

A SLIDING MODE APPROACH TO VISUAL MOTION
ESTIMATION

by
BURAK YILMAZ

Submitted to the Graduate School of Engineering and Natural Sciences
in partial fulfillment of
the requirements for the degree of
Master of Science
Sabanci University
Spring 2005

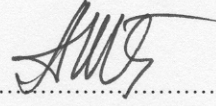
A Sliding Mode Approach to Visual Motion Estimation

Burak Yilmaz

APPROVED BY

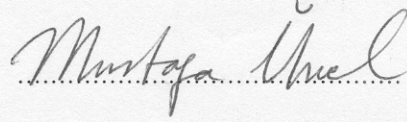
Prof. Dr. Asif SABANOVIC

(Thesis Supervisor)

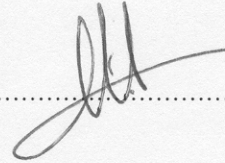


Assoc. Prof. Dr. Mustafa UNEL

(Thesis Coadvisor)



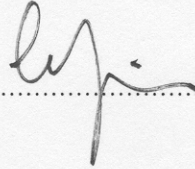
Assoc. Prof. Dr. Mahmut F. AKSIT



Assist. Prof. Dr. Ahmet ONAT



Assist. Prof. Dr. Cem GUNERI



DATE OF APPROVAL: 14.07.2005

©Burak Ylmaz 2005
All Rights Reserved

to my mother

Acknowledgments

I would like to thank many people who made significant contributions to the completion of this thesis. First of all, I would like to express my gratitude to Prof. Dr. Asif Sabanovic, who helped me many times to find my way when I was lost, and more importantly who always had some trust in me, which was and which is very precious to know.

Also I would like to thank Assoc. Prof. Dr. Mustafa Unel, for all those hours he spent for me, discussing every aspect of the work created in this thesis. Without him, not only this thesis would be unfinished, it could not even start.

Among my friends, who were ready for support any time, and who made this campus a place to live in for me, I am happy to acknowledge the following names; Ibrahim Eden, who is my full time roommate and friend ever, and who is one of the smartest people I have ever known, Nusrettin Gulec, who has been an enormous friend for more than 5 years now, Cagdas Onal, who is truly ‘a beautiful mind’ that I envy, Sakir Kabadayi, who is a true friend and who is one of the most caring people I have ever known, Onur Ozcan, who is hopefully my future company partner, Eray Dogan, who was very helpful in conducting the experiments, Arda of CafeDorm, who supplied lots of cigarettes, food and friendship, and all the others, Izzet, Eray, Ozer, Firuze, Esra, Hande, Cagrihan, Celal, Nevzat, Fazil, Shahzad...

Finally my greatest thanks will go to my family, whose support and trust I cannot compare to anything else.

A Sliding Mode Approach to Visual Motion Estimation

by Burak Yilmaz

Abstract

The problem of estimating motion and structure from a sequence of images has been a major research theme in machine vision for many years and remains one of the most challenging ones. In this work, a new approach to this problem is presented; using sliding mode observers to estimate the motion and structure of a moving body with the aid of a CCD camera. A variety of dynamical systems which may arise in machine vision applications is considered and a novel identification procedure is developed for the estimation of both constant and time varying parameters. The basic procedure introduced for parameter estimation is to recast image feature dynamics linearly in terms of unknown parameters and construct a sliding mode observer to produce asymptotically correct estimates of the observed image features, and then use 'equivalent control' to explicitly compute parameters. Much of the procedure presented in this work has been substantiated by computer simulations and real experiments.

Özet

Bir dizi görüntü bilgisi kullanılarak hareket ve yapı tahmini, uzun yıllar boyunca önemli bir araştırma konusu olmuştur ve olmaya devam etmektedir. Bu çalışmada, hareket kestirimi problemine yeni bir yaklaşım sunulmaktadır: bir CCD kamera yardımıyla hareket kestirimini, kayma kipli gözlemciler kullanılarak gerçekleştirilmesi. Görüntülü makineler alanında karşılaşılabilecek çeşitli dinamik sistemler göz önüne alınmakta ve hareketli nesnelerin sabit ya da zamanla değişen hareket parametrelerinin bulunması/tanımlanması için yeni bir yöntem geliştirilmektedir. Bu yeni yöntemin uygulanışındaki temel yaklaşım, görüntü hareket denklemlerinin bilinmeyen parametreler cinsinden doğrusal olacak şekilde yeniden yazılması, alınan görüntülerdeki ayıklanabilir nokta koordinatlarını asimptotik olarak doğru takip edecek bir kayma kipli gözlemcinin oluşturulması, ve bu gözlemciden elde edilecek eşdeğer kontrol sinyali ile parametrelerin gerçek değerlerinin bulunması şeklinde özetlenebilir. Çalışmada bahsedilen yöntemler, bilgisayar simülasyonları ve gerçek deneylerle denenmiş, yöntemin başarısı incelenmiştir.

Table of Contents

Acknowledgments	v
Abstract	vi
Ozet	vii
1 Introduction	1
1.1 Parameter Estimation in General	1
1.2 Estimation of Motion Parameters	3
2 A Survey on Parameter Estimation Methods	5
2.1 Predictors Based on Output Error	5
2.2 Methods for Estimating Motion and Structure	7
2.2.1 Optical Flow Based Methods	7
2.2.2 Feature Point Based Methods	8
3 Sliding Mode Variable Structure Control	10
3.1 Introduction	10
3.2 Sliding-Mode in Variable Structure Systems	11
3.3 The Idea of Equivalent Control	13
3.4 Remarks	16
4 Using Sliding Mode to Estimate Motion Parameters	17
4.1 Motion Models	17
4.2 Estimation of Rigid Motion	17
4.3 Problem Formulation	18
4.4 Parameter Estimation Problem; Redefined	19
4.5 Sliding Mode Based Solution for Rigid Motion	21
4.6 Expanding the System	23
4.7 Singularity in the Solution for Rigid Motion	24
4.8 Simulation Results for Rigid Motion	25
4.9 Estimation of Affine Motion	34
4.10 Simulation Results for Affine Motion	35

5	Experimental Results	40
5.1	Experimental Setup - Vision System	40
5.1.1	The Code Briefly	43
5.1.2	Results for Translational Motion	45
5.1.3	Results for Rotational Motion	47
5.2	Conclusions	48
	Appendix	50
A	C++ Codes For the Experiment	50
	Bibliography	58

List of Figures

1.1	The system identification loop	2
1.2	A possible vision scenario	3
2.1	Optical Flow (a)Frame 1 (b)Frame 2 (c) 1 iteration (d) 10 iterations	8
3.1	Two intersecting switching surfaces	12
3.2	Phase portrait of a sliding motion	14
3.3	Discontinuous control action	14
3.4	Equivalent Control	15
4.1	Vision Setup	18
4.2	Simulink Model	26
4.3	Trajectory of the object	27
4.4	ω and ω -estimate	28
4.5	ω and ω -estimate, zoomed	28
4.6	b_1 and b_1 -estimate	29
4.7	b_2 and b_2 -estimate	30
4.8	Object Trajectory	30
4.9	$\omega(t)$ and $\omega(t)$ -estimate	31
4.10	$\omega(t)$ and $\omega(t)$ -estimate, zoomed	31
4.11	Reaching to manifold	32
4.12	$b_1(t)$ and $b_1(t)$ -estimate	32
4.13	$b_2(t)$ and $b_2(t)$ -estimate	33
4.14	Trajectory of the Points on the Object	35
4.15	a_1 and a_1 estimated	36
4.16	a_2 and a_2 estimated	36
4.17	a_3 and a_3 estimated	37

4.18	a_4 and a_4 estimated	37
4.19	b_1 and b_1 estimated	38
4.20	b_2 and b_2 estimated	38
5.1	Experimental Setup	40
5.2	PI NanoCube	41
5.3	Micrometer	42
5.4	Rotational motion of micrometer	42
5.5	Linear Motion along x-axis	45
5.6	Linear Motion along y-axis	46
5.7	Rotational Velocity	46
5.8	Rotational Velocity	47
5.9	Estimated Trajectory	49
5.10	Angular Velocity	49

List of Abbreviations

PEM:	Prediction-Error Identification Method
LS:	Least Squares
ML:	Maximum Likelihood
VSCS:	Variable Structure Control Systems
VSS:	Variable Structure Systems
VSC:	Variable Structure Control
SMC:	Sliding Mode Control
SMO:	Sliding Mode Observer
LTV:	Linear Time Varying
det:	determinant
fps:	frames per second

Chapter 1

Introduction

Inferring models from observations and studying their properties is really what science is about. The models may be of more or less formal character, but they have the basic feature that they attempt to link observations together into some pattern. System identification deals with the problem of building mathematical models of dynamical systems based on observed data from the system. The subject is thus part of basic scientific methodology, and since dynamical systems are abundant in our environment, the techniques of system identification have a wide application area [1].

1.1 Parameter Estimation in General

For any control analysis and synthesis, it is desirable to be able to obtain a model of the plant to allow complete off-line analysis with minimum interference to the process [2]. Many engineering systems of interest to the control engineer are partially known in the sense that the system structure, together with some system parameters are known, but some system parameters are unknown. This gives rise to a problem of parameter estimation when values for the unknown parameters are to be determined from experimental data comprising measurements of system inputs and outputs. There is considerable literature in the area. Parameter estimation and identification are usually described within probabilistic and statistical frameworks. It is possible to identify the steps in a typical system identification procedure as [1]:

- *The data record.* To record or generate the input-output data so that the data become maximally informative for system identification

- *The model structure.* A model with some unknown parameters is constructed from basic physical laws and other well-established relationships. This is no doubt the most important and the most difficult choice of the system identification procedure. It is here that *a priori* knowledge and engineering intuition and insight have to be combined. Generally speaking, a model structure is a parameterized mapping from past inputs and outputs to the space of the model outputs [1].
- *Determining the "best" model in the set, guided by the data.* This is the identification method. The assessment of model quality is typically based on how the models perform when they attempt to reproduce the measured data.

This system identification procedure can be summarized by the following flow chart:

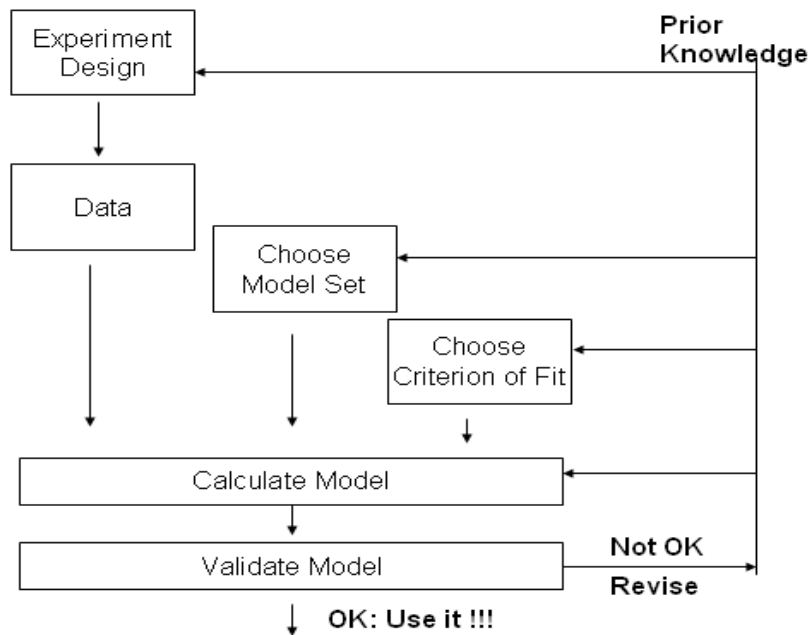


Figure 1.1: The system identification loop

1.2 Estimation of Motion Parameters

The interest in motion estimation using image sequences has been growing rapidly in many fields. Motion estimation using image data has many application areas such as mobile robotics, vision guided navigation, automatic target detection and recognition systems. Let us consider the following sample problem and example applications:

Problem : Suppose that an object undergoes some kind of rigid and/or affine motion with possibly time varying parameters in a plane perpendicular to the optical axis of a CCD camera, as depicted in the following figure. Estimate the shape and

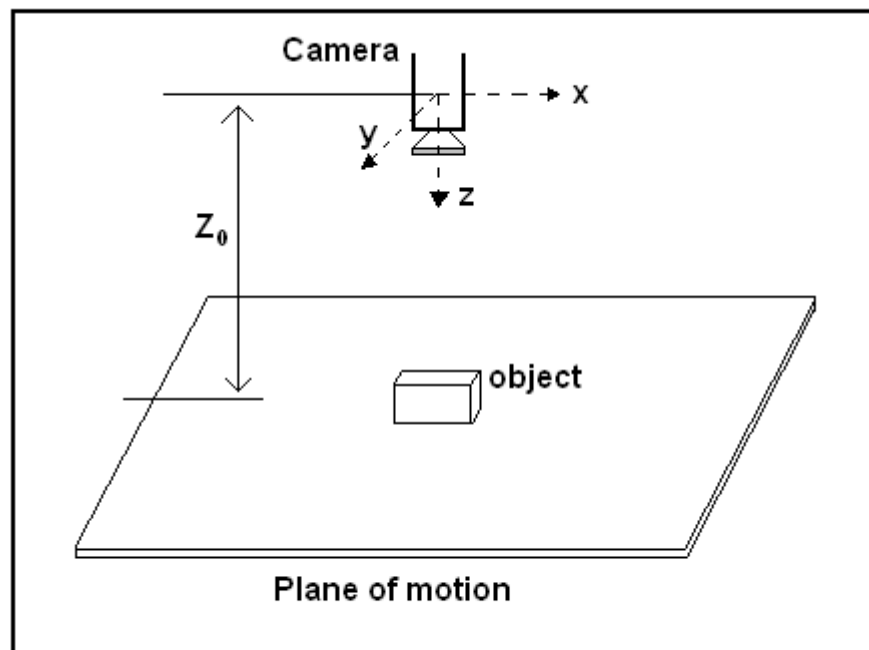


Figure 1.2: A possible vision scenario

motion parameters of the object from the observed time varying images produced by the camera. We can cite several examples which fall into this problem category: a mobile robot maneuvering on a flat horizontal surface and being viewed from a camera whose optical axis is down to the surface; a robot arm which picks a free-form moving part on a conveyor belt using images taken from a camera mounted to the ceiling; tracking the motion of a microorganism whose shape deforms during its route

under a composite vision system which consists of an optical microscope plus a CCD camera, and lip tracking or lip reading for speech recognition in noisy environments. The first two examples are related to the identification of rigid motion parameters whereas the last two examples are related to the affine motion estimation. However, in all these examples, the perspective projection of the CCD camera reduces to a scaled orthographic projection due to a constant depth.

Several solutions for estimating rigid scene structure and the relative 3D motion of a camera from image sequences have been proposed, based on different measurements and different estimation algorithms. These solutions can be classified into two categories depending upon what is measured from the scene. If the brightness pattern is the data observed from images, a well known approach is based on analyzing the optical flow (see [3],[4],[5],[6]). On the other hand if the data observed are the discontinuity-curves in the image brightness pattern, a possible approach is to identify the correspondence of various features such as points, lines and curves between consecutive frames (see [7],[8],[9]). The former approach assumes that the image intensity is a smooth function and considers only the smooth part of the image. The latter approach assumes that the image intensity is a piecewise smooth function and concentrates onto the image discontinuity curves.

In this work, a variety of dynamical systems which arise in machine vision applications will be considered and a novel identification procedure for the estimation of both constant and time varying parameters will be developed. As the main approach, ‘feature based analysis’ will be used. The basic procedure introduced for parameter estimation is to recast image feature dynamics linearly in terms of unknown parameters and construct a sliding mode observer to produce asymptotically correct estimates of the observed image features, and then use *equivalent control* to explicitly compute parameters.

Chapter 2

A Survey on Parameter Estimation Methods

The problem of parameter estimation can be summarized as follows: Suppose a set of candidate models has been selected, and it is parameterized as a model structure, using a parameter vector Θ . The search for the best model within the set then becomes a problem of determining or estimating Θ . There are many different ways of organizing such a search and some example methods will be discussed in the sequel [1].

2.1 Predictors Based on Output Error

Suppose that a batch of data from the system is collected as:

$$Z^N = [y(1), u(1), y(2), u(2), \dots, y(N), u(N)] \quad (2.1)$$

A test by which the different models' ability to describe the observed data can be evaluated is sought. Since a model's essence is its prediction aspect, this can be used to judge its performance in this respect. Define the prediction error given by a certain model $M(\Theta_*)$ as:

$$\epsilon(t, \Theta_*) = y(t) - \hat{y}(t|\Theta_*) \quad (2.2)$$

When the data set is Z^N is known, these errors can be computed for $t = 1, 2, \dots, N$. Thus guiding principle for parameter estimation using output error becomes: Based on Z^t , prediction error $\epsilon(t, \Theta)$ can be computed using (2.2). At time $t = N$, select $\hat{\Theta}_N$ so that the prediction errors $\epsilon(t, \hat{\Theta}_N), t = 1, 2, \dots, N$, become as small as possible.

