Motion Estimation of Planar Curves and Their Alignment Using Visual Servoing

by

Ahmet Yasin Yazıcıoğlu

Submitted to the Graduate School of Sabancı University
in partial fulfillment of the requirements for the degree of
Master of Science

Sabanci University

August, 2009

to my beloved family

Motion Estimation of Planar Curves and Their Alignment Using Visual Servoing

Ahmet Yasin Yazıcıoğlu
Mechatronics Engineering, Master's Thesis, 2009
Thesis Supervisor: Assoc. Prof. Dr. Mustafa Ünel

Keywords: Implicit Polynomials, Algebraic Curve Fitting, Robotics, Control, Computer Vision, Motion Estimation, Visual Servoing.

Abstract

Motion estimation and vision based control have been steadily improving research areas recently. Visual motion estimation is the determination of underlying motion parameters by using image data. Visual servoing on the other hand refers to the closed loop control of robotic systems using vision. Solving these problems with objects that have simple geometric features, such as points and lines is rather easy. However, these problems may imply certain challenges when we deal with curved objects that lack such simple features.

This thesis proposes novel vision based estimation and control techniques that use object boundary information. Object boundaries are represented by planar algebraic curves. Decomposition of algebraic curves are used to extract features for motion estimation and visual servoing. Motion estimation algorithm uses the parameters of line factors resulting from the decomposition of the curve whereas visual servoing method employs the intersections of lines. Motion estimation algorithm is verified with several simulations and

experiments. Visual servoing algorithm developed for the arbitrary alignment of a planar object is tested both with simulations on a 6 DOF Puma 560 robot and experiments on a 2 DOF SCARA robot. Results are quite promising.

Düzlemsel Eğrilerin Hareket Kestirimi
ve Görme Tabanlı Kontrol Kullanılarak Hizalanması

Ahmet Yasin Yazıcıoğlu
Mekatronik Master Tezi, 2009
Tez Danışmanı: Doç. Dr. Mustafa Ünel

Keywords: Örtük Polinomlar, Cebirsel Eğri Oturtma, Robotik, Kontrol, Bilgisayar Görmesi, Hareket Kestirimi, Görme Tabanlı Kontrol.

Özet

Hareket kestirimi ve görme tabanlı kontrol yakın zamanda hızla ilerleme gösteren araştırma alanlarıdır. Görsel hareket kestirimi hareket paremetrelerinin görsel veri kullanılarak belirlenmesidir. Öte yandan, görme tabanlı kontrol ise görüntünün geri beslemeli robot kontrolünde kullanımını ifade eder. Noktalar, doğrular gibi basit geometrik özniteliklere sahip nesnelerle bu problemlerin çözümü daha kolaydır. Ancak, bu tür basit özniteliklerden yoksun eğrisel nesnelerle uğraşırken bu problemler belirli zorluklar ortaya çıkarmaktadır.

Bu tezde nesne sınır verisini kullanan özgün yöntemler önerilmektedir. Nesne sınırları düzlemsel cebirsel eğrilerle modellenmektedir. Hareket kestirimi ve görme tabanlı kontrolde kullanılan öznitelikleri elde etmek için cebirsel eğrilerin ayrıştırılmasından yararlanılmaktadır. Hareket kestirimi algoritması ayrıştırma sonucunda elde edilen doğru çarpanlarının paramaterelerini kullanırken, görme tabanlı kontrol yöntemi ise doğruların kesişim noktalarından yararlanmaktadır. Hareket kestirimi algoritması benzetim ve deneylerle desteklenmiştir. Düzlemsel nesnelerin hizalanması için geliştirilen görme tabanlı

kontrol algoritması ise hem 6 serbestlik dereceli Puma 560 robotuyla gerçekleştirilen benzetimler hem de 2 serbestlik dereceli SCARA robot üzerinde gerçekleştirilen deneylerle test edilmiştir. Elde edilen sonuçlar oldukça başarılıdır.

Acknowledgements

I would like to express my sincere gratitude to Assoc. Prof. Dr. Mustafa Ünel for his priceless academic guidance and enlightening me with his brilliant ideas. He conveyed a spirit of enthusiasm for research and scholarship, and an excitement in regard to teaching. I am pleased to state my appreciation to him for spending so much effort on my improvement and for his endless support. Without him, it would not have been possible to achieve my goals for my Masters and my academic future.

I am grateful to Assist. Prof. Dr. Kemalettin Erbatur, Assist. Prof. Dr. Hakan Erdoan, Assist. Prof. Dr. Selim Balcısoy and Assist. Prof. Dr. Gözde Ünal for participating in my thesis jury and for their constructive comments. Also I would like to thank Assist. Prof. Dr. Kemalettin Erbatur for giving me the possibility to conduct experiments on SCARA robot, Berk Çallı for showing me how to use it and Erol Özgür for sharing some of his C++ codes which have been quite helpful. In addition, I am glad to acknowledge TUBİTAK BİDEB for their financial support throughout my Masters.

Furthermore I would like to thank all my friends for so many good memories we have together. I am happy to acknowledge Ahmetcan Erdoğan, Berk Çallı, Kaan Öner, Evrim Taşkıran, Özer Koca, Hümay Esin, Utku Seven, Murat Ergun and all my other friends...

Finally, my greatest thanks go to my family, whose endless love, trust and support can not be compared to anything else. I feel extremely lucky to have amazing parents who are always there whenever I need them and an excellent brother who has been my best friend, making me feel that I am never alone since the moment he was born. No matter how far I am they will always be with me in my heart.

Contents

List of Figures

Chapter 1

Introduction

Algebraic curves and surfaces have been used in various branches of engineering for a long time, but in the past two decades, they have proven very useful in many model-based applications. Various algebraic and geometric invariants obtained from implicit models of curves and surfaces have been studied rather extensively in computer vision, especially for single computation pose estimation, shape tracking, 3D surface estimation from multiple images and efficient geometric indexing of large pictorial databases [1]- [11].

Algebraic curves/surfaces have great modelling power for complicated shapes and can represent acquired data very well. One problem about algebraic models is the possible sensitivity of their their coefