepsilon_{\omega_k}\right)\left((1-\lambda\Delta t)\left(\omega_k - E\{\omega_k\}\right) - \lambda\Delta t \varepsilon_{\omega_k}\right)^T\right\} \\
&= (1-\lambda\Delta t)^2 \operatorname{cov}(\omega_k, \omega_k) + \lambda^2 \Delta t^2 G_v G_P \Sigma_{s,k} G_P{}^T G_v{}^T,
\end{aligned} \qquad (4.98)$$

which is solved similar to (4.71), i.e,

$$\operatorname{cov}(\omega_k, \omega_k) = \frac{\lambda \Delta t}{(2 - \lambda \Delta t)}\left(1 - e^{\left(\lambda^2 \Delta t - 2\lambda\right)t}\right) G_v G_P \Sigma_{s,k} G_P{}^T G_v{}^T. \qquad (4.99)$$

Finally, it is worth mentioning that error covariance of other pose definitions (e.g., ω_c^w) is easily obtainable from the error of the pose defined in (4.96),

$$\varepsilon_{\rho\vartheta} = G_{\rho\vartheta} G_\pi \varepsilon_{\pi_o^c} = G_{\rho\vartheta} G'_\pi \varepsilon_{\pi_c^w}, \qquad (4.100)$$

$$\varepsilon_{t_c^w} = R_{c^*}^w \varepsilon_{t_c^{c^*}}. \qquad (4.101)$$

4.5 Simulation Results

The efficiency of the proposed methods for error covariance calculation is verified through multiple simulations. These simulations have been carried under MATLAB® 2011b environment from Mathworks (Natick, MA, USA). The simulations are based on Monte Carlo simulations with large number of samples (10,000 samples). In these simulations, the image noise covariance is assumed to be known. The closed-loop error covariance is calculated based on this image noise covariance.

In the first simulation, the validity of the proposed method for a closed-loop SISO system is evaluated. Two systems with proportional and second-order controllers are simulated by adding Gaussian noise to the input process (x) with the covariance of 0.01. A time step of 1 ms is considered for these systems. The resultant error covariance of the process is compared with the predictions, formulated in this chapter. Figures 4.2 and 4.3 show the results of these comparisons. As it can be seen, the predictions closely match the simulation results in the case of proportional controller, while the predictions from [4.11] are only approximating the results. The results also match closely for the second-order system, where the previous method was not able to make any predictions. It should be noted that the stability of the controllers are not of concern in this work and they are assumed to be stable (by selecting proper gains).

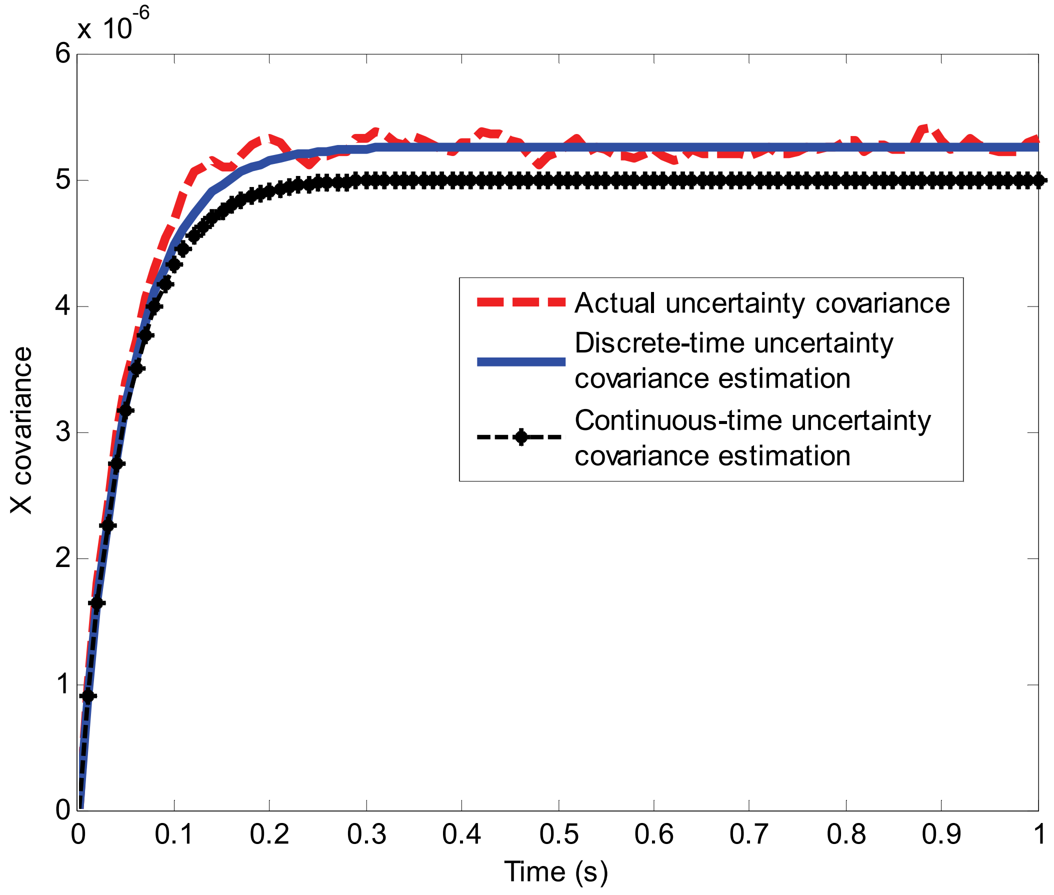


Figure 4.2 First simulation: Process error covariance using a proportional controller.

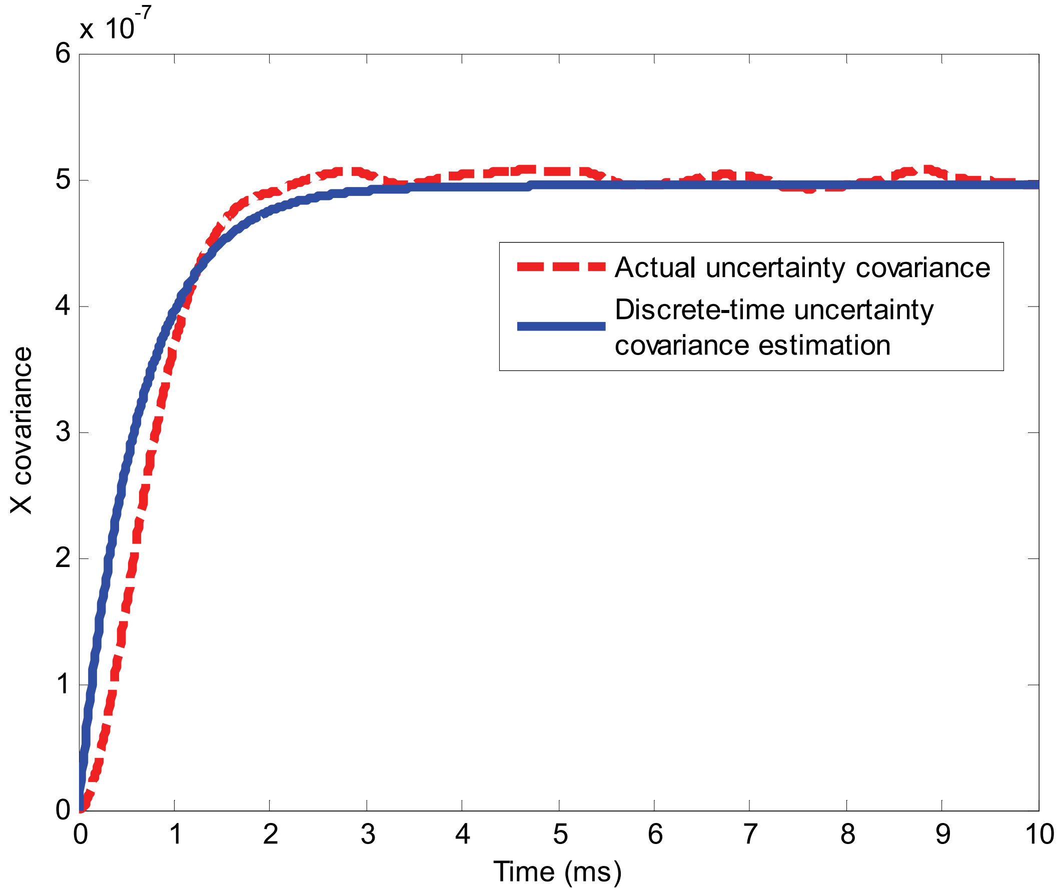


Figure 4.3 First simulation: Process error covariance using proportional derivative (second-order) controller.

In the second simulation, a closed-loop IBVS is tested to verify the efficiency of the proposed method for an IBVS system. An IBVS system with proportional controller is simulated in MATLAB® 2011b environment. The camera is servoed from a distance to half a meter above the object of interest. Four coplanar feature points have been exploited for this purpose. Figures 4.4-4.6 show the simulation results in image space, in Cartesian space, and for camera velocities. In image space, u and v are the image coordinates in pixel. The initial image is shown by dashed lines and the initial points are shown by circles. The final feature points are shown by crosses. The trajectories of the feature points are shown by solid lines. In Cartesian space, the orientations of the camera are shown by their frame coordinates for its initial and final poses and the camera trajectory is shown by a solid curve. The closed-loop image features error covariance is then obtained based on (4.78) and are compared to those from the simulations. The diagonal elements of the error covariance matrices are selected for sake of brevity. The result of this comparison is shown in Figure 4.7. As it is apparent, there is a close match between the simulations and the predictions throughout the simulation period. Similarly, the error covariance of the camera poses with respect to the object and the error covariance of the camera velocities are calculated and compared with their predictions through (4.93) and (4.95). Once again, the diagonal elements of the error covariance matrices are chosen. The outcomes of these comparisons are presented in Figures 4.8 and 4.9. Once again, the close estimation of the error covariance for camera pose and velocity is observed.

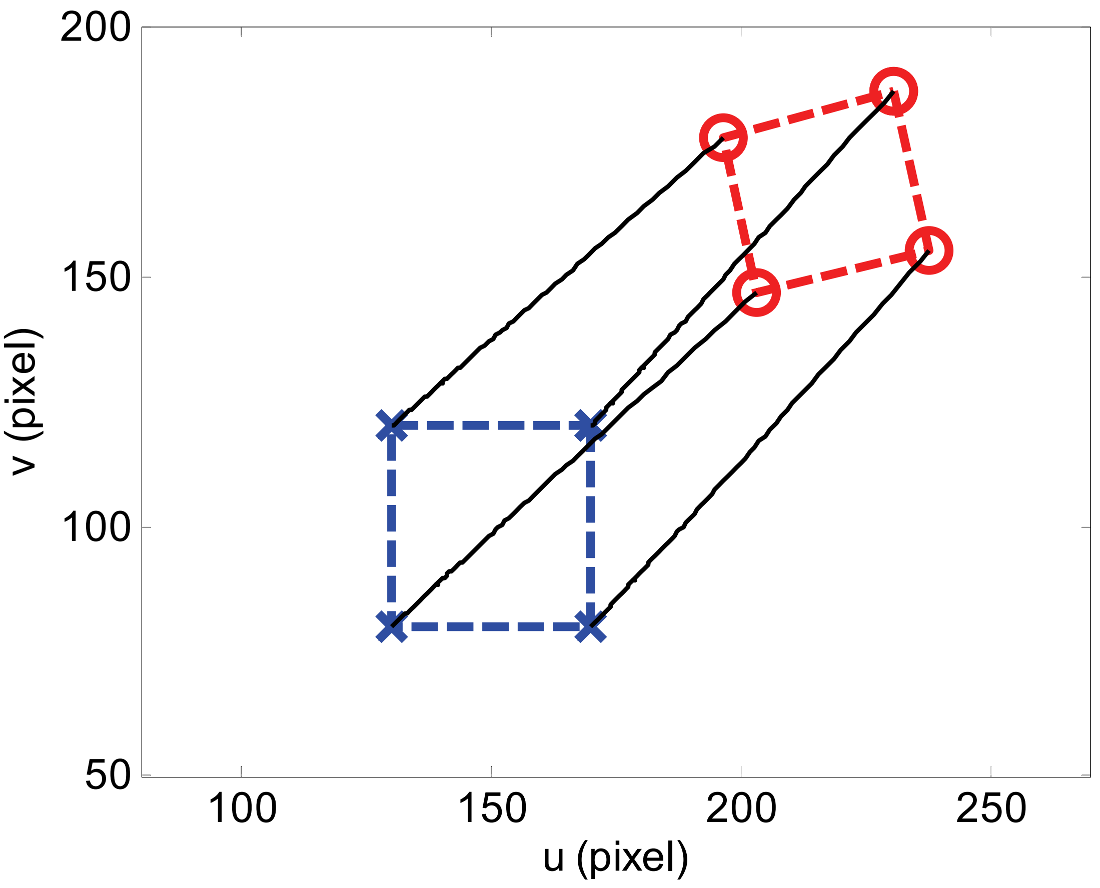


Figure 4.4 Second simulation: Simulated IBVS method in image space.

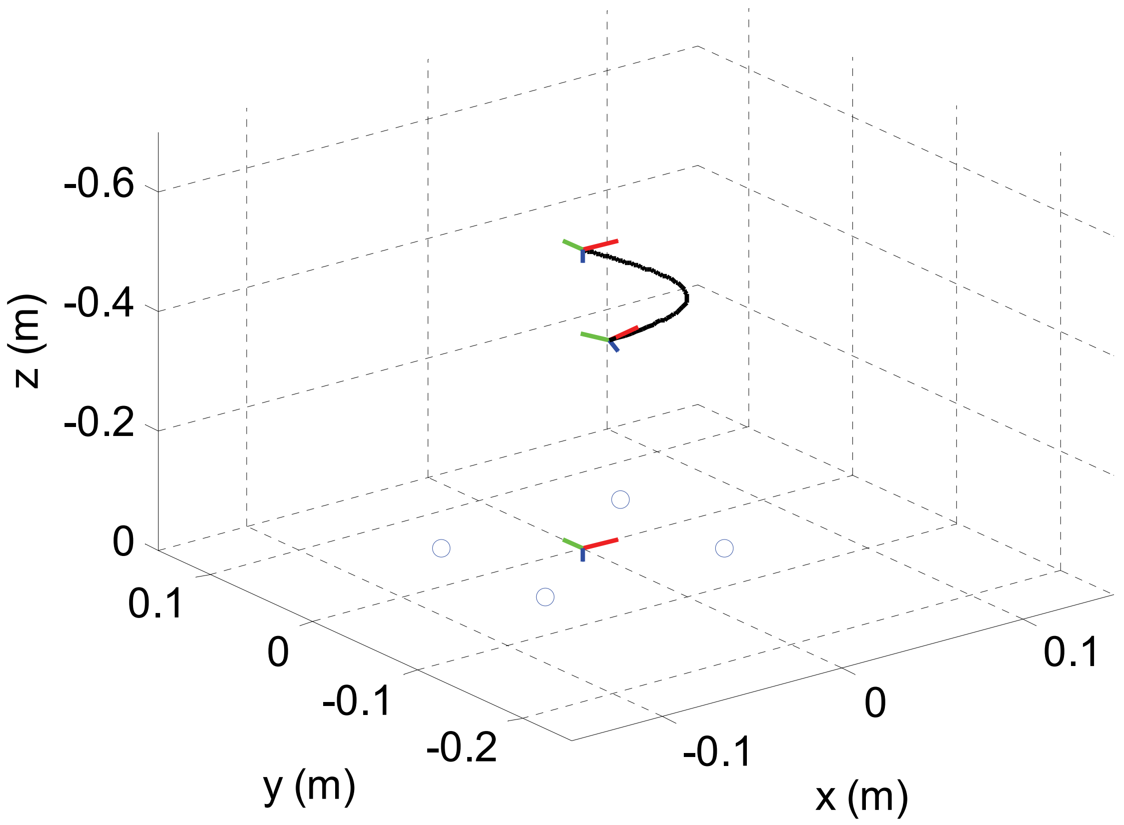


Figure 4.5 Second simulation: Simulated IBVS method in Cartesian space.

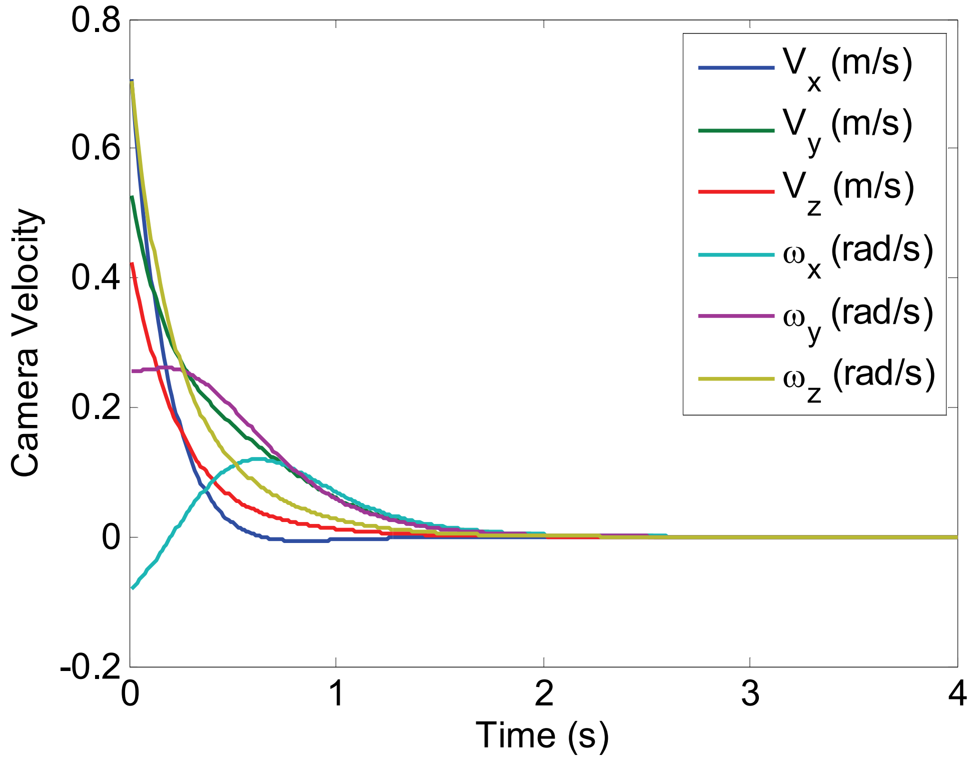


Figure 4.6 Second simulation: Camera velocities in simulated IBVS method.

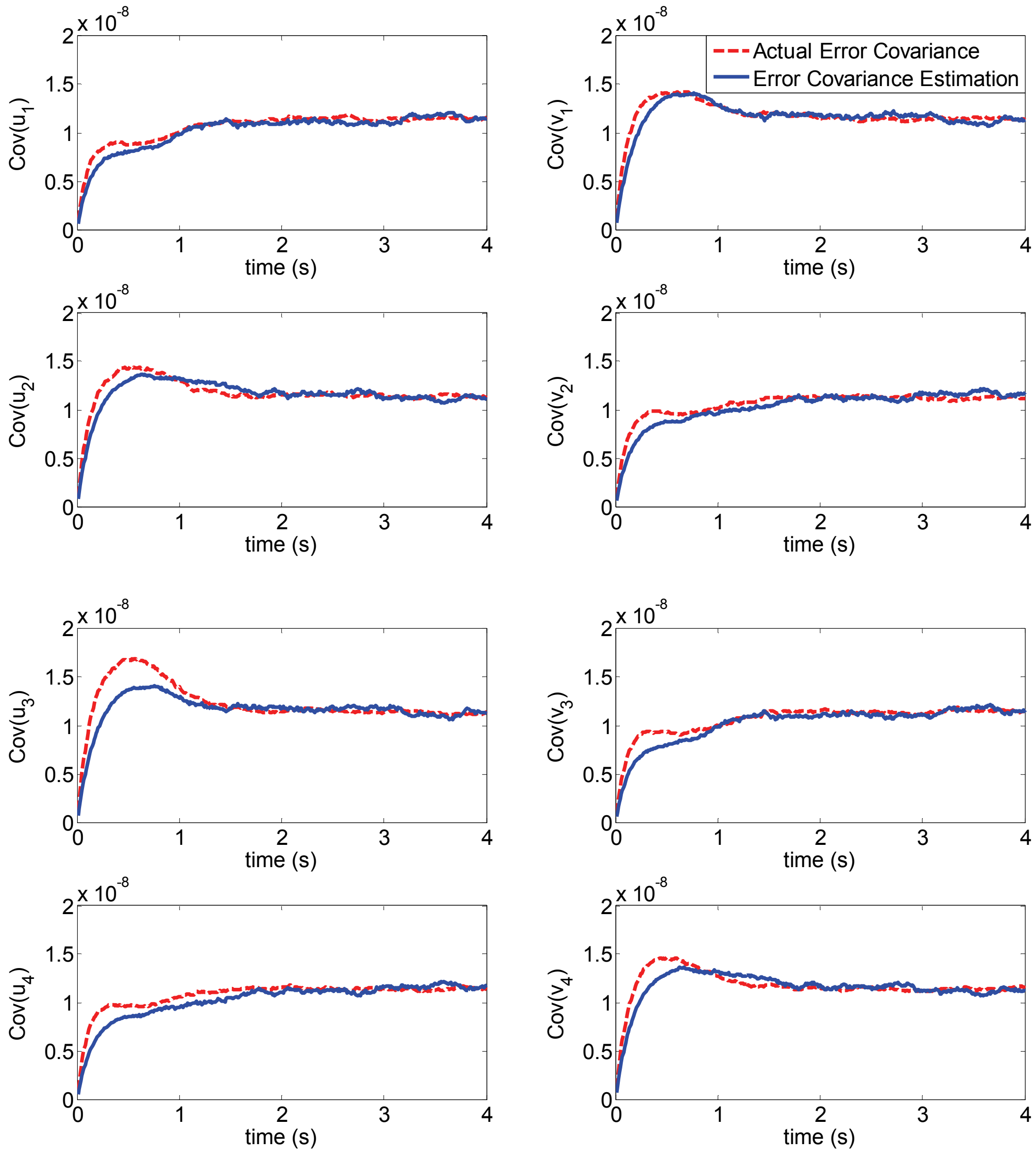


Figure 4.7 Second simulation: Image features error covariance in comparison with its estimation.

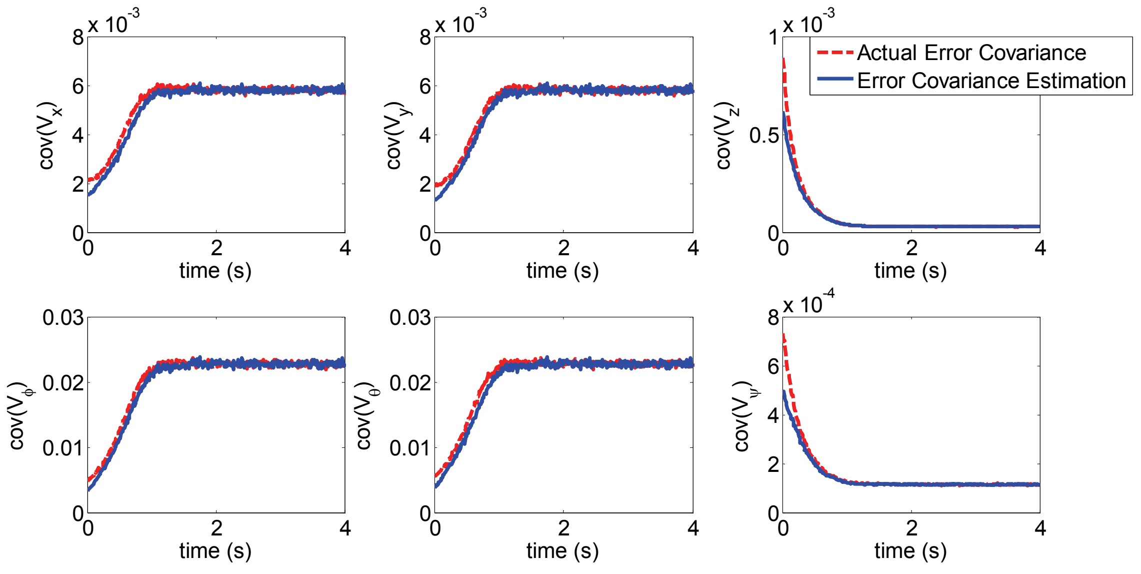


Figure 4.8 Second simulation: Camera velocity error covariance in comparison with its estimation.

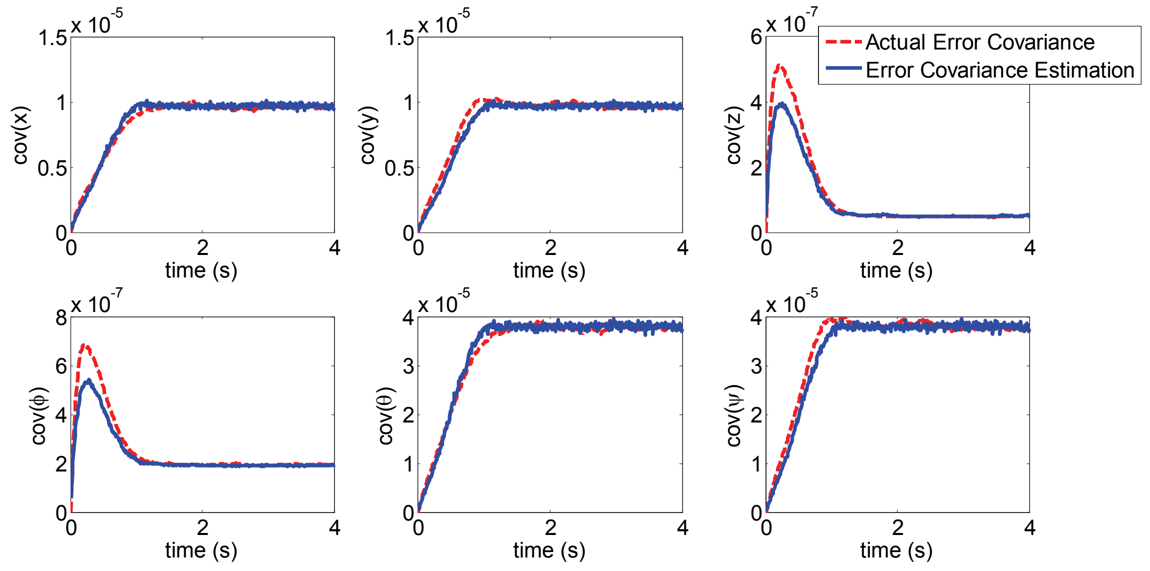


Figure 4.9 Second simulation: Camera pose error covariance in comparison with its estimation.

4.6 Summary

In this chapter, the uncertainty modeling in visual servoing systems was developed and a novel method was introduced to model the closed-loop uncertainty in visual servoing systems. The proposed method accounted for discrete-time nature of visual servoing systems and could entail more control laws than previous works. The effectiveness of the method was then proven by simulations. The method was applied to IBVS and PBVS systems and the error covariance of image features, camera velocities, and camera pose were found. The simulation results were presented to certify the accuracy of error covariance estimations. As was shown, there was a good match between the actual error covariance and its estimation.

The development of uncertainty models can help the designer of visual servoing systems to have a good understanding of the effect of uncertainties on different parts of the system and improve the systems accuracy and robustness with respect to this knowledge. In next chapter, robust controllers based on the developed error modeling are presented to enhance the accuracy of the system and add to its robustness.

Chapter 5

Robust and Constrained Visual Servo Control

5.1 Introduction

The practicality of any RVS approach for real world applications highly depends on its robust and accurate performance in presence of system uncertainties and constraints. The limitations imposed by the robot or the camera are usually known a priori and therefore can be avoided in RVS systems. However, system uncertainties are usually unknown and cannot be measured. Thus, robust schemes are required to alleviate the effect of uncertainties on system performance. Previously, path planning methods and image-based predictive control (IBPC) schemes were proposed to address the constraints in RVS systems. However, IBPC had only local stability and its numerical feasibility was not certified. Moreover, those methods were prone to system uncertainties. Robust methods were previously proposed as a remedy to this problem. Yet, most of those robust methods focused on robust convergence of the system and accuracy of the system was not considered. In addition, those approaches were either deterministic and/or conservative (e.g., path planning), or negligent of system constraints (e.g., robust controllers).

In this chapter, a set of controllers are introduced to address the aforementioned shortcomings. In particular, position-based predictive control (PBPC) and hybrid predictive control (HPC) methods are proposed as alternative solutions to IBPC. These controllers can manage the system constraints properly, while providing the system with global stability and flexibility in the design. The feasibility of the proposed controllers is improved by minimizing their computational cost and the subsequent stability issues are compensated through proper modifications of the cost function and the constraints. In addition, a two-step control scheme is proposed for robust control. By using this scheme, the rate of convergence is decoupled from uncertainty measures (namely mean and covariance of the error). Thus unlike previous controllers, the effect of uncertainties may be minimized without affecting the convergence rate. This methodology is then exploited in conjunction with the developed uncertainty propagation model and predictive controllers to robustly guide the robot towards its destination, without any constraint violation.

The chapter is organized as follows. First, a survey of previous works is presented in Sec. 5.2 and most related works are reviewed. The design of PBPC and HPC are explained in Sec. 5.3. The two-stage robust control design is developed in Sec. 5.4 and is exploited towards robust and constraint-aware controllers with model predictive structures. The simulation and experimental results are brought after in Sec. 5.5 and the chapter is concluded with a discussion over the theoretical and experimental results in Sec. 5.6.

5.2 Literature Survey

Successful RVS necessitates effective constraint handling in the system. Various methods were previously proposed to deal with the constraints. One effective way to manage the system's constraints was to employ path planning techniques [5.1-5.3]. These methods formed image space constraint-aware trajectories prior to servoing, which were then tracked closely to reach the desired pose. Global stability was obtained through this method. Several techniques were exploited to achieve the constraint-aware trajectories. Potential field method, once popular in collision-free path planning, was employed for path planning [5.1]. Optimization-based techniques were alternatively proposed to generate robust trajectories [5.3]. Nevertheless, such techniques could not cope with the uncertainties of the system. On a separate line, constrained controllers were employed to deal with the system limitations during the servoing. Specifically, model predictive controllers (MPC) were employed for this matter. These controllers were capable of system output optimization, while considering the limitations of the system.