The prediction-error sequence in (2.2) can be seen as a vector in R^N so the size of this vector can be measured using any norm in R^N . Let the prediction-error

sequence be filtered through a stable linear filter $L(q)$:

$$\epsilon_F(t, \Theta) = L(q)\epsilon(t, \Theta) \quad (2.3)$$

Typically the following norm can be used:

$$V_N(\Theta, Z^N) = \frac{1}{N} \sum_{t=1}^N l(\epsilon_F(t, \Theta)) \quad (2.4)$$

where $l(\cdot)$ is a scalar-valued (typically positive) function. The estimate $\hat{\Theta}$ is then defined by minimization of (2.4). There are several methods at this point to minimize the sequence of model prediction error:

- The *prediction-error identification approach* (PEM) defined above contains well-known procedures, such as the least-squares (LS) method and the maximum-likelihood (ML) method.
- The *subspace approach to identify state-space models* consists of three steps: (1) estimating the k -step ahead predictors using an LS-algorithm, and (2) selecting the state vector from these, and finally (3) estimating the state-space matrices using these stated and the LS-method.
- There is also another approach named *correlation approach*, which contains the instrumental-variable (IV) technique, as well as several methods for rational transfer function models.

Having obtained a proper cost function, any optimization method can be applied to update the estimates of Θ . Abundant literature is available on various techniques of optimization, i.e [10];

1. Unconstrained Gauss-Newton
2. Bounded-Variable Gauss-Newton
3. Levenberg-Marquardt
4. Simplex Method of Nelder and Mead
5. Subspace Simplex method of Rowan
6. Powell's method, conjugate directions

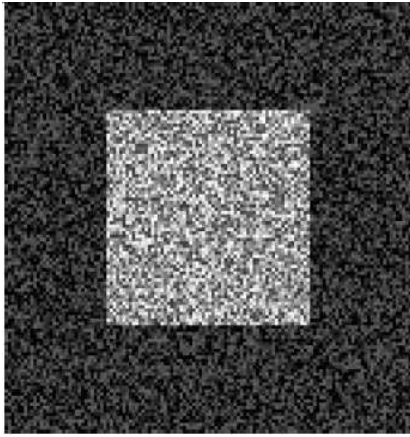
7. Jacob's method of heuristic search

Although these procedures are explained for time domain parameter estimation, it is also possible to extend the search for parameters to the frequency domain, mainly by using Fourier Transforms. In actual practice, most data are collected as samples of the input and output time signals. There are occasions when it is natural and fruitful to consider the Fourier transforms of the inputs and the outputs to be the primary data (e.g. data are collected by a frequency analyzer). This view has been less common in the traditional system identification literature, but has been of great importance in the Mechanical Engineering community, vibrational analysis, and so on. There is a very close relationship between time domain methods and frequency domain methods for estimating linear models [1].

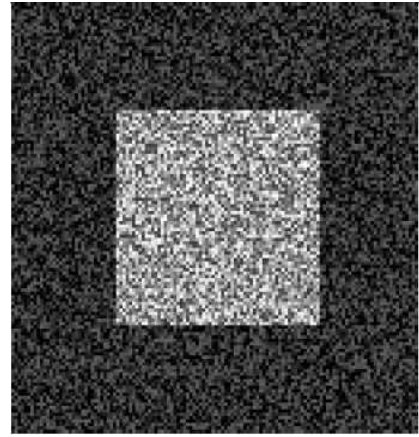
2.2 Methods for Estimating Motion and Structure

2.2.1 Optical Flow Based Methods

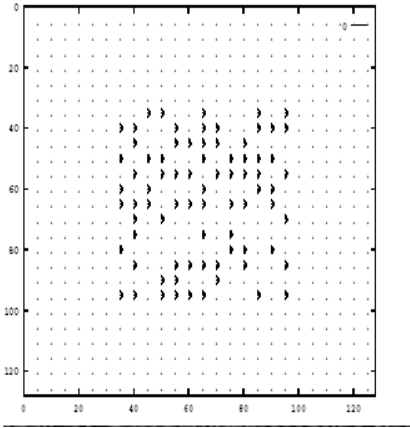
The motion of objects in 3D induces the 2D motion in the image plane. That motion is called *optical flow*. There are several methods to compute optical flow. The optical flow can be used to compute 3D motion, i.e. translation and rotation, and 3D shape. Previous approaches in this class have dealt with the simplified problems involving some assumptions related to the motion of the object, e.g., the assumption of translation motion only, rotation motion only, known depth of objects, and planar surfaces. Recently, Heeger and Jepson have proposed a general method for computing 3D motion and depth from optical flow. The method can be summarized as follows: Optical flow (u, v) is given by some dynamics under the assumption of perspective projection of the camera where in this dynamics system matrices are consisting of known image coordinates and there are unknown motion parameters to be estimated. For each point in the image, a separate equation can be written and can be combined into a matrix equation. When large number of flow vectors are used in this manner, resulting system of equations can be solved using least squares estimates. Their method first computes translation, followed by rotation, and then depth. The optical flow computation is illustrated in the following figure [11]:



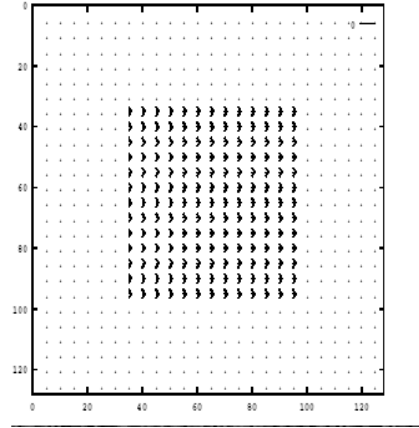
(a)



(b)



(c)



(d)

Figure 2.1: Optical Flow (a)Frame 1 (b)Frame 2 (c) 1 iteration (d) 10 iterations

2.2.2 Feature Point Based Methods

The other general approach to the problem of motion and shape estimation considers the discontinuity curves in the brightness pattern of the image; a possible approach is to identify the correspondance of various features such as points, lines and curves between consecutive frames.

This class of methods for estimating the motion field is also known as *matching techniques*, which estimate the motion field at feature points only. The class can

be subdivided into two main categories as *two-frame methods: feature matching* and *multiple-frame methods: feature tracking*. Both methods use Kalman filtering techniques extensively [12].

By identifying feature points in the sequence of images, it is possible to develop a model of the motion using various techniques, e.g. using the approaches described at the beginning of the chapter. The approach that will be developed in this work also falls in this category where as features, enough number of easy-to-extract points are selected on the object.

Chapter 3

Sliding Mode Variable Structure Control

3.1 Introduction

Sliding mode control is a particular type of *Variable structure control*. Variable structure control systems (VSCS) are characterised by a suite of feedback control laws and a decision rule. The decision rule, termed the switching function, has as input some measure of the the current system behaviour and produces as an output the particular feedback controller which should be used at that instant in time. The result is a variable structure system (VSS), which may be regarded as a combination of subsystems where each subsystem has a fixed control structure and is valid for specified regions of system behaviour.

Variable structure systems first appeared in the late fifties in Russia, as a special class of nonlinear systems. At the very beginning, VSS were studied for solving several specific control tasks in second-order linear and nonlinear systems. The most distinguishing property of VSS is that the closed loop system is completely insensitive to system uncertainties and external disturbances. However, VSS did not receive wide acceptance among engineering professionals until the first survey paper was published by Utkin, [13]. Since then, and especially during later 80s, the control research community has shown significant interest in VSS. This increased interest is explained by the fact that robustness has become a major requirement in modern control applications. Due to its excellent invariance and robustness properties, variable structure control has been developed into a general design method and extended to a wide range of system types including multivariable, large-scale, infinite-dimensional and stochastic systems. The applications include control of aircraft and spacecraft flight, control of flexible structures, robot manipulators, electrical drives,

electrical power converters and chemical engineering systems.

3.2 Sliding-Mode in Variable Structure Systems

Sliding mode control (SMC), which is sometimes known as variable structure control (VSC), is characterized by a discontinuous control action which changes structure upon reaching a set of predetermined switching surfaces. This kind of control may result in a very robust system and thus provides a possibility for achieving the goals of high-precision and fast response. Some promising features of SMC are listed below:

- The order of the motion can be reduced
- The motion equation of the sliding mode can be designed linear and homogeneous, despite that the original system may be governed by non-linear equations.
- The sliding mode does not depend on the process dynamics, but is determined by parameters selected by the designer.
- Once the sliding motion occurs, the system has invariant properties which make the motion independent of certain system parameter variations and disturbances. Thus the system performance can be completely determined by the dynamics of the sliding manifold.

Consider the system defined below:

$$\dot{x} = f(x, t) + B(x, t)u(x, t) \quad (3.1)$$

here $x \in \mathfrak{R}^n$, $u \in \mathfrak{R}^m$, $f(x, t)$ and $B(x, t)$ are assumed continuous and bounded and the rank of $B(x, t)$ is m . The discontinuous control is given by:

$$u_i = \begin{cases} u_i^+ & \text{for } \sigma_i(x) > 0 \\ u_i^- & \text{for } \sigma_i(x) < 0 \end{cases} \quad (3.2)$$

for $i = 1, 2, \dots, m$, $\sigma(x) = Gx$, $\sigma \in \mathfrak{R}^m$ whose components are m smooth functions and $G \in \mathfrak{R}^{m \times n}$, yielding

$$\sigma(x) = \left[\sigma_1(x) \quad \sigma_2(x) \quad \cdots \quad \sigma_m(x) \right]^T \quad (3.3)$$

here u_i^+ , u_i^- , and $\sigma_i(x)$ are continuous functions with $u_i^+ \neq u_i^-$. Sliding mode may appear on the manifold $\sigma(x) = 0$, which is the intersection of m hyperplanes defined by the m components of $\sigma(x)$ as $\sigma_i(x) = 0$, $i = 1, 2, \dots, m$. Note that $\sigma(x)$ is called the “switching function” and if sliding mode exists, $\sigma(x) = 0$ is called the “sliding manifold” or “sliding hyperplane” of m dimensions, since i th control u_i faces discontinuities on the i th surface $\sigma_i(x)$ in terms of switching according to (3.2), $i = 1, 2, \dots, m$. If, for any initial condition x_0 , there exists a time t_0 such that $x(t)$ is on the manifold $\sigma(x) = 0$ for $t \geq t_0$, then $x(t)$ is a “sliding mode” of the system, in which the motion is determined by the manifold equation only and therefore, note that motion order is reduced to the order of control inputs, namely m . The order reduction means that system model of the n th order is decomposed into two modes, one is the so-called “reaching mode” which is defined by a motion

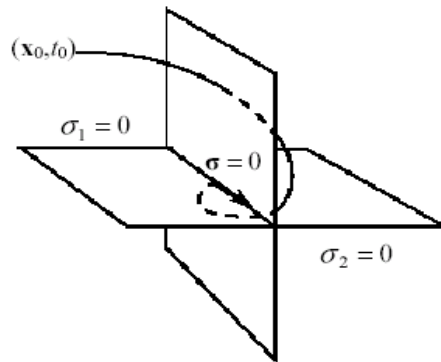


Figure 3.1: Two intersecting switching surfaces

of $(n - m)$ th order and the other is the sliding mode defined by the motion on the sliding manifold of m th order. Decoupled motion equations of the system could be written as

$$\dot{x}_1 = f_1(x_1, \sigma(x)) \quad (3.4)$$

$$x_2 = \sigma(x) \quad (3.5)$$

for $x_1, f_1 \in \mathfrak{R}^{n-m}$ and $x_2 \in \mathfrak{R}^m$. If $\sigma(x) = 0$ is appropriately designed in such a way that it satisfies the control objectives (e.g. x follows x^{ref}), then SMC is realized.

In real implementations, the trajectories are confined to some vicinity of the switching line. The deviation from the ideal model may be caused by imperfections of switching devices such as small delays, dead zones and hysteresis, which may lead to high-frequency oscillations. The same phenomenon may appear due to small time constants of sensors and actuators having been neglected in the ideal model. This phenomenon, called *chattering*, was a serious obstacle to the use of sliding modes in control systems [14].

3.3 The Idea of Equivalent Control

The notion of equivalent control, which will be used extensively in this work, will be explained on a simple example system in this section [15]. For the purpose of illustration consider the double integrator given by

$$\ddot{y}(t) = u(t) \quad (3.6)$$

with the control law

$$u(t) = \begin{cases} -1 & \text{if } s(y, \dot{y}) > 0 \\ 1 & \text{if } s(y, \dot{y}) < 0 \end{cases} \quad (3.7)$$

where the switching function is defined by

$$s(y, \dot{y}) = m\dot{y} + \ddot{y} \quad (3.8)$$

where m is a positive design scalar. For values \dot{y} satisfying the inequality $m|\dot{y}| < 1$, then

$$s\dot{s} = s(m\dot{y} + \ddot{y}) = s(m\dot{y} - \text{sgn}(s)) < |s|(m|\dot{y}| - 1) < 0$$

Consequently the system trajectories on either side of the line

$$L_s = \{(y, \dot{y}) : s(y, \dot{y}) = 0\} \quad (3.9)$$

point toward the line. The system described by (3.6) is simulated for $m = 1$, and

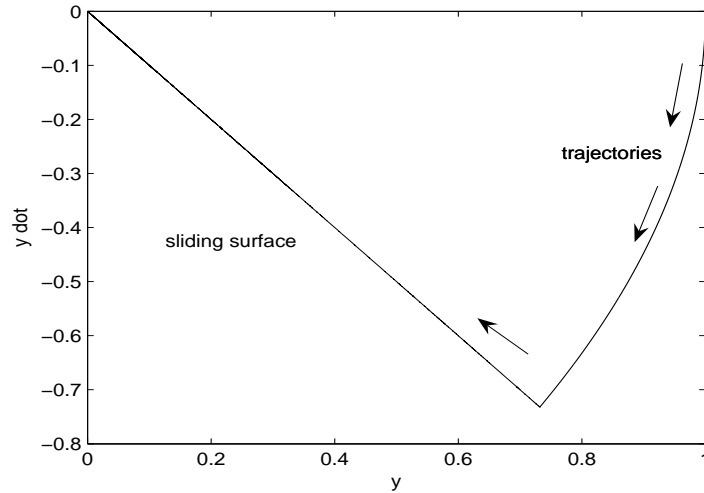


Figure 3.2: Phase portrait of a sliding motion

the initial conditions are given by $y = 1$ and $\dot{y} = 0$. The two stage nature of the dynamics is readily observed in Figure 3.2 : the initial (parabolic) motion towards the sliding surface, followed by a motion along the line $\dot{y} = -y$ towards the origin. The control action associated with this simulation is given in Figure 3.3. It can be seen that sliding takes place after about 0.7 seconds when high frequency switching takes place. Now suppose that at time t_s the switching surface is reached and an

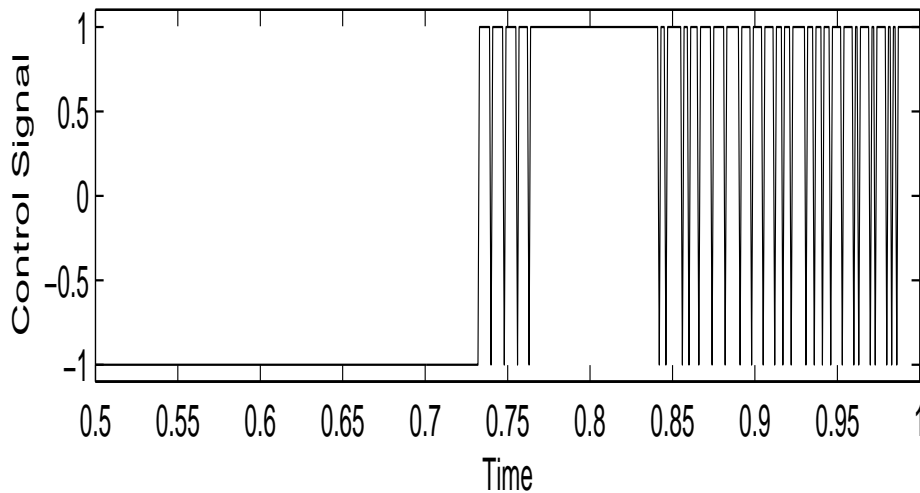


Figure 3.3: Discontinuous control action

ideal sliding motion takes place. It follows that the switching function satisfies $s(t) = 0$ for all $t > t_s$, which in turn implies that $\dot{s}(t) = 0$ for all $t \geq t_s$. From

equations (3.6) and (3.8)

$$\dot{s}(t) = m\dot{y}(t) + u(t) \quad (3.10)$$

and thus since $\dot{s}(t) = 0$ for all $t \geq t_s$, it follows from (3.10) that a control law which maintains the motion on L_s is

$$u(t) = -m\dot{y}(t) \quad (3.11)$$

This control law is referred as the *equivalent control*. This is not the control signal which is actually applied to the plant but may be thought of as the control signal which is applied on average. This can be demonstrated by passing the discontinuous control signal (in Figure 3.3) through a low pass filter to obtain the low frequency component of the control, u_{low} . Figure 3.4 shows $u_{low}(t)$, together with the associ-

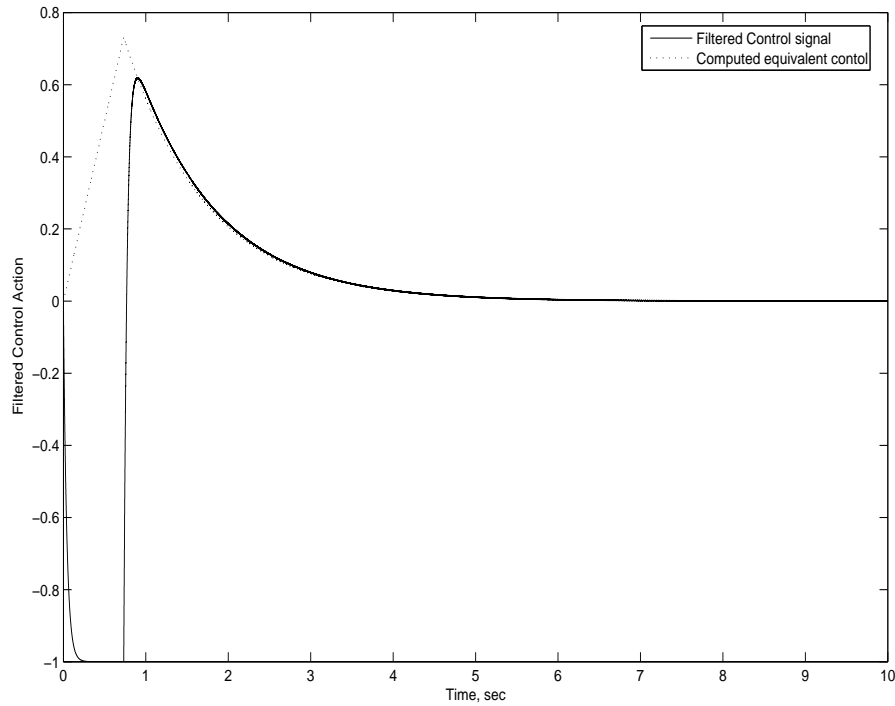


Figure 3.4: Equivalent Control

ated equivalent control. It can be seen that the filtered control signal agrees with the equivalent control action defined in (3.11) once sliding is established - in this case after about 0.7 seconds.

3.4 Remarks

The ideas presented in this chapter will be used extensively in the sequel. In the context of parameter estimation, we will use basic concepts of sliding mode control, without actually controlling any plant physically (Sliding Mode Observers). This will reveal us from considering the practical issues related to the application of sliding mode control, i.e. chattering. Also most of the time, having a finite bound on state velocities will be enough to ensure the reachability condition, thus existing of sliding modes. This is simply because our control signals will run on computers rather than being actual currents, meaning that they can attain any finite value.

Chapter 4

Using Sliding Mode to Estimate Motion Parameters

4.1 Motion Models

If a rigid body is moving with instantaneous translational velocity, T , and rotational velocity, Ω , then 3D instantaneous velocity of points on the surface is given by

$$\frac{d}{dt} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \Omega \times \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \Rightarrow \frac{d}{dt} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} + \begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix} \quad (4.1)$$

where $\Omega = (\omega_1, \omega_2, \omega_3)^T$ and $T = (t_1, t_2, t_3)^T$

An affine motion in 2D, on the other hand, can be described as

$$\frac{d}{dt} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} a_1 & a_2 \\ a_3 & a_4 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} \quad (4.2)$$

In case the 2×2 matrix

$$M = \begin{pmatrix} a_1 & a_2 \\ a_3 & a_4 \end{pmatrix} \quad (4.3)$$

happens to be a skew-symmetric matrix, i.e. $M + M^T = 0 \Leftrightarrow M = \begin{pmatrix} 0 & \omega \\ -\omega & 0 \end{pmatrix}$, the motion will be termed as a rigid motion.

4.2 Estimation of Rigid Motion

The problem definition for rigid motion estimation is as follows; If the motion of the object is assumed to be rigid in the setup above, how can one determine the parameters of the assumed motion model using visual information? An algorithm

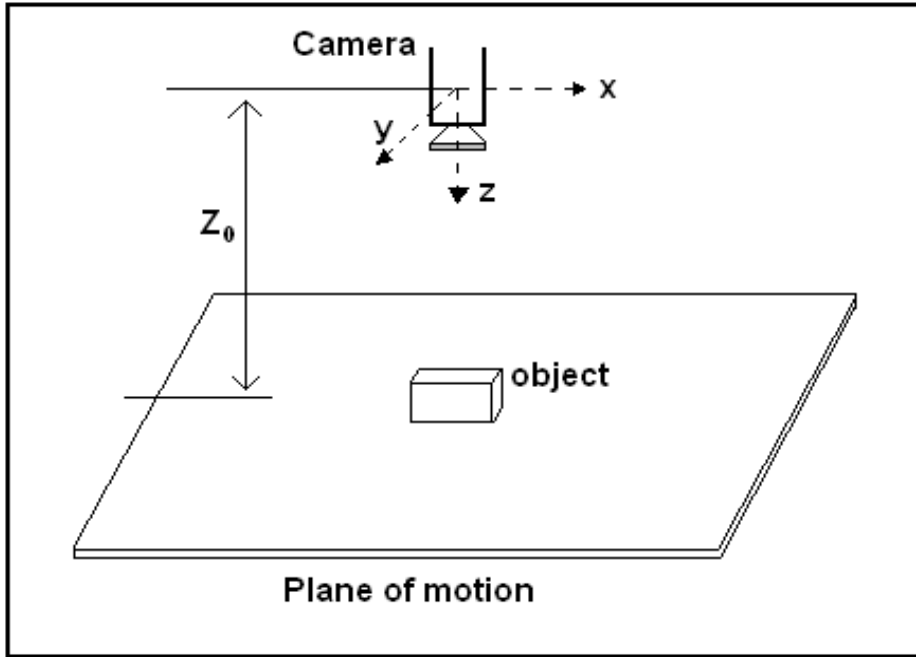


Figure 4.1: Vision Setup

that will provide a fast estimation for these parameters, thus providing an online identification of the rigid motion of the object is searched. These parameters are not restricted to be constant, they can also be time varying. The main approach is different from collecting visual data for some time interval and then trying to fit a motion model to the gathered data; instead, proposed algorithm tries to produce a running estimation of the unknown parameters, hence the meaning of online identification.

4.3 Problem Formulation

Suppose a moving object is performing a rigid motion with unknown parameters. The dynamics of the motion, as mentioned before, will be given by :

$$\frac{d}{dt} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} + \begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix} \quad (4.4)$$

If the motion of the object is confined to a plane, then the above dynamics reduces to:

$$\frac{d}{dt} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} 0 & -\omega & 0 \\ \omega & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \\ 0 \end{pmatrix} \quad (4.5)$$

where ω is the rotational velocity along optical axis, $b_1 = t_1/Z_0$ and $b_2 = t_2/Z_0$, with Z_0 being the distance between camera and the object plane. So in this setting of the problem, there are three unknown parameters to determine, namely ω , b_1 and b_2 .

4.4 Parameter Estimation Problem; Redefined

To use sliding mode idea in this framework, we redefine the problem of *rigid motion estimation* as a usual parameter estimation problem of a *linear time-varying system* (LTV), a hot topic among control community. Consider a LTV system of the following form :

$$\frac{d}{dt}(x) = A(t)x + b(t), \quad y = x \quad (4.6)$$

Here, $x \in R^n$ is the state vector, $A(t) \in R^{n \times n}$ is the system matrix, $b(t) \in R^n$ is an unknown vector field, $y \in R^n$ is the measurement vector. Note that full state information is assumed here. Let us first consider the very general case. Suppose that the number of unknown parameters in matrix A is k , and the number of unknown parameters in vector b is p with $0 \leq k \leq n^2$, $0 \leq p \leq n$ and $k + p > 0$. Note that totally we are looking for $k + p$ parameters. (4.6) can be recast linearly in terms of the unknown parameters as follows:

$$\frac{d}{dt}(x) = (B(x) | C) \begin{pmatrix} q_1 \\ q_2 \end{pmatrix} + m(x), \quad y = x \quad (4.7)$$

where

$q_1 \in R^k$: column vector consisting of unknown parameters in A

$q_2 \in R^p$: column vector consisting of unknown parameters in vector b

$B \in R^{n \times k}$: a matrix which is a function of the states to be constructed appropriately

$C \in R^{n \times p}$: a matrix consisting of 1's and 0's to be constructed appropriately

$m \in R^n$: a vector which is a function of the states and known parameters

The following example is given to illustrate this procedure:

Dynamic System :

$$\frac{d}{dt} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 1 & 3 & a_1 & a_2 \\ 0 & a_3 & 5 & 0 \\ a_4 & 1 & a_5 & 2 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} + \begin{pmatrix} 4 \\ b_1 \\ b_2 \\ 0 \end{pmatrix} \quad (4.8)$$

Here in this example, we have both known and unknown parameters in $A(t)$ and $b(t)$ of the system given in (4.6). Since q_1 and q_2 of (4.7) consist of unknown parameters in matrix A and b , following is the case for this example system :

$$q_1 = [a_1 \ a_2 \ a_3 \ a_4 \ a_5], \quad q_2 = [b_1 \ b_2]$$

So when all the unknown parameters are combined in a vector, the system (4.6) is represented by the following set of differential equations:

$$\frac{d}{dt} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} x_3 & x_4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & x_2 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & x_1 & x_3 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ b_1 \\ b_2 \end{pmatrix} + \begin{pmatrix} x_1 + 3x_2 + 4 \\ 5x_3 \\ x_2 + 2x_1 \\ 0 \end{pmatrix} \quad (4.9)$$

Note here that $n = 4$, $k = 5$ and $p = 2$, so $B(x) \in R^{4 \times 5}$, $C \in R^{4 \times 2}$ and $m \in R^4$. The system can also be rewritten in a more compact way, by expanding the unknown parameter vector as:

$$\Xi = [a_1 \ a_2 \ a_3 \ a_4 \ a_5 \ b_1 \ b_2 \ 1]^T$$

The equation describing the system (4.8) becomes:

$$\frac{d}{dt} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} x_3 & x_4 & 0 & 0 & 0 & 0 & 0 & x_1 + 3x_2 + 4 \\ 0 & 0 & x_2 & 0 & 0 & 1 & 0 & 5x_3 \\ 0 & 0 & 0 & x_1 & x_3 & 0 & 1 & x_2 + 2x_4 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \times \Xi \quad (4.10)$$

Thus, by this procedure, we can rewrite the equations of the system in such a way that all the unknown parameters are put together in a vector. This format can always be achieved as long as the system equations are linear with respect to parameters, whether the system equations are linear or non-linear with respect to states.

Next section will present the motivation behind this redefinition of the system equations.

4.5 Sliding Mode Based Solution for Rigid Motion

As depicted in previous sections, motion estimation problem can be considered as a parameter estimation problem of a LTV system. This section will present how previously explained *equivalent control* idea in SMC can be used to estimate the motion parameters of an object, when the motion is of rigid type and confined to a plane.

More precisely, suppose Fig. 4.1 is the system in consideration, thus equations describing the motion of the object will be given as in (4.5). The moving object is being viewed by a stationary CCD camera on the top, so that a sequence of images is available. By processing the gathered images, the image coordinates of a easy-to-track feature point on the object will be computed at every frame. So when the image plane dynamics is considered, these image coordinates of the feature point, namely x and y will be the states, and the following equations will be the description of the rigid motion dynamics projected on image plane:

$$\frac{d}{dt} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 & -\omega \\ \omega & 0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} \quad (4.11)$$

Suppose all three parameters (ω, b_1, b_2) are possibly time-varying unknown param-

eters. As explained in previous section, one can rewrite this system as follows:

$$\underbrace{\frac{d}{dt} \begin{pmatrix} x \\ y \end{pmatrix}}_X = \underbrace{\begin{pmatrix} -x & 1 & 0 \\ y & 0 & 1 \end{pmatrix}}_\Omega \underbrace{\begin{pmatrix} \omega \\ b_1 \\ b_2 \end{pmatrix}}_T \Rightarrow \dot{X} = \Omega T \quad (4.12)$$

Though the state values are obtained through CCD camera and are known at every frame, an observer, \hat{X} , whose state follows the state of the composite image feature dynamics given in (4.11) as closely as possible will be constructed. In doing so, sliding mode control will be employed to stabilize the state estimation error around zero. Basically image feature dynamics will be copied and then this copied version will be controlled by SMC. More precisely, let our observer be

$$\dot{\hat{X}} = u \quad (4.13)$$

where u will be designed using SMC so that $\tilde{X} = \hat{X} - X \rightarrow 0$ as $t \rightarrow \infty$. Let us define the sliding mode manifold as

$$\sigma = X - \hat{X}$$

which, in light of (4.12), then implies that

$$\dot{\sigma} = \dot{X} - \dot{\hat{X}} = \Omega T - u \quad (4.14)$$

To guarantee global asymptotic stability, Lyapunov theory can be employed, by selecting an appropriate Lyapunov function candidate as

$$V = \frac{1}{2} \sigma^T \sigma \quad (4.15)$$

whose time derivative is

$$\dot{V} = \sigma^T \dot{\sigma} \quad (4.16)$$

which can be made negative definite by setting $\dot{\sigma}$ to either $-MSgn(\sigma)$, where $M > 0$ and $Sgn(\cdot)$ is the signum function, or $-D\sigma$, where D is a positive definite matrix. If $-MSgn(\sigma)$ is selected, all components of the control are switching between lower and upper bound of control. This may cause unnecessary chattering in the system especially in the discrete-time implementations of the control algorithm. Combination of the $\dot{\sigma} = -MSgn(\sigma)$ and $\dot{\sigma} = -D\sigma$ by selecting $\dot{\sigma} = -D\sigma - \rho(x, t)Sgn(\sigma)$

yields a solution that may combine good properties of both solutions and allows selecting $\rho(x, t)$ small enough to minimize chattering and at the same time to guarantee the existence of sliding mode.

So by selecting $\dot{\sigma} = -D\sigma - \rho(x, t)Sgn(\sigma)$, \dot{V} then becomes

$$\dot{V} = -\sigma^T D\sigma - \rho(x, t)\sigma^T Sgn(\sigma) = -\sigma^T D\sigma - \rho(x, t) \|\sigma\|,$$

which is clearly negative definite since $D > 0$ and $\rho > 0$. Therefore for stability,

$$\dot{\sigma} = -D\sigma - \rho Sgn(\sigma) \Rightarrow \dot{\sigma} + D\sigma + \rho Sgn(\sigma) = 0 \quad (4.17)$$

must be satisfied.

When the sliding manifold is reached, the system will be governed by $\sigma = 0$ and $\dot{\sigma} = 0 = \Omega T - u$, from which the *equivalent control* that keeps the system on the manifold can be computed as

$$u_{eq} = \Omega T \quad (4.18)$$

There are different methods to compute equivalent control. If $\dot{\sigma} = -D\sigma - \rho(x, t)Sgn(\sigma)$ is selected, u goes to u_{eq} as soon as the sliding manifold is reached. If $\dot{\sigma} = -MSgn(\sigma)$ is selected, u_{eq} can be obtained by passing the control signal u through a low pass filter. Both approaches has its own advantages that will be discussed later. In any case, when equivalent control is obtained and the system is moving on the manifold, (4.18) will be valid in which the only unknown is parameter vector T . The solution will yield unknown parameters as

$$T = \Omega^{-1} u_{eq} \quad (4.19)$$

4.6 Expanding the System

An important problem to be addressed here is the existence of a solution for (4.19), or in other words following is a crucial question: "Is matrix $\Omega(x)$ invertible?" Not considering any singularities that $\Omega(x)$ may have, at least for the time being, there is an immediate conclusion here; $\Omega(x)$ should be at least a square matrix, which can be translated as *number of states measured should be at least equal to the number of unknown parameters*. As far as the vision system given in (4.11) is concerned (or similar vision systems), this restriction is not a vital one. Remember that for system

given in (4.11), states are the coordinates of the feature points on the object. So by choosing appropriate number of feature points, it possible to make sure that $\Omega(x)$ is at least a square matrix (or even better; an overdetermined system).