Model predictive control was previously employed for visual servoing. One of the early applications of MPC for visual servoing was proposed in [5.4], where a generalized predictive control (GPC) form of MPC was employed in image space. The method was numerically stable and implementable for real-time applications, while providing optimal results. Yet, this work was not concerned with constraint handling. Later, the camera and robot constraints were considered in the optimisation of the MPC to address the shortcomings of the classical IBVS [5.5]. A more general constrained image-based MPC was introduced in [5.6] with a different cost function. It was shown that the proposed MPC was capable of handling the constraints, yet its stability and numerical feasibility were questionable. Similar image-based MPCs were employed by other works such as [5.7]. The reference trajectory used in MPC was modified in [5.8] to ensure the stability of the controller. In addition, a collision-free visual servoing was practiced via MPC in [5.9], which used collision-free planning techniques (such as probabilistic road map) for this matter. This technique was later used as a part of robot programming scheme in cluttered workspaces [5.10]. Moreover, a MPC which exploited image moments was proposed to improve the speed and stability of image-based MPC controllers in visual servoing systems. Similarly, a quasi min-max MPC controller was developed for visual servoing which modeled the system as a polytopic linear time-varying (LTV) system [5.11]. This controller could handle the system constraints by constrained optimizations carried through LMI. Yet, many of these controllers suffered from local stability and numerical feasibility problems. Moreover, the effect of system uncertainties was not considered in any of these works.

Robust RVS was followed in many of the previous works as a remedy to problems caused by system uncertainties. Many of these works focused on uncertain parameters of camera's calibration and robot's dynamics. An adaptive technique was employed to estimate robot's parameters online [5.12]. A robust control scheme was proposed by [5.13] and its stability in presence of camera/robot's uncertainties was proven via Lyapunov method. In addition, robustness with respect to calibration errors in terms of stability and error boundness was achieved in [5.14]. Separate compensators were designed in [5.15] for robot and camera uncertain parameters. A quaternion-based visual servoing was proposed by [5.16] to provide robustness against camera calibration parameters. Moreover, a controller robust to robot's dynamics parameter uncertainties was proposed for a 2 DOF robot in [5.17]. Robot's parametric uncertainties were considered in [5.18]. The robustness to visual servoing parametric uncertainties was also provided in [5.19] through LMI optimisation. Time-varying uncertainties of the robot was considered in [5.20] via an adaptive controller, which used a function approximation technique. A PBVS scheme robust to camera parameter uncertainties was proposed in [5.21] which had the benefit of global stability. In addition to that, various control schemes were proposed to counter the effects of uncertainties in the system. Sliding-mode control techniques were practiced in a couple of works including [5.22-5.25]. Robust filter-based techniques were employed to reduce the uncertainties of the system [5.26, 5.27]. Optimization techniques based on LMI were also introduced in [5.28, 5.29]. The inherent robustness of LQG and optimal controllers were exploited in [5.30], [5.31]. Yet, most of these methods were not concerned with the constraints of the system. In addition, the accuracy of the system was usually not the main concern.

This work proposed several control schemes that can handle the constraints of the system properly, offer global stability and improved numerical feasibility, and are robust against the uncertainties of the system. The details of these controllers are presented in the sequel.

5.3 Predictive Controllers for Constrained Visual Servoing

The design methodology of robust and constrained visual servo controllers is the topic of this section. The required knowledge of the system is presented first, followed by constrained and robust control schemes for visual servoing.

5.3.1 Preliminaries

In this subsection, the structure of the system, the definitions and assumptions required by the proposed control designs are discussed as follows.

System Configuration

Similar to previous chapters, this chapter assumes a system composed of a six degree-of-freedom manipulator and a camera which is mounted on the end-effector of the manipulator (i.e., eye-in-hand camera configuration). The camera initially has the object of interest in its FOV. The camera images of the object are processed during the servoing and the image features are extracted. In this thesis, point image features are of interest. The image plane coordinates of image points are used as image features,

$$s_k = \begin{bmatrix} u_{1,k} & v_{1,k} & . & . & u_{n,k} & v_{n,k} \end{bmatrix}^T , \tag{5.1}$$

where n is the number of point features, s_k is the vector of image features at time k and,

$$\begin{bmatrix} u_{i,k} & v_{i,k} \end{bmatrix} = f \begin{bmatrix} \frac{X^c_{o,i,k}}{Z^c_{o,i,k}} & \frac{Y^c_{o,i,k}}{Z^c_{o,i,k}} \end{bmatrix}, \tag{5.2}$$

are the image coordinates of i^{th} point of the object in the camera frame (i.e., $P^c_{o,i,k} = \begin{bmatrix} X^c_{o,i,k} & Y^c_{o,i,k} & Z^c_{o,i,k} \end{bmatrix}$), projected on the image plane. In (5.2), the camera's focal length is denoted by *f*. Image features are indirectly connected to the relative pose of the object with respect to the camera,

$$P^c_{o,i,k} = R^c_{o,k} P^o_i + t^c_{o,k}, \tag{5.3}$$

where $R^c_{o,k}$, and $t^c_{o,k}$ are the rotation and translation between the object and camera frame, respectively. Here, $P^o_i = \begin{bmatrix} X^o_i & Y^o_i & Z^o_i \end{bmatrix}$ is the i^{th} point of the object in object frame and is supposed to be known through the object's geometry. Both image features and pose parameters may be used for servoing, as is explained in the sequel.

System Servoing Errors

The image features may be directly used for control purposes (e.g., in IBVS methods). In such scenarios, the image features are compared with the features at the desired pose,

$$s^* = \begin{bmatrix} u^*_1 & v^*_1 & . & . & u^*_n & v^*_n \end{bmatrix}^T, \tag{5.4}$$

and the image space error at time step k is formed as,

$$e_{s,k} = s_k - s^*. \tag{5.5}$$

It is assumed that the robot will reach its desired pose once this error is reduced to zero. Yet, this assumption may not always hold due to local minima of the system.

Alternatively, image features may be used to estimate the pose of the object with respect to the camera, which is represented by the transformation matrix between the object and camera frame,

$$H^{c}_{o,k} = \begin{bmatrix} R^{c}_{o,k} & t^{c}_{o,k} \\ 0 & 1 \end{bmatrix}. \tag{5.6}$$

The pose of the object with respect to the camera may be estimated accurately through iterative methods (as was discussed in chapter 2). The pose error is then defined based on the transformation between the current and the desired camera frames,

$$H^{c^*}_{c,k} = H^{c^*}_{o} \left(H^{c}_{o,k}\right)^{-1}. \tag{5.7}$$

Three different pose errors are defined based on this transformation. The translation error is defined as follows,

$$e_{t,k} = t^{c^*}_{c,k}, \tag{5.8}$$

where $t^{c^*}_{c,k}$ is the translation between the current and the desired camera frame. As for the orientation error, two different choices are taken. In the first orientation error, the angle and axis representation of the rotation matrix between the current and desired camera frames ($R^{c^*}_{c,k}$) is exploited,

$$e_{R_1,k} = \rho^{c^*}_{c,k} \vartheta^{c^*}_{c,k}, \tag{5.9}$$

where $\vartheta^{c^*}_{c,k}$ and $\rho^{c^*}_{c,k}$ are the angle and axis related to the $R^{c^*}_{c,k}$, respectively. The second orientation error is composed of the Euler angles, representing the rotation matrix, $R^{c^*}_{c,k}$,

$$e_{R_2,k} = \phi^{c^*}_{c,k} = \begin{bmatrix} \varphi^{c^*}_{c,k} & \theta^{c^*}_{c,k} & \psi^{c^*}_{c,k} \end{bmatrix}^T, \tag{5.10}$$

where $\varphi^{c^*}_{c,k}$, $\theta^{c^*}_{c,k}$, and $\psi^{c^*}_{c,k}$ are the roll, pitch, and yaw angles related to the rotation matrix, $R^{c^*}_{c,k}$.

The third error is defined as the logarithm of depth difference for each feature point between the current and desired camera frames,

$$e_{z,k} = \left[\log Z^{c}_{o,1,k} - \log Z^{c*}_{o,1} \quad . \quad . \quad \log Z^{c}_{o,i,n} - \log Z^{c*}_{o,n} \right], \tag{5.11}$$

where $Z^{c*}_{o,i}$ is the depth of i^{th} point of the object in the desired camera frame ($P^{c*}_{o,i}$). As it is apparent, the camera will reach its desired pose once the first and second introduced pose errors are reduced to zero. The third pose error may be used as a complement to enhance the stability of the system [5.32]. In the sequel, it will be shown how a combination of image space and pose errors may be used in visual servoing to make the robot reach its desired pose.

System Dynamics

Each time the controller generates the camera velocity, the robot is moved in compliance. As a result, the pose of the object with respect to the camera and subsequently the pose errors are updated as,

$$e_{t,k+1} = e_{t,k} + \Delta t R^{c^*}_{c,k} \upsilon_{c,k} , \tag{5.12}$$

$$e_{R,k+1} = e_{R,k} + \Delta t J_{\theta,k} \varpi_{c,k} , \tag{5.13}$$

$$e_{z,k+1} = e_{z,k} + \Delta t J_{z,k} V_{c,k} . \tag{5.14}$$

Here, $V_{c,k} = \left[\upsilon^{T}_{c,k} \quad \varpi^{T}_{c,k} \right]^{T}$ is the camera velocity in camera frame, Δt is the sampling time, $J_{\theta,k}$ and $J_{z,k}$ are the angle and depth Jacobian, defined in [5.32].

Similarly, the features error is updated as follows,

$$e_{s,k+1} = e_{s,k} + \Delta t J_{s,k} V_{c,k} , \tag{5.15}$$

where $J_{s,k}$ is the image Jacobian, defined in [5.33]. The error update formulation, shown in (5.12) to (5.15), is used in constraint optimization of predictive controllers, which are the focus of the coming subsections.

The control goal in visual servoing is to reduce the error of the system, whether in image space or Cartesian space, to zero. Traditionally, a simple proportional controller was employed to guarantee exponential decree of the error,

$$\dot{e}_G = -\lambda e_G, \tag{5.16}$$

where e_G is the general error term and λ is the controller gain. However such controller could not handle the robot/camera constraints due to its simplicity. The design of MPC to handle the constraints is the topic of this subsection. It will be shown that the errors introduced in the previous subsection are reduced to zero without violation of the constraints. Image-based, position-based, and hybrid controllers are discussed separately and the constraints of the system are explained afterwards.

5.3.2 Image-Based Predictive Control

This controller was introduced in [5.6] and is composed of an optimization problem solver which finds the appropriate control to minimize the cost function defined as follows,

$$L_{s,k} = \sum_{i=1}^{N_c} e_{s,k+i}{}^T \mathrm{B}_s e_{s,k+i}, \tag{5.17}$$

where B_s is the image error weighting matrix and N_c is the control horizon. As was discussed in [5.6], the choice of N_c is not trivial. Large values for N_c result in increase of computation time, which makes this controller not suitable for real-time and fast applications, such as visual servoing. On the other hand, small values of N_c cause discontinuities in the control and instability. In this work, N_c is chosen as unity to minimize the computational time. Then the problem of control discontinuities caused by this selection is solved by modifying the cost function as,

$$L_{s,k} = e_{s,k+1}{}^T \mathrm{B}_s e_{s,k+1} + V_{c,k}{}^T \Upsilon V_{c,k}, \tag{5.18}$$

where $V_{c,k}$ and Υ are the camera velocity and its related weighting matrix, respectively. This choice of cost function reduces the discontinuities of the control as it penalizes large control actions. The stability of the system is ensured through an additional constraint, which is explained in the sequel.

The control actions are found by minimizing the cost, subject to system constraints,

$$\begin{aligned} &V_{c,k} = \underset{V_{c,k}}{\arg\min}\, L_{s,k} \\ &s.t. \quad e_{s,k+1} = e_{s,k} + \Delta t J_{s,k} V_{c,k} \quad s_k \in S \quad V_{c,k} \in \Lambda, \end{aligned} \tag{5.19}$$

where S is the set of admissible image features, Λ is the set of acceptable camera velocities, and $J_{s,k}$ is the image Jacobian at time step k, respectively. As can be seen, the first constraint of the system is the linearized model of the system, while the other conditions set boundaries on future features and system controls. The velocity of the camera is then found by solving this optimization problem and is applied to the robot's end-effector.

5.3.3 Position-Based Predictive Control

This controller, proposed by this work, is similar to its image-based counterpart in optimization and the constraints. In this controller, the reduction of the position error in the presence of constraints is considered. As a result, the cost function is defined as,

$$L_{p,k} = e_{p,k+1}{}^T \mathrm{B}_p e_{p,k+1} + V_{c,k}{}^T \Upsilon V_{c,k}, \tag{5.20}$$

where B_p is the error weighting matrix. The position error is defined as,

$$e_{p,k} = \begin{bmatrix} e_{t,k}^T & e_{R_2,k}^T \end{bmatrix}^T. \tag{5.21}$$

The control action is found similar to image-based MPC,

$$\begin{aligned} &V_{c,k} = \underset{V_{c,k}}{\arg\min}\, L_{p,k} \\ &s.t. \quad e_{p,k+1} = e_{p,k} + \Delta t J_{p,k} V_{c,k} \quad s_k \in S \quad V_{c,k} \in \Lambda, \end{aligned} \tag{5.22}$$

where $J_{p,i}$ is the position Jacobian [5.34] at time step i and is defined as,

$$J_{p,i} = \begin{bmatrix} I_3 & 0 \\ 0 & T(\phi_c^{c^*})^{-1} \end{bmatrix}. \tag{5.23}$$

Here I_3 is the 3 by 3 identity matrix and $T(\phi_c^{c^*})$ is the transformation matrix between angular velocities and time derivatives of Euler angles,

$$\omega_c^{c^*} = T(\phi_c^{c^*})\dot{\phi}_c^{c^*}. \tag{5.24}$$

As can be inferred from (5.20) and (5.22), this is a MPC with unity control horizon, which has set penalties on the control to prevent discontinuities in the output. Setting the control horizon to unity has the advantage of minimal computational cost, as was discussed before. The velocity of the camera is obtained by solving (5.22) online and is then applied to the robot for servoing. The constraints of the system are explained in the sequel.

5.3.4 Hybrid Predictive Control

Image-based predictive controllers were proposed previously to reduce the image space error, subject to system constraints. However, image-based algorithms were prone to local stability as was discussed in [5.6]. Hybrid predictive controllers are proposed to solve this problem. In such controllers, a suitable combination of image space and pose errors are selected as the error of the system and is reduced to zero using the optimization techniques. For this reason, the cost function of the system is defined as follows,

$$L_{H,k} = e_{H,k+1}^T \mathrm{B}_H e_{H,k+1} + V_{c,k}^T \Upsilon V_{c,k}, \tag{5.25}$$

where $e_{H,k+1}$ is the hybrid error at time step *k+1*, $V_{c,k}$ is the camera error at time step *k*, and B_H is the error weighting matrix. Such cost function set penalties on current camera velocity and the error of next time step, which is related to the camera velocity through (5.12) to (5.15). A suitable selection of control weighting matrix (Υ) is important as it affect the smoothness and convergence of the system. It is also worth mentioning that the weighting matrices in (5.25) are in close relationship as they determine the contribution of error and control in the cost function.

The camera velocity is found by minimizing the cost function, subject to system constraints,

$$\begin{aligned} &V_{c,k} = \arg\min_{V_{c,k}} L_{H,k} \\ &s.t \quad s_{k+1} \in S \quad V_{c,k} \in \Lambda, \end{aligned} \tag{5.26}$$

where S and Λ are the sets of permissible image features and camera velocities, respectively. The choice of cost function in this minimization makes this controller equivalent to a model predictive controller with unity control horizon. Short control horizon in this controller helps its online solution to be feasible. As a result, the stability of this controller is not guaranteed. Moreover, the controller may result in severe control discontinuities as a result of fast control actions. This work proposes an additional constraint to guarantee the stability of the system, while introducing the penalties on control action (i.e., the camera velocities) to prevent erratic control decisions, caused by short control horizon. The control penalties are explained in (5.25), whereas the stability constraint is discussed later, along with the other constraints.

The hybrid error selection for the cost function has an important impact on both image space and Cartesian trajectories. This work proposes two hybrid controllers with different error definitions for cost minimization. The image error is common between these hybrid errors, while the pose errors are selective. The first hybrid error is defined similar to hybrid visual servoing controller proposed in [5.32],

$$e_{H_1,k} = \begin{bmatrix} e_{s,k}^T & e_{R,k}^T & e_{z,k}^T \end{bmatrix}^T. \tag{5.27}$$

This selection of hybrid error renders the orientation control to orientation error, while the translational motion is directed by the image and depth error minimization. As a result, both image space and Cartesian space trajectories are smooth. The error weighting matrix is chosen as follows,

$$\mathrm{B}_{H_1} = \begin{bmatrix} \mathrm{B}_s & 0 & 0 \\ 0 & \mathrm{B}_R & 0 \\ 0 & 0 & \mathrm{B}_z \end{bmatrix}, \tag{5.28}$$

which changes the cost function in (5.25) to,

$$L_{H_1,k} = e_{s,k+1}^T \mathrm{B}_s e_{s,k+1} + e_{R,k+1}^T \mathrm{B}_R e_{R,k+1} + e_{z,k+1}^T \mathrm{B}_z e_{z,k+1} + V_{c,k}^T \Upsilon V_{c,k}. \tag{5.29}$$

It is worth mentioning that an equal weighting between different error in (5.29) results in a balanced motion, while priorities might be given to a specific error, depending on the servoing scenario.

Alternatively, the second hybrid error is defined as,

$$e_{H_2,k} = \begin{bmatrix} e_{s,k}^T & e_{R,k}^T & e_{t,k}^T \end{bmatrix}^T, \tag{5.30}$$

which is a combination of image and pose errors, each being used in IBVS and PBVS, respectively. The weighting matrix is selected as,

$$\mathrm{B}_{H_2} = \begin{bmatrix} \mathrm{B}_s & 0 & 0 \\ 0 & \mathrm{B}_R & 0 \\ 0 & 0 & \mathrm{B}_t \end{bmatrix}, \tag{5.31}$$

which results in a cost function as follows,

$$L_{H_2,k} = e_{s,k+1}^T \mathrm{B}_s e_{s,k+1} + e_{R,k+1}^T \mathrm{B}_R e_{R,k+1} + e_{t,k+1}^T \mathrm{B}_t e_{t,k+1} + V_{c,k}^T \Upsilon V_{c,k}. \tag{5.32}$$

As it can be inferred from (5.32), this choice of hybrid error provides an interesting tool to compromise between the error reduction in image space and Cartesian space. The controller will became purely image-based by choosing $\mathrm{B}_R = \mathrm{B}_t = 0$, while a totally position-based control is experienced by selecting $\mathrm{B}_s = 0$. Therefore, these controllers are subsets of the second hybrid controller. It is worth mentioning that the weighting matrices may be selected to minimize the chance of constraint violation, which is handled by the predictive controller. The system constraints are explained as follows.

5.3.5 System Constraints

The constraints of the system are categorized into three groups and are discussed as follows. In order to have a continuous feedback from the camera, the image features of the object should be visible at all times. This means that the image features should not leave a bounded area in the image space. Such constraint is considered in the admissible image feature's set,

$$S = \{ s \mid s_{\min} \leq s \leq s_{\max} \}, \tag{5.33}$$

where $s_{\min}$ and $s_{\max}$ are the minimum and maximum allowable values for image features.

The workspace of the robot is limited and the velocity at which it can move is also restricted. Therefore the velocity of the camera should be limited so that the optimization solution results in feasible robot motion. These constraints are incorporated in the acceptable camera velocity set as follows,

$$\Lambda = \{ V \mid \dot{q}_{r,\min} \leq J_R^{\dagger} R_c^b V_c \leq \dot{q}_{r,\max} \} \cap \{ V \mid q_{r,\min} \leq q_r + \Delta t J_R^{\dagger} R_c^b V_c \leq q_{r,\max} \}, \tag{5.34}$$

where q_r and $\dot{q}_r$ are the robot's joint angles and velocities, respectively. The robot Jacobian is denoted by J_R and the rotation between camera and robot's base is shown by R_c^b .The first set in (5.34) is based on the limitation of joint velocities and the second set implies the joint limits. Alternately, the set of admissible camera velocities may be defined as,

$$\Lambda = \{ V_{c,k} \mid V_{c,\min} \leq V_{c,k} \leq V_{c,\max} \} \cap \{ V_c \mid \omega_{\min} \leq \omega_{c,k}^o + \Delta t R_{c,k}^o J_{p,k} V_{c,k} \leq \omega_{\max} \}. \tag{5.35}$$

The new set definition is based on camera position and velocity limits.

In order to enforce the stability of the system, the energy of the error signal is bound by the previous energy of the system to ensure the convergence,

$$e_{G,k+1}{}^T \Psi e_{G,k+1} \leq e_{G,k}{}^T \Psi e_{G,k}, \tag{5.36}$$

where Ψ is a positive definite matrix. Such constraint may be set by amending the velocity constraint,

$$\begin{aligned} \Lambda = &\{ V \mid \dot{q}_{r,\min} \leq J_R^{\dagger} R_c^b V_c \leq \dot{q}_{r,\max} \} \cap \\ &\{ V \mid q_{r,\min} \leq q_r + \Delta t J_R^{\dagger} R_c^b V_c \leq q_{r,\max} \} \cap \\ &\{ V \mid V_c^T J_p^T \Psi e_{G,k} + e_{G,k}^T \Psi J_p V_c + \Delta t V^T J_p^T \Psi J_p V_c \leq 0 \}, \end{aligned} \tag{5.37}$$

where the stability constraint is included as the third set or alternatively,

$$\begin{aligned}\Lambda = &\left\{V_{c,k} \mid V_{c,\min} \le V_{c,k} \le V_{c,\max}\right\} \cap \\ &\left\{V_c \mid P_{\min} \le P_{c,k}^o + \Delta t R_{c,k}^o J_k V_{c,k} \le P_{\max}\right\} \cap \\ &\left\{V \mid V^T J^T P e_k + e_k^T P J V + \Delta t V^T J^T P J V \le 0\right\}.\end{aligned} \tag{5.38}$$

The error term in (5.37) and (5.38) may be selected as the pose error ($e_{p,k}$) or either of the hybrid errors ($e_{H,k}$). Enforcing the complementary stability constraint causes the optimal solver to reduce the error constantly and therefore the convergence of the system is guaranteed. The design of the constrained MPC controllers is now complete. The effectiveness of these methods is verified through multiple simulations in Sec. 5.5.

5.4 Two-Stage Robust Control Design

Robust control schemes are the focus of this section. A two-stage control scheme is introduced for this purpose. First, it will be shown that this controller has the ability of decoupling the mean and covariance of the propagated error from the convergence rate. Hence, the mean and covariance of the errors may be reduced without sacrificing the convergence rate. Second, the discussed MPC is employed at either of control stages to enable the system of constraint management. As a result, two robust schemes are proposed and the unique properties of each are highlighted.

5.4.1 Decoupled Controller

It was shown in chapter 4 that the traditional single degree of freedom (DOF) P type controller is not sufficient to reduce the propagated error without sacrificing the speed. As a remedy, a two degree of freedom (DOF) controller is proposed in this section that minimizes the rate of error growth, while maintaining the servoing convergence rate. This controller decouples the adjustment of the mean and the covariance of image features through a two stage control scheme. An offline controller is employed to maintain the convergence of the system, while an online ancillary controller is introduced to minimize the effects of propagated uncertainties. As a result, the online controller is needless of depth estimation and inverse Jacobian calculation, leading to a faster controller. A novel control structure is proposed to decouple the dependency of mean and covariance of the features. For this purpose, a two DOF controller is proposed. In this scheme, an offline controller is employed as the base for visual servoing. Initially the feature trajectories are formed offline through a proportional controller, starting from the initial features ($\overline{s}_0 = s_0$),

$$\dot{\overline{s}}_k = -\lambda_1(\overline{s}_k - s^*)\,, \tag{5.39}$$

$$\overline{s}_{k+1} = \overline{s}_k + \Delta t \dot{\overline{s}}_k\,, \tag{5.40}$$

where λ_1 is the offline controller gain, $\overline{s}_k$ and $\dot{\overline{s}}_k$ are the offline features and their time derivatives. The important feature of this offline controller is its deterministic nature. This character comes as a consequence of offline computation, based on initial and desired features. This may be considered as an open-loop controller. The features are then updated based on (5.39) and (5.40),

$$\overline{s}_{k+1} = (1-\lambda_1\Delta t)\overline{s}_k + \lambda_1 \Delta t s^*\,. \tag{5.41}$$

Next, an online (closed-loop) controller is proposed to keep the features close to the determined trajectory. For this purpose, the time derivative of online features is calculated as,

$$\dot{s}_k = \dot{\overline{s}} - \lambda_2(s_k - \overline{s}_k)\,, \tag{5.42}$$

where λ_2 is the online controller gain. As can be seen from (5.42), the feature update has two parts to it. The first part is the pre-computed feature update from the aforementioned open-loop controller. The second part is similar to typical IBVS controller having the pre-computed features ($\overline{s}_k$) as its desired features. By employing the integration on the feature vector,

$$s_{k+1} = s_k + \Delta t \hat{\dot{s}}_k\,, \tag{5.43}$$

the trajectory of the features are updated online as follows,

$$s_{k+1} = (1-\lambda_2\Delta t)s_k + (\lambda_2 - \lambda_1)\Delta t \overline{s}_k + \lambda_1 \Delta t s^* + \Delta t m - \lambda_2 \Delta t \varepsilon_{s,k}\,, \tag{5.44}$$

where m is the deterministic error caused by calibration and is assumed to be constant for simplicity. It is worth mentioning that a choice of $\lambda_1 = \lambda_2$ reduces the controller to the conventional proportional controller, discussed in previous section. Having (5.44) ready, the mean and covariance of features are calculated in the sequel. It should be reminded that the pre-computed features ($\overline{s}_k$) are deterministic and therefore have zero covariance.

The mean of features are calculated by applying the expectation function to (5.44),

$$\mathrm{E}\left\{s_{k+1}\right\} = (1-\lambda_2 \Delta t)\mathrm{E}\left\{s_k\right\} + (\lambda_2 - \lambda_1)\Delta t \mathrm{E}\left\{\overline{s}_k\right\} + \lambda_1 \Delta t s^* + \Delta t m. \tag{5.45}$$

The mean of the pre-computed features is computed as follows,

$$\mathrm{E}\left\{\overline{s}_k\right\} = \left(s_0 - s^*\right)\exp(-\lambda_1 k \Delta t) + s^* . \tag{5.46}$$

Now the mean of features is found with the same approach. It is assumed that,

$$g_m(k\Delta t) = \mathrm{E}\left\{s_k\right\}, \tag{5.47}$$

$$\mathrm{E}\left\{s_{k+1}\right\} = g_m(k\Delta t) + \frac{\partial g_m}{\partial t}\Delta t, \tag{5.48}$$

which lead to,

$$\frac{\partial g_m}{\partial t} + \lambda_2 g_m(k\Delta t) = \lambda_2 s^* + m + (\lambda_2 - \lambda_1)\left\{\left(s_0 - s^*\right)\exp(-\lambda_1 k \Delta t)\right\}. \tag{5.49}$$

The solution to this partial differential equation is given by,

$$g_m(k\Delta t) = \left(s_0 - s^*\right)\exp(-\lambda_1 k \Delta t) + s^* + \frac{m}{\lambda_2}\left(1 - \exp(-\lambda_2 k \Delta t)\right). \tag{5.50}$$

The achieved mean is very similar to mean of the simple proportional controller, calculated in chapter 3. The only difference is the separate control of the convergence rate and the steady-state error by the offline and online gains, respectively.

By definition, the covariance of the features are expressed as,

$$g_c(k\Delta t) = \mathrm{cov}\left\{s_k\right\}, \tag{5.51}$$

$$\operatorname{cov}\{s_{k+1}\} = g_c(k\Delta t) + \frac{\partial g_c}{\partial t}\Delta t \,. \tag{5.52}$$

By exploiting (5.44), (5.51), and (5.52), one can show,

$$g_c(k\Delta t) + \frac{\partial g_c}{\partial t}\Delta t = \left(1-\lambda_2\Delta t\right)^2 g_c(k\Delta t) + \lambda_2{}^2\Delta t^2\sigma^2 \,, \tag{5.53}$$

which has a solution as follows,

$$g_c(k\Delta t) = \left(\frac{1-\exp\left(\left(-2\lambda_2 + \lambda_2{}^2\Delta t\right)k\Delta t\right)}{2-\lambda_2\Delta t}\right)\lambda_2\Delta t\sigma^2 \,. \tag{5.54}$$

As can be seen, the convergence rate and final value of the covariance of the features in the new controller depends only on the gain of the online controller. Therefore, the proposed controller has the advantage of reducing the covariance and the rate of its growth, while keeping the original convergence rate of the mean trajectories.

Based on the proposed scheme, an IBVS controller has been developed. In such a system, the velocity of the camera is related to the time derivation of features via image Jacobian,

$$\dot{s}_k = J_{s,k}V_{c,k} \,, \tag{5.55}$$

where $J_{s,k}$ is the image Jacobian discussed in [5.33]. Then the offline camera velocity may be calculated as,

$$\overline{V}_{c,k} = -\lambda_1\overline{J}_{s,k}{}^{\dagger}(\overline{s}_k - s^*) \,. \tag{5.56}$$

Here $\overline{J}_{s,k}{}^{\dagger}$ is the inverse Jacobian of the offline controller, $\overline{J}_{s,k}$ .The feature trajectories ($\overline{s}_k$) and their respective Jacobian ($\overline{J}_{s,k}$) are recorded offline. In the next step, the online controller is employed and the camera velocity is found as,

$$V_{c,k} = \overline{V}_{c,k} - \lambda_2 J_{s,k}{}^{\dagger}(s_k - \overline{s}_k) \,. \tag{5.57}$$

The first part of this controller guides the robot towards the desired position based on offline calculations and the second part works as an ancillary IBVS controller with $\overline{s}_k$ as its desired features, assuring the current features to follow the pre-computed feature trajectories closely. As was shown in [5.33], for such controller the Jacobian of desired features may be used alternatively and hence (5.57) can be reformulated as,

$$V_{c,k} = \overline{V}_{c,k} - \lambda_2 \overline{J}_{s,k}{}^{\dagger}(s_k - \overline{s}_k)\,. \tag{5.58}$$

The proposed controller in (5.56) has several advantages over the conventional controller. First, this controller has two degrees of freedom. As was shown previously, such structure allows the decoupling of the rate of convergence from the uncertainty control. Second, the Jacobian used in (5.58) is pre-computed and is needless of online depth estimation. Third, a big portion of the velocity is computed online and therefore current and computed features are very close. As a result the online gain (λ_2) may be selected sufficiently small. Finally, since most components of such controller are pre-computed and readily available at servoing, the control will perform faster compared to conventional controller.

It is worth computing the mean and covariance of the camera velocity through the new controller. The mean of the velocity is obtained through (5.50), (5.57), and (5.58) as follows,

$$\mathrm{E}\left\{V_{c,k}\right\} = \overline{J}_{s,k}{}^{\dagger}\left(\lambda_1\left(s^* - s_0\right)\exp(-\lambda_1 k\Delta t)\right) - \overline{J}_{s,k}{}^{\dagger} m\left(1-\exp(-\lambda_2 k\Delta t)\right). \tag{5.59}$$

Similarly, the covariance is computed as follows,

$$\operatorname{cov}\left\{V_{c,k}\right\} = \left(\frac{1-\exp\left(\left(-2\lambda_2 + \lambda_2{}^2\Delta t\right)k\Delta t\right)}{2-\lambda_2\Delta t}\right)\lambda_2{}^3\Delta t\sigma^2\left(\overline{J}_{s,k}{}^T\overline{J}_{s,k}\right)^{-1}. \tag{5.60}$$

As can be seen from (5.60), increasing the online gain will increase the covariance of the camera velocity and consequently the uncertainty on robot position will rise. Therefore, it is important to select the online gain as small as possible, but not too small so that the steady-state error of features remains reasonable. The effectiveness of this proposed method is verified through simulations in Sec 5.5.

As it was shown in the previous chapter, the uncertainty affects the visual feature trajectories, which in turn propagates to the robot's trajectory. To alleviate this issue, a two-stage control scheme is proposed. The controller is composed of an offline controller, through which the trajectories of camera velocity, visual features, and their associated image Jacobian are estimated, and an online controller which minimizes the uncertainties by relying on the estimated trajectories. The constraint handling is possible in either of these steps. Based on this fact, two robust predictive controllers are proposed. In the first robust controller, MPC is employed as the online controller. Such scheme has the advantage of online constraint handling and uncertainty minimization at the price of increased computational cost. The global stability of this controller is guaranteed similar to path planning schemes (e.g., [5.1]). The second robust controller exploits the MPC as its offline controller, rendering the online control to a simple proportional controller. This scheme has the advantage of guaranteed constraint management, even in the presence of uncertainties. Moreover, the choice of offline MPC reduces the computational cost dramatically. The stability of the system is guaranteed, similar to the proposed predictive controllers in Sec. 5.3. The control scheme as explained as follows.

5.4.2 Robust Control with Online Predictive Controller (RCONPC)

As was discussed, this controller has two stages. At the offline stage, the control problem is solved assuming no uncertainties. Therefore the model of the system is reduced to,

$$\bar{e}_{G,k+1} = \bar{e}_{G,k} + \Delta t \bar{J}_{G,k} \bar{V}_{c,k}\,, \tag{5.61}$$

where the estimated values are denoted by an "over-bar". The problem is initiated by setting,

$$e_0 = s_0 - s^*\,, \tag{5.62}$$

and is solved by selecting an appropriate controller (e.g., $\bar{V}_{c,k} = -\lambda \bar{J}_{G,k}{}^{\dagger} \bar{e}_{G,k}$). Since this problem is solved offline, the parameters such as object's pose and features' depths are readily available at each time step. Two different error sets may be selected to calculate the offline camera velocity. In the first set, the error of each image feature is selected as the servoing error,

$$e_{G,k} = e_{s,k}\,, \tag{5.63}$$

In the second set, a combination of pose and image feature errors is selected, similar to hybrid controller discussed in Sec. 5.3, i.e.,

$$e_{G,k} = \begin{bmatrix} e_{s,k}^T & e_{R,k}^T & e_{z,k}^T \end{bmatrix}^T . \tag{5.64}$$

The choice of the first set of features has the advantage of simplicity, while the second set ensures smooth Cartesian trajectory and global stability [5.32]. Choosing only the point features simplifies the overall servoing scheme. This is because these features are easy to extract and do not require any post-calculations (as is the case for depth and rotation angle). Therefore, regardless of the feature choice, only the trajectories of the point features and their associated Jacobian are recorded. In addition to that, the velocity profile of the offline controller is recorded and exploited in the online controller.

Once the estimated trajectories of the system are available, an online controller is employed to track the estimated trajectories. An optimal controller is designed by setting a cost function and minimizing the cost,

$$V_{c,k} = \arg\min \sum_{i=1}^{N_c} (e_{G,k+i} - \overline{e}_{G,k+i}) \mathrm{B}_G (e_{G,k+i} - \overline{e}_{G,k+i})^T , \tag{5.65}$$

where B_G is the weighting matrix and N_c is the control horizon. The controller is a typical model predictive control (MPC) adopted for VS, e.g., the one proposed in [5.6], with the desired features being replaced by the offline controller trajectories. As it can be inferred from (5.65), the covariance of the features is minimized if the mean of the features is estimated by their offline trajectories.

The control horizon is directly related with the computation time of controllers and therefore cannot be large for real-time applications. For this purpose, the control horizon is chosen to be minimal (i.e., equal to one). As a result, the acquired velocities and feature trajectories might face discontinuities that would cause erratic robot motions, as was pointed out in [5.6]. In order to alleviate this problem, the cost function is modified as follows

$$V_{c,k} = \arg\min \left[(e_{G,k+1} - \overline{e}_{G,k+1})^T \mathrm{B}_G (e_{G,k+1} - \overline{e}_{G,k+1}) + (V_{c,k} - \overline{V}_{c,k})^T \Upsilon (V_{c,k} - \overline{V}_{c,k}) \right]. \tag{5.66}$$

The new cost function penalizes the control deviations from the offline control and therefore minimizes the discontinuity in velocities. The choice of Υ in the new cost function must be done cautiously, as small values in Υ cause the system to be subjected to unwanted vibrations, while large values may prevent the system to converge. A reasonable design will start with higher values for Υ and reduce it gradually to let the system converge.

In the absence of constraints, the control may be calculated analytically similar to a linear quadratic regulator (LQR) as follows,

$$V_{c,k} = -(\Upsilon + \Delta t^2 J_{s,k}{}^T \mathrm{B}_s J_{s,k})^{-1} \left[\Delta t J_{s,k}{}^T \mathrm{B}_s (e_{s,k} - \overline{e}_{s,k}) - (\Upsilon + \Delta t^2 J_{s.k}{}^T \mathrm{B}_s \overline{J}_{s,k}) \overline{V}_{c,k} \right]. \quad (5.67)$$

The image Jacobian is not accessible directly and may only be estimated. Here, we approximate this Jacobian with its offline counterpart, $\overline{J}_{s,k}$. Then, (5.67) may be rewritten as follows,

$$V_{c,k} = \overline{V}_{c,k} - \Delta t (\Upsilon + \Delta t^2 \overline{J}_{s,k}{}^T \mathrm{B}_s \overline{J}_{s,k})^{-1} \overline{J}_{s,k}{}^T \mathrm{B}_s (e_{s,k} - \overline{e}_{s,k}). \quad (5.68)$$

As can be inferred from (5.68), the unconstrained online control is composed of the offline control and a damped-least-square controller that regulates the features along their pre-computed trajectories. The use of offline Jacobian has several advantages. First, this Jacobian is not affected by the noise and therefore does not amplify the propagated noise level. Second, as this Jacobian replaces the real-time Jacobian, there is no need for online Jacobian computation which requires partial pose estimation. Finally, by having most part of (5.68) pre-computed, this choice of Jacobian speeds up the control calculation.

Alternatively, the minimization problem may be solved as a constrained optimization problem to entail the restrictions of the system such as camera's visibility and robot's reachability constraints, similar to [5.6], i.e.,

$$\begin{aligned} &V_{c,k} = \arg\min \left[(e_{s,k+1} - \overline{e}_{s,k+1})^T \mathrm{B}_s (e_{s,k+1} - \overline{e}_{s,k+1}) + (V_{c,k} - \overline{V}_{c,k})^T \Upsilon (V_{c,k} - \overline{V}_{c,k}) \right] \\ &s.t. \quad s_k \in S \quad V_{c,k} \in \Lambda, \end{aligned} \quad (5.69)$$

where S and Λ are the sets of admissible image features and velocity trajectories, respectively. This empowers the proposed controller to be capable of minimizing the uncertainties, while handling the constraints. The unconstrained analytical solution is used as an initial guess to speed up the optimization procedure.

The stability of the offline controller was proven in the previous works [5.33]. While the stability of the point features is only local, adding the depth and rotation to the features results in global stability. Therefore, the feature trajectories in offline phase will end in desired features.

As for the online controller, the stability may be followed separately for constrained and unconstrained problems. In case of unconstrained system, the analytical solution is available for the controller and the stability is proven using the energy of the error as the positive definite Lyapunov function,

$$\ell = \frac{1}{2}\left(e_{s,k} - \overline{e}_{s,k}\right)^T \left(e_{s,k} - \overline{e}_{s,k}\right), \tag{5.70}$$

The derivate of this function is computed as follows,

$$\dot{\ell} = \left(e_{s,k} - \overline{e}_{s,k}\right)^T \left(J_{s,k} V_{c,k} - \overline{J}_{s,k} \overline{V}_{c,k}\right). \tag{5.71}$$

By using (5.68) in (5.71) and assuming $J_{s,k} \approx \overline{J}_{s,k}$, it can be shown that,

$$\dot{\ell} = -\Delta t\left(e_{s,k} - \overline{e}_{s,k}\right)^T \left[\overline{J}_{s,k}(\Upsilon + \Delta t^2 \overline{J}_{s,k}{}^T \mathrm{B}_s \overline{J}_{s,k})^{-1} \overline{J}_{s,k}{}^T \mathrm{B}_s\right](e_{s,k} - \overline{e}_{s,k}), \tag{5.72}$$

which is always negative as B_s and Υ are positive definite symmetric matrices. One way to assure that the Jacobian matrices are the same is to run the offline controller at each time step and find the velocity and the trajectory for the coming step.

5.4.3 Robust Control with Offline Predictive Controller (RCOFPC)

Prior to servoing, an error-free model of the system is developed as a reference, i.e.,

$$\overline{s}_{k+1} = \overline{s}_k + \Delta t \overline{J}_k \overline{V}_{c,k}, \tag{5.73}$$

where the error-free values are denoted by the “over-bar”. The error-free feature trajectories are decided by selecting a proper camera velocity profile, which guides the features towards the desires ones. In order to reduce the uncertainties of the system, the actual features are guided as close as possible to the error-free features. For this matter, the error between the actual and error-free features is to be minimized. This difference of features is shown by,

$$e_{s,k} = \hat{s}_k - \overline{s}_k \, . \tag{5.74}$$

Then it can be shown that,

$$e_{s,k+1} = e_{s,k} + \Delta t \dot{e}_{s,k} \, , \tag{5.75}$$

where,

$$\dot{e}_{s,k} = J_{s,k} V_{c,k} - \overline{J}_{s,k} \overline{V}_{c,k} \, . \tag{5.76}$$

The difference of features may reduce to zero exponentially by choosing,

$$\dot{e}_{s,k} = -\lambda e_{s,k} \, . \tag{5.77}$$

The velocity of the camera is then found by employing (5.77) in (5.76),

$$V_{c,k} = J_{s,k}{}^{\dagger} (\overline{J}_{s,k} \overline{V}_{c,k} - \lambda e_{s,k}) \, . \tag{5.78}$$

The image Jacobian may be estimated by the reference image Jacobian (i.e., $J_{s,k} \approx \overline{J}_{s,k}$), which reduces the control to,

$$V_{c,k} = \overline{V}_{c,k} - \lambda \overline{J}_{s,k}{}^{\dagger} e_{s,k} \, . \tag{5.79}$$

It can be inferred from (5.77) that the proposed camera velocity is composed of two parts. The first part is a pre-computed camera velocity which is the velocity reference. The second part is a regulatory controller that minimizes the deviation of features from the reference ones. It is worth mentioning that since the error-free components of (5.79) are computed prior to servoing, the computation of camera velocity through (5.79) is very fast in action. The computation of the reference velocities ($\bar{V}_{c,k}$), image Jacobians ($\bar{J}_{s,k}$), and image features ($\bar{s}_k$) are explained in the sequel.

In order to handle the constraints of the system, a MPC controller is developed. Since this controller is based on the error-free model of the system, it has been introduced to the reference system shown in (5.70). The cost of the system is defined as,

$$L_s = \sum_{i=1}^{N_c} \left(\bar{s}_{k+i} - s^*\right)^T \mathrm{B}_s \left(\bar{s}_{k+i} - s^*\right) + \bar{V}_{c,k+i}^T \Upsilon \bar{V}_{c,k+i} \,. \tag{5.80}$$

Here B_s and Υ are the features and velocity weighting matrices and N_c is the control horizon, which is selected as one to minimize the computational costs. As a result, the system may face velocity discontinuities, which is alleviated by selecting proper weighting matrices. Alternatively, the control horizon may be selected larger at the cost of extra computational time.

The control actions of the system is predicted over the control horizon by minimizing the cost function, subject to system constraints, i.e.,

$$\begin{aligned} U = & \underset{\bar{V}_{c,k+1},\ldots,\bar{V}_{c,k+N}}{\arg\min} \; L_s \\ s.t. \quad & \bar{s}_{k+i} \in S, \qquad \bar{V}_{c,k+i} \in \Lambda, \end{aligned} \tag{5.81}$$

where S and Λ are the admissible features and camera velocities sets, respectively. The first element of U is selected as the camera velocity and is applied to the reference model (5.73) to form the features of the next step. The process is repeated until the reference image features are close enough to the desired features. Next, the trajectories of camera velocity, image features and their relevant image Jacobian are recorded to be used for servoing through (5.79).

The conditions defined in (5.33) through (5.38) prevent the system from constraint violation. However, the constraints may not be met if the system is uncertain. One way to alleviate this problem is to tighten the constraints by setting new limits including,

$$S = \left\{ s_k \mid s_{\min} + \tau_s \le s_k \le s_{\max} - \tau_s \right\}, \tag{5.82}$$

$$\Lambda = \left\{ V_{c,k} \mid V_{c,\min} + \tau_V \le V_{c,k} \le V_{c,\max} - \tau_V \right\} \bigcap \\ \left\{ V_c \mid \omega_{\min} + \tau_P \le \omega_{c,k}^o + \Delta t R_{c,k}^o J_{p,k} V_{c,k} \le \omega_{\max} - \tau_P \right\}, \tag{5.83}$$

where τ_s, τ_V, and τ_P are the constraint thresholds of image features, camera velocity, and camera position, respectively. A proper selection of the uncertainty thresholds is important, since they may cause the system to be conservative or inadequate to handle the constraints. In this work, the uncertainty thresholds are selected as three times of the uncertainties standard deviation,

$$\tau_s = 3\sigma_s, \quad \tau_V = 3\sigma_V, \quad \tau_P = 3\sigma_P, \tag{5.84}$$

where σ_s, σ_V, and σ_P are the standard deviations of the uncertainties related to image features, camera velocity and camera position, respectively. Assuming the noise to be Gaussian, this choice of thresholds ensures satisfactory constraint handling for %99.7 of times. However, the standard deviations of the uncertainties are required for this matter, which is available from Chapter 3.

5.5 Simulation and Experimental Results

In this section, numerous simulations and experiments are conducted to demonstrate the effectiveness of the proposed methods for constraint and uncrtainty handling. Testing the system through simulations is performed under controlled situations (e.g., known noise parameters, system dynamics, etc.), while experimental results confirm the practicality of the proposed methods under actual system conditions. In the simulations, an object with four planar feature points, located on the vertices of a square that has a side of 10 cm, is considered. The camera is servoed from a distant pose to half a meter above the object. The sampling time is 30 milliseconds ($\Delta t = 0.03$ seconds) and the error weighting matrices are selected as the identity matrix. The simulations are carried under Matlab® 2011b software from Mathworks (Natick, MA, USA).

5.5.1 Constrained Predictive Controllers

In this subsection, the proposed predictive controllers (i.e., PBPC and HPC) are verified and compared with previously proposed image-based predictive controller. In the first simulation, the constraint handling property of the hybrid and position-based predictive controllers with the proposed cost functions, in which the control actions are penalized, are verified and compared with that of image-based predictive controller. For this purpose, the image features are bounded between 125 and 175 pixels in u direction and between 75 and 125 pixels in v direction. The position of the camera is restricted by 0.3 meters from the object in x and y direction and between 0.5 and 0.6 meters away from the object in z direction in the object frame. The camera velocities are limited to 0.3 (m/s or rad/s). The control weighting matrix is selected hundred times smaller than the identity matrix ($\Upsilon = 10^{-2} I_6$). The result of servoing with these controllers in image space and Cartesian space, along with camera velocities and system error energy are shown in Figures 5.1-5.4. The initial and final images of the object are demonstrated in image space by red dashed and blue dot-dashed lines, respectively. The image and Cartesian boundaries are depicted by black and blue dotted lines, respectively. The image space and Cartesian space results confirm the ability of all proposed controller to successfully handle the constraints of the system. The trajectories in both spaces are relatively smooth and all controllers are capable of guiding the robot to its desired pose in spite of the constraints in image and Cartesian space. The camera velocities, resulted from each of these controllers, are smooth despite the minimal selection of the control horizon. This smoothness is due to the penalties set on them in the cost function. In addition, the gradual decrease of error energy was depicted in all controllers and therefore the stability of the systems is maintained. The terminal constraint set on the camera velocities has a major role in this steady reduction of the error. It is worth mentioning that the image-based predictive controller has the longest convergence rate as a result of velocity penalizing, while the second hybrid method has the smoothest trajectories of all.

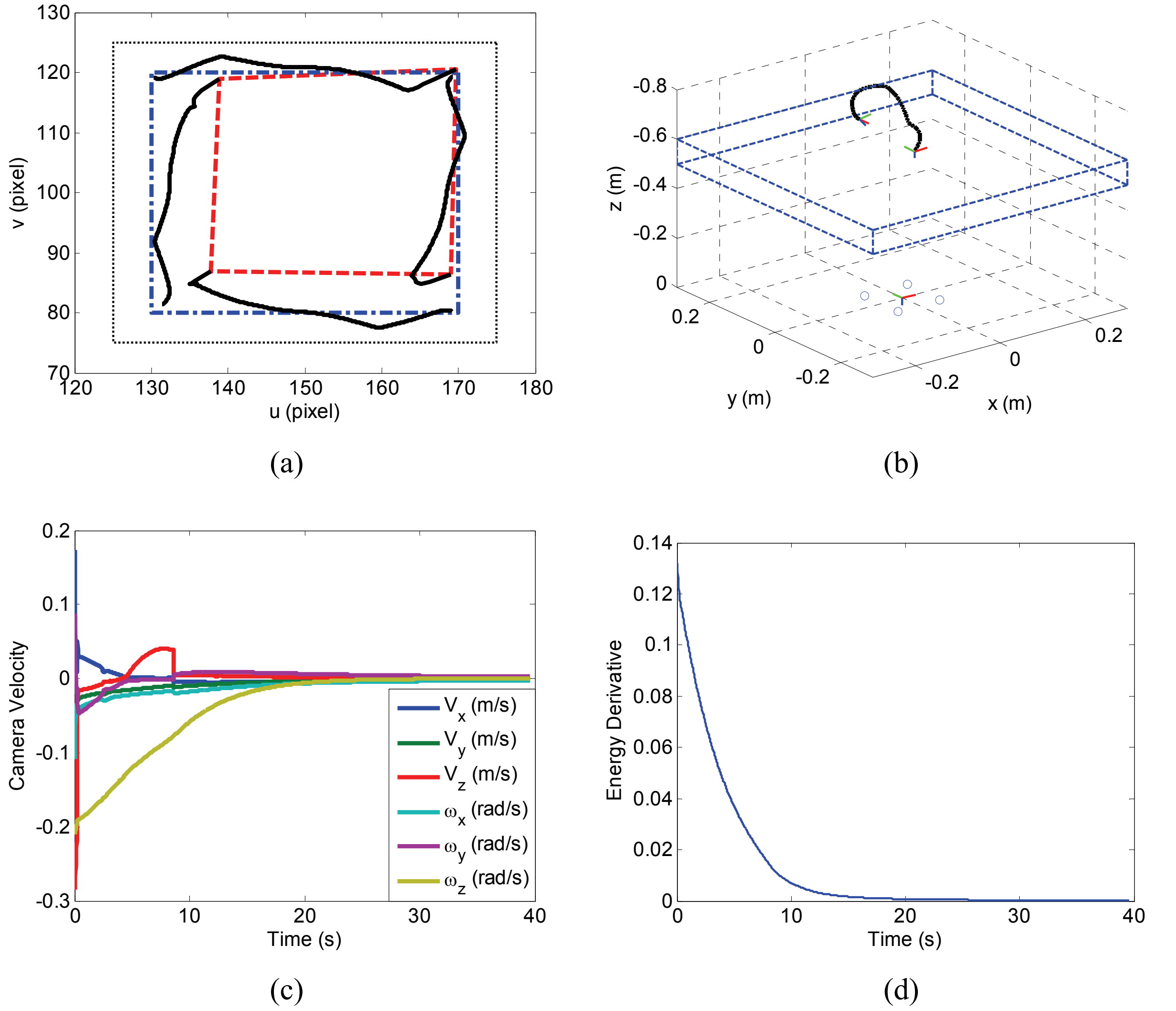
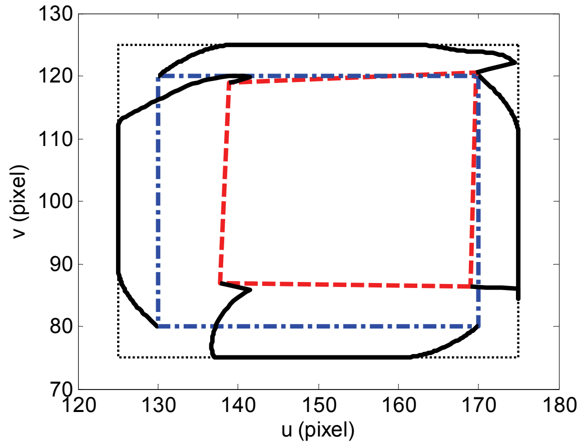
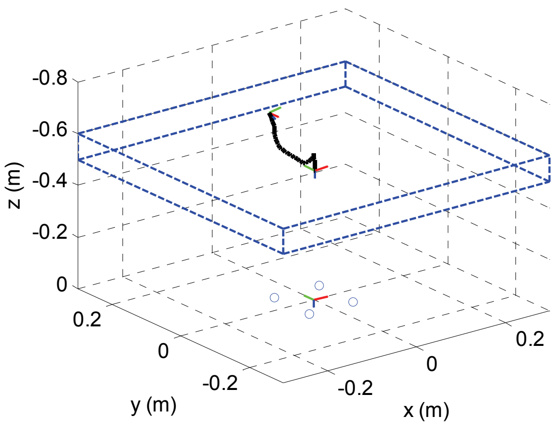


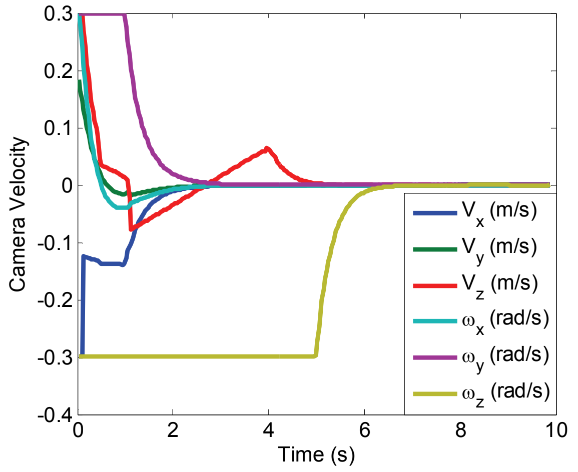
Figure 5.1 First simulation: Visual servoing via image-based MPC with velocity penalties, (a) Image space, (b) Cartesian space, (c) Camera velocities, and (d) Error energy.



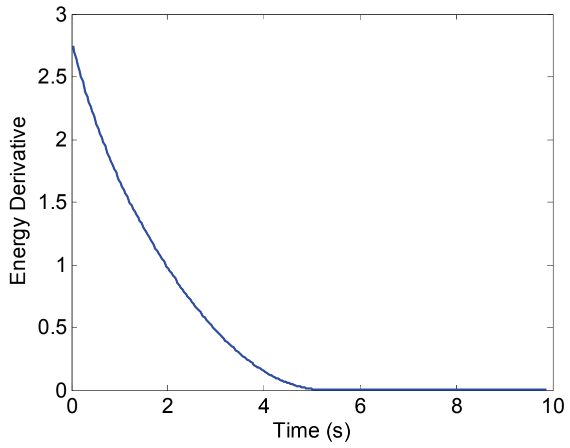
(a)



(b)

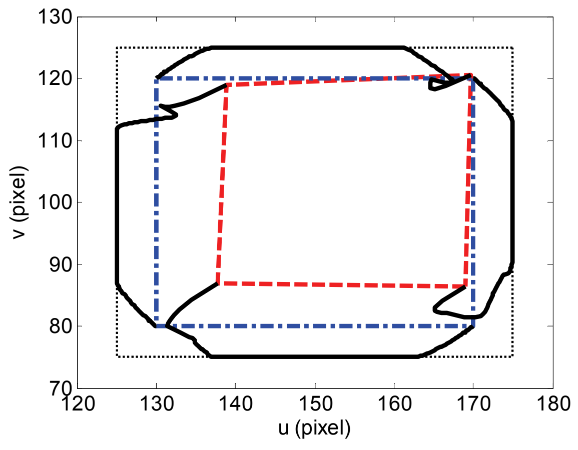


(c)

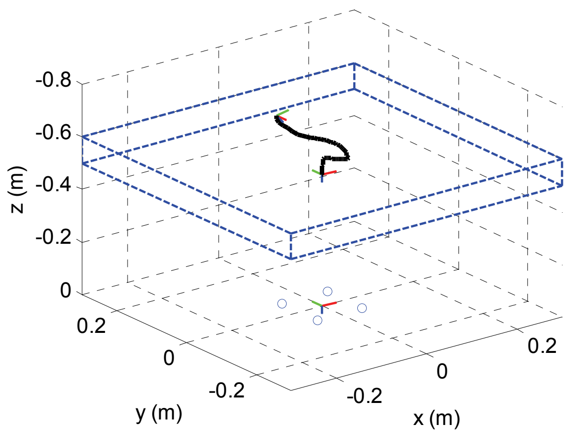


(d)

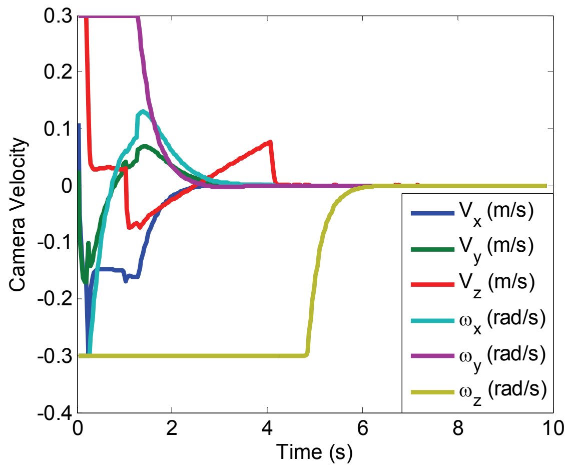
Figure 5.2 First simulation: Visual servoing via position-based MPC with velocity penalties, (a) Image space, (b) Cartesian space, (c) Camera velocities, and (d) Error energy.