In the case of estimation of rigid motion, 3 parameters are unknown, meaning that 2 feature points selected on the object, (x_1, y_1) and (x_2, y_2) will be more than enough, creating an overdetermined system. Overall expanded system dynamics will be given as:

$$\frac{d}{dt} \begin{pmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \end{pmatrix} = \begin{pmatrix} 0 & -\omega & 0 & 0 \\ \omega & 0 & 0 & 0 \\ 0 & 0 & 0 & -\omega \\ 0 & 0 & \omega & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \\ b_1 \\ b_2 \end{pmatrix} \quad (4.20)$$

which can be rewritten as:

$$\frac{d}{dt} \begin{pmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \end{pmatrix} = \begin{pmatrix} y_1 & 1 & 0 \\ -x_1 & 0 & 1 \\ y_2 & 1 & 0 \\ -x_2 & 0 & 1 \end{pmatrix} \begin{pmatrix} \omega \\ b_1 \\ b_2 \end{pmatrix} \quad (4.21)$$

Now with this expanded system, $\Omega(x) \in \mathfrak{R}^{4 \times 3}$, (4.19) becomes an overdetermined set of equations, where the solution can be obtained using pseudo-inverse.

Generally speaking, if the number of unknown parameters is k , and the number of feature points that is being extracted from the image is m , then $2m \geq k$ should be satisfied.

4.7 Singularity in the Solution for Rigid Motion

Since the number of parameters to be estimated is 3 for rigid motion, it is enough to measure 3 states, resulting in a square $\Omega(x)$ matrix so that a solution can be sought as in (4.19). Suppose we expand the system as follows:

$$\frac{d}{dt} \begin{pmatrix} x_1 \\ y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} y_1 & 1 & 0 \\ -x_1 & 0 & 1 \\ -x_2 & 0 & 1 \end{pmatrix} \begin{pmatrix} \omega \\ b_1 \\ b_2 \end{pmatrix} \quad (4.22)$$

where x_1, y_1 are image coordinates of one feature point and y_2 is one of the coordinates of a second feature point. To analyze the existence of the solution, one must look at the determinant of the Ω matrix:

$$\det(\Omega) = x_1 - x_2 \quad (4.23)$$

This result shows that whenever the line that connects the two feature points becomes parallel to the y -axis, the above determinant will assume the value 0 and the solutions will jump to infinity. The matrix gets ill-conditioned as x_1 and x_2 assume close values. This is indeed verified via simulations.

To overcome this difficulty, creating an overdetermined system using 4 state values for 3 parameters is considered. Consider again $\Omega(x)$ in (4.21). To solve this overdetermined system, following steps are taken:

$$\begin{aligned} u_{eq} &= \Omega(x)T \\ \Omega^T(x)u_{eq} &= \Omega^T(x)\Omega(x)T \\ (\Omega^T(x)\Omega(x))^{-1}\Omega^T(x)u_{eq} &= T \end{aligned}$$

So to analyze the solutions, one must check the determinant of $\Omega^T(x)\Omega(x)$:

$$\det(\Omega^T(x)\Omega(x)) = 2((x_1 - x_2)^2 + (y_1 - y_2)^2) \quad (4.24)$$

This quadratic equation assumes 0 value only when $x_1 = x_2$ and $y_1 = y_2$ are satisfied simultaneously. But this contradicts the fact that two feature points are distinct points taken on the object. So by using 4 states and pseudo-inversion, it is guaranteed that a solution will always exist.

4.8 Simulation Results for Rigid Motion

The approach presented in this chapter is simulated using Matlab 7.0 and Simulink 6.0. The following simulink model is created:

- Block 1 : Experimental data is generated for simulation purposes. The image data is generated with 50 frames per second.
- Block 2 : Control input u is computed, using state values and observer error σ

- Block 3 : Takes as input u_{eq} and state values, and computes the unknown parameter vector T using (4.19)
- Block 4 : Low pass filter block to generate u_{eq} from u
- Block 5 : Filter the results

The simulation data is as follows:

- Run Time : 10 s
- Simulation Sample Time: 0.0001 s
- Time constant of filter in Block 4, $\tau_c=0.001$ s
- Time constant of filter in Block 5, $\tau_p=0.01$ s

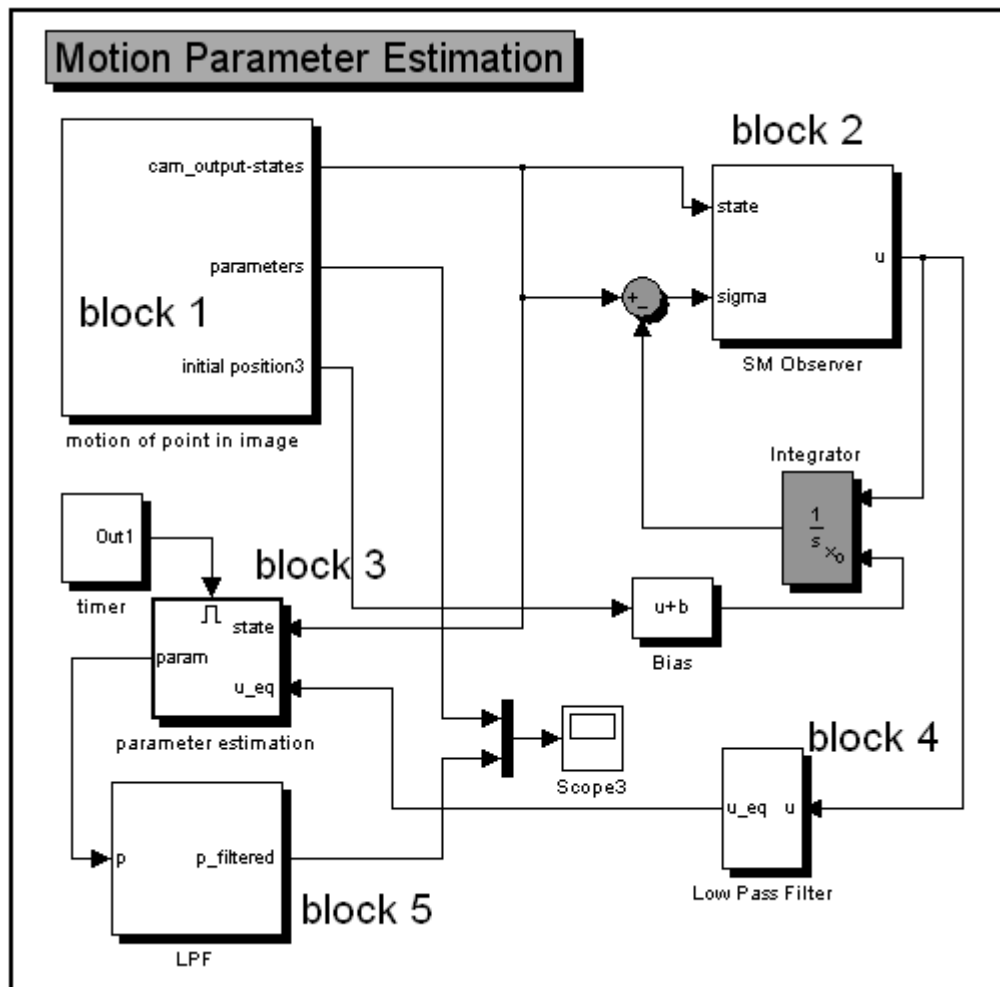


Figure 4.2: Simulink Model

Case 1: Pure Rotational Motion, Constant Parameters :

The motion of the object is generated with the following parameters:

$$\omega = 2$$

$$b_1 = 0$$

$$b_2 = 0$$

The resulting motion of the object in the image plane is depicted in the following figure:

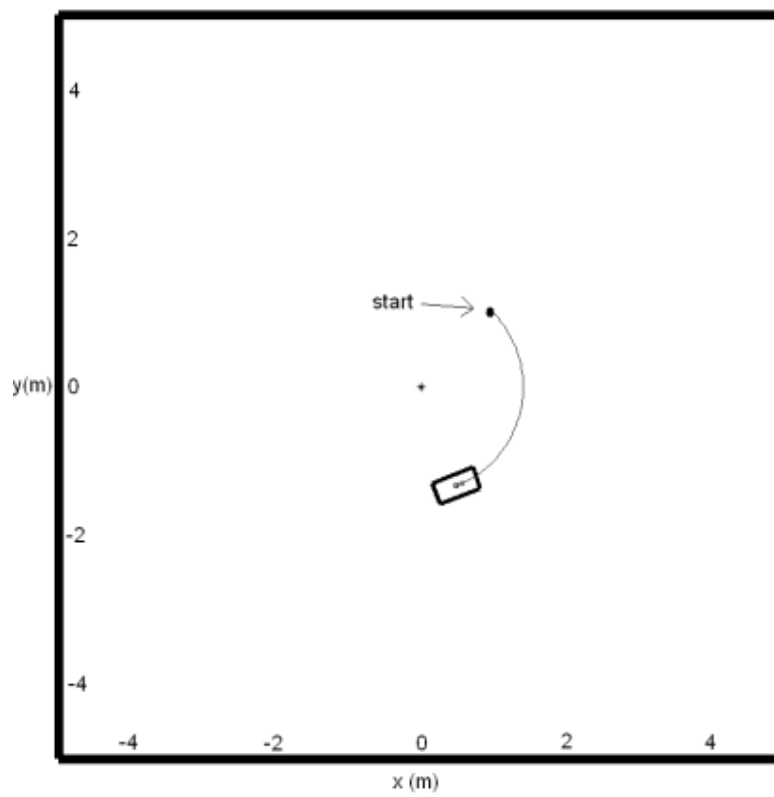


Figure 4.3: Trajectory of the object

Results for estimation of ω is as follows:

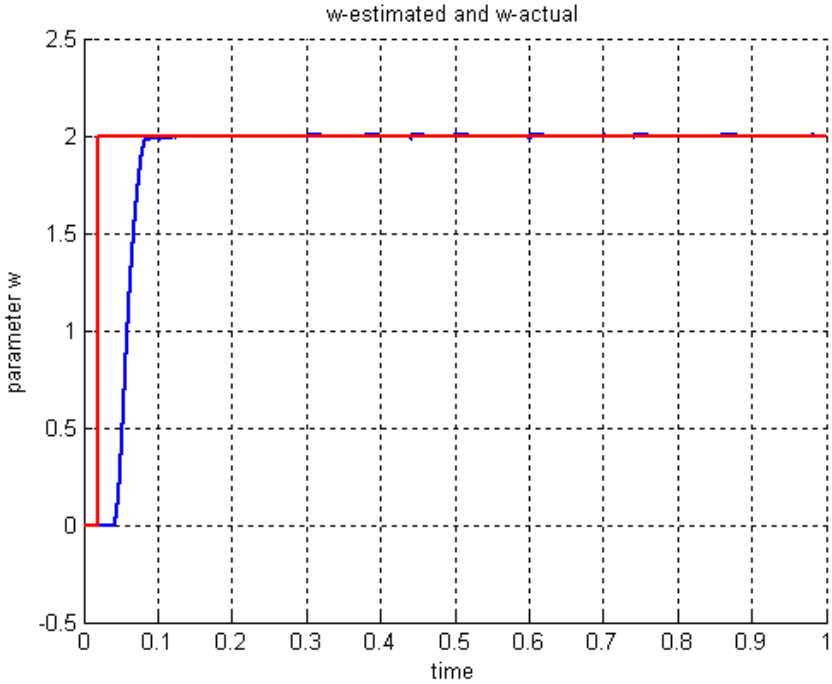


Figure 4.4: ω and ω -estimate

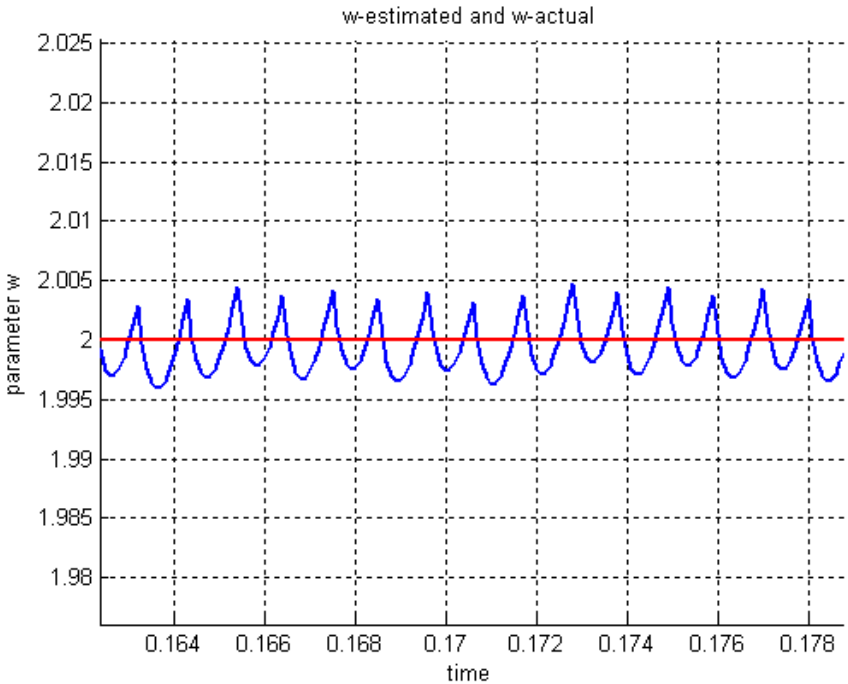


Figure 4.5: ω and ω -estimate, zoomed

As it is seen in the figures, the estimate value of ω catches the true value as soon as the sliding manifold is reached. Fig. 4.5 gives a more detailed view of the estimation performance. As seen in the figure, estimation is oscillatory. This oscillation is due to the well known problem in sliding mode control, so called chattering problem, which is the result of the high frequency switching between different control structures as the system trajectories repeatedly cross the sliding surfaces. A possible remedy could be implementing higher order sliding modes, or different control structures so that a continuous control signal is employed instead of a discontinuous one.

Results for estimation of b_1 and b_2 is as follows:

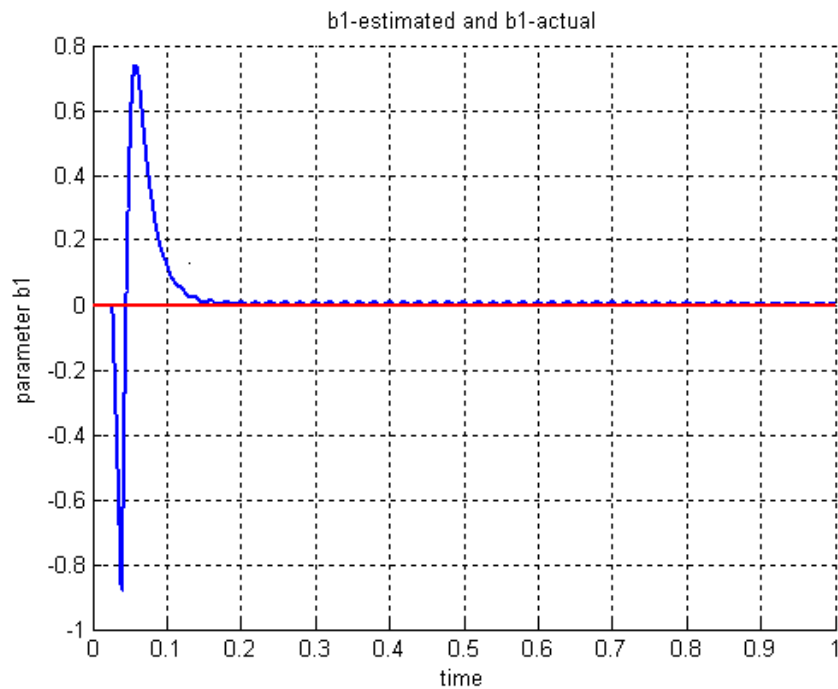


Figure 4.6: b_1 and b_1 -estimate

As seen in the figure, the estimate value of b_1 catches the true value after the reaching phase to the sliding manifold. The next figure illustrates the estimation performance for b_2 .

Figures 4.4 to 4.20 show that SMO algorithm works fine, and fast estimation of the parameters with acceptable accuracy is achieved for constant parameters.