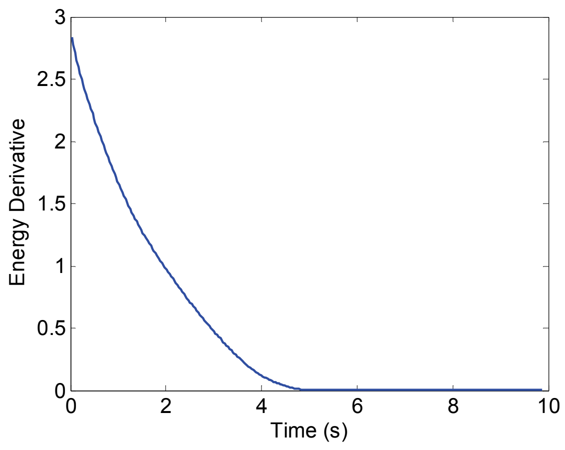
(a)



(b)

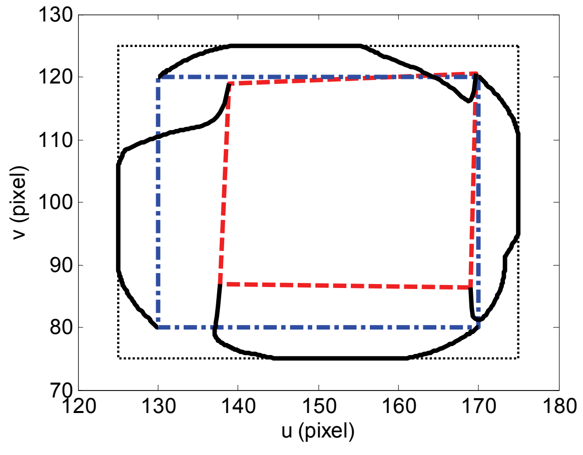


(c)

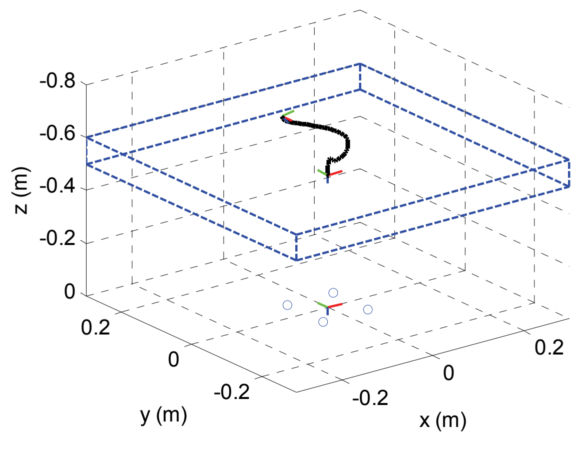


(d)

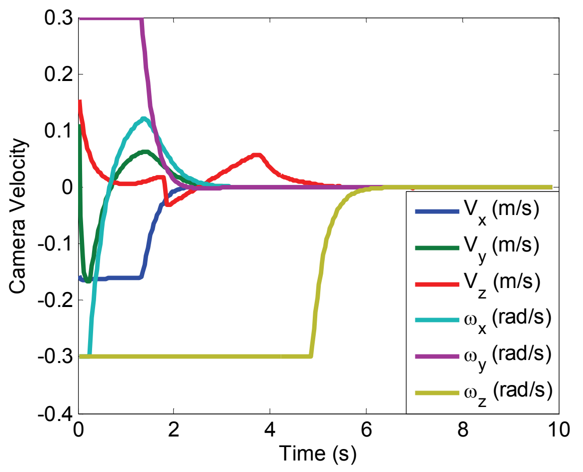
Figure 5.3 First simulation: Visual servoing via first hybrid controller with velocity penalties, (a) Image space, (b) Cartesian space, (c) Camera velocities, and (d) Error energy.



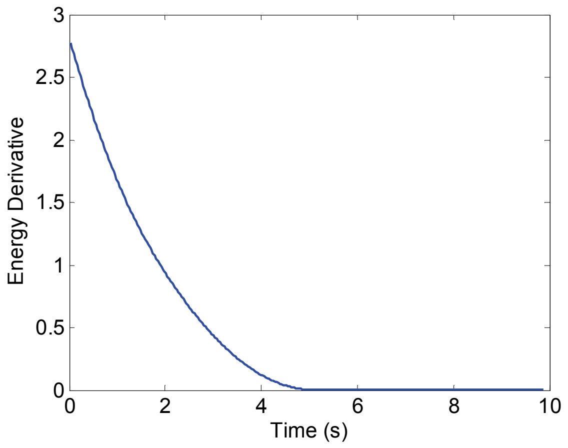
(a)



(b)



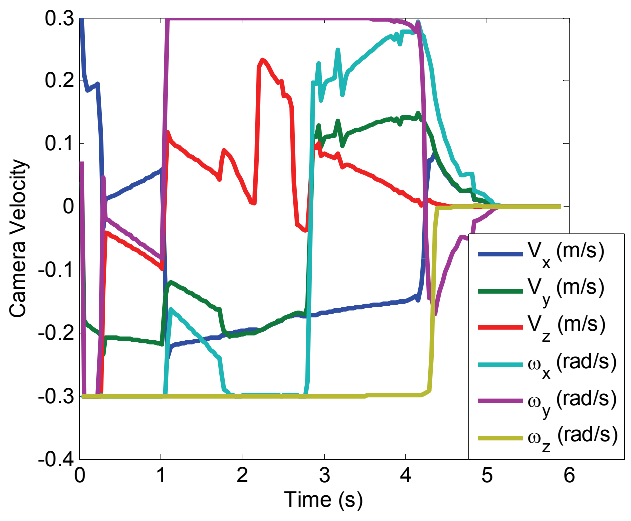
(c)



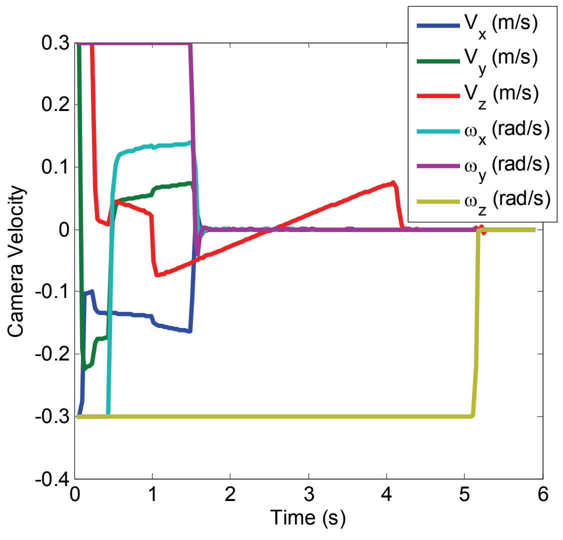
(d)

Figure 5.4 First simulation: Visual servoing via second hybrid controller, (a) Image space, (b) Cartesian space, (c) Camera velocities, and (d) Error energy.

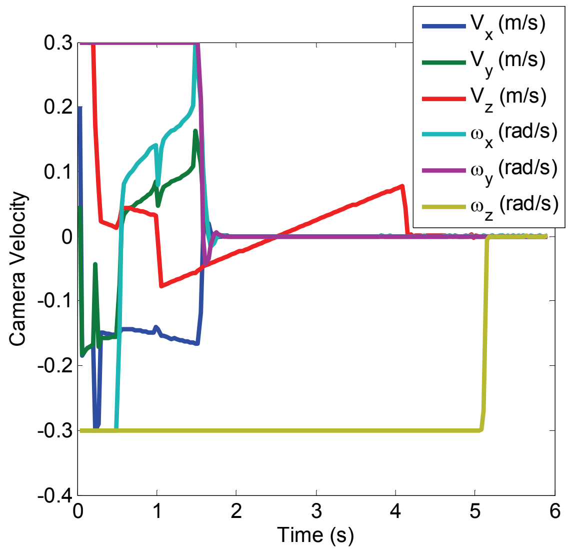
In the second simulation, the role of velocity penalizing in the cost function is investigated. The goal is to show how removing the velocity penalty affects the system trajectories. For this matter, the proposed controllers are tested without any velocity penalties (i.e., $\Upsilon = 0$) and their camera velocities are compared with those of image-based MPC. The results of this comparison are demonstrated in Figure 5.5. As it can be seen, all controllers are affected by the short control horizon. Control discontinuities are caused as a matter of numerical optimization and seem to be more severe in case of image-based predictive control. Position-based predictive controller seems to be the least affected. It is also worth mentioning that all controllers have converged at a higher rate, compared to their velocity penalized counterparts.



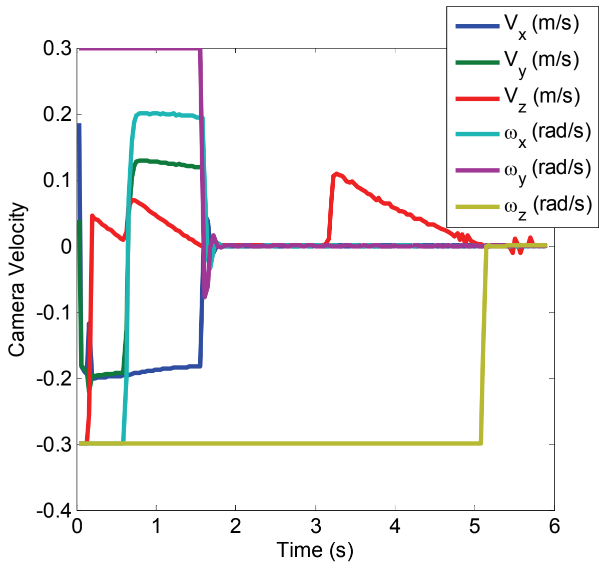
(a)



(b)



(c)



(d)

Figure 5.5 Second simulation: Camera velocities of different MPC controllers without velocity penalties, (a) image-based controller, (b) position-based controller, (c) first hybrid controller, and (d) second hybrid controller.

In the third simulation, the superiority of the proposed position-based and hybrid MPCs over their image-based competitor in terms of stability is put into test. For this matter, the camera is rotated 180 degrees around the camera's *Z* direction from its desired orientation, while its position is kept the same as the desired one. This particular pose selection causes IBVS systems to retract the camera to infinity, which will lead to system failure. All controllers are engaged separately to bring the camera back to its desired pose and their results are compared through Figures 5.6 to 5.9. As it was expected, the image-based MPC is unable of converging to the desire pose, while the position-based and hybrid MPCs converge with minimal translational move. It should be noted that the robot motion in *Z* direction comes as a result of constraint handling. This simulation confirms the improvement of system's stability in the proposed controllers, compared to image-based MPC.

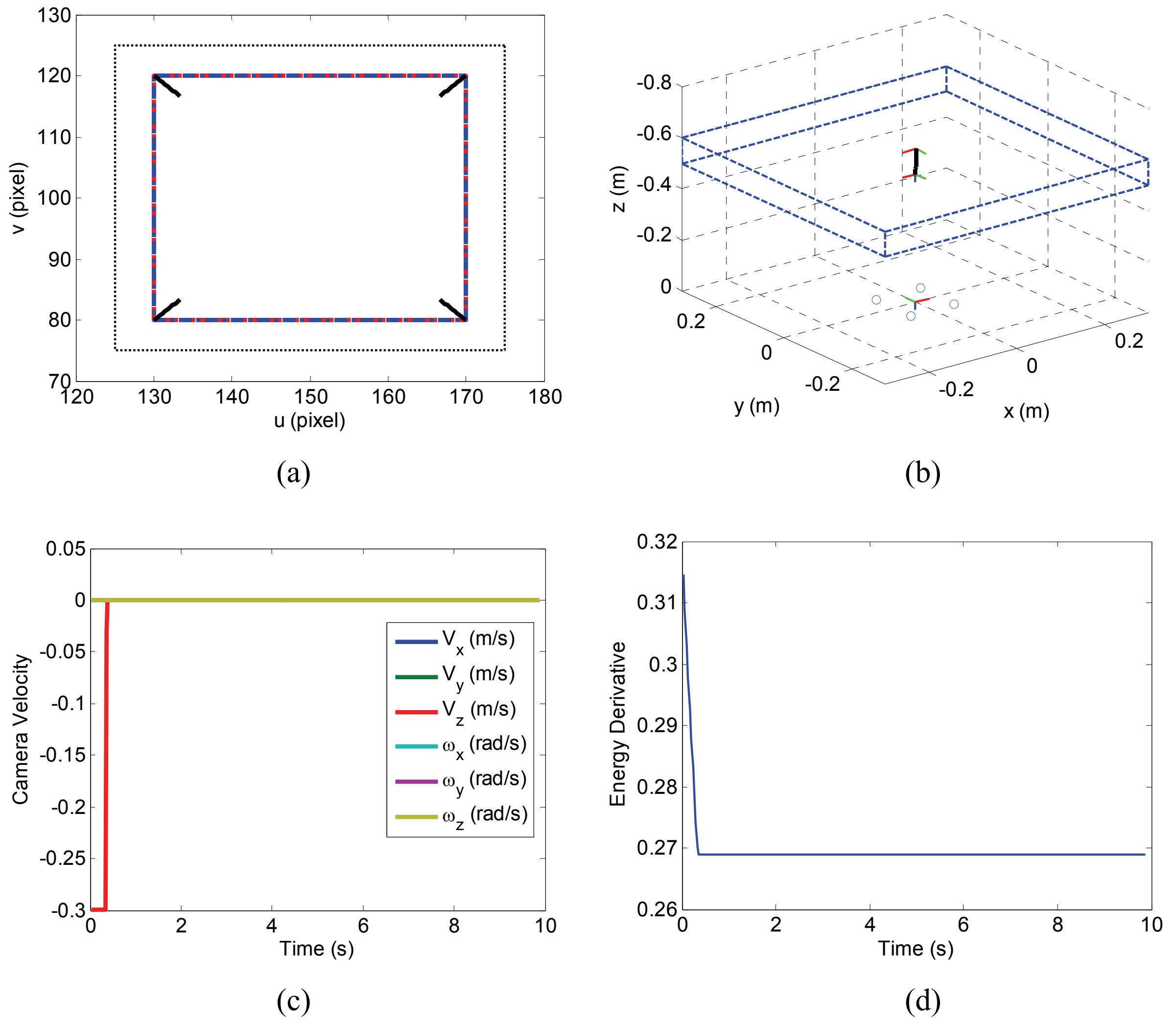


Figure 5.6 Third simulation: Visual servoing via image-based MPC with a half turn around the camera axis, (a) Image space, (b) Cartesian space, (c) Camera velocities, and (d) Error energy.

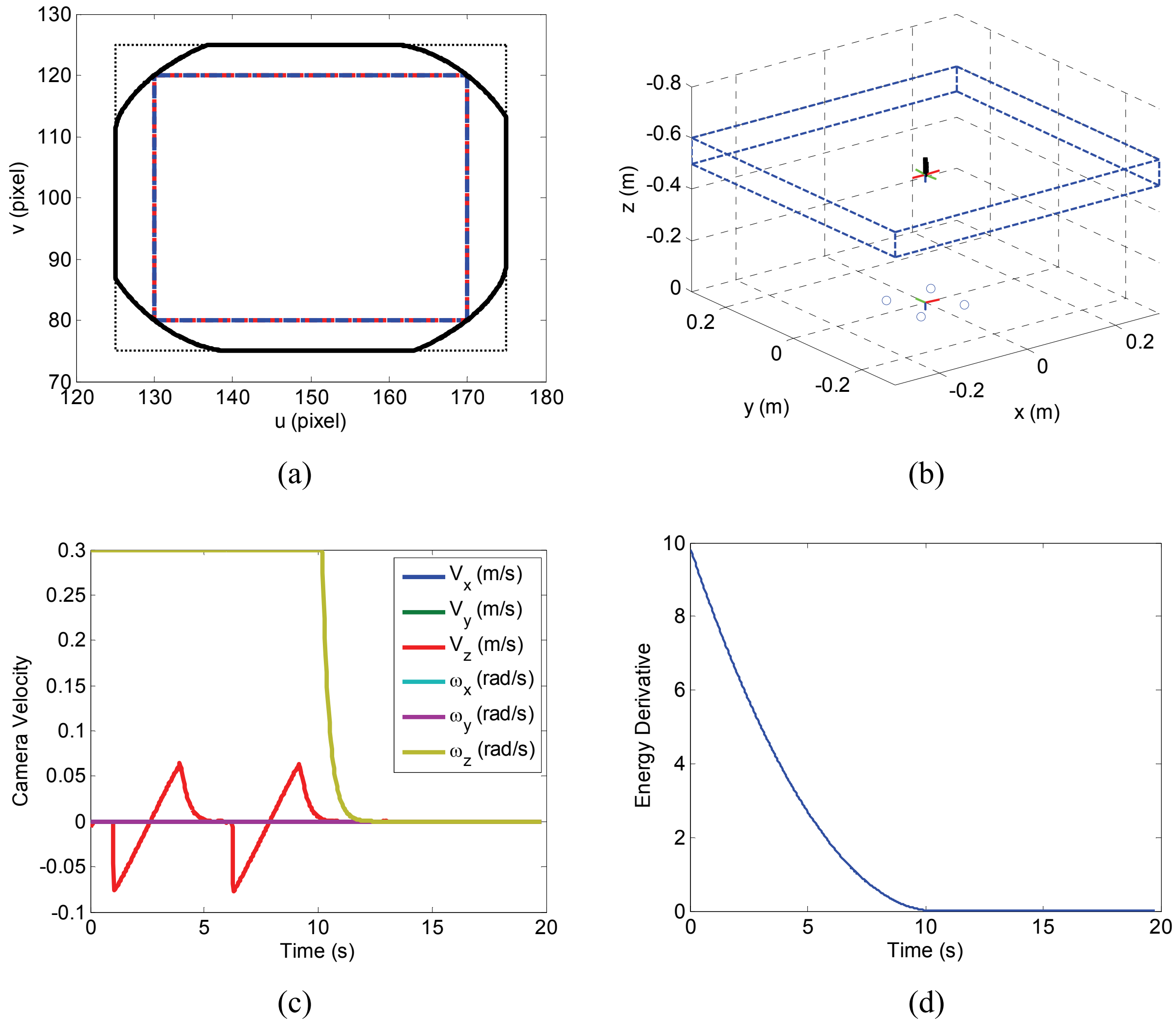
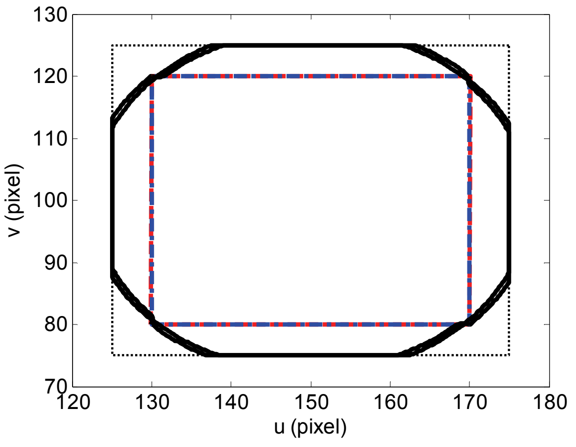
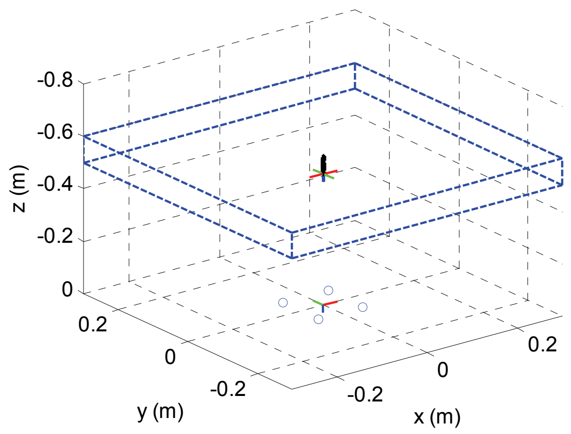


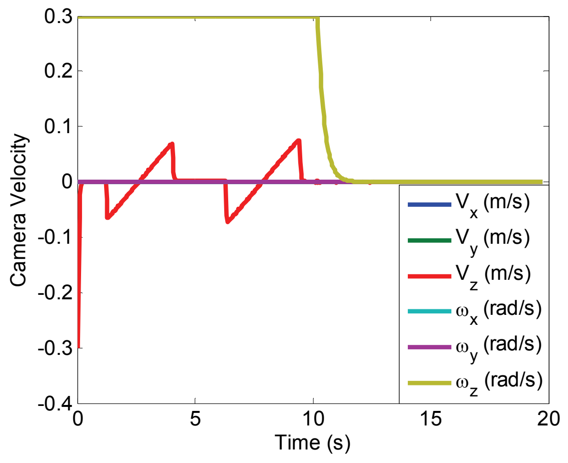
Figure 5.7 Third simulation: Visual servoing via position-based MPC with a half turn around the camera axis, (a) Image space, (b) Cartesian space, (c) Camera velocities, and (d) Error energy.



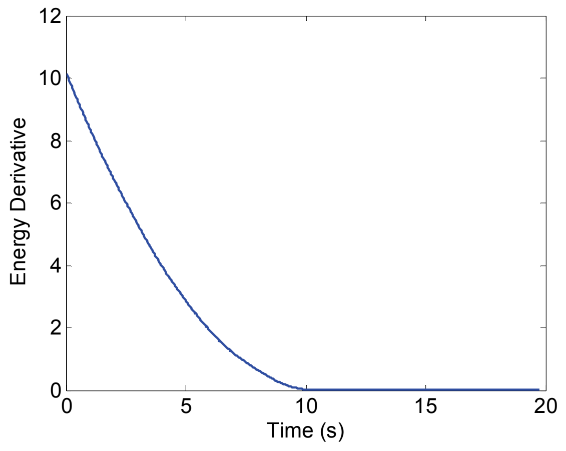
(a)



(b)



(c)



(d)

Figure 5.8 Third simulation: Visual servoing via first hybrid MPC with a half turn around the camera axis, (a) Image space, (b) Cartesian space, (c) Camera velocities, and (d) Error energy.

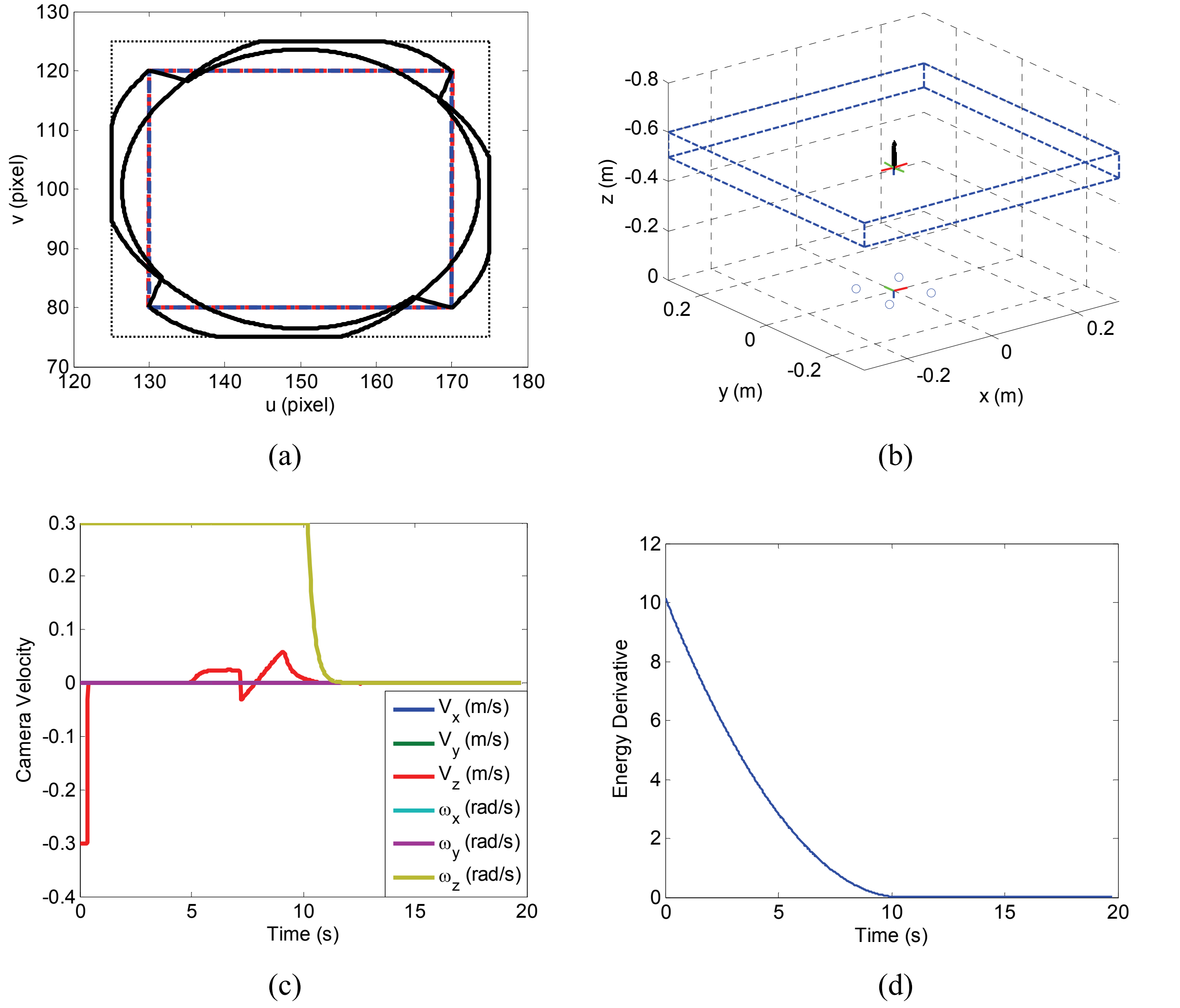


Figure 5.9 Third simulation: Visual servoing via second hybrid MPC with a half turn around the camera axis, (a) Image space, (b) Cartesian space, (c) Camera velocities, and (d) Error energy.

5.5.2 Decoupled Controller

In this subsection, the performance of the proposed decoupled controller is verified through multiple simulations and experiments. In the simulation 4, the effect of controller gains over image features is investigated through a Monte Carlo simulation with 1000 samples. A single feature system is chosen for this purpose. The initial value of the feature is selected as 10. The mean and the covariance of feature error are both selected to be 0.1. The traditional single DOF proportional controller is employed to control the feature towards the desired value, which is selected as zero. The mean and covariance of the feature are shown in Figure 5.10 for three different gain values. As it can be inferred, the convergence rate, steady-state error, and error covariance are all changing with the gain.

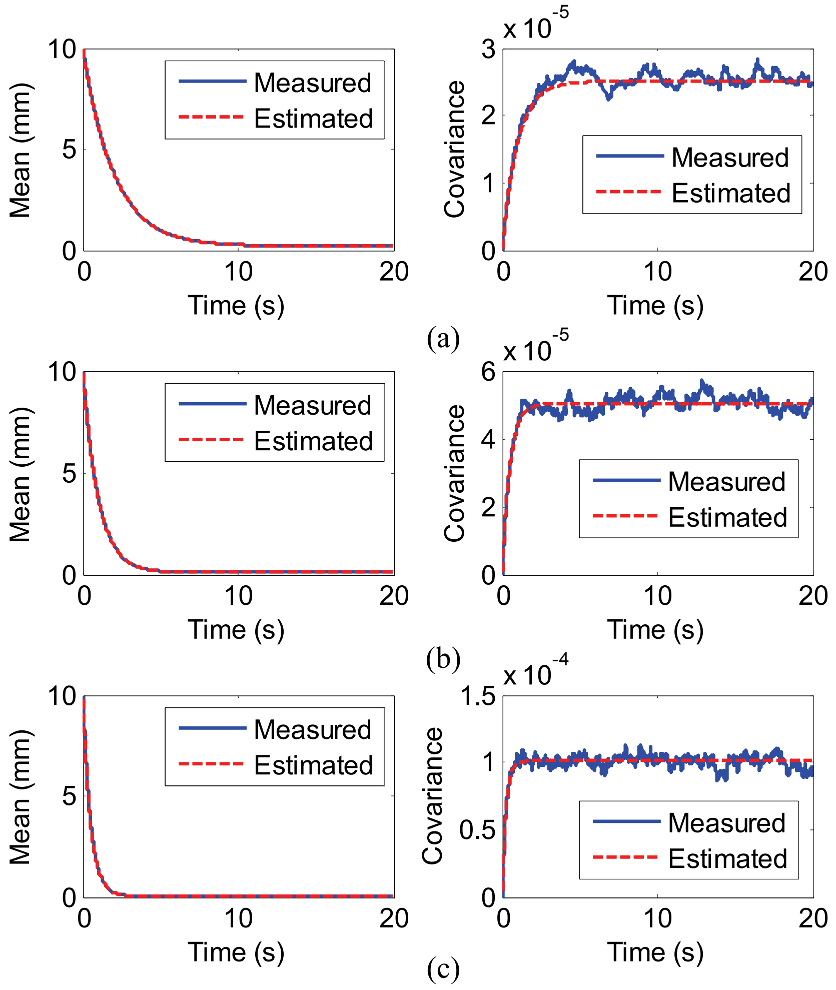
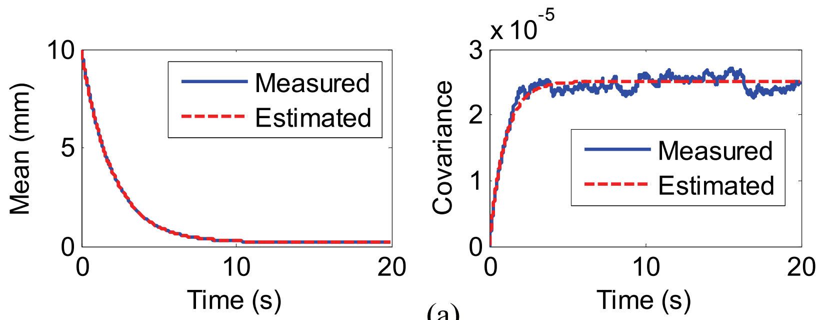
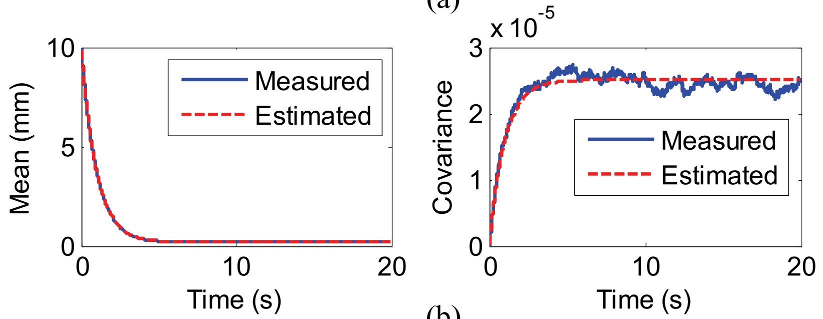


Figure 5.10 Fourth simulation: The mean and covariance of single feature, using a traditional proportional controller with (a) $\lambda = 0.5$, (b) $\lambda = 1$, and (c) $\lambda = 2$.