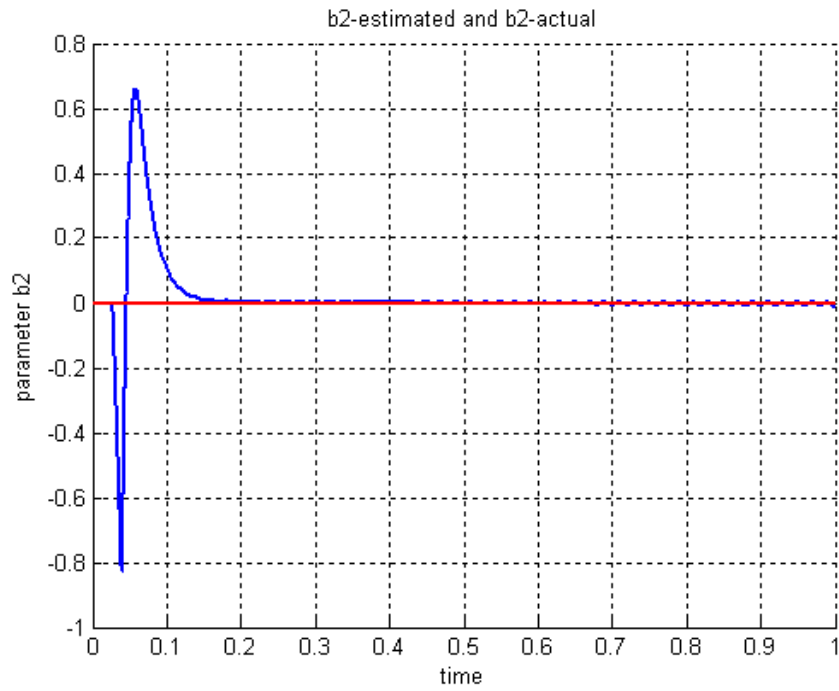


Figure 4.7: b_2 and b_2 -estimate

Case 2: Rigid Motion with Time Varying Parameters

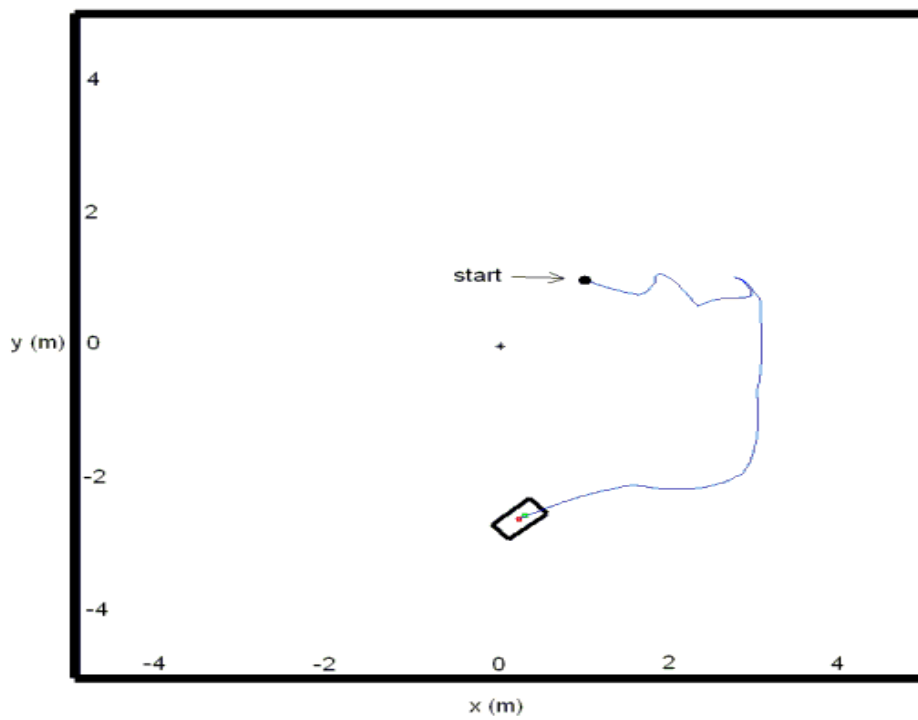


Figure 4.8: Object Trajectory

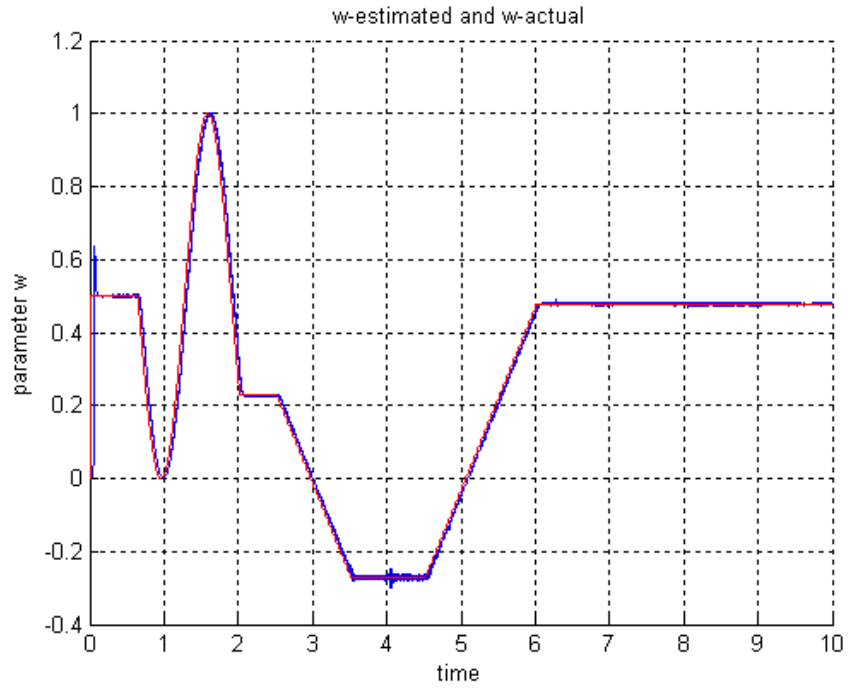


Figure 4.9: $\omega(t)$ and $\omega(t)$ -estimate

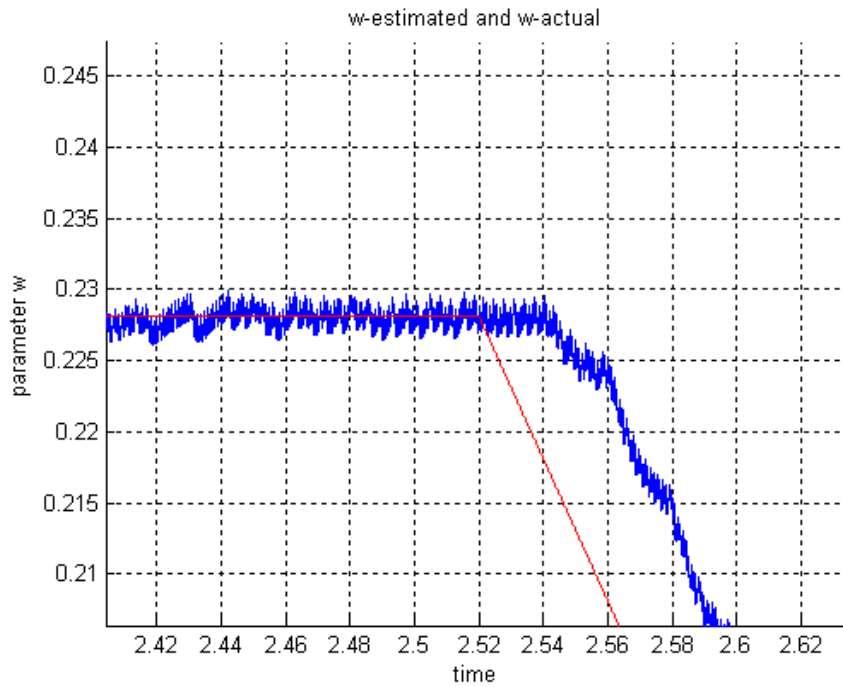


Figure 4.10: $\omega(t)$ and $\omega(t)$ -estimate, zoomed

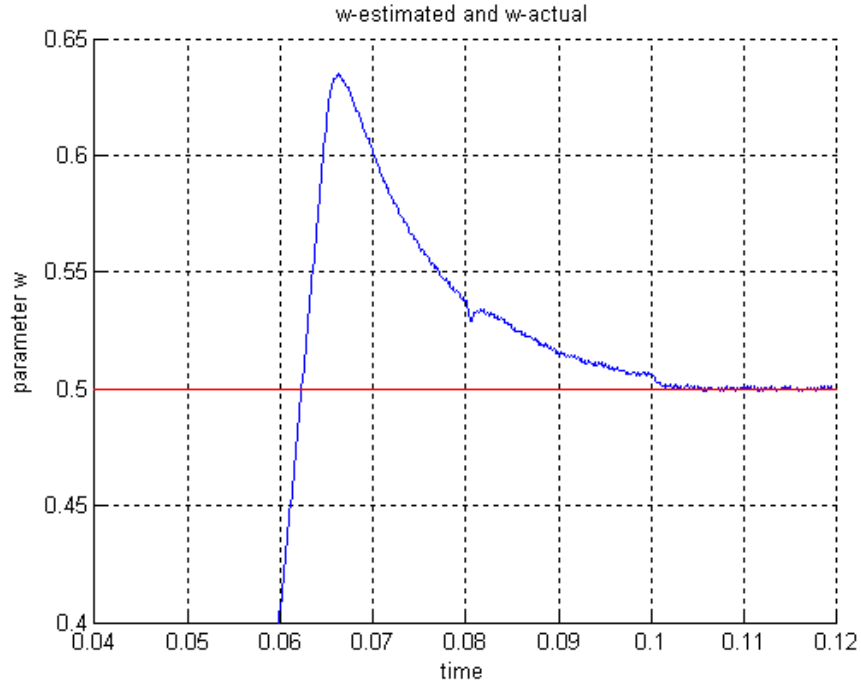


Figure 4.11: Reaching to manifold

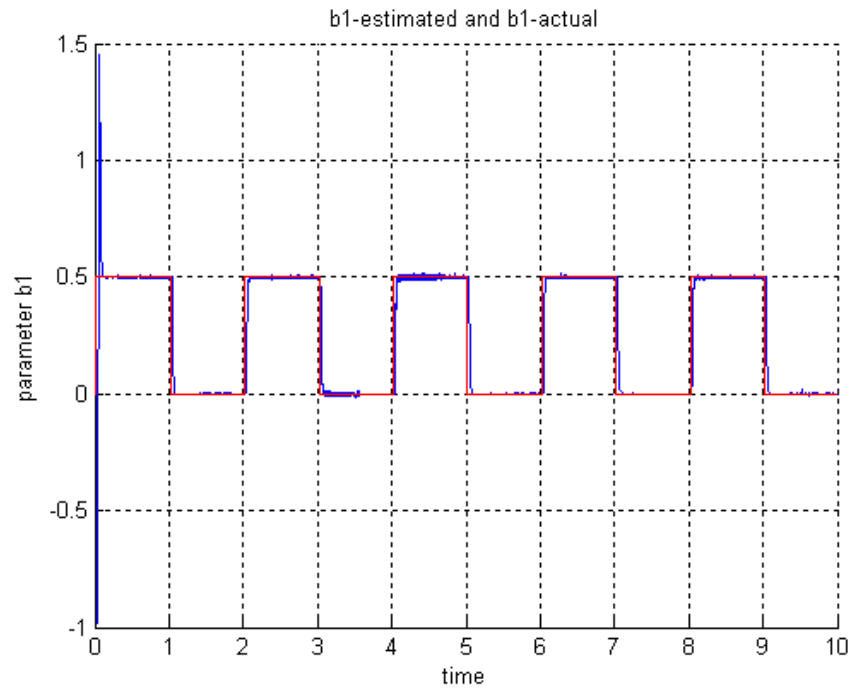


Figure 4.12: $b_1(t)$ and $b_1(t)$ -estimate

As seen in Figures 4.9 to 4.13, SMO algorithm works fine for time varying parameters also. There is a computational problem to be addressed here, which is

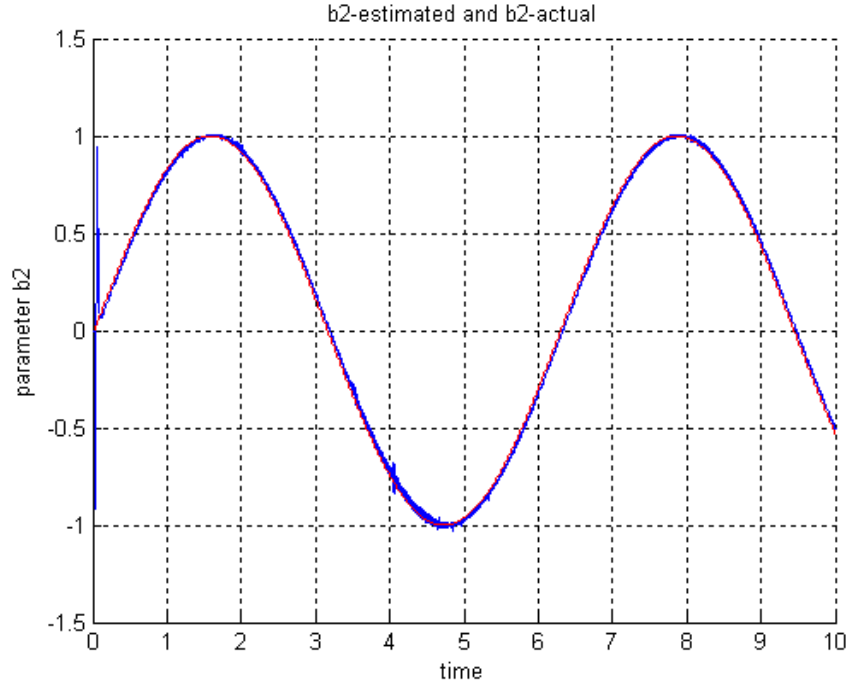


Figure 4.13: $b_2(t)$ and $b_2(t)$ -estimate

visualized in Fig. 4.10. As seen in the figure, estimation has a lag with respect to the original parameter. This may cause a problem when the parameters are time varying. Since image data is available in every $1/\text{fps}$ seconds (which is 0.02 seconds, and bigger than the simulation sample time), a linear interpolation block inside Block 1 of Fig 4.2 is constructed, that interpolates the state values between two consecutive frames. The lag problem is introduced by this interpolator. To implement such an interpolator, the estimator should be started no earlier than the time the second frame is captured, which means that the estimator will follow the time varying parameters with a lag equal to $1/\text{fps}$ (Note in Figure 10 that lag between signals is approximately 0.02 seconds, and the simulation is run with 50fps). So interpolation comes with its price, but this price is worth to pay, especially when low frame rates are considered. The convergence of the estimates to their true values is depicted in more detail in Fig 4.11.

4.9 Estimation of Affine Motion

One of the motivations for studying affine dynamics in 2D image plane is that it is the projection of 3D rigid dynamics under the weak perspective camera model. An almost planar object performing rigid motion in 3D space has the dynamics given in (4.4). The weak perspective equations are

$$x = f \frac{X}{Z} = \alpha X \quad \text{and} \quad y = f \frac{Y}{Z} = \alpha Y$$

where \bar{Z} is the average scene depth. Thus the resulting image dynamics in terms of (x, y) are obtained from the dynamics of (X, Y) , by using weak perspective equations and scene depth substitutions, $Z = pX + qY + r$ where required as:

$$\frac{d}{dt} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \omega_2 p & -\omega_3 + \omega_2 q \\ \omega_3 - \omega_1 p & -\omega_1 q \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} (b_1 + \omega_2 r)\alpha \\ (b_2 - \omega_2 r)\alpha \end{pmatrix} \quad (4.25)$$

which are clearly in the form of affine dynamics in 2D space, verifying the practical importance of 2D affine motion and related work.

Thus the practical categorization of a motion into rigid or affine groups is mainly determined by the position and the orientation of the camera viewing the moving object. Suppose that an affine motion in 2D of the form is given:

$$\frac{d}{dt} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} a_1 & a_2 \\ a_3 & a_4 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} \quad (4.26)$$

Suppose all six parameters are possibly time-varying unknown parameters. As explained in previous sections, one can rewrite this system as follows:

$$\frac{d}{dt} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x & y & 0 & 0 & 1 & 0 \\ 0 & 0 & x & y & 0 & 1 \end{pmatrix} \Phi \quad (4.27)$$

where $\Phi = [a_1, a_2, a_3, a_4, b_1, b_2]^T$

A Note on Existence

It is readily seen at this point that the matrix $\Omega(x)$ is in $\mathfrak{R}^{2 \times 6}$, so that the system should be expanded before a solution can be obtained as in (4.19). Obviously, number of feature points needed is 3, and the resulting expanded system for affine

motion will be as follows:

$$\frac{d}{dt} \begin{pmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \\ x_3 \\ y_3 \end{pmatrix} = \begin{pmatrix} x_1 & y_1 & 0 & 0 & 1 & 0 \\ 0 & 0 & x_1 & y_1 & 0 & 1 \\ x_2 & y_2 & 0 & 0 & 1 & 0 \\ 0 & 0 & x_2 & y_2 & 0 & 1 \\ x_3 & y_3 & 0 & 0 & 1 & 0 \\ 0 & 0 & x_3 & y_3 & 0 & 1 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ b_1 \\ b_2 \end{pmatrix} \quad (4.28)$$

4.10 Simulation Results for Affine Motion

The procedure taken for estimating the affine motion of the object is the same as done for rigid motion. With the same controllers and simulation data, the following results are obtained for time varying parameters case:

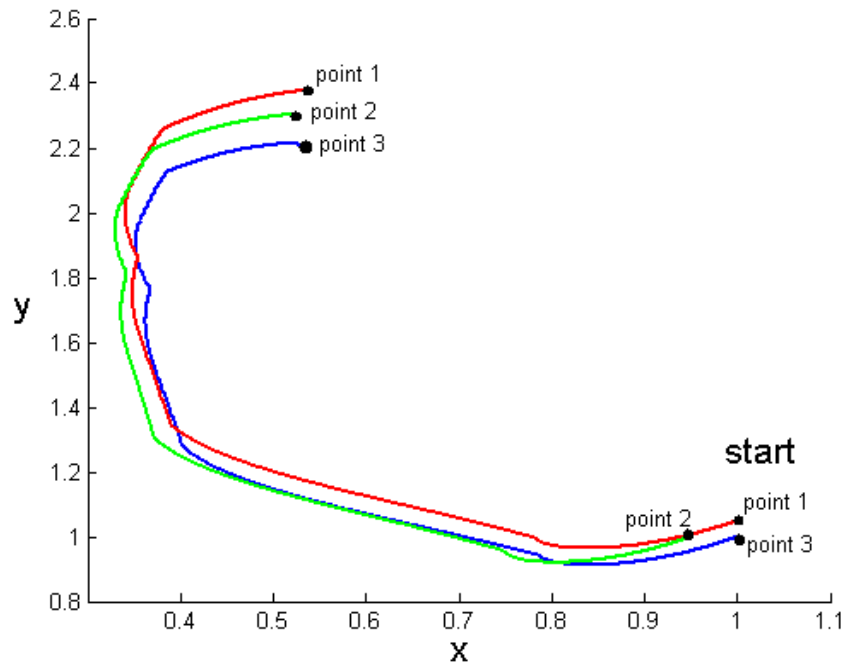


Figure 4.14: Trajectory of the Points on the Object

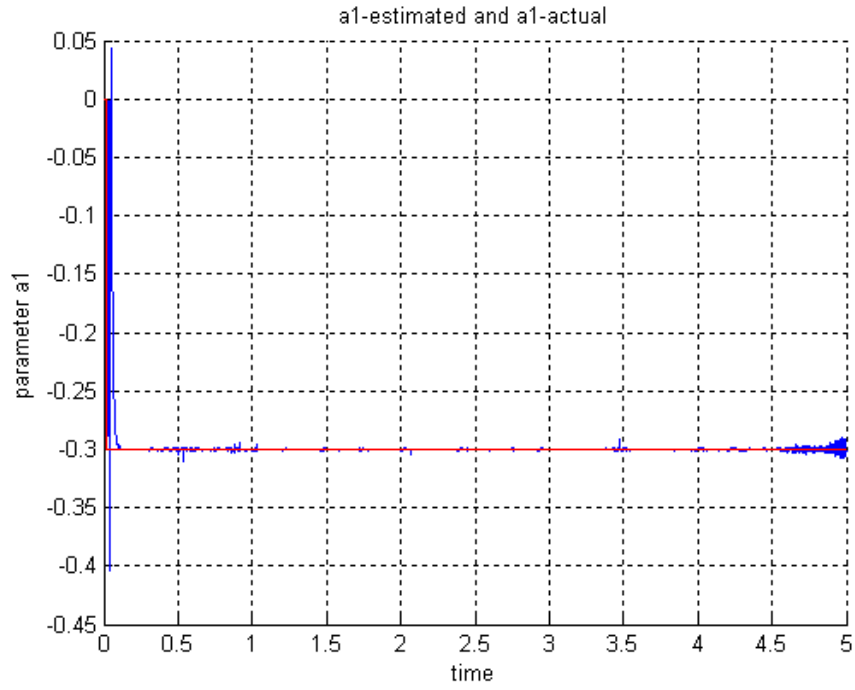


Figure 4.15: a_1 and a_1 estimated

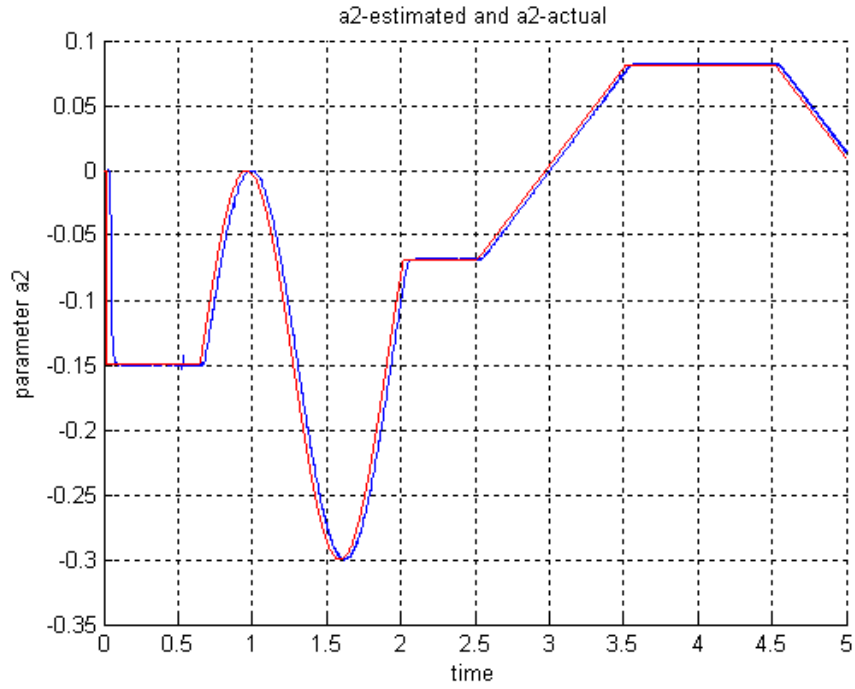


Figure 4.16: a_2 and a_2 estimated

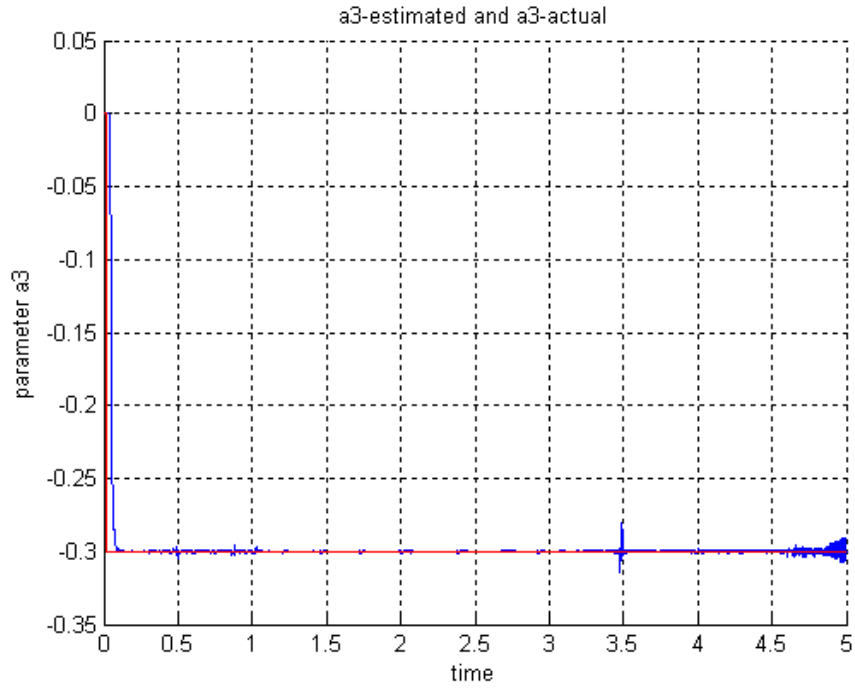


Figure 4.17: a_3 and a_3 estimated

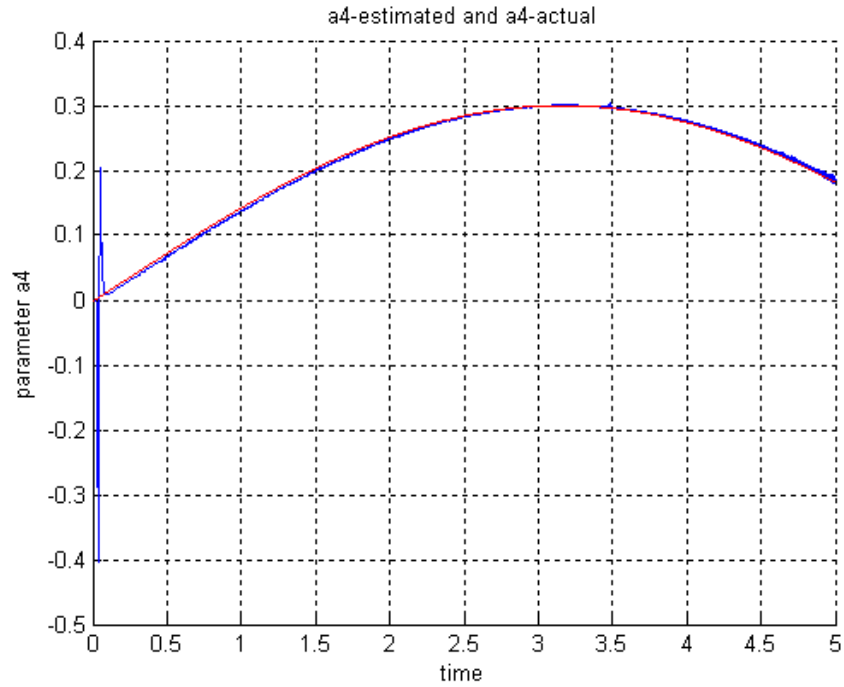


Figure 4.18: a_4 and a_4 estimated

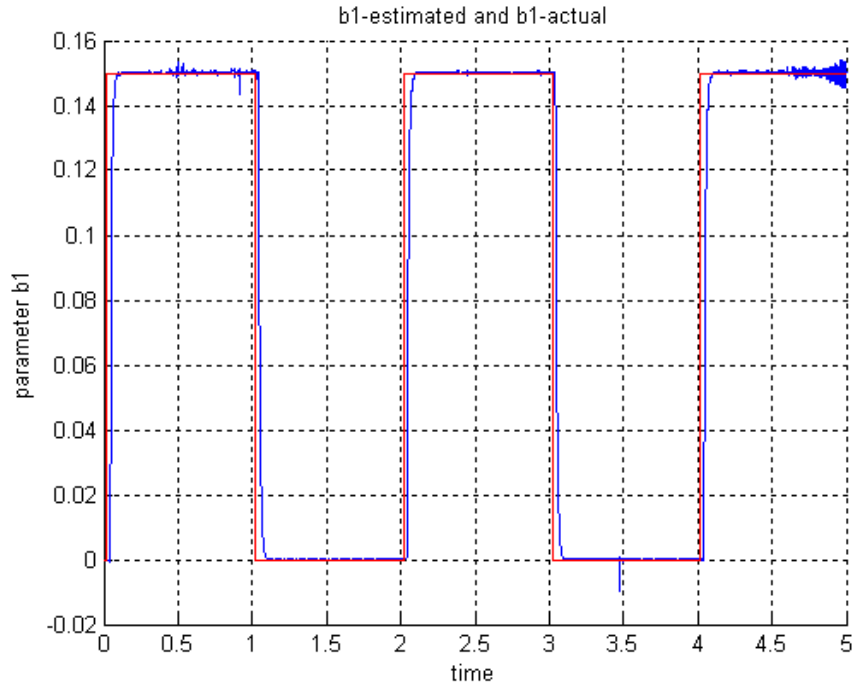


Figure 4.19: b_1 and b_1 estimated

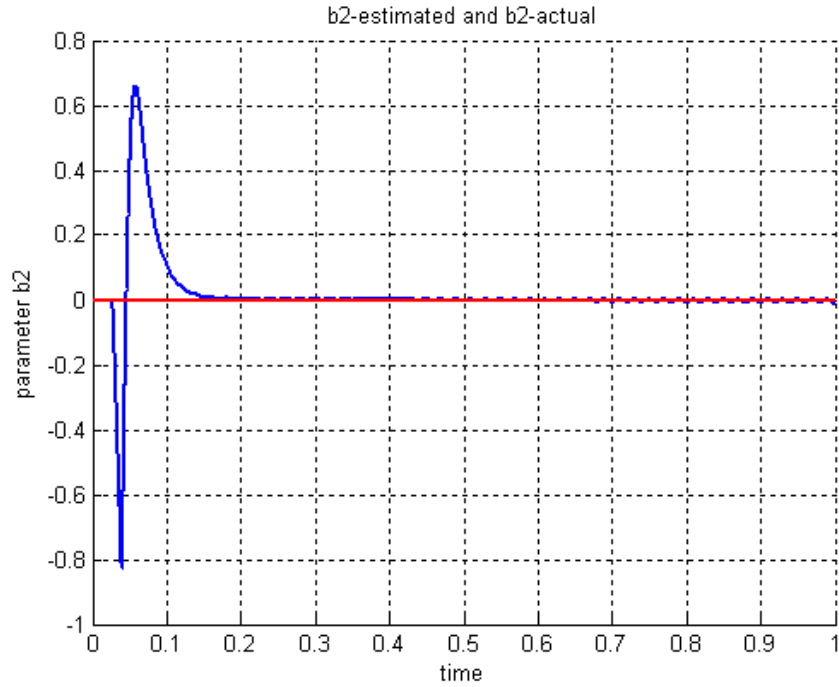


Figure 4.20: b_2 and b_2 estimated

The problem considered in this section is the affine motion of feature points in the image plane (which may be the projection of rigid motion in 3D onto the image plane). Fig 4.14 demonstrates what is meant by affine motion of points in 2D. As seen in the figure, the distances between point pairs do not stay constant, as would be the case in a rigid motion. The figures 4.15 to 4.20 prove that fast and accurate estimation of affine motion parameters is also achieved, with the same problems and possible reasons discussed for rigid motion also being valid for this case.

Chapter 5

Experimental Results

5.1 Experimental Setup - Vision System

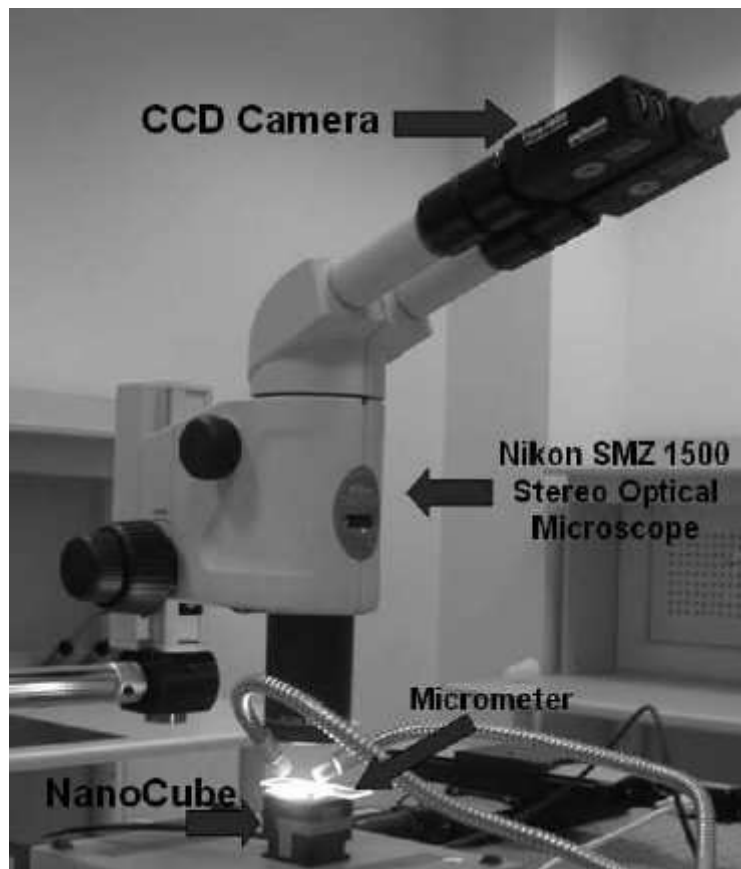


Figure 5.1: Experimental Setup

The algorithms proposed in this work are tested in the setup above. The setup is composed of four main parts:

- The object in motion \rightarrow A NanoCube
- The microscope
- CCD Camera
- A PC where image processing and calculations are done

The Nikon SMZ 1500 Stereo Optical Microscope and a firewire CCD camera is used to acquire fast and high quality images of the NanoCube. The CCD camera is capable of getting 30 frames per second.