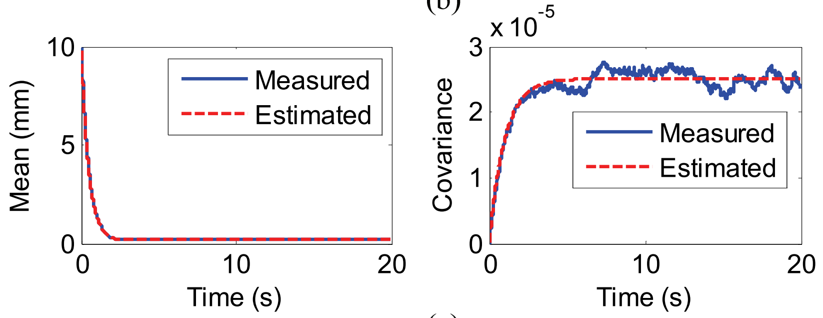
Next in simulation 5, the proposed decoupled controller is employed for the same purpose as simulation 4. To demonstrate the role of the offline controller, the gain of this controller is altered while keeping the online gain constant ($\lambda_2 = 0.5$) in the fifth simulation. The changes of mean and covariance of the feature are shown in Figure 5.11. It is clear that the rate of convergence changes with the offline control gain, while the steady-state error and error covariance are remaining unchanged due to the online gain being constant.



(a)



(b)



(c)

Figure 5.11 Fifth simulation: The mean and covariance of single feature, using the proposed decouple controller with constant online gain ($\lambda_2 = 0.5$) and variable offline gain, (a) $\lambda_1 = 0.5$, (b) $\lambda_1 = 1$, and (c) $\lambda_1 = 2$.

In the sixth simulation, the mean and covariance of the feature is calculated for multiple online gains, having the offline gain constant ($\lambda_1 = 0.5$). The results of this experiment are shown in Figure 5.12. It is obvious that the rate of convergence remains the same, while the steady-state error and error covariance change due to modification of the online controller gain. Though these parameters are decoupled and can be designed separately. Moreover, it can be inferred from Figures 5.10-5.12 that the mean and covariance of the feature are estimated correctly.

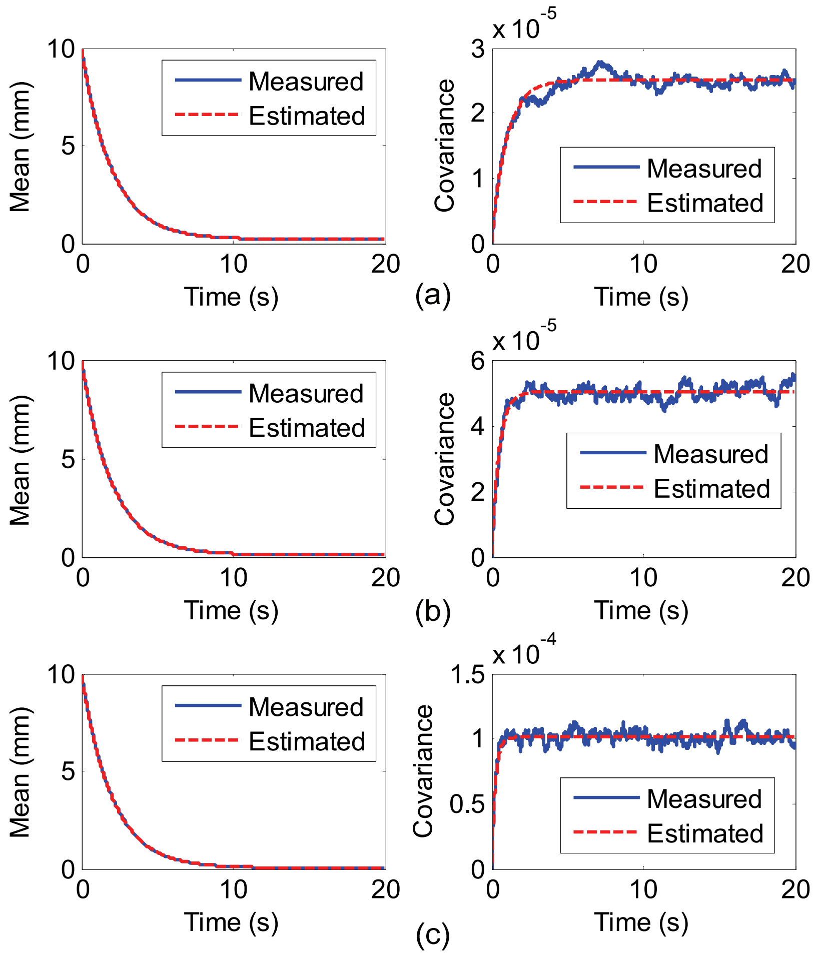
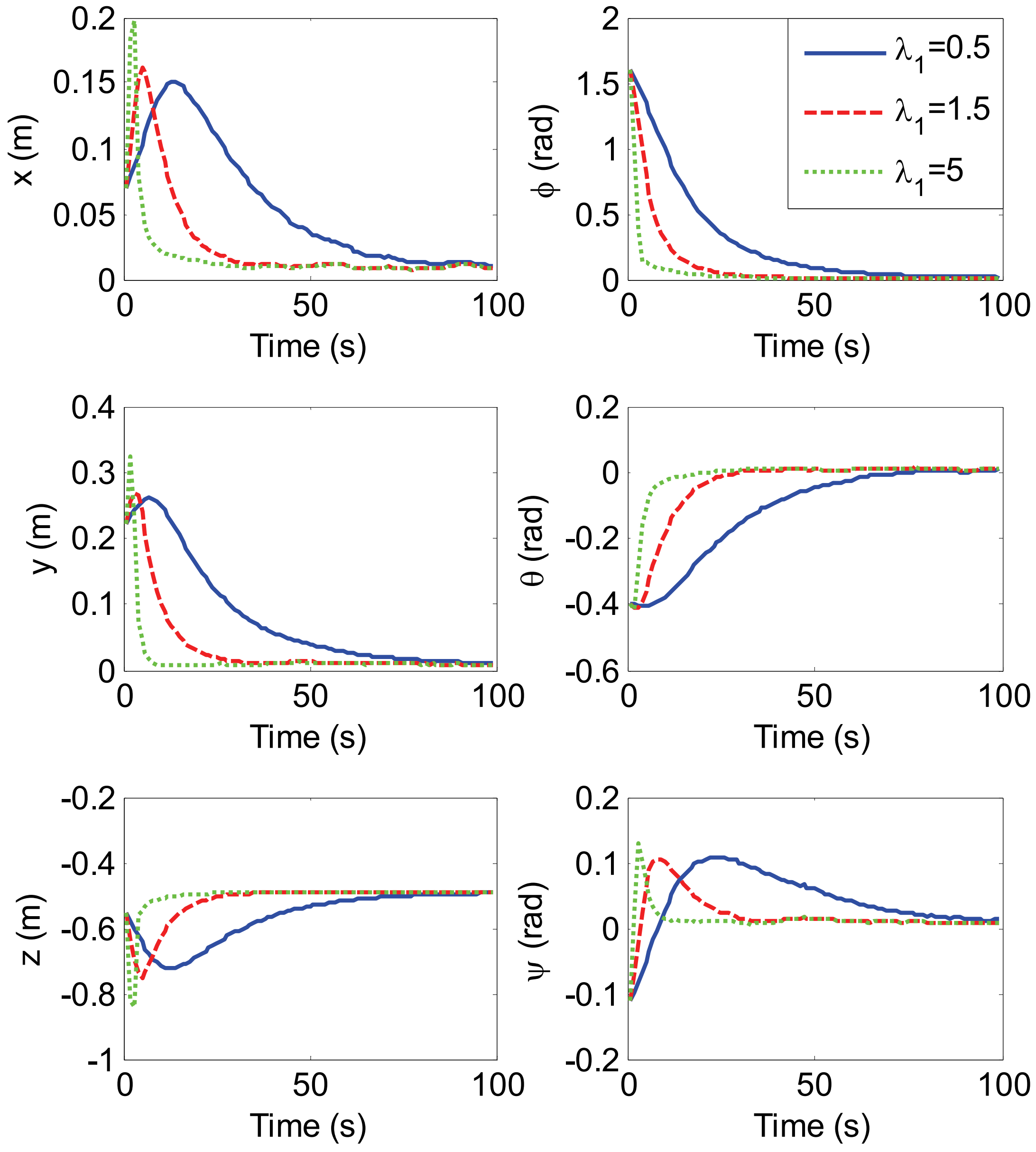
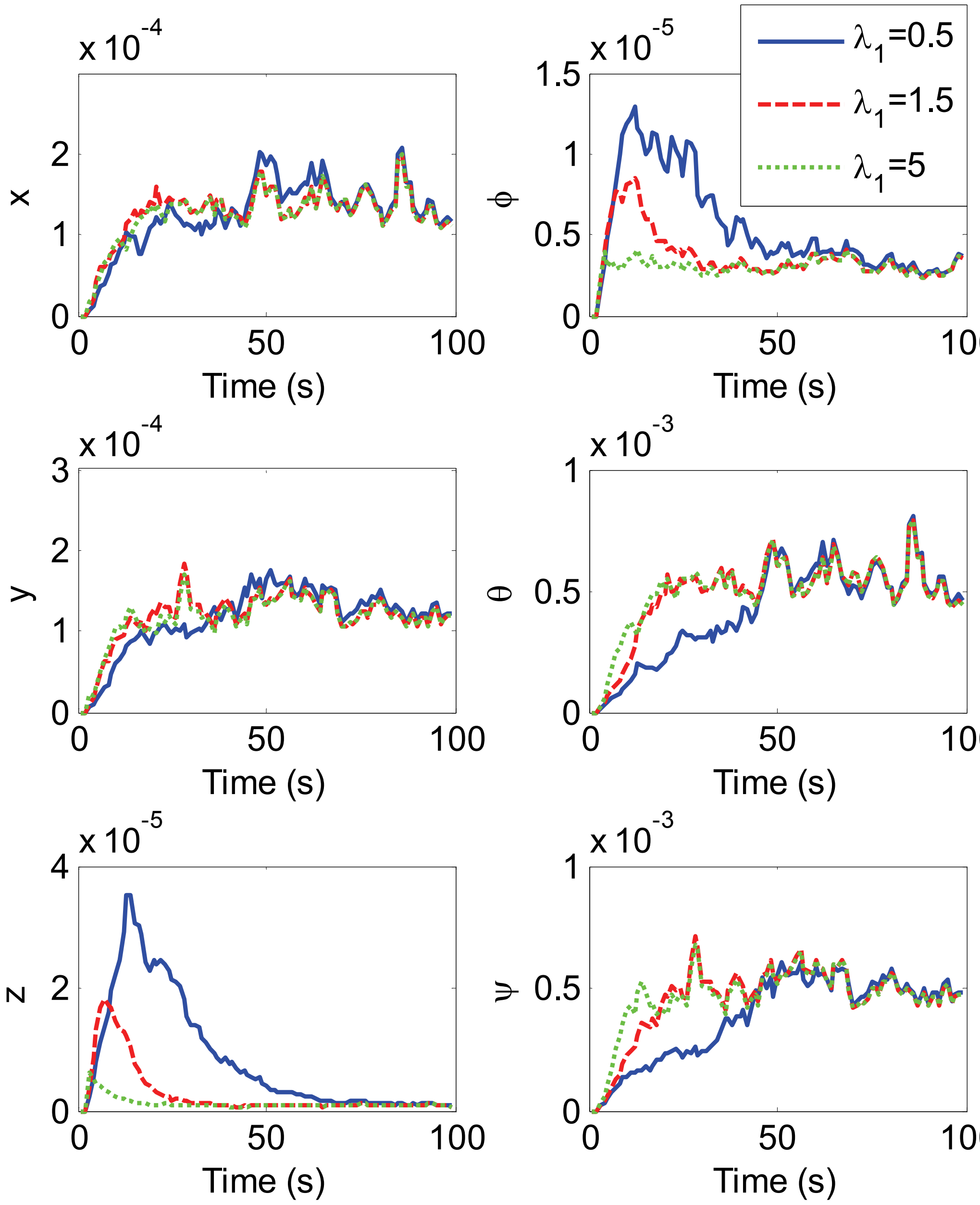


Figure 5.12 Sixth simulation: The mean and covariance of single feature, using the proposed decouple controller with constant offline gain ($\lambda_1 = 0.5$) and variable offline gain: (a) $\lambda_2 = 0.5$, (b) $\lambda_2 = 1$, and (c) $\lambda_2 = 2$.

In the seventh simulation, the proposed controller is used in a visual servoing scenario. The desired pose of the camera-mounted robot is selected half a meter away from the object. Four point features are selected on the object and are exploited for servoing. The goal of simulations 7 and 8 is to show the decoupling effect of the proposed controller. Initially, the gain of the offline controller is changed, while the gain of the online controller is kept constant. The mean and covariance of features is shown in Figure 5.13. As it can be inferred, the convergence rate of the features changes with the offline gain. As it was predicted, the steady-state error remains unchanged, while the covariance of the features changes slightly, due to the change of Jacobian matrix. Nevertheless, the steady-state of the covariance is fixed as well.



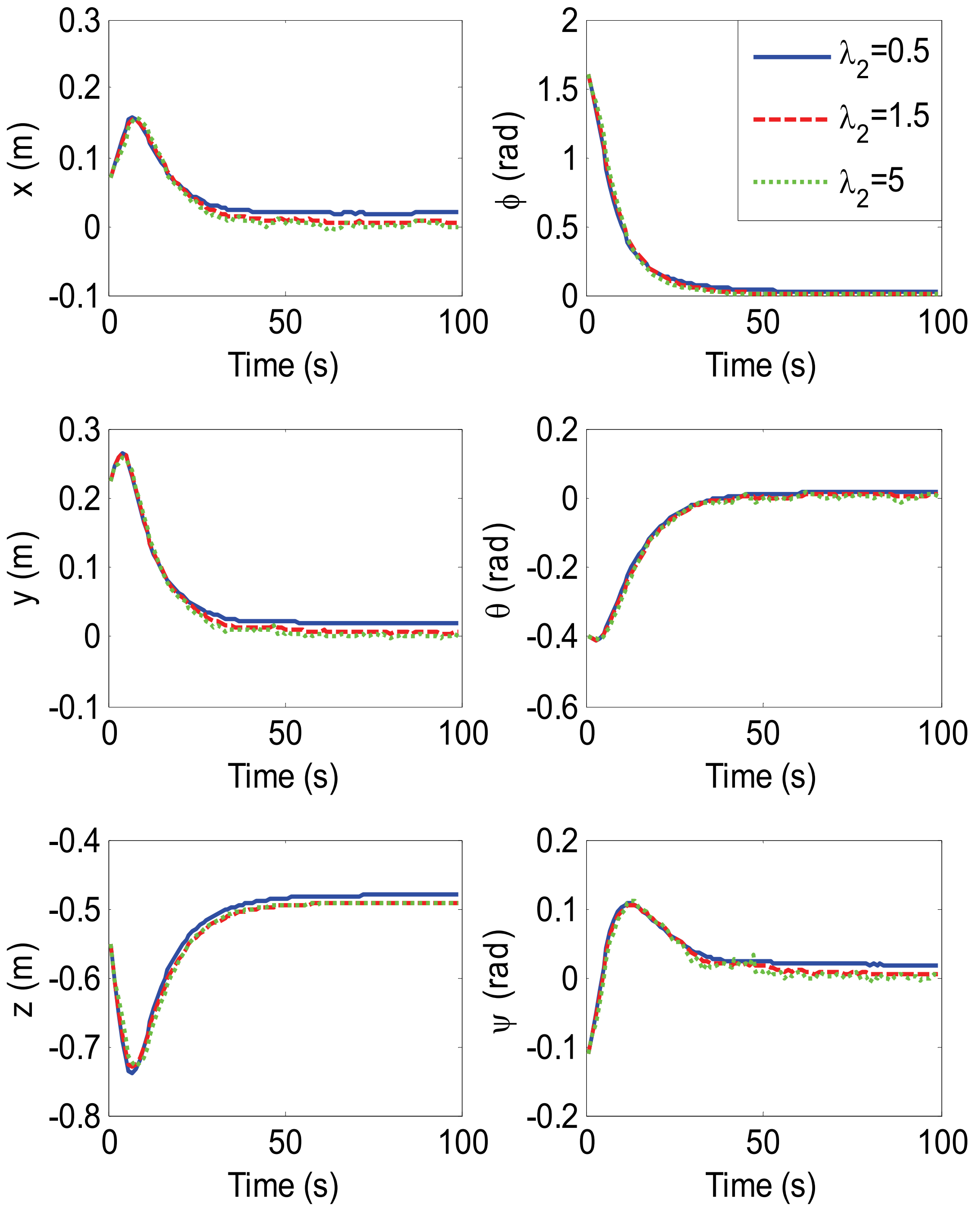
(a)



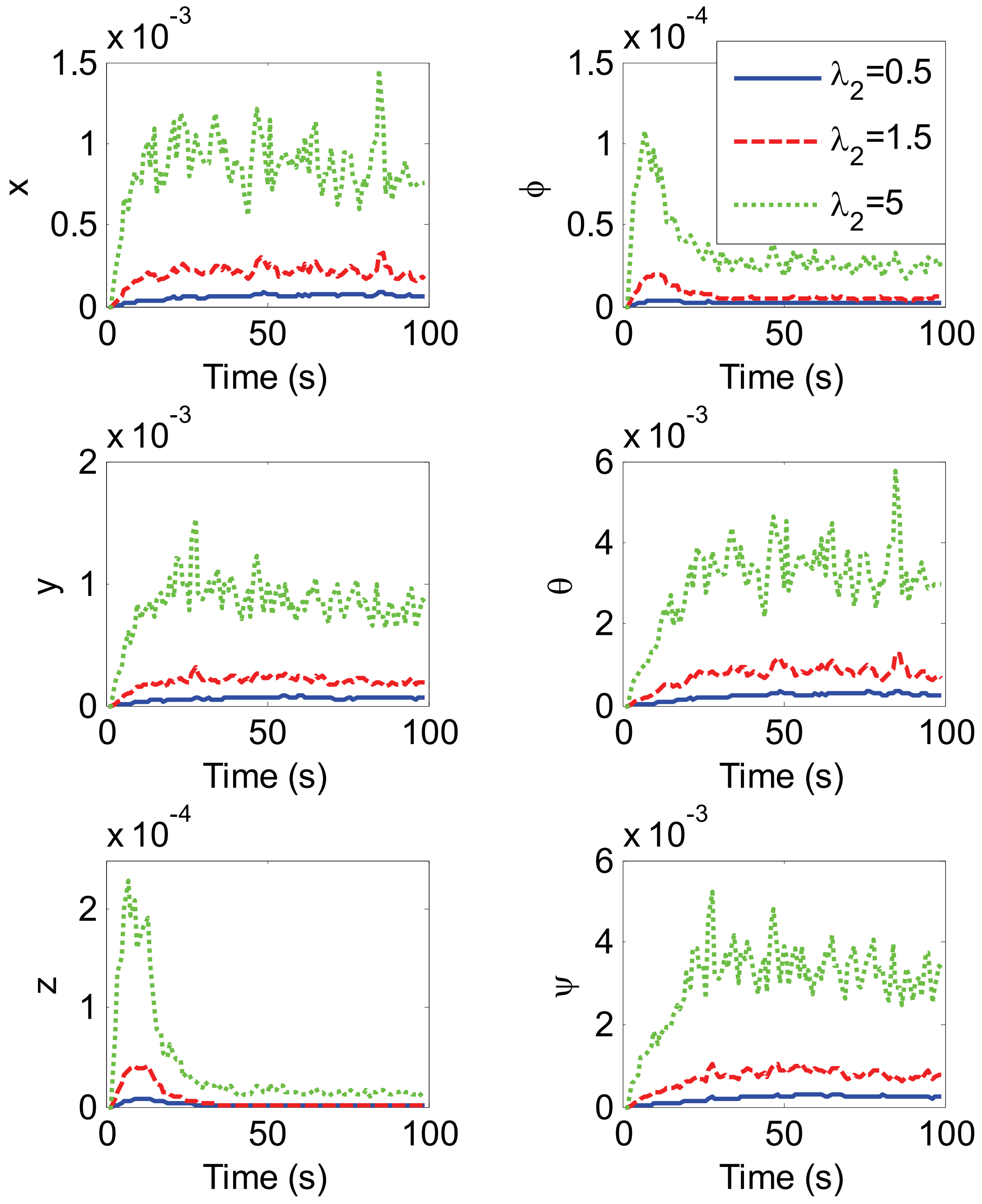
(b)

Figure 5.13 Seventh simulation: The statistical measurements of visually servoed system, using the proposed controller with different offline controller gains, (a) mean of the pose parameters, and (b) covariance of the pose parameters.

Next, the gain of the offline controller is kept constant and the online gain is varied in the eighth simulation. The results of these changes on the mean and covariance of the features are demonstrated in Figure 5.14. As it was expected, the rate of convergence remains the same, while the steady-state error decreases and the error covariance increases as the online gain rises.



(a)



(b)

Figure 5.14 Eigth simulation: The statistical measurements of visually servoed system, using the proposed controller with multiple online controller gains, (a) mean of the pose parameters, and (b) covariance of the pose parameters.

Next, the decoupling properties of the proposed controller are demonstrated through experiments. The experimental setup used for this matter is composed of a 6-degree of freedom

(DOF) robot from Denso robotics (Long beach, CA, USA), equipped with a *Firefly* camera from Point Grey (Richmond, BC, Canada). The robot operates in open-architecture mode, which is made possible through MATLAB® SIMULINK® from Mathworks (Natick, MA, USA) and the Quarc® control software from Quanser (Markham, ON, Canada). The camera is working with the speed of 60 fps, and the maximum velocity of the robot is 3900 mm/s, which is reachable at its end-effector. An object with four circles (as features) is used for servoing. The experimental setup is shown in Figure 5.15.

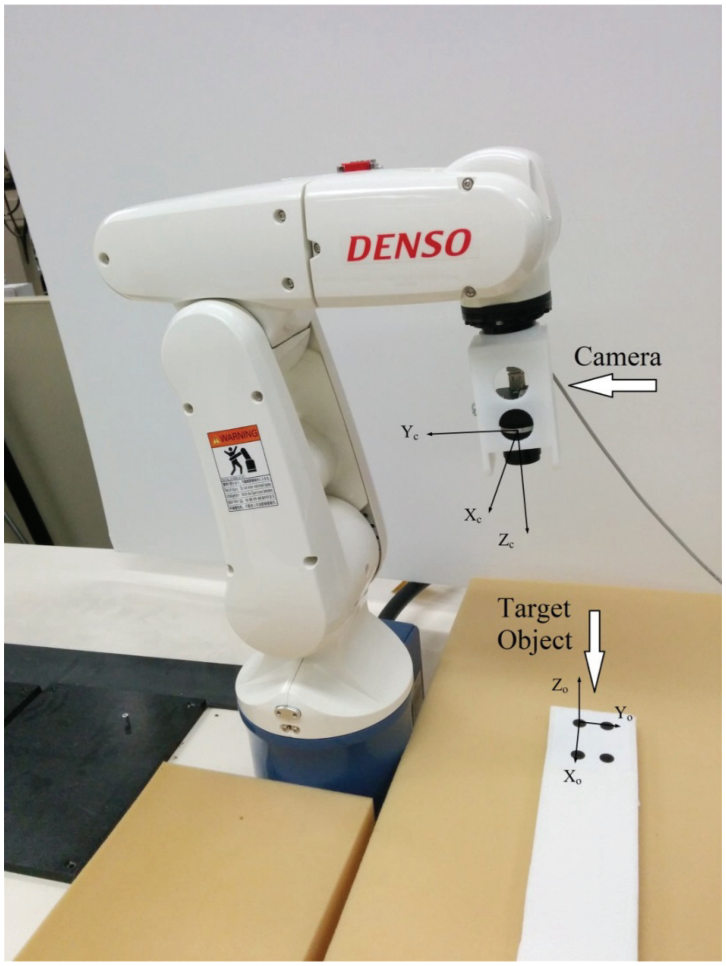
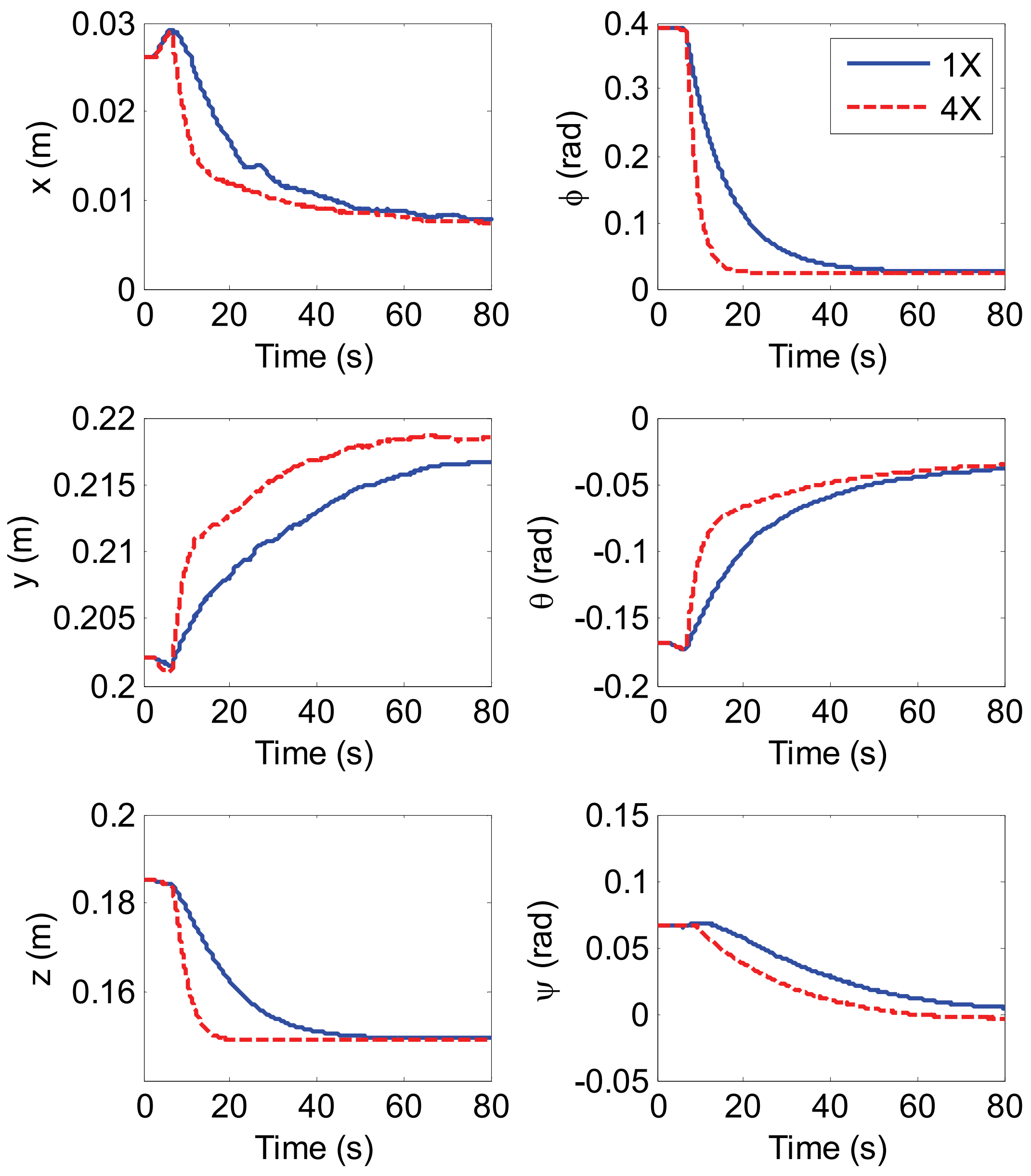
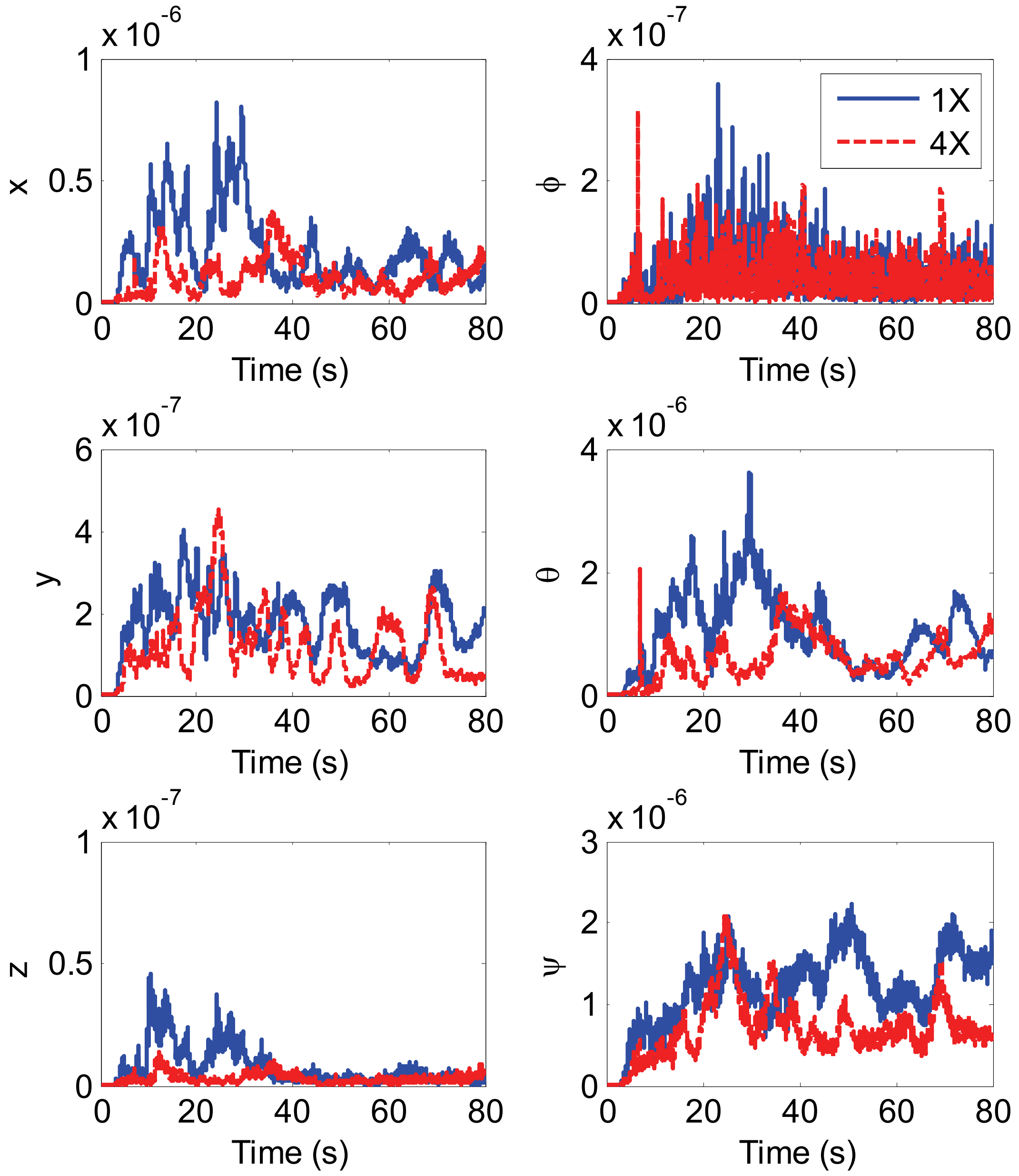


Figure 5.15 The experimental setup.

In the first experiment, the effect of offline gain in the proposed controller is tested. For that matter, the online gain of the controller is kept constant, while the offline gain is increased four times. The camera is servoed to almost 40 cm above the object and the experiment is repeated 10 times. The mean and covariance of the camera pose are depicted in Figure 5.16. It can be seen that the change of offline gain results in faster convergence, while the covariance of the system remains almost the same.



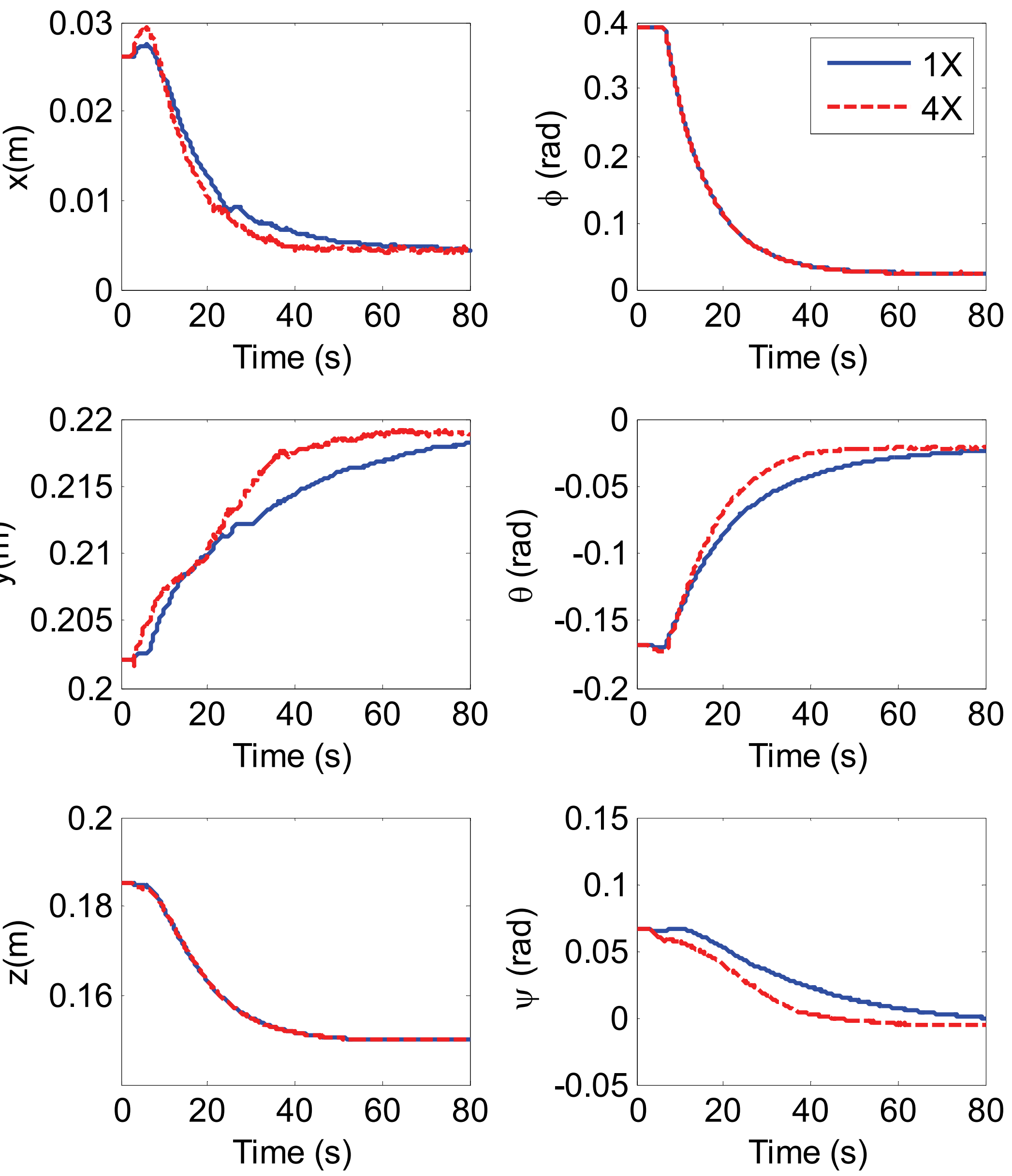
(a)



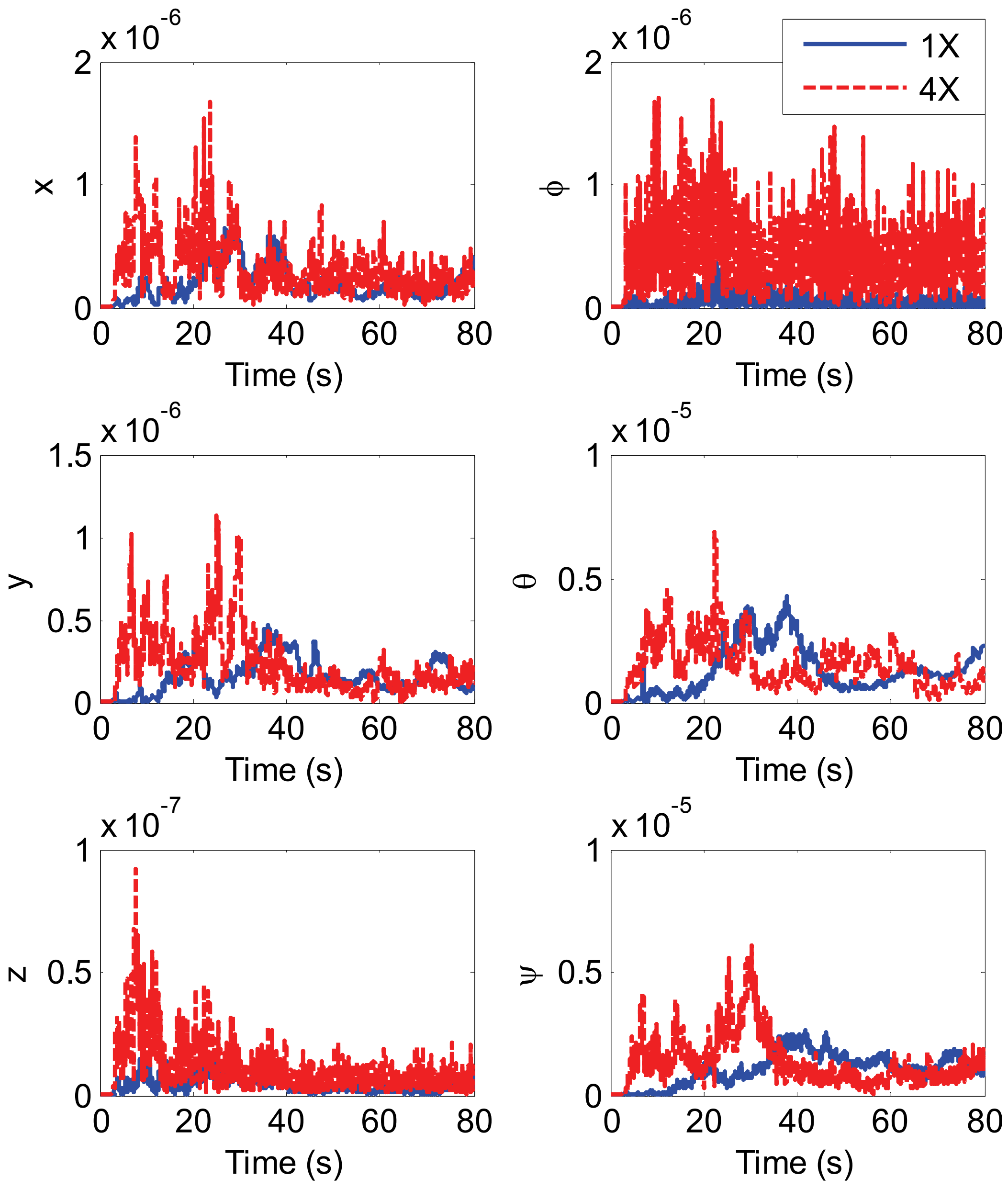
(b)

Figure 5.16 First experiment: The statistical measurements of visually servoed system, using the proposed controller with different offline controller gains, (a) mean of the pose parameters, and (b) covariance of the pose parameters.

In the second experiment, the effect of online gain on the mean and covariance of the camera pose is investigated. For that matter, the offline gain is selected as constant, while the online gain is increased four times. Similar to previous experiment, the camera is servoed to 40 cm above the object repeatedly for 10 times. The results of this experiment are shown in Figure 5.17. It can be seen from thos figure that the mean values of both controller are almost similar, while the covariance of the controller with increased online gain is higher. It is noticeable that the covariance of some degrees of freedom becomes similar for both cases once the camera reaches its destination. The reason behind this matter is the small changes in velocity, which are not followed by robot's motors.


x(m)
0.03
0.02
0.01
0
Time (s)
φ (rad)
1X
4X
y(m)
0.22
0.215
0.21
0.205
0.2
θ (rad)
z(m)
0.18
0.16
ψ (rad)
0
20
40
60
80

(a)



(b)

Figure 5.17 Second experiment: The statistical measurements of visually servoed system, using the proposed controller with two online controller gains, (a) mean of the pose parameters, and (b) covariance of the pose parameters.

5.5.3 Robust Controller with Online Predictive Controller

The simulation and experimental results of the robust controller with online MPC are presented in this subsection. In the ninth simulation, the features error is used for the offline controller to test the constraint handling capability. For this matter, the trajectory of the camera in Cartesian space is bounded. The camera was banned to move more than 10 cm in *x* direction, 25 cm in *y* direction, and 60 cm in *z* direction, away from the object. Moreover, the camera velocity is set to be less than 0.5 m/s for translational and 0.5 rad/s for rotational velocities. The control measurement matrix is initialized at $\Upsilon = 10^{-3} I$, where *I* is the identity matrix, and is reduced 3 percent at each time step after 3 seconds. The results of this simulation are demonstrated in Figure 5.18. The initial and final images are shown by dashed red and dot-dashed blue lines in the image space. As it can be depicted, the propose controller is fully capable of handling the given constraints, while delivering the camera to its desired pose. The camera velocities are not facing any unusual discontinuities. In addition, the time derivative of error energy is negative throughout the servoing, which is indicative of system's stability.

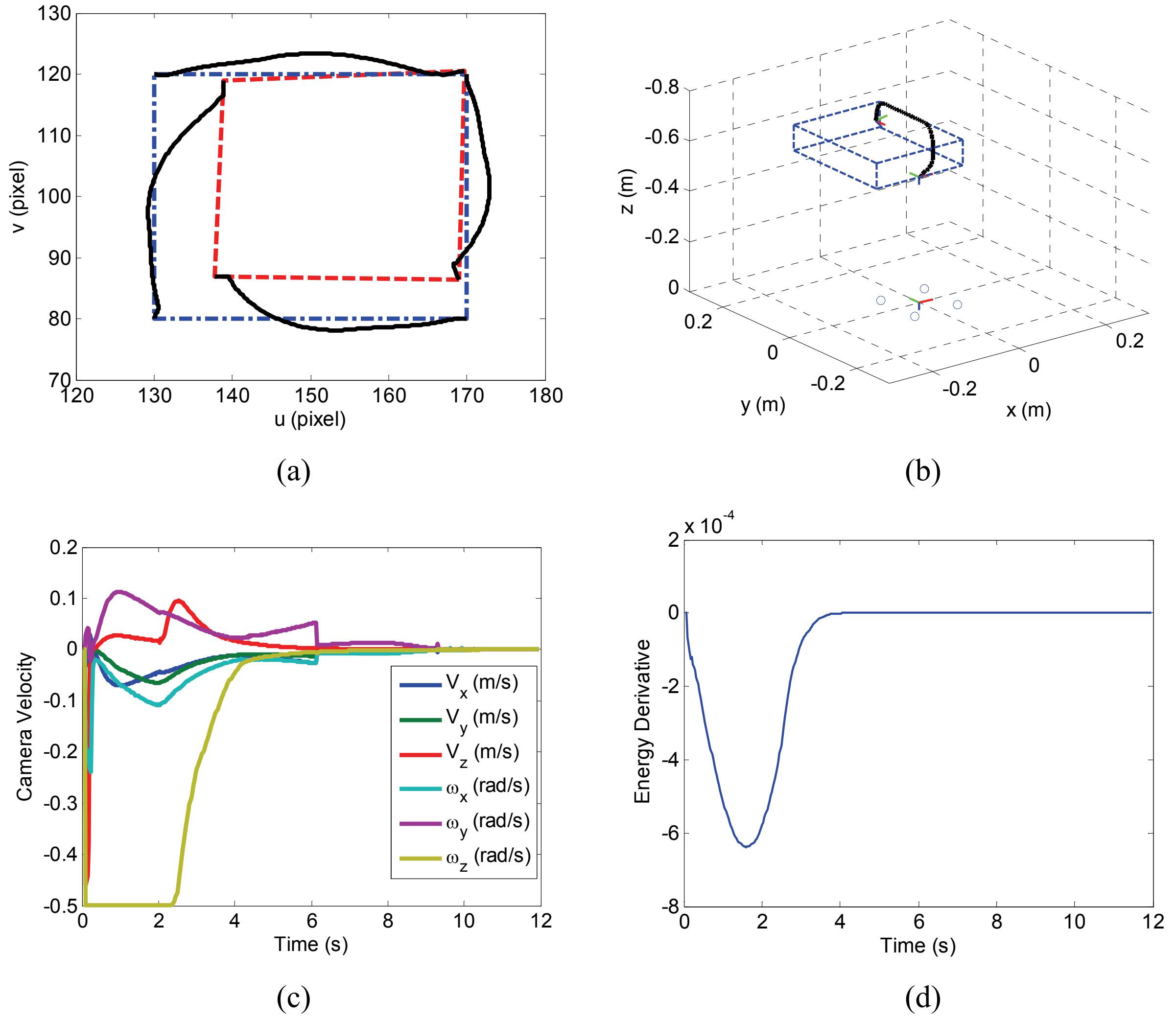


Figure 5.18 Ninth simulation: Visual servoing via two-stage controller with IBVS offline controller and online MPC, (a) Image space, (b) Cartesian space, (c) Camera velocities, and (d) Error energy time derivative.

In simulation 10, the hybrid error (e_{H_1}) is employed for the offline controller to test the constraint handling capabilities of this controller. The weighting matrices and the velocity boundaries are selected the same as the previous simulation. The image features are bounded between 125 and 175 pixels in *u* direction and between 70 and 130 in *v* direction. The results of this simulation are shown in Figure 5.19. The image boundaries are shown with the black dotted rectangle in the image space. It is clear that the image features are kept within the boundaries, while moving toward the desired features. The velocities are smoothly converging towards zero without violating the limit. Moreover, the stability of the system is shown through consistent negativity of the error energy time derivative.

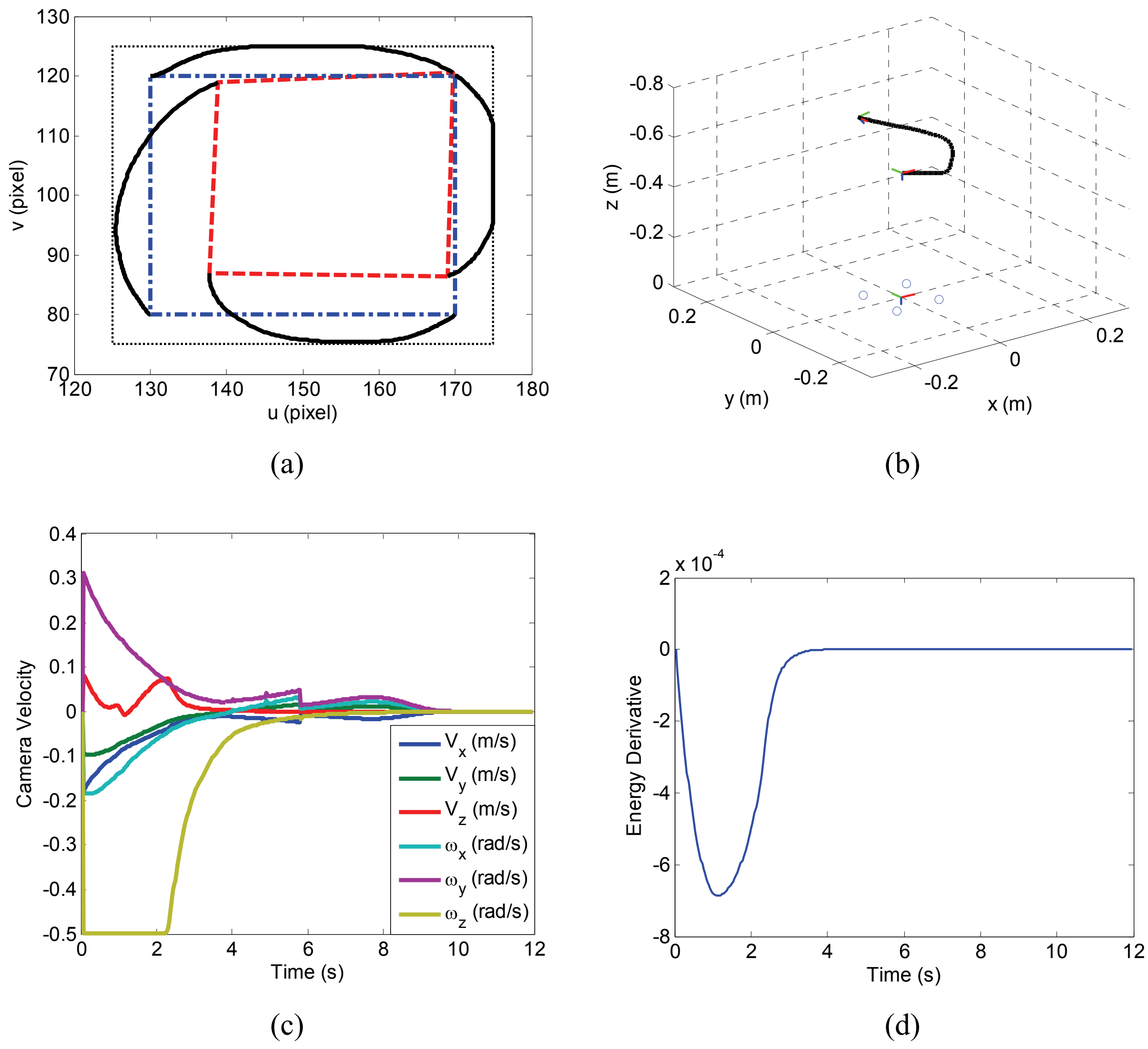


Figure 5.19 Tenth simulation: Visual servoing via two-stage controller with HVS offline controller and online MPC, (a) Image space, (b) Cartesian space, (c) Camera velocities, and (d) Error energy time derivative.

In simulation 11, the uncertainty levels of the proposed methods are compared with that of a typical IBVS to show the superiority of the proposed controller in terms of accuracy. The norm of the position error covariance is exploited as a measure of uncertainty. Two Monte Carlo simulations with 20 iterations are conducted for constrained and unconstrained visual servoing and their results are compared with that of a typical single-stage visual servoing with a gain of

0.8. The point features were used for the offline controller. The weighting matrices are selected as $\mathrm{B} = I$ and $\Upsilon = 10^{-4} I$. Figure 5.20 entails the results of this comparison. As it can be seen, the uncertainty levels of the camera position's trajectory with the proposed controllers are much less than that of a typical controller. It is also worth mentioning that the uncertainty level is dropped during the constraint handling, since the controller keeps the trajectories at the limits and therefore position variations are diminished.

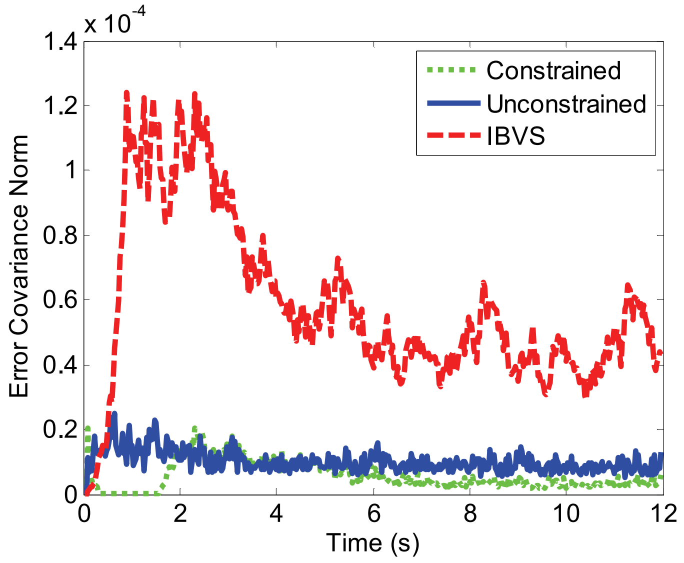


Figure 5.20 Eleventh simulation: The norm of the position error covariance for the proposed constrained and unconstrained two-stage controller with online MPC versus a simple proportional IBVS controller.

In the twelfth simulation, the effect of control weighting matrix (Υ) on the uncertainty level is investigated. Three different values are selected and the norm of position error covariance is measured in an unconstrained visual servoing scenario. The results of this simulation are depicted in Figure 5.21. From this figure, it can be inferred that increasing the weights on the control results in uncertainty decrease. However, the weights cannot be selected too high, since it prevents the system to fully converge, as discussed before.

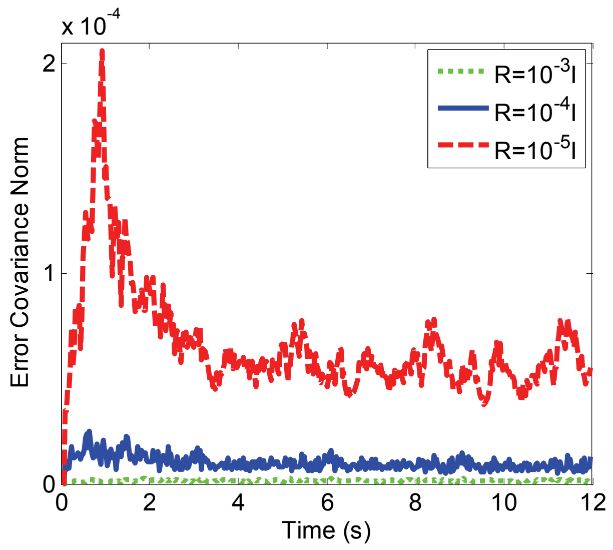


Figure 5.21 Twelfth simulation: The norm of the position error covariance for the proposed constrained and unconstrained two-stage controller with different control weightings.

In order to demonstrate the applicability of the proposed control methods in real world, the controller has been tested with experimental setup shown in Figure 5.15. In experiment 3, the accuracy of the proposed controller is compared with that of classic IBVS. For this matter, the norm of the camera position in 10 repeated servoing scenarios is measured separately for the proposed controller and IBVS. Then, the covariance of these norms is computed. Figure 5.22 demonstrates the results of this experiment. As it can be seen, the covariance of the proposed controller is much less than that of IBVS.

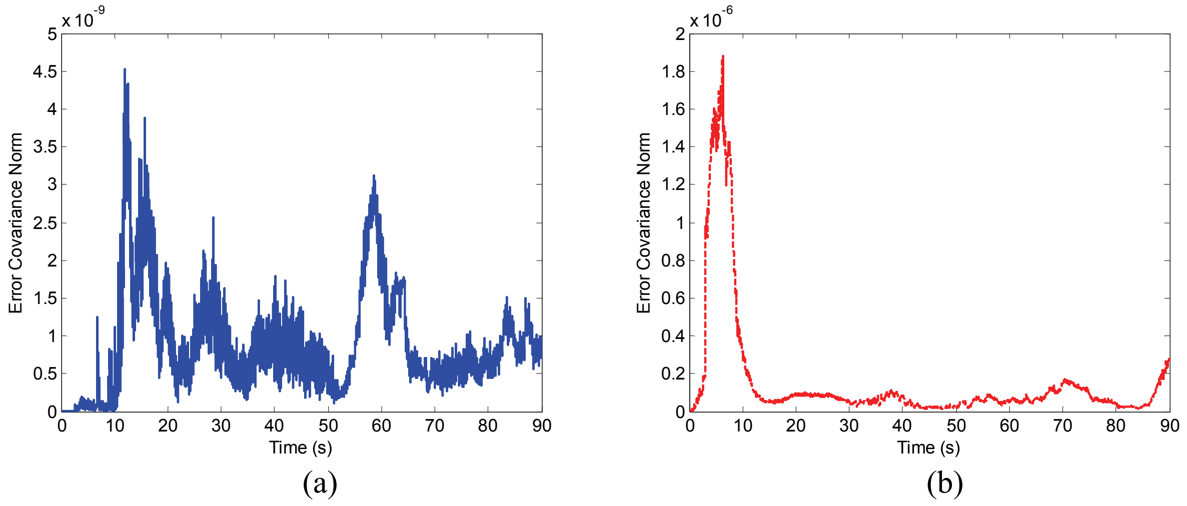
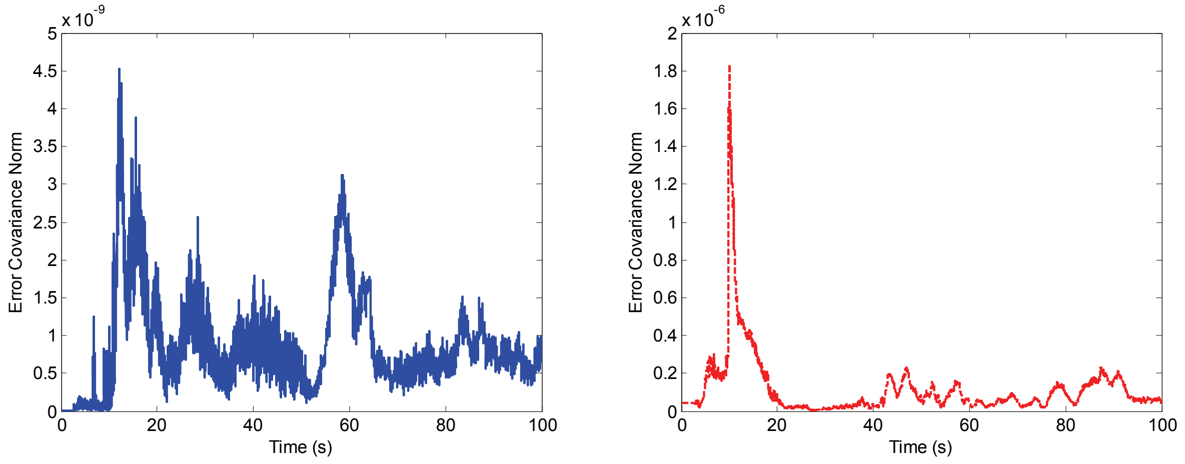


Figure 5.22 Third experiment: The norm of the position error covariance for, (a) two-stage controller with online MPC, and (b) a simple proportional IBVS controller.

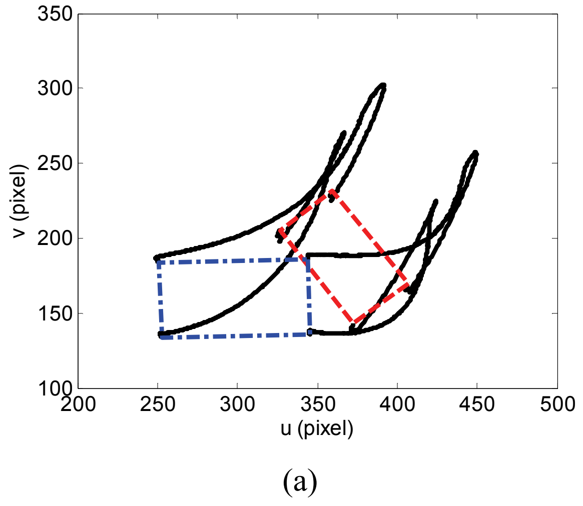
In experiment 4, the effect of velocity penalizing on the accuracy of the system is demonstrated. For that matter, the camera is servoed from a distant pose to 40 cm above the object. Two controllers with $\Upsilon = 10^{-3} I$ and $\Upsilon = 10^{-5} I$ are each used 10 times to bring the camera to its desired location. The covariance of camera position norm is then calculated. Figure 5.23 shows the results of this experiment. It is obvious that higher penalities in the controller lead to decreased covariance of the robot trajectory.



(a) (b)

Figure 5.23 Fourth experiment: The norm of the position error covariance for the two-stage controller with online MPC, (a) $\Upsilon = 10^{-3} I$, and (b) $\Upsilon = 10^{-5} I$.

In experiment 5, the camera is servoed from a distance to its desired pose, which is 15 cm above the object. The control weighting matrix is selected as $\Upsilon = 10^{-3} I$ to reduce the image noise effects. Image features servoing error is used for the offline controller. The results of this maneuver are presented in Figure 5.24. As it can be seen, camera is successfully guided towards the desired pose with a smooth motion. This is mainly due to proper selection of Υ matrix.