The object in motion is the PI (Physik Instrumente) NanoCube shown in Fig. 5.2, which is a multi-axis (XYZ) Piezo-NanoPositioning system.

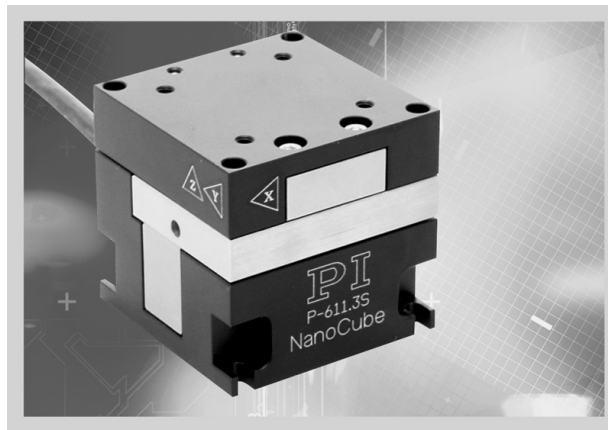


Figure 5.2: PI NanoCube

The closed loop NanoCube provides high accuracy $100 \times 100 \times 100 \mu m$ XYZ positioning. To drive the nanocube with desired velocities, dSPACE 1103 board on a separate PC is used.

The image processing part is done using OpenCV library under Visual C++. The OpenCV implements a wide variety of tools for image interpretation. It is mostly a high level library implementing algorithms for calibration techniques, feature detection and tracking, shape analysis, motion analysis, 3D reconstruction, object

segmentation and recognition. The essential feature of the library is performance. More than that, the OpenCV Library is a way of establishing an open source vision community to apply computer vision in the PC environment.

To conduct the experiments, various motion profiles are applied to the NanoCube, and gathered images are processed to extract a feature point which will be used in identifying the motion parameters. As the feature point, a micrometer is placed on top of the NanoCube, and the tip of the markings on the micrometer is traced. The code consists of three major parts:

- Image capture and enhancement
- Extraction of feature points
- Sliding Mode Observers algorithm

Following figures illustrate the motion that is traced in this experiment:

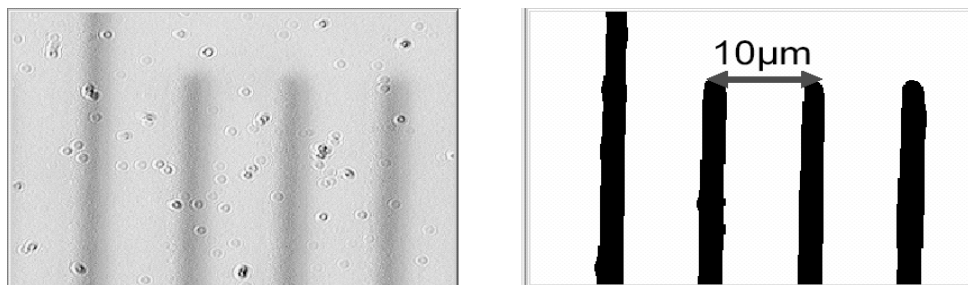


Figure 5.3: Micrometer

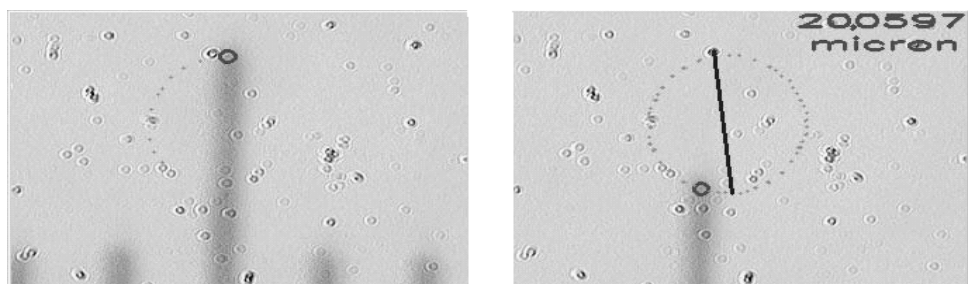


Figure 5.4: Rotational motion of micrometer

5.1.1 The Code Briefly

```
void get_image()
{
image_timer = GetCPUCount( lo, hi );
theCamera1.CaptureImage();
theCamera1.getDIB(buf1);
cvSetImageData(img1, buf1, img1->widthStep);
cvFlip(img1,img1,0);
cvCvtColor(img1, gray1, CV_RGB2GRAY);
cvSmooth( gray1, gray1, CV_GAUSSIAN, 31, 31, 0 );
black_counter=0;
x_1=0;
y_1=0;
for (int i2=0;i2<image_height*0.95;i2++)
{ for(int j2=0;j2<image_width*0.95;j2++)
{   colorGray1 = (unsigned char)gray1->imageData[i2*gray1->widthStep + j2];
if (colorGray1<200)
{
x_1=x_1+j2;  y_1=y_1+i2; black_counter++ ;
}}
x_1=(x_1/black_counter)-center_x;
y_1=(y_1/black_counter)-center_y;
Counter1++;
while (GetCPUCount( lo, hi )<image_timer+0.03333*SLK)
{}}
```

Above is the part where image gathering, enhancement and processing is carried out. The frame which is taken is smoothed by a Gaussian Filter and then the tip point of the micrometer is extracted by comparing pixel's gray level to a threshold value. The end of the code fragment is responsible for synchronizing the image processing time with the desired 30 frames per second camera speed, by suspending the program until the desired time for one frame is consumed when necessary.

The following is the part responsible for sliding mode observer:

```
void controller()
{
sigma1=x_1_int-x1_est;
sigma2=y_1_int-y1_est;
u1_prev=u1;
```

```

u2_prev=u2;
u1=(reach_time+v_max)*((sigma1>0)-0.5)*2;
u2=(reach_time+v_max)*((sigma2>0)-0.5)*2;
x1_est=x1_est+step_time*(u1+u1_prev)/2;
y1_est=y1_est+step_time*(u2+u2_prev)/2;
u_eq_1=(step_time*a_c*(u1+u1_prev)+u_eq_1
*(2-step_time*a_c))/(2+step_time*a_c);
u_eq_2=(step_time*a_c*(u2+u2_prev)+u_eq_2
*(2-step_time*a_c))/(2+step_time*a_c);}

```

First σ values for both x_1 and y_1 (states) are computed. Then the controller given in ... is employed digitally, and estimated state values are computed. Finally a low pass filter is implemented to get equivalent control from the discontinuous control signal u . Between two consecutive frames, the well known interpolater $G(s) = \frac{w_n^2}{s^2+2\zeta w_n s+w_n^2}$ with $w_n = 50$ and $\zeta = 1$ is used, which creates smooth values of states.

The solver used for pure rotational motion is given as an example:

```

void solver()
{
w_prev=w;
w=(u_eq_1*y_1_int-u_eq_2*x_1_int)/(pow(x_1_int,2)+pow(y_1_int,2));
w_filt  = (step_time*a_p*(w+w_prev)+w_filt*
(2-step_time*a_p))/(2+step_time*a_p);
}

```

Here ω value is computed using (4.19). The matrix inversion is carried out using pseudoinverse. The results are filtered.

So the overall code can be summarized as follows:

```

int main(int argc, char** argv)
{
get_image();
for(Counter3=0;Counter3<NumOfInterpolations;Counter3++)
{
interpolater();
controller();
solver();
}
return 0;
}

```


5.1.2 Results for Translational Motion

In this experiment the NanoCube moves along x -axis with $6 \mu\text{m}/\text{s}$ for a while and stops, and then turns back to opposite direction again with a $6 \mu\text{m}/\text{s}$. Since there is no motion along y -axis, the estimated linear velocity along y -axis is around zero. Furthermore, since there is no rotational motion, the estimated angular velocity is again around zero. The following figures indicate above results.

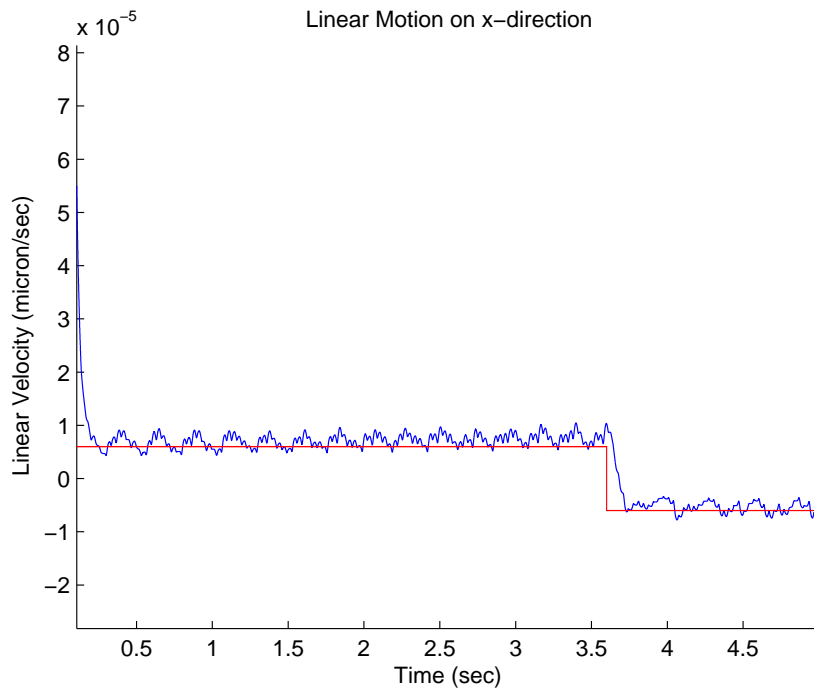


Figure 5.5: Linear Motion along x-axis

As seen in the Figures 5.5-5.7, the preliminary experimental results agree with the simulation results. The motion parameters are computed fast and accurately. As noted in Fig. 5.5, the estimated parameter value catches the true value as soon as the sliding manifold is reached. The experimental results seem less smooth than the simulation results, which is due to both noise in the images taken and low frame rates compared to the simulated camera.

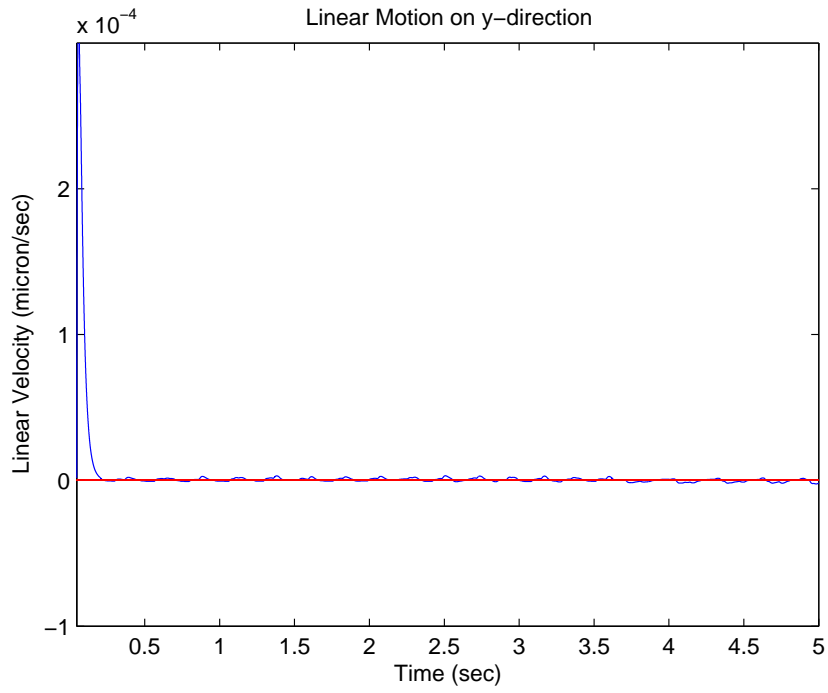


Figure 5.6: Linear Motion along y-axis

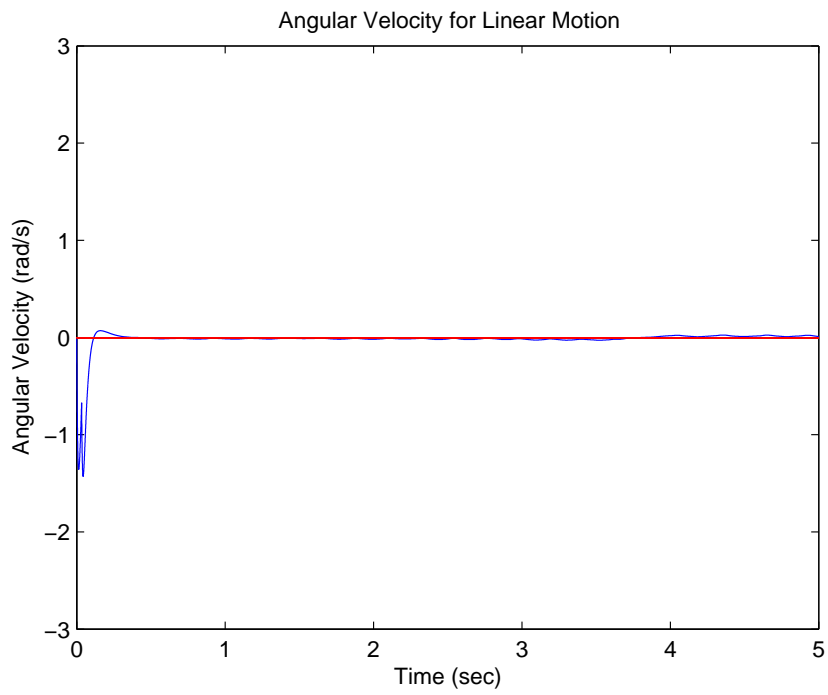


Figure 5.7: Rotational Velocity

5.1.3 Results for Rotational Motion

In this case NanoCube undergoes a pure rotational motion with a constant angular velocity, $\pi \text{ rad/s}$ in counterclockwise. The radius of the circular motion is $10 \mu\text{m}$. Actual and estimated angular velocities are superimposed in the following figure. Although, the reference motion is pure rotation, the angular velocity becomes very large for small time interval at very beginning. Then it converges to reference value very quickly. At the same time, the estimated trajectory seems to undergo linear motion (See Fig. 5.9). The reason is that the initial points, which are given as initial conditions for estimation, are different from the actual points on the circle. Therefore estimated points show some linear behavior until the dynamic of the system reaches the sliding manifold. When the system is on the sliding manifold, estimated values converge to the actual values and hence the motion becomes circular. The following figures indicate above results.

Fig. 5.10 shows another rotational motion with constant angular velocity, $\pi/5 \text{ rad/s}$ in counterclockwise.

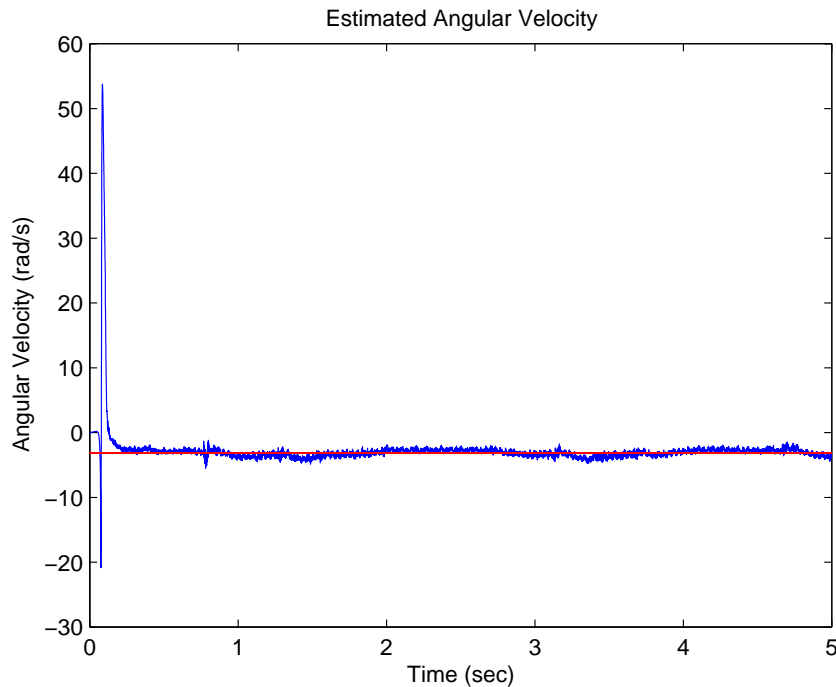


Figure 5.8: Rotational Velocity

5.2 Conclusions

The problem of estimating motion from a sequence of images has been investigated and a novel identification procedure for the estimation of both constant and time varying motion parameters is developed in this work. The proposed algorithm is substantiated by computer simulations and real experiments.