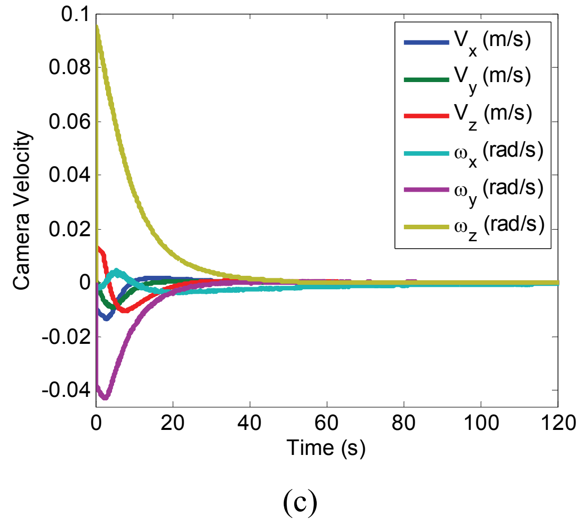
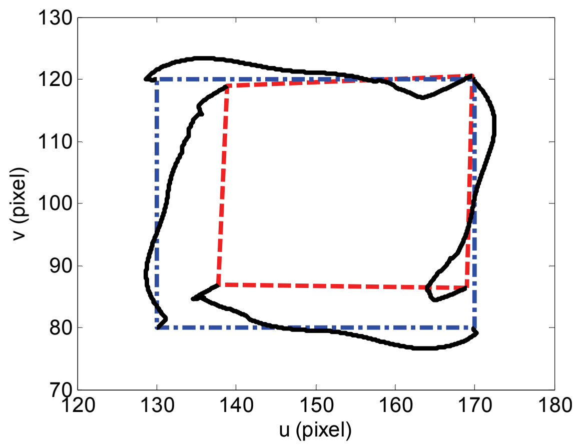


Figure 5.24 Fifth experiment: Visual servoing via two-stage controller with offline MPC, (a) Image space, (b) Cartesian space, and (c) Camera velocities.

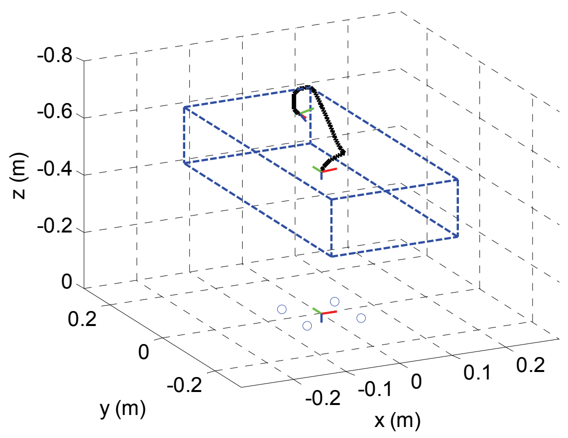
5.5.4 Robust Controller with Offline Predictive Controller

In this subsection, the effectiveness of the robust control method with offline MPC is put into test. In simulation 13, the MPC is employed initially to form the reference trajectories. The image features are not constrained; however, the camera's distance from the object is bounded. The weighting matrices, $\mathbf{B}$ and Υ, are selected to be the identity matrix and one tenth of the identity matrix, respectively. The velocity weighting matrix is gradually decreased to allow the system to fully converge. The goal of this simulation is to show the performance of this controller in presence of system constraints. The results of this simulation are depicted in Figure

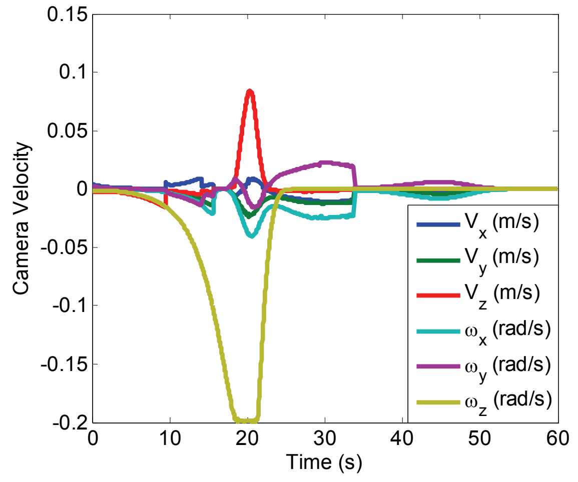
5.25. The image trajectories are smooth and the Cartesian trajectory is bounded as expected. The camera velocities are mostly smooth and converge to zero as the camera approaches its desired pose.



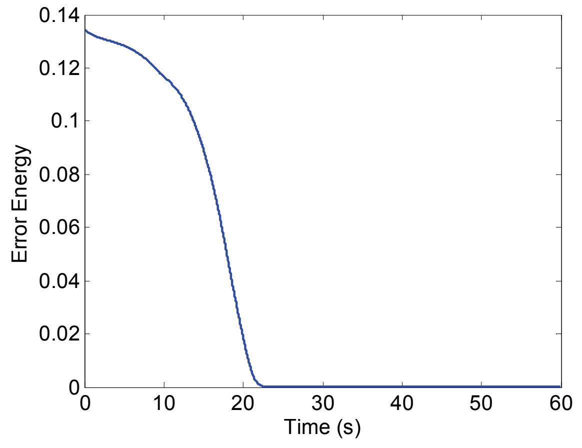
(a)



(b)



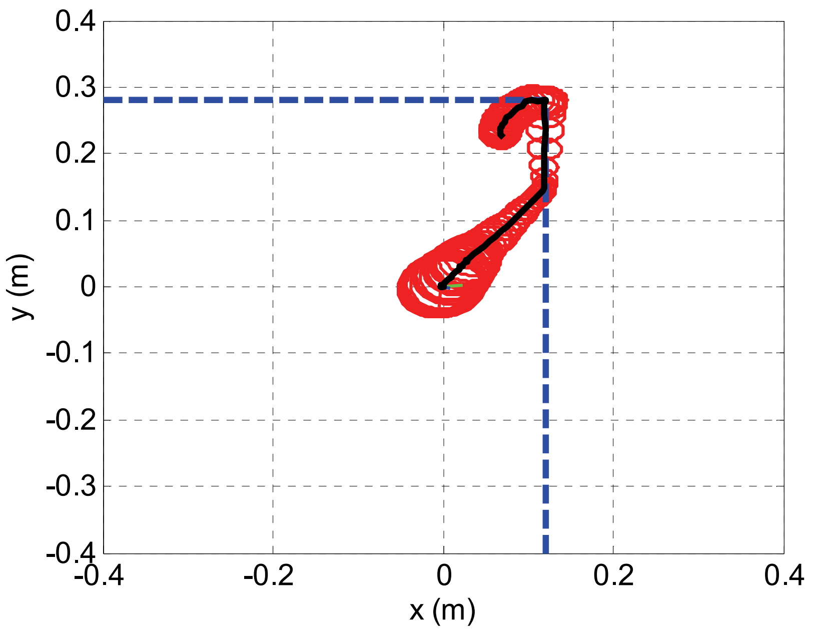
(c)



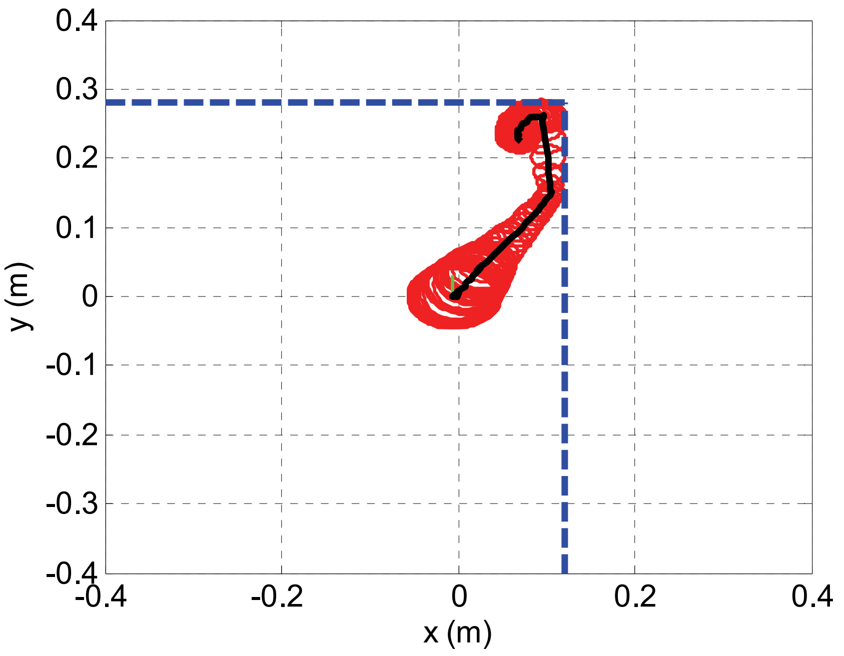
(d)

Figure 5.25 Thirteenth simulation: Visual servoing via two-stage controller with offline MPC, (a) Image space, (b) Cartesian space, (c) Camera velocities, and (d) Error energy.

In the fourteenth simulation, the effect of constraint tightening in presence of system uncertainty is demonstrated. First, the system with ordinary constrained MPC is exposed to the image noise. Next, the system with the proposed constraint handling is tested under the same conditions. Both systems are simulated for 100 times and their trajectories are compared. The results of this comparison are presented in Figure 5.26. In this figure, the position of the camera is shown in two dimensions for better demonstration of constraint handling. The tubes encompassing the uncertain camera position trajectories are shown by the red ellipsoids and the mean of these trajectories are shown by the solid black line. The constraints are shown by dashed blue lines. As it can be seen, only the mean of trajectories of the system with ordinary MPC satisfies the constraints. In fact, many of the trajectories of such system that are included in the red tubes have violated the constraints. On the other hand, none of the trajectories of the system with the proposed controller has violated the boundaries of the system, due to correct uncertainty estimation and constraint handling.



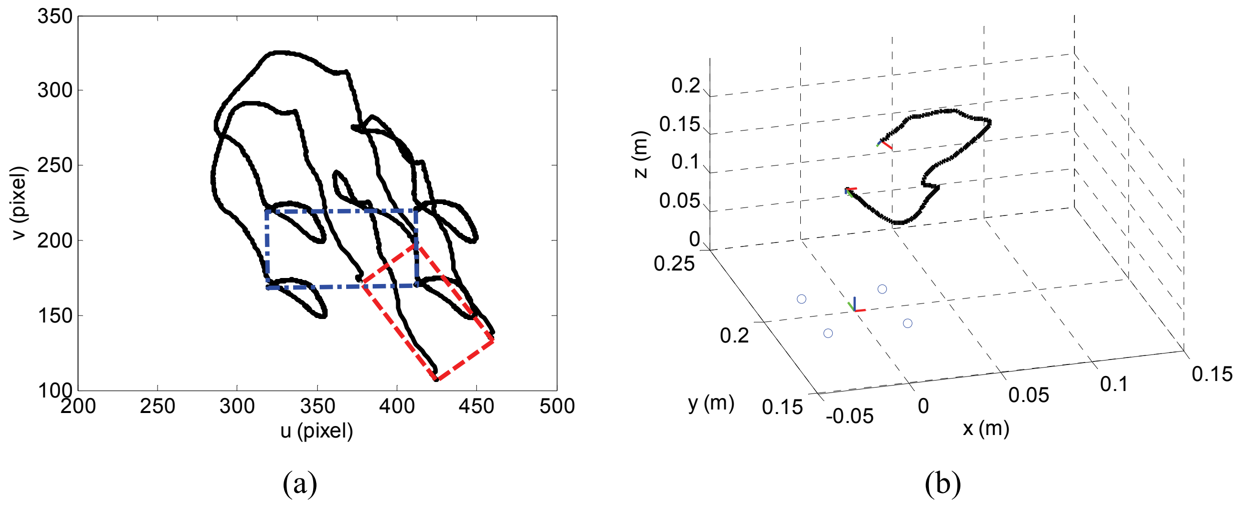
(a)



(b)

Figure. 5.26 Fourteenth simulation: Two dimensional view of Constraint visual servoing in the presence of uncertainties with, (a) uncompensated controller, and (b) the proposed controller.

In the experiment 6, the camera is moved from its desired pose and is returned back to the desired pose through the proposed visual servoing controller. The same experimental setup used for RCONPC is exploited for that matter. The camera is bound not to be higher than 0.2 meters above the object. The weighting matrices are similar to the first simulation and the proportional gain is selected as 0.1. The results of this maneuver are demonstrated in Figure 5.26. As it can be inferred, the controller is fully capable of guiding the robot to its desired pose, while handling the given constraints. The camera velocities are mostly smooth and converge to zero, similar to the simulations.



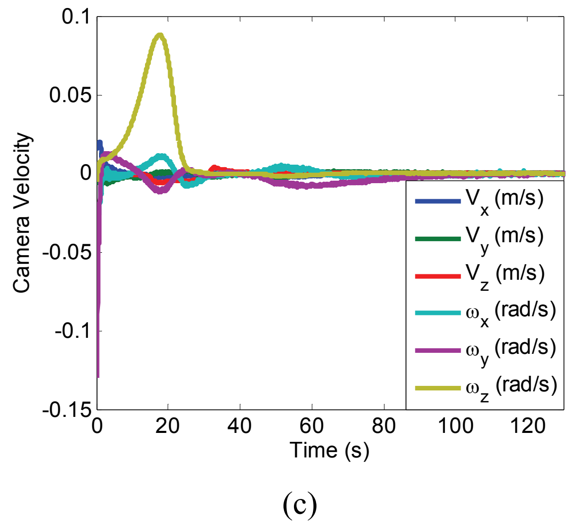


Figure 5.27 Sixth experiment: Visual servoing via two-stage controller with offline MPC, (a) Image space, (b) Cartesian space, and (c) Camera velocities.

5.6 Summary

Successful RVS requires proper constraint handling and treatment of uncertainty effects. In this chapter, several control schemes were proposed to address these concerns. Initially, a PBPC and two HPC were proposed to handle the constraints of the visual servoing while improving the stability and numerical feasibility of the system. It was shown that the proposed controllers were fully capable of handling the constraints, while their convergence was guaranteed through a supplementary constraint. The HPC with full pose error offered the smoothest trajectories of all. In addition, a novel two-DOF control structure was introduced for robust RVS. Unlike the conventional proportional controllers, this control scheme enabled the system to control the rate of convergence, without impacting the steady-state error and the error covariance of the system. The controller was needless of online depth estimation and inverse Jacobian calculation, since it relied on offline calculation for that matter. Moreover, the controller allowed the system to run faster, since most of controller components were pre-computed during the offline phase. Based on this structure two controllers were presented for robust RVS. The first controller was capable of minimizing the system uncertainties, while handling the camera/robot constraints. The second controller was proposed for constraint handling in the presence of system uncertainties. Moreover, a model of system uncertainties was developed and exploited in constraint handling to minimize the effects of uncertainties. It was shown through simulations and experiments that the proposed methods were capable of constraint handling and were robust to image noise.

It is worth mentioning that unlike their image-based counterpart, the proposed MPC controllers require accurate estimations of pose from the camera images. The computation of pose may impose an extra computational burden to the system, which is supposed to be minimal. As a solution to this problem, the pose of previous time step may be used in optimization-based estimators to achieve fast pose estimations, similar to VVS explained in Chapter 2. In addition, the object of interest was supposed to be static which could limit the applicability of the proposed robust controllers. However, the same methodology may be practiced for moving objects by shifting the offline controller to the online stage to predict the next time step trajectories. Yet, this change comes at a price of increased computational cost.

Chapter 6

Conclusions

6.1 Summary of the Thesis

Successful RVS in unstructured environments entails apt techniques to handle the system limitations and errors caused by uncertainties. This thesis was mainly focused on robust visual servoing schemes to address such requirements. In addition to that, enhancing the accuracy of the system was of interest. Three major steps were taken towards robust and accurate RVS design. In the first phase, pose estimation was targeted as one of the most sensitive parts of most RVS systems. It was known that image uncertainties could degrade the accuracy of pose estimations, which could lead to inferior performance or even task failure. Sensor fusion techniques were proposed to alleviate this sensitivity by employing multiple cameras. A comprehensive study on sensor fusion for pose estimation was conducted under various system conditions. It was shown that the proposed centralized fusion techniques could response to system uncertainties and faults (e.g., image occlusion) better than the previous methods. Since this superior performance (in terms of accuracy and robustness) came at the price of increased computational cost of the system, decentralized fusion techniques were proposed as alternatives. These schemes were capable of lowering the computational complexity at the price of accuracy reduction. Yet, they could facilitate the fault detection and isolation. Since the centralized and decentralized fusion techniques required particular vessels for pose estimation (e.g., Kalman-based structure), they could not be used for other pose estimation methods. In order to address this shortcoming, a pre-processing fusion scheme was introduced. This scheme could offer enhanced accuracy and be engaged with any available pose estimation technique.

In the second step towards robust control of RVS systems, a novel uncertainty model for RVS systems was proposed. In this model, the error covariance of different signals in the system was estimated. For that purpose, the closed-loop and discrete-time nature of the RVS systems were considered. In addition, a general methodology was offered which could be applied to a vast group of controllers. The modeling was particularly developed for first and second order controllers, which were very popular in RVS systems. It was shown later that the model could approximate the uncertainties of the system closely.

Finally, several robust and constrained control schemes were introduced to cope with system limitations. Initially, novel predictive controllers were proposed to enhance the stability of the system, while handling the constraints. Then, a two-stage control scheme was introduced for robust and accurate RVS. The proposed control structure had the capability of decoupling the uncertainty measures from the system's convergence rate, which could help in uncertainty minimization without sacrificing the speed. In addition, the proposed uncertainty model was exploited along with the developed predictive controllers to robustly guide the system to its destination, while avoiding the system constraints, even in the presence of system uncertainties. Moreover, the accuracy of the system was improved by minimizing the effects of uncertainties.

In conclusion, the proposed robust techniques were capable of achieving reasonable results, in the presence of uncertainties mainly originated from image noise. The proposed fusion techniques can pave the way for more accurate and robust pose estimation to be used not only in visual servoing systems, but also in many other applications such as object recognition and tracking. The developed IAUKF scheme may be used for accurate estimations in many nonlinear applications. In addition, the developed uncertainty model adds extra insight to the response of RVS systems in presence of uncertainties, which in turn may lead to optimal control schemes. Last but not least, the proposed robust controllers open a new horizon in robust visual servoing by exploiting the developed uncertainty model to handle the constraints efficiently. In summary, the proposed techniques make RVS was step closer to wide applicability in industrial tasks, while offering new paths to expand the robustness of the system even further.

6.2 Contributions

The contributions of the work were many-fold. Some of the most important contributions are listed as follows.

- Novel sensor fusion techniques: Three fusion structures were proposed to enhance the accuracy and robustness of pose estimation in presence of system uncertainties. Novel centralized fusion, namely IAEKF, IAUKF, and VVS were proposed for the first time and their superior accuracy was demonstrated in the presence of system uncertainties. The close relation between the VVS and Gauss-Newton pose estimation methods was shown and the superiority of the former was proven. Decentralized fusion methods were proposed to reduce the computational cost of the system. A novel pre-processing fusion was introduced for the first time which could be used with any available pose estimation to enhance the accuracy of the pose estimation.

- Novel uncertainty modeling for RVS systems: A novel methodology was developed to model the effect of image noise in RVS systems. Unlike the previous models, this model accounted for the closed-loop nature of the system and could be used with a wide range of controllers and systems. In this model, the RVS system was treated as a discrete-time system.
- Robust and constraint-aware controller design: Multiple constrained controllers such as PBPC and HPC were developed. It was shown that these controllers provide enhanced stability and numerical feasibility, compared to previous predictive controllers. In addition, a novel two-stage control scheme was proposed to decouple the effect of image noise from the system's rate of convergence. It was shown that the effect of errors could be minimized, while maintaining the same convergence rate. Next, the constrained control design was used in conjunction with the two-stage control scheme to handle the system's constraints, while minimizing the effect of image noise on the system's accuracy. Finally the developed error model was used to robustly handle the constraints in the presence of image noise. The chance of conservatism was reduced by engaging the knowledge from the developed uncertainty model.

6.3 Future Works

The work proposed in this thesis may be expanded in many directions, some of which are listed as follow.

- **Parameter Tuning:** Many of the proposed techniques are dependent on parameters that need to be adjusted accordingly. While these settings may be done on a case by case basis, an autonomous system is desirable which could maximize the performance of the system. This system may take the required specifications of the system and tune the parameters through an optimization technique.

- **Sensor Fusion Expansion:** Several sensor fusion techniques were proposed in this work. However the propose fusion techniques were limited to pose estimation. One way to enhance the performance of the system is to expand the fusion algorithms to other parts of the system. Fusion at imaging and control stage may be beneficial as the redundant information provide by multiple cameras may add to system robustness.
- **Uncertainty Modeling Expansion:** The proposed uncertainty modeling was shown to be very useful. However, this model only accounts for the uncertainties imposed by the image noise. In addition, the image noise was assumed to be Gaussian. It would be desirable to expand this model to entail more uncertainties from the system (e.g., camera calibration or robot dynamics). This change will enhance the applicability of RVS in actual tasks.
- **Predictive Control:** The usefulness of predictive controllers for constraint handling was shown in this work. However, this was just the beginning. There are different structures for predictive controllers which could benefit the RVS systems in different ways. Simpler predictive controllers may be sought to reduce the computational complexity of the optimization algorithms. In addition, different models of the system and reference trajectories may be taken to improve the accuracy or the robustness of the system.
- **Comprehensive Study:** Several robust algorithms were proposed for various purposes. However their careful investigation and applicability was beyond the scope of this work. The applications of the robust estimator (i.e., IAEKF and IAUKF) in other systems are yet to be investigated. Moreover, the proposed robust controller may find useful applications in other systems, where accuracy and robustness are key features.

Appendices

A. Velocity Transformation Matrix

The velocity transformation matrix, $\Omega_e^{c_i}$, is calculated separately for eye-in-hand and eye-to-hand cameras.

Eye-in-hand

The rotation between the camera and the end-effector is expressed as,

$$R_c^e = R_o^e R_c^o \, . \tag{A.1}$$

The time derivation of both sides yields,

$$S(\varpi_o^e) R_c^e = -S(R_o^e \varpi_c^o) R_c^e \, . \tag{A.2}$$

A change of coordinates results in,

$$\varpi_c^o = \varpi_e^o \, , \tag{A.3}$$

which is equivalent to,

$$\varpi_c = R_e^c \varpi_e \, . \tag{A.4}$$

As for the translational velocity, one can show,

$$t_c^e = R_o^e t_c^o + t_o^e \, . \tag{A.5}$$

A time derivative of both sides of the equation results in,

$$S(\varpi_o^e) R_o^e (t_e^o - t_c^o) + R_o^e \dot{t}_e^o = R_o^e \dot{t}_c^o \, , \tag{A.6}$$

which can be reordered as,

$$\dot{t}_c^o = \dot{t}_e^o + R_c^o S(R_e^c \varpi_o^e) t_e^c \, . \tag{A.7}$$

A simple change of coordinates yields,

$$\dot{t}_c = R_e^c \dot{t}_e + S(t_e^c) R_e^c \varpi_e . \tag{A.8}$$

One can show that (A.4) and (A.8) are equivalent to (2.89).

Eye-to-hand

The time derivation of (A.1) yields,

$$S(\varpi_c^e) R_c^e = S(\varpi_o^e) R_c^e . \tag{A.9}$$

A change of coordinates results in,

$$\varpi_c^e = -R_o^e \varpi_e^o , \tag{A.10}$$

which is equivalent to,

$$\varpi_c = -R_e^c \varpi_c . \tag{A.11}$$

In case of translational velocity, time derivation of (A.5) yields,

$$\dot{t}_c^e = S(\varpi_o^e) R_o^e (t_c^o - t_e^o) - R_o^e \dot{t}_e^o . \tag{A.12}$$

Once again, a change of coordinates yields,

$$\dot{t}_c = -R_e^c \dot{t}_e - S(t_e^c) R_e^c \varpi_e . \tag{A.13}$$

It is easy to show that (A.11) and (A.13) are equivalent to (2.91).

B. Iterative Pose Estimation

In this section the iterative pose method proposed by Dementhon in [2.16] is explained. This method is selected especially, since it is usually difficult to modify this method to entail measurements from multiple cameras. Pre-processing fusion is beneficial to algorithms of this type. Also, since the algorithm has an iterative fashion, it provides the fusion level with more accurate estimate of object depth. The algorithm is described as follows. Based on (2.4) and (2.5),

$$u_i^c = \frac{x_i^c}{z_i^c} = \frac{\begin{bmatrix}1 & 0 & 0\end{bmatrix}\left(R_o^c P_i^o + t_o^c\right)}{\begin{bmatrix}0 & 0 & 1\end{bmatrix}\left(R_o^c P_i^o + t_o^c\right)}, \tag{B.1}$$

$$v_i^c = \frac{y_i^c}{z_i^c} = \frac{\begin{bmatrix}0 & 1 & 0\end{bmatrix}\left(R_o^c P_i^o + t_o^c\right)}{\begin{bmatrix}0 & 0 & 1\end{bmatrix}\left(R_o^c P_i^o + t_o^c\right)}. \tag{B.2}$$

If point P_0^o of the object is selected as the origin of the object frame (i.e., $t_o^c = P_0^c$) and the rotation matrix is rewritten as follows,

$$R_o^c = \begin{bmatrix} a_i \\ a_j \\ a_k \end{bmatrix}, \tag{B.3}$$

then (B.1) and (B.2) can be presented as follows,

$$u_i^c = \frac{a_i P_i^o + x_0^c}{a_k P_i^o + z_0^c} = \frac{\left(P_i^o\right)^T \frac{\left(a_i\right)^T}{z_0^c} + u_0^c}{\frac{a_k P_i^o}{z_0^c} + 1} = \frac{\left(P_i^o\right)^T X_i + u_0^c}{\varepsilon_i' + 1}, \tag{B.4}$$

$$v_i^c = \frac{a_j P_i^o + y_0^c}{a_k P_i^o + z_0^c} = \frac{\left(P_i^o\right)^T \frac{\left(a_j\right)^T}{z_0^c} + v_0^c}{\frac{a_k P_i^o}{z_0^c} + 1} = \frac{\left(P_i^o\right)^T X_j + v_0^c}{\varepsilon_i' + 1}, \tag{B.5}$$

where:

$$\varepsilon_i' = \frac{a_k P_i^o}{z_0^c}. \tag{B.6}$$

Therefore if ε_i' are available, linear equations (B.4) and (B.5) yield,

$$X_i = \begin{bmatrix} \left(P_1^o\right)^T \\ \cdot \\ \cdot \\ \left(P_l^o\right)^T \end{bmatrix}^{\dagger} \begin{bmatrix} u_1^c\left(\varepsilon_1'+1\right)-u_0^c \\ \cdot \\ \cdot \\ u_l^c\left(\varepsilon_l'+1\right)-u_0^c \end{bmatrix}, \tag{B.7}$$

$$X_j = \begin{bmatrix} \left(P_1^o\right)^T \\ \cdot \\ \cdot \\ \left(P_l^o\right)^T \end{bmatrix}^{\dagger} \begin{bmatrix} v_1^c\left(\varepsilon_1'+1\right)-v_0^c \\ \cdot \\ \cdot \\ v_l^c\left(\varepsilon_l'+1\right)-v_0^c \end{bmatrix}. \tag{B.8}$$

The first two rows of the rotation matrix are extracted through normalization,

$$a_i = \frac{X_i}{\left\|X_i\right\|}, \tag{B.9}$$

$$a_j = \frac{X_j}{\left\|X_j\right\|}. \tag{B.10}$$

The third row of the rotation matrix is calculated through its orthogonality property,

$$a_k = a_i \times a_j, \tag{B.11}$$

and the translation element of the pose is calculated as follows,

$$t_o^c = P_0^c = z_0^c p_0^c, \tag{B.12}$$

where,

$$z_0^c = \frac{2}{\left(\|a_i\| + \|a_j\|\right)}. \tag{B.13}$$

If an initial value is used for ε_i' values (e.g., $\varepsilon_i' = 0$) an approximate method for pose estimation is achieved, which is known as pose from orthography and scaling (POS). A more accurate estimation of pose is obtained by using the approximate pose gained by POS algorithm to recalculate the values of ε_i' through (B.6), and re-estimating the pose using the new ε_i' values. Iterating through this process leads to an accurate pose estimation method known as POS with iterations (POSIT), which is considered in this work for pose estimation.

References

[1.1] F. Janabi-Sharifi, “Visual Servoing: Theory and Applications,” Chapter in *Opto-Mechatronic Systems Handbook*, Ed. H. Cho, CRC Press, Boca Raton, FL, 2002, pp. 15-1—15-24.

[1.2] F. Chaumette, S. Hutchinson, “Visual servo control I: basic approaches,” *IEEE Robotics and Automation Magazine*, vol. 13, no. 4, pp. 82–90, Dec. 2006.

[1.3] F. Chaumette, “Potential problems of stability and convergence in imagebased and position-based visual servoing,” in *The Confluence of Vision and Control,* D. Kriegman, G. D. Hager, and A. Morse, Eds. Springer-Verlag, Lecture Notes in Control and Information Sciences, 1998, vol. 237, pp. 66–78.

[1.4] F. Janabi-Sharifi, L. Deng, W.J. Wilson, “Comparison of basic visual servoing methods,” *ASME/IEEE Transactions on Mechatronics*, vol. 16, no. 5, pp. 967–983, Oct. 2011.

[1.5] E. Malis, F. Chaumette, S. Boudet, “2-1/2-D visual servoing,” *IEEE Transactions on Robotics and Automation*, vol. 15, no. 2, pp. 238–250, Oct. 1999.

[1.6] F. Janabi-Sharifi, M. Ficocelli, “Formulation of radiometric feasibility measures for feature selection and planning in visual servoing,” *IEEE Transactions on Systems, Man, and Cybernetics: Part B,* vol. 34, no. 2, pp. 978–987, Apr. 2004.

[1.7] J. E. Slotine, W. Li, “Adaptive manipulator control: A case study,” *IEEE Transactions on Automatic Control*, vol. 33, no. 11, pp.995–1003, Aug. 1988.