It is shown that most of the dynamics encountered in machine vision applications can be recast as dynamical systems which are linear in terms of parameters to be estimated. Then an appropriate sliding observer whose output follows the output of the original dynamics asymptotically is constructed. As the given results suggest, the SMO algorithm is a promising method that provides fast and accurate estimation of motion parameters. The ideas developed in this work are proven to be useful in both rigid and affine type of motion identification where the parameters can be constant or time-varying.

The convergence rate of the parameters to their true values is a key issue in system identification, where a fast convergence rate implies better performance when the parameters are time varying. Regarding this property, SMO algorithm seems to be more reliable when compared to other statistical based methods in the literature, where convergence for time varying cases is usually a hard issue.

The work carried out in this thesis can be thought of as a preliminary step in computer vision applications to merge ideas from control theory with image analysis.

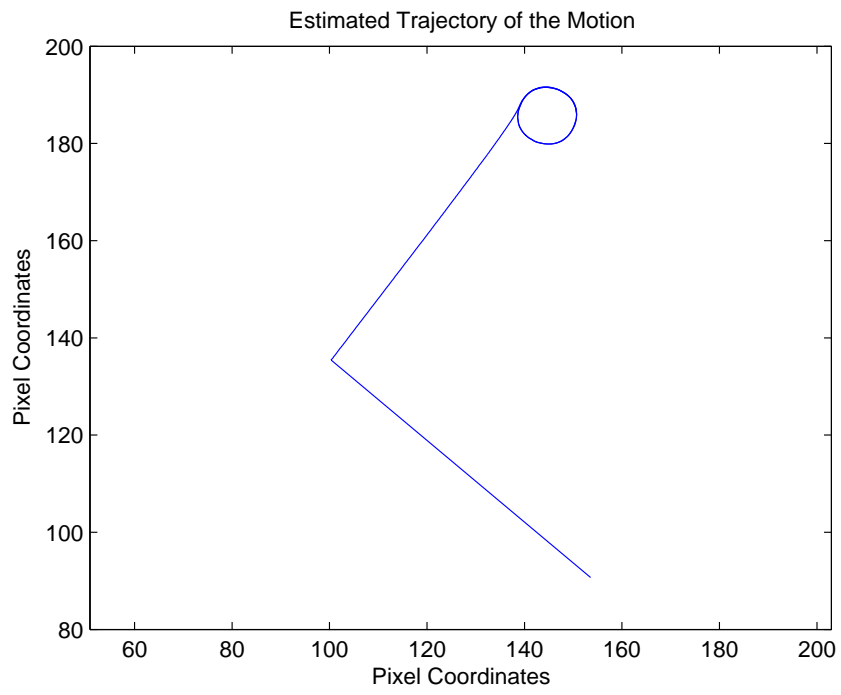


Figure 5.9: Estimated Trajectory

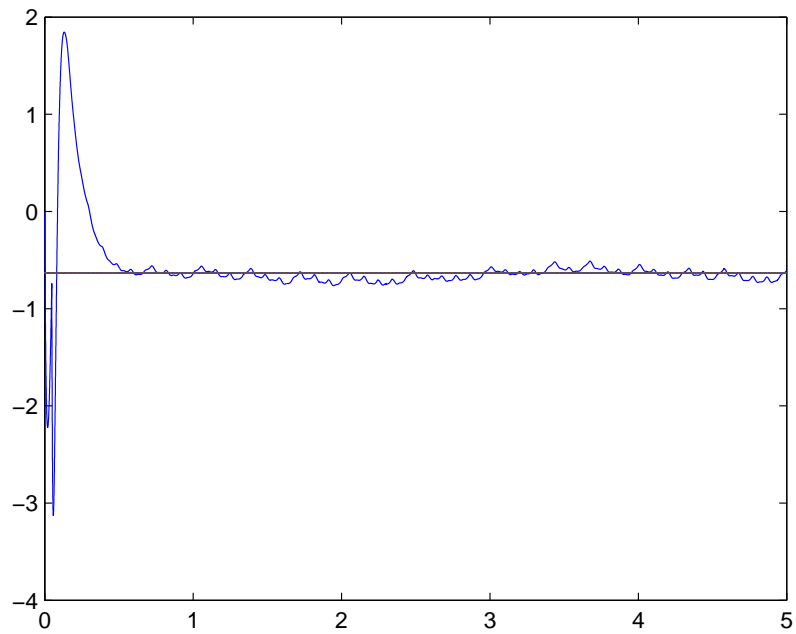


Figure 5.10: Angular Velocity

Appendix A

C++ Codes For the Experiment

```
#ifdef _CH_
#pragma package <opencv>
#endif

#ifdef _EiC
#include "cv.h"
#include "highgui.h"
#endif

#include <windows.h>
#include <1394Camera.h>
#include <iostream>
#include <fstream>
#include <math.h>
#include <time.h>
#include <stdlib.h>

using namespace std;

int image_width = 320;
int image_height = 240;
char wndname1[] = "Actual Image";

int black_counter=0;

double SLK=2260000000;

int Counter1=0;
int Counter2=0;
int Counter3=0;
int Counter5=0;
int Counter4=0;

double R=45/2.0;

float image_timer=0;
```

```

float frame_counter=0;
IplImage *img1 = 0, *gray1 = 0, *edge = 0;

IplImage * eigImage = cvCreateImage( cvSize(image_width,image_height), IPL_DEPTH_8U, 3);
IplImage * tempImage = cvCreateImage( cvSize(image_width,image_height), IPL_DEPTH_8U, 3);
int cornerCount=1;

CvPoint2D32f corners[1];
int edge_tresh = 20;

int temp=0;

float w_filt_old_old=0;
float w_filt_old=w_filt_old_old;

int colorGray1=0;

int minGray1=255;

ofstream file;
ofstream file1;
ofstream file2;
ofstream file3;
ofstream file4;
ofstream file_time;

const int NumOfInt =200;
const int NumOfFrames=1;

float step_time=0.03333/NumOfInt;
float mokkar=0;

C1394Camera theCamera1;
unsigned char buf1[320*240*3];

float xpoints[1000];
float ypoints[1000];

float w_fil [NumOfInt*NumOfFrames*500];
float dummy1 [NumOfInt*NumOfFrames*500];
float dummy2 [NumOfInt*NumOfFrames*500];
float dummy3 [NumOfInt*NumOfFrames*500];
float dummy4 [NumOfInt*NumOfFrames*500];

float wf=50;
float zeta=1;
float delta_x=0;
float delta_y=0;

```

```

float delta_x1=0;
float delta_y1=0;
float delta_x1_old=0;
float delta_y1_old=0;

double bias=0.001;
double center_x = 208;
double center_y = 105;

float u1,u2,u3,u4=0;
float u1_prev,u2_prev,u3_prev,u4_prev=0;
float x1_est_filt,y1_est_filt,x1_est_prev,y1_est_prev=0;
float u_eq_1,u_eq_2,u_eq_3,u_eq_4=0;
float u_eq_1_prev,u_eq_2_prev=0;
float x_1=0;
float y_1=0;
float x1temp=0;
float x_1_old=0;
float y_1_old=0;
float x_1_old_old=0;
float y_1_old_old=0;

float x_1_int=0;
float y_1_int=0;
float x_1_int_old=0;
float y_1_int_old=0;
float x_1_int_old_old=0;
float y_1_int_old_old=0;
float x1_est=0;
float y1_est=0;

static v_max=10;
static reach_time=0;
double tao_p=0.1;
double tao_c=0.001;
double a_p=1/tao_p;
double a_c=1/tao_c;
float sigma1,sigma2,sigma3,sigma4=0;
float sigma1_old,sigma2_old,sigma3_old,sigma4_old=0;
float w,b_1,b_2=0;
float w_prev,b_1_prev,b_2_prev=0;
float w_filt,b_1_filt,b_2_filt;

float der1=0;
float der2=0;
float der3=0;
float der4=0;
float K=0.005;

```



```

float D=150;

double t=0;
unsigned int hi = 0, lo = 0;

__int64 GetCPUCount ( unsigned int loword, unsigned int hiword )
{
    _asm
    {
        _emit 0x0f // insert rtdsc opcode
        _emit 0x31
        mov hiword , edx
        mov loword , eax
    }
    return ( (__int64) hiword << 32 ) + loword;
}

void controller()

{
    sigma1=x_1_int-x1_est;
    sigma2=y_1_int-y1_est;

    der1=(sigma1-sigma1_old)/step_time;
    der2=(sigma2-sigma2_old)/step_time;

    u1=(reach_time+v_max)*((sigma1>0)-0.5)*2;
    u2=(reach_time+v_max)*((sigma2>0)-0.5)*2;

    x1_est=x1_est+step_time*(u1+u1_prev)/2;
    y1_est=y1_est+step_time*(u2+u2_prev)/2;

    u_eq_1=u1;
    u_eq_2=u2;

    u1_prev=u1;
    u2_prev=u2;

    u_eq_1=(step_time*a_c*(u1+u1_prev)+u_eq_1
    *(2-step_time*a_c))/(2+step_time*a_c);
    u_eq_2=(step_time*a_c*(u2+u2_prev)+u_eq_2
    *(2-step_time*a_c))/(2+step_time*a_c);

    sigma1_old=sigma1;
    sigma2_old=sigma2;
}

```

```

}

void solver()
{
w_prev=w;
w=(u_eq_1*y_1_int-u_eq_2*x_1_int)/(pow(x_1_int,2)+pow(y_1_int,2));
w_filt  = (step_time*a_p*(w+w_prev)+w_filt*
(2-step_time*a_p))/(2+step_time*a_p);
}

void cam_init()
{
if (theCamera1.CheckLink())
printf("No cams found");

theCamera1.SelectCamera(2);
theCamera1.InitCamera();
theCamera1.SetVideoFormat(0);
theCamera1.SetVideoMode(1);
theCamera1.SetVideoFrameRate(4);
theCamera1.SetWhiteBalance(100,200);
theCamera1.SetSharpness(10);
theCamera1.StartImageAcquisition();

}

void get_image()
{
image_timer = GetCPUCount( lo, hi );

theCamera1.CaptureImage();
theCamera1.getDIB(buf1);
cvSetImageData(img1, buf1, img1->widthStep);
cvFlip(img1,img1,0);
cvCvtColor(img1, gray1, CV_RGB2GRAY);
cvSmooth( gray1, gray1, CV_GAUSSIAN, 31, 31, 0 );

black_counter=0;
x_1=0;
y_1=0;

for (int i2=0;i2<image_height*0.95;i2++)

{

```

```

for(int j2=0;j2<image_width*0.95;j2++)
{
colorGray1 = (unsigned char)gray1->imageData[i2*gray1->widthStep + j2];

if (colorGray1<200)
{
x_1=x_1+j2;
y_1=y_1+i2;
black_counter++ ;
}
}

x_1=(x_1/black_counter)-center_x;
y_1=(y_1/black_counter)-center_y;

cvCircle(img1, cvPoint(x_1+center_x, y_1+center_y), 5, CV_RGB(255,0,0), 1);
cvShowImage(wndname1, img1);
Counter1++;
}

void interpolater()
{

x_1_int=(step_time*step_time*wf*wf*x_1+x_1_int_old*(2*step_time*zeta*wf+2)-x_1_in
y_1_int=(step_time*step_time*wf*wf*y_1+y_1_int_old*(2*step_time*zeta*wf+2)-y_1_in

x_1_int_old_old=x_1_int_old;
x_1_int_old=x_1_int;

y_1_int_old_old=y_1_int_old;
y_1_int_old=y_1_int;/**/

    controller();
solver();
}

void initialize()
{

theCamera1.CaptureImage();
theCamera1.getDIB(buf1);
cvSetImageData(img1, buf1, img1->widthStep);
cvCvtColor(img1, gray1, CV_RGB2GRAY);

```

```

cvSmooth( gray1, gray1, CV_GAUSSIAN, 45, 45, 0 );

black_counter=0;

for (int i2=0;i2<image_height*0.95;i2++)

{
for(int j2=0;j2<image_width*0.95;j2++)
{
colorGray1 = (unsigned char)gray1->imageData[i2*gray1->widthStep + j2];

if (colorGray1>190)
{
xpoints[Counter1]=xpoints[Counter1]+j2;
ypoints[Counter1]=ypoints[Counter1]+i2;
black_counter++ ;
}
}

}

xpoints[Counter1]=(xpoints[Counter1]/black_counter)-center_x;
ypoints[Counter1]=(ypoints[Counter1]/black_counter)-center_y;

cvCircle(img1, cvPoint(xpoints[Counter1], ypoints[Counter1]), 5, CV_RGB(255,0,0),

x1_est=xpoints[Counter1]+bias;
y1_est=ypoints[Counter1]+bias;
x_1_int_old_old=xpoints[Counter1];
y_1_int_old_old=ypoints[Counter1];
x_1_int_old=xpoints[Counter1];
y_1_int_old=ypoints[Counter1];
x_1_int=xpoints[Counter1];
y_1_int=ypoints[Counter1];

interpolator();
Counter1++;
}

int main(int argc, char** argv)
{

cvNamedWindow(wndname1, 1);
theCamera1.AcquireImage();
cam_init();

img1 = cvCreateImage(cvSize(image_width,image_height),IPL_DEPTH_8U, 3);
gray1 = cvCreateImage(cvSize(image_width,image_height), IPL_DEPTH_8U, 1);

```

```

edge = cvCreateImage(cvSize(image_width,image_height), IPL_DEPTH_8U, 1);

initialize();
initialize();
t = GetCPUCount( lo, hi );

while (GetCPUCount( lo, hi )<t+10*SLK)
{

get_image();
if (temp<2)
{
x1_est=x_1+bias;
y1_est=y_1+bias;
x_1_int=x_1;
y_1_int=y_1;
temp++;
}

for(Counter3=0;Counter3<NumOfInt;Counter3++)
{
interpolater();

}

step_time=(GetCPUCount(lo,hi)-image_timer)/(SLK*NumOfInt);
}

return 0;
}

```

Bibliography

- [1] Ljung, L. (1999). *System Identification*. Prentice-Hall, Upper Saddle River, N.J.
- [2] Porch, H.R. and Mo, S.H. (1990). Toolkit for plant system identification. *Multifrequency Testing for System Identification, IEE Colloquium on*, page(s):6/1-6/6
- [3] Calway, A. (2005) Recursive Estimation of 3D Motion and Surface Structure from Local Affine Flow Parameters. *IEEE Transactions on PAMI*, Volume 27, No. 4, pages 562-574
- [4] Horn, B.K.P. (1986). *Robot Vision*. The MIT Press.
- [5] Adiv, G. (1985). Determining three dimensional motion and structure from optical flow generated by several objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 7, pages 384-401.
- [6] Aloimonos, J. Brown, C.M. (1986). Preception of structure from motion, I: Optical flow v.s. discrete displacements. *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*.
- [7] Liu, Y. and Huang, T. S. (1986). Estimation of rigid body motion using straight line correspondence. *Proc. Workshop on Motion: Representation and Analysis, IEEE Computer Society Press*.
- [8] Liu, Y. and Huang, T. S. (1988). A linear algorithm for motion estimation using straight line correspondences. *Computer Vision, Graphics, and Image Process*, Volume 44.

- [9] Faugeras, O.D., Lustman, F., Toscani, G. (1987). Motion and structure from point and line matches. *INRIA, BP105*, Domain de Volucen, rocquencourt, 78153 Le Chesnay, France.
- [10] Jategaonkar, R. and Thielecke, F. (2002). ESTIMA an integrated software tool for nonlinear parameter estimation. *Aerospace Science and Technology*, Volume 6, Issue 8, pages 565-578
- [11] Shah, M. (1997) *Fundamentals of Computer Vision*. (electronic edition).
- [12] Trucco, E., Verri, A. (1998) *Introductory Techniques for 3-D Computer Vision*. Prentice-Hall, Upper Saddle River, N.J.
- [13] Utkin, V. I. (1977). Variable Structure Systems with Sliding-Modes. *IEEE Transactions on Automatic Control*, Volume 22, No. 2, pages 212-222
- [14] Utkin, V., Guldner, J. and Shi, J. (1999). *Sliding Modes in Electromechanical Systems*, Taylor and Francis, London.
- [15] Edwards, C. and Spurgeon, S. (1998). *Sliding Mode Control*, Taylor and Francis, Bristol, PA.
- [16] Gawthrop, P.J. and Wang, L. (2005) Data compression for estimation of the physical parameters of stable and unstable linear systems. *Automatica*, Volume 41, Issue 8, pages 1313-1321
- [17] Shtessel, Y.B. and Poznyak, A.S. (2004). Parameter Identification of Linear Time Varying Systems via Traditional and High Order Sliding Modes. *Conference on Variable Structure Systems*