[1.8] R. Kelly, “Robust asymptotically stable visual servoing of planar robots,” *IEEE Transactions on Robotics and Automatation*, vol. 12, pp.759–766, Oct. 1996.

[1.9] S. A. Hutchinson, G. D. Hager, P. I. Corke, “A tutorial on visual servo control,” *IEEE Transactions on Robotics and Automation*, vol. 12, no. 5, pp. 651–670, Oct. 1996.

[1.10] G. Chesi, Y.S. Hung, "Image noise induced errors in camera positioning," *IEEE Transaction on Pattern Analysis and Machine Intelligence,* vol.29, no. 8, pp. 1476–1480, Aug. 2007.

[1.11] V. Kyrki, "Control uncertainty in image-based visual servoing," in *Proceeding of IEEE International Conference on Robotics and Automation*, Kobe, Japan, pp. 1516–1521, May 2009.

[1.12] B. Siciliano, L. Sciavicco, L. Villani, G. Oriolo, *Robotics: Modeling, Planning and Control*, Springer-Verlag: London, 2009.

[2.1] L. Quan, Z. Lan, "Linear N-point camera pose determination," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 8, pp. 774-780, Aug. 1999.

[2.2] A.H. Zahraee, J.K. Paik, J. Szewczyk, G. Morel, "Toward the development of a hand-held surgical robot for laparoscopy," *IEEE/ASME Transactions on Mechatronics*, vol. 15, no. 6, pp. 851-863, Dec. 2010.

[2.3] Q. He, C. Hu, W. Liu, N. Wei, M.Q.H. Meng, L.Liu, C. Wang, "Simple 3-D point reconstruction methods with accuracy prediction for multiocular system," *IEEE/ASME Transactions on Mechatronics*, vol. 18, no. 1, pp. 366-375, Feb. 2013.

[2.4] P.J. Flynn, A.K. Jain, "BONSAI: 3D object recognition using constraint search," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 10, pp. 1066-1075, Oct. 1991.

[2.5] Y. Shen, D. Sun, Y.H. Liu, K. Li, "Asymptotic trajectory tracking of manipulators using uncalibrated visual feedback," *IEEE/ASME Transactions on Mechatronics*, vol. 8, no. 1, pp. 87-98, March 2003.

[2.6] W.J. Wilson, C. Williams Hulls, F. Janabi-Sharifi, "Robust Image Processing and Position-Based Visual Servoing," Chapter in *Robust Vision for Vision-Based Control of Motion*, Ed. M. Vincze, and G. D. Hager, IEEE Press, New York, NY, pp. 163-201, 2000.

[2.7] E. Mouragnon, M. Lhuillier, M. Dhome, F. Dekeyser, P. Sayd, “Monocular vision based SLAM for mobile robots,” in *Proceedings of IEEE Internation Conference on Pattern Recognition (ICPR)*, vol. 3, pp. 1027-1031, Honk Kong, 2006.

[2.8] J.F. Vasconcelos, C. Silvestre, P. Oliveira, “A nonlinear GPS/IMU based observer for rigid body attitude and position estimation,” in *Proceedings of IEEE Conference on Decision and Control*, pp. 1255-1260, Cancun, Mexico, Dec. 2008.

[2.9] S. Panzieri, F. Pascucci, G. Ulivi, “An outdoor navigation system using GPS and inertial platform,” *IEEE/ASME Transactions on Mechatronics*, vol. 7, no. 2, pp. 134-142, Jun. 2002.

[2.10] M.A. Fischler, R.C. Bolles, “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography,” *Journal of Graphics and Image Processing*, vol. 24, no. 6, pp. 381-395, 1981.

[2.11] W. Forstner, “Reliability analysis of parameter estimation in linear models with applications to mensuration problems in computer vision,” *Journal of Computer Vision, Graphics and Image Processing*, vol. 40, pp. 273-310, 1987.

[2.12] R.M. Haralick, C. Lee, K. Ottenberg, M. NoÈ lle, “Analysis and solutions of the three point perspective pose estimation problem,” in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 592-598, Maui, Hawaii, USA, Jun. 1991.

[2.13] J.J. Wu, R.E. Rink, T.M. Caelli, V.G. Gourishankar, “Recovery of the 3-D location and motion of a rigid object through camera image (an extended Kalman filter approach,” *International Journal of Computer Vision*, vol. 2, no. 4, pp. 373-394, Apr. 1989.

[2.14] N. Y. Chen, J. Birk, R. Kelley, “Estimating workpiece pose using the feature points method”, *IEEE Transactions of Automatic Control*, vol. 25, no. 6, pp. 1027-1041, 1980.

[2.15] P. D. Fiore, “Efficient linear solution of exterior orientation”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 2, pp. 140-148, 2001.

[2.16] D. DeMenthon, L. S. Davis, “Exact and approximate solutions of the perspective-three point problem,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 11, pp. 1100-1105, Nov. 1992.

[2.17] D.G. Lowe, “Fitting parameterized three-dimensional models to images,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 5, pp. 441-450, May 1991.

[2.18] R. M. Haralick, H. Joo, C. Lee, X. Zhuang, V. G. Vaidya, M. B. Kim, “Pose estimation from corresponding point data,”, *IEEE Transactions on Systems, Man and Cybernetics*, vol. 19, no. 6, pp. 1426-1445, 1989.

[2.19] P. L. Rosin, “Robust pose estimation,” *IEEE Transactions on Systems, Man and Cybernetics-Part B*, vol. 29, no. 2, pp. 297-303, 1997.

[2.20] E. Marchand, F. Chaumette, “Virtual visual servoing: A framework for real-time augmented reality,” in *Proceedings of Computer Graphics Forum*, vol. 21, no. 3, pp. 289-298, Saarebrücken, Germany, Sep. 2002.

[2.21] A.I. Comport, E. Marchand, M. Pressigout, F. Chaumette, “Real-time markerless tracking for augmented reality: the virtual visual servoing framework,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 4, pp. 615-628, July-Aug. 2006.

[2.22] A.I. Comport, E. Marchand, F. Chaumette, “Robust model-based tracking for robot vision”, in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 1, pp. 692-697, Sendal, Japan, Sep. 2004.

[2.23] J.J. Wu, R.E. Rink, T.M. Caelli, V.G. Gourishankar, “Recovery of the 3-D location and motion of a rigid object through camera image (an extended Kalman filter approach,” *International Journal of Computer Vision*, vol. 2, no. 4, pp. 373-394, Apr. 1989.

[2.24] C. Fagerer, D. Dickmanns, E. D. Dickmanns, “Visual grasping with long delay time of a free floating object in orbit”, *Journal of Autonomous-Robots*, vol.1, no. 1, pp. 53-68, March 1994.

[2.25] W.J. Wilson, C.C.W. Hulls, G.S. Bell, “Relative end-effector control using Cartesian position based visual servoing,” *IEEE Transactions on Robotics and Automation*, vol. 12, no. 5, pp. 684-696, Oct. 1996.

[2.26] L. Jetto, S. Longhi, and G. Venturini, “Development and experimental validation of an adaptive extended Kalman filter for the localization of mobile robots,” *IEEE Transactions on Robotics and Automation*, vol. 15, no. 2, pp. 219–229, Apr. 1999.

[2.27] P.S. Maybeck, R.L. Jensen, D.A. Harnly, “An adaptive extended Kalman filter for target image tracking,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 17, no. 2, pp. 173–180, Mar. 1981.

[2.28] V. Lippiello, B. Siciliano, L. Villani, “Adaptive extended Kalman filtering for visual motion estimation of 3D objects,” *Journal of Control Engineering Practice*, vol. 15, no. 1, pp. 123–134, Jan. 2007.

[2.29] Ficocelli, M., and Janabi-Sharifi, F., “Adaptive filtering for pose estimation in visual servoing,” in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems,* Maui, Hawaii, pp. 19-24, Oct. 2001.

[2.30] A. Shademan, F. Janabi-Sharifi, “Sensitivity analysis of EKF and iterated EKF pose estimation for position-based visual servoing,” in *Proceedings of IEEE Conference on Control Applications.*, Toronto, Canada, pp. 755–780, Aug. 2005.

[2.31] F. Janabi-Sharifi, M. Marey, “A Kalman-filter-based method for pose estimation in visual servoing,” *IEEE Transactions on Robotics*, vol. 26, no. 5, pp. 939–947, Oct. 2010.

[3.1] E. Malis, G. Morel, F. Chaumette, “Robot control using disparate multiple sensors,” *The International Journal of Robotic Research*, vol. 20, no. 5, pp. 364-377, May 2001.

[3.2] O. Kermorgant, F. Chaumette, “Multi-sensor data fusion in sensor-based control: application to multi-camera visual servoing,” in *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 4518-4523, Shanghai, China, May. 2011.

[3.3] D. Hershberger, R. Burridge, D. Kortenkamp, R. Simmons, “Distributed visual servoing with a roving eye,” in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 1, pp. 441-447, Takamatsu, Japan, Oct. 2000.

[3.4] C. Dune, E. Marchand, C. Leroux, “One click focus with eye-in-hand/eye-to-hand coopration,” in *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 2471-2476, Roma, Italy, Apr. 2007.

[3.5] L. Qiu, Q. Song, J. Lei, Y. Yu, Y. Ge, “Multi-camera-based robot visual servoing,” in *Proceedings of IEEE International Conference on Mechatronics and Automation*, pp. 1509-1514, Luoyang, China, June 2006.

[3.6] N. Garcia-Aracil, C. Perez-Vidal, J.M. Sabater, R. Morales, F. J. Badesa, “Robust and cooperative image-based visual servoing system using a redundant architecture,” *Journal of Sensors*, vol. 11, pp. 11885-11900, Dec. 2011.

[3.7] A. Y. Mulayim, U. Yilmaz, V. Atalay, “Silhouette-based 3-D model reconstruction from multiple images,” *IEEE Transactions on Systems, Man and Cybernetics-Part B*, vol. 33, no. 4 pp. 582-591, 2003.

[3.8] F. Farshidi, S. Sirouspour, T. Kirubarajan, “Active multicamera object recognition in presence of occlusion,” In *Proceedings of IEEE International Conference on Intelligent Robots and Systems*, pp. 2718-2723, Edmonton, Canada, 2005.

[3.9] J. Stavnitzky, D. Capson, “Multiple camera model-based 3-D visual servo,” *IEEE Transactions on Robotics and Automation*, vol. 16, no. 6, pp. 732-739, 2000.

[3.10] B. Triggs, P.F. McLauchlan, R.I. Hartley, A.W. Fitzgibbon, “Bundle adjustment-a modern synthesis*,” Lecture Notes in Computer Science*, vol. 1883, pp. 298-375, 2000.

[3.11] V. Lippiello, B. Siciliano, L. Villani, “Eye-in-hand/eye-to-hand multicamera visual servoing,” in *Proceedings of 44th IEEE Conference on Decision and Control,* and *the European Control Conference*, Seville, Spain*, ,pp.* 5355-5359, 2005.

[3.12] -----, “Position-based visual servoing in industrial multirobot cells using a hybrid camera configuration”, *IEEE Transactions of Robotics*, vol. 23, no. 1, pp. 73-86, 2007.

[3.13] S.L. Sun, Z.L. Deng, “Multi-sensor optimal information fusion kalman filter,” *Journal of Automatica*, vol. 40, no. 6, pp. 1017-1023, 2004.

[3.14] J.B. Gao, C.J. Harris, “Some remarks on kalman filters for multisensor fusion,” *Journal of Information Fusion*, vol. 3, no. 3, pp. 191-201, 2002.

[3.15] R. C. Luo, C. C. Chang, and C. C. Lai, “Multisensor fusion and integration: theories, applications, and its perspectives,” *IEEE Sensors Journal*, vol. 11, no. 12, pp. 3122–3138, Dec. 2011.

[3.16] H. Zhao, and Z. Wang, “Motion measurement using inertial sensors, ultrasonic sensors, and magnetometers with extended Kalman filter for data fusion,” *IEEE Sensors Journal*, vol. 12, no. 5, pp. 943–953, May 2012.

[3.17] M. Carminati, G. Ferrari, R. Grassetti, and M. Sampietro, “Real-time data fusion and MEMS sensors fault detection in an aircraft emergency attitude unit based on Kalman filtering,” *IEEE Sensors Journal*, vol. 12, no. 10, pp. 2984-2992, Oct. 2012.

[3.18] M. Hoshino, Y. Gunji, S. Oho, and K. Takano, “A Kalman filter to estimate direction for automotive navigation,” in *Proceedings of IEEE/SICE/RSJ International Conference on Multisensor Fusion and Integration for Intelligent Systems*, Washington, USA, pp. 145–150, Dec. 1996.

[3.19] G. Bleser, and D. Stricker, “Advanced tracking through efficient image processing and visual-inertial sensor fusion,” *Journal of Computer & Graphics*, vol. 33, no. 1, pp. 59–72, Feb. 2009.

[3.20] V. Sazdovski, and P. Silson, “Inertial navigation aided by vision-based simultaneous localization and mapping,” *IEEE Sensors Journal*, vol. 11, no. 8, pp. 1646–1656, Aug. 2011.

[3.21] H. Zhao, and Z. Wang, “Motion measurement using inertial sensors, ultrasonic sensors, and magnetometers with extended Kalman filter for data fusion,” *IEEE Sensors Journal*, vol. 12, no. 5, pp. 943–953, May. 2012.

[3.22] A. Nemra, and N. Aouf, “Robust INS/GPS sensor fusion for UAV localization using SDRE nonlinear filtering,” *IEEE Sensors Journal*, vol. 10, no. 4, pp. 789–798, Apr. 2010.

[3.23] S. J. Julier, and J. K. Uhlmann, “A new extension of the Kalman filter to nonlinear systems,” in *Proceedings of SPIE, Signal Processing, Sensor Fusion, and Target Recognition VI*, Orlando, Florida, USA, pp. 182–193, Apr. 1997.

[3.24] Q. Fang, and S. X. Huang, “UKF for integrated vision and inertial sensors based on three-view geometry,” *IEEE Sensors Journal*, vol. 13, no. 7, pp. 2711–2719, Jul. 2013.

[3.25] W. Li, and H. Leung, “Simultaneous registration and fusion of multiple dissimilar sensors for cooperative driving,” *IEEE Transactions on. Intelligent Transportation Systems*, vol. 5, no. 2, pp. 84–98, Jun. 2004.

[3.26] S. J. Julier, and J. K. Uhlmann, “Unscented filtering and nonlinear estimation,” in *Proceedings of the IEEE*, vol. 92, no. 3, pp. 401–422, Mar. 2004.

[3.27] J.J. Laviola, “A comparison of unscented and extended Kalman filtering for estimating quaternion motion,” in *Proceedings of IEEE American Control Conference*, Denver, Colorado, USA, pp. 2435–2440, Jun. 2003.

[3.28] L. Jetto, S. Longhi, and G. Venturini, “Development and experimental validation of an adaptive extended Kalman filter for the localization of mobile robots,” *IEEE Transactions on Robotics and Automation*, vol. 15, no. 2, pp. 219–229, Apr. 1999.

[3.29] P.S. Maybeck, R.L. Jensen, D.A. Harnly, “An adaptive extended Kalman filter for target image tracking,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 17, no. 2, pp. 173–180, Mar. 1981.

[3.30] V. Lippiello, B. Siciliano, L. Villani, “Adaptive extended Kalman filtering for visual motion estimation of 3D objects,” *Journal of Control Engineering Practice*, vol. 15, no. 1, pp. 123–134, Jan. 2007.

[3.31] Ficocelli, M., and Janabi-Sharifi, F., “Adaptive filtering for pose estimation in visual servoing,” in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems,* Maui, Hawaii, pp. 19-24, Oct. 2001.

[3.32] Q. Song, Y. He, “Adaptive unscented Kalman filter for estimation of modelling errors for helicopter,” in *Proceedings of IEEE International Conference on Robotics and Biomimetics*, Guilin, China, pp. 2463–2467, Dec. 2009.

[3.33] Q. Song, J. D. Han, “An adaptive UKF algorithm for the state and parameter estimations of a mobile robot,” *Journal of Acta Automatica Sinica*, vol. 34, no. 1, pp. 72–79, Jan. 2008.

[3.34] A. Shademan, F. Janabi-Sharifi, “Sensitivity analysis of EKF and iterated EKF pose estimation for position-based visual servoing,” in *Proceedings of IEEE Conference on Control Applications.*, Toronto, Canada, pp. 755–780, Aug. 2005.

[3.35] K. Spingarn, “Passive position location estimation using the extended Kalman filter,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 23, no. 4, pp. 557–567, Jul. 1987.

[3.36] L. Ling, E. Cheng, I.S. Burnett, “An iterated extended Kalman filter for 3D mapping via Kinect camera,” in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing.*, Vancouver, Canada, pp. 1773–1777, May. 2013.

[3.37] R. Zhan, J. Wan, “Iterated unscented Kalman filter for passive target tracking,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 43, no. 3, pp. 1155–1163, Jul. 2007.

[3.38] M. Zhong-Kai, S. Li-Fen, “Improvement of UKF algorithm and robustness study,” in *Proceedings of IEEE International Workshop on Intelligent Systems and Applications*, Wuhan, China, pp. 1–4, May. 2009.

[3.39] Y. A. Zhang, D. Zhou, G. R. Duan, “An adaptive iterated Kalman filter,” in *Proceedings of IEEE Multiconference on Computational Engineering in Systems Applications*, Beijing, China, pp. 1727-1730, Oct. 2006.

[3.40] F. Janabi-Sharifi, M. Marey, “A Kalman-filter-based method for pose estimation in visual servoing,” *IEEE Transactions on Robotics*, vol. 26, no. 5, pp. 939–947, Oct. 2010.

[3.41] D.J. Jwo, S. H. Wang, “Adaptive fuzzy strong tracking extended Kalman filtering for GPS navigation,” *IEEE Sensors Journal,* vol. 7, no. 5, pp. 778–789, May 2007.

[3.42] J. Z. Sasiadek, Q. Wang, M. B. Zeremba, “Fuzzy adaptive Kalman filtering for INS/GPS data fusion,” in *Proceedings of IEEE Internationa Symposium on Intelligent Control*, Rio Patras, Greece, pp. 181–186, July 2000.

[3.43] R. van der Merwe, E. A. Wan, S. J. Julier, “Sigma-point Kalman filters for nonlinear estimation and sensor fusion: Applications to integrated navigation,” in *Proceedings of AIAA Guidance, Navigation, and Control Conference Exhibition*, Providence, RI, AIAA 2004-5120, Aug. 2004.

[3.44] Y. Gao, W. J. Jia, X. J. Sun, Z. L. Deng, “Self-tuning multisensor weighted measurement fusion Kalman filter,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 45, no. 1, pp. 179–191, Jan. 2009.

[3.45] F. Janabi-Sharifi, M. Ficocelli, “Formulation of radiometric feasibility measures for feature selection and planning in visual servoing,” *IEEE Transactions on System, Man, and Cybernetics-Part B*, vol. 34, no. 2, pp. 978-987, 2004.

[3.46] A.S. Willsky, “A survey of design method for failure detection in dynamic systems,” *Journal of Automatica*, vol. 12, pp. 601-611, 1976.

[4.1] M. De Santo, C. Liguori, A. Pietrosanto, “Uncertainty characterization in image-based measurements: a preliminary discussion”, *IEEE Transaction on Instrumentation and Measurement*, vol.49, no. 5, pp. 1101–1107, 2000.

[4.2] G. Chesi,Y.S. Hung, “Image noise induced errors in camera positioning,” *IEEE Transaction on Pattern Analysis and Machine Intelligence,* vol.29, no. 8, pp. 1476–1480, 2007.

[4.3] G. Chesi, H.L. Yung, “Performance limitation analysis in visual servo systems: bounding the location error introduced by image points matching,” in *Proc. IEEE International Conference on Robotics and Automation*, Kobe, Japan, pp. 695–700, 2009.

[4.4] G. Chesi, “Optimal object configuration to minimize the positioning error in visual servoing,” *IEEE Transactions on Robotics*, vol. 26, no. 3, pp. 584–589, 2010.

[4.5] -------, ”Visual servoing path-planning via homogeneous forms and LMI optimizations,” *IEEE Transactions on Robotics*, vol.25, no.2, pp 281–291, 2009.

[4.6] -------, “LMI conditions for time-varying uncertain systems can be non-conservative,” *Journal of Automatica*, vol. 47, pp. 621-624, 2011.

[4.7] V. Kyrki, D. Kragic, H.I. Christensen, “Measurement errors in visual servoing,” in*Proc. IEEE International Conference on Robotics and Automation*, vol. 2, New Orleans, LA, USA, 2004, pp. 1861–1867.

[4.8] V. Kyrki, D. Kragic, and H. Christensen, “Measurement errors in visual servoing,” *Robotics and Autonomous Systems*, vol. 54, no. 10, pp. 815–827, 2006.

[4.9] E. Malis, F. Chaumette, and S. Boudet, “2-1/2-D visual servoing,” *IEEE Transaction on Robotics and Automation*, vol. 15, no. 2, pp. 238–250, Oct. 1999.

[4.10] R.M. Haralick, “Propagating covariance in computer vision,” *International Journal of Pattern Recognition and Artificial Intelligence,* vol.10, no.5, pp 561–572, 1996.

[4.11] V. Kyrki, “Control uncertainty in image-based visual servoing,” *In Proceeding of IEEE Internatonal Conference on Robotics and Automation*, Kobe, Japan, pp. 1516–1521, 2009.

[4.12] A. Chao, M. Athans, “Stability robustness to unstructured uncertainty for linear time invariant systems,” *The Control Handbook* (Editor, W.S. Levine), CRC Press and IEEE Press, 1996.

[4.13] J. Raisch, B. Francis, “Modeling deterministic uncertainty,” *The Control Handbook* (Editor, W.S. Levine), CRC Press and IEEE Press, 1996.

[4.14] B. Siciliano, L. Sciavicco, L. Villani, G. Oriolo, *Robotics: Modeling, Planning and Control*, Springer, Verlag, London, 2009.

[5.1] Y. Mezouar, F. Chaumette, “Path planning for robust image-based control,” *IEEE Transactions on Robotics and Automation*, vol. 18, no. 4, pp. 534–549, Aug. 2002.

[5.2] M. Kazemi, K. Gupta, M. Mehrandezh, “Global path planning for robust visual servoing in complex environments,” in *Proceeding of IEEE International Conference on Robotics and Automation*, Kobe, Japan, pp. 326–332, May, 2009.

[5.3] G. Chesi, “Visual servoing path-planning via homogeneous forms and LMI optimizations,” *IEEE Transactions on Robotics*, vol.25, no.2, pp. 281–291, Apr. 2009.

[5.4] J.A. Ganglo, M.F. De Mathelin, “High speed visual servoing of a 6 DOF manipulator using multivariable predictive control,” *Journal of Advanced Robotics*, vol. 17, no. 10, pp. 993–1021, 2003.

[5.5] M. Sauv´ee, P. Poignet, E. Dombre, E. Courtial, “Image based visual servoing through nonlinear model predictive control,” in *Proceedings of 45th IEEE Conference on Decision and Control*, San Diego, California, USA, pp.1776–1781, Dec. 2006.

[5.6] G. Allibert, E. Courtial, F. Chaumette, “Predictive control for constrained image-based visual servoing,” *IEEE Transactions on Robotics*, vol. 26, no. 5, pp. 933–939, Oct. 2010.

[5.7] C. Lazar, A. Burlacu, “Visual servoing of robot manipulators using model-based predictive control,” in *Proceedings of IEEE International Conference on Industrial Informatics*, Cardiff, Wales, pp. 690–695, Jun. 2009.

[5.8] C. Lazar, A. Burlacu, C. Copot, “Predictive control architecture for visual servoing of robot manipulators,” in *Proceedings of 18th IFAC Congress*, Milan, Italy, pp. 9464–9569, Aug. 2011.

[5.9] A. Chan, S. Leonard, E.A. Croft, J.J. Little, “Collision-free visual servoing of an eye-in-hand manipulator via constraint-aware planning and control,” in *Proceedings of American Control Conference*, San Francisco, California, USA, pp. 4642–4648, Jun. 2011.

[5.10] T. Wang, W.Xie, G. Liu, Y. Zhao, “Quasi-min-max model predictive control for image-based visual servoing,” in *Proceedings of IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, Kaohsiung, Taiwan, pp. 98–103, July 2012.

[5.11] C. Copot, C. Lazar, A. Burlacu, “Predictive control of nonlinear visual servoing systems using image moments,” *IET Control Theory and Applications*, vol. 6, no. 10, pp. 1486–1496, July 2012.

[5.12] H. Wang, M. Jiang, W. Chen, Y.H. Liu, “Visual servoing of robots with uncalibrated robot and camera parameters,” *Journal of Mechatronics*, vol. 22, no. 6, pp. 661–668, Sept. 2012.

[5.13] R. Kelly, “Robust asymptotically stable visual servoing of planar robots,” *IEEE Transations on Robotics and Automation*, vol. 12, no. 5, pp. 759–776, Oct. 1996.

[5.14] G. Morel, P. Zanne, F. Plestan, “Robust visual servoing: bounding the task function tracking errors,” *IEEE Transactions on Control Systems Technology*, vol. 13, no. 6, pp. 998–1009, Nov. 2005.

[5.15] C.S. Kim, E.J. Mo, S.M. Han, M.S. Jie, K.W. Lee, “Robust visual servo control of robot manipulators with uncertain dynamics and camera parameters,” *International Journal of Control, Automation and Systems*, vol. 8, no. 2, pp.308–313, Apr. 2010.

[5.16] G. Hu, N. Gans, and W. Dixon, “Quaternion-based visual servo control in the presence of camera calibration error,” *Intertional Journal of Robust and Nonlinear Control*, vol. 20, no. 5, pp. 489–503, Mar. 2010.

[5.17] E. Zergeroglu, D.M. Dawson, M.S. de Queiroz, P. Setlur, “Robust visual-servo control of robot manipulators in the presence of uncertainty,” *Journal of Robotic Systems*, vol. 20, no. 2, pp. 93–106, Feb. 2003.

[5.18] C.S Kim, K.W. Lee, “Image-based robust control of robot manipulators using dynamic compensator,” in *Proceeings of American Control Conference*, Baltimore, Maryland, USA, pp. 5266–5271, June 2010.

[5.19] M.U. Khan, I. Jan, N. Iqbal, “Robustness analysis of uncalibrated eye-in-hand visual servo system in the presence of parametric uncertainty,” *Industrial Robot: an International Journal*, vol. 39, no. 2, pp. 154 –161, 2012.

[5.20] M.C. Chien, A.C. Huang, “Design of a Fat-based adaptive visual servoing for robots with time varying uncertainties,” *International Journal of Optomechatronics*, vol. 4, no. 2, pp. 93–114, Jun. 2010.

[5.21] D.H. Park, J.H. Kwon, I.J. Ha, “Novel position-based visual servoing approach to robust global stability under field-of-view constraint,” *IEEE Transactions on Industrial Electronics*, vol. 59, no. 12, pp. 4735–4752, Dec. 2012.

[5.22] H. M. Becerra, C. Sagues, “A sliding mode control law for epipolar visual servoing of differential-drive robots,” in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, Nice, France, pp. 358–363, Sept. 2008.

[5.23] T.R. Oliveira, A.J. Peixoto, A.C. Leite, L. Hsu,” Sliding mode control of uncertain multivariable nonlinear systems applied to uncalibrated robotics visual servoing,” in *Proceedings of American Control Conference*, St. Louis, Missouri, USA, pp. 71–76, Jun. 2009.

[5.24] T.R. Oliveira, A.J. Peixoto, L. Hsu, “Sliding mode control of uncertain multivariable nonlinear systems with unknown control direction via switching and monitoring function,” *IEEE Transactions on Automatic Control*, vol. 55, no. 4, pp. 1028–1034, Apr. 2010.

[5.25] X. Liang, X. Huang, M. Wang, X. Zeng, “Adaptive task-space tracking control of robots without task-space- and joint-space-velocity measurements,” *IEEE Transactions on Robotics*, vol. 26, no. 4, pp. 733–742, Aug. 2010.

[5.26] M.G. Ortega, M. Vargas, F.R. Rubio, “H∞ controller for a visual servoing system,” in *Proceedings of European Control Conference*, Karlsruhe, Germany, pp. 217–220, 1999.

[5.27] A.I. Comport, E. Marchand, F. Chaumette, “Statistically robust 2-D visual servoing”, *IEEE Transactions on Robotics*, vol. 22, no. 2, pp. 416–421, Apr. 2006.

[5.28] D. Bellot, P. Danes, “Handling visual servoing schemes through rational systems and LMIs,” in *Proceedings of IEEE Conference on Decision and Control*, Orlando, Florida, USA, pp. 3601–3606, Dec. 2001.

[5.29] D. Bellot, P. Danes, “An LMI solution to visual-based localisation as the dual of visual servoing,” in *Proceedings of IEEE Conference on Decision and Control*, Maui, Hawaii, USA, pp. 5420–5425, Dec. 2003.

[5.30] N. P. Papanikolopoulos, P.K. Khosla, T. Kanade, “Visual tracking of a moving target by a camera mounted on a robot: a combination of control and vision,” *IEEE Transactions on Robotics and Automation*, vol. 9, no. 1, pp. 14–35, Feb 1993.

[5.31] K. Hashimoto, T. Ebine, H. Kimura, “Visual servoing with hand-eye manipulator-optimal Control approach,” *IEEE Transactions on Robotics and Automation*, vol. 12, no. 5, pp. 766–774, Oct. 1996.

[5.32] E. Malis, F. Chaumette, S. Boudet, “2-1/2-D visual servoing,” *IEEE Transactions on Robotics and Automation*, vol. 15, no.2, pp.238–250, Apr. 1999.

[5.33] F. Chaumette, S. Hutchinson, “Visual servo control I: Basic approaches,” *IEEE Robotics and Automation Magazine*, vol. 13, no. 4, pp. 82–90, Dec. 2006.

[5.34] B. Siciliano, L. Sciavicco, L. Villani, G. Oriolo, *Robotics: Modeling, Planning and Control*, pp.128–132, Springer, Verlag, London, 2009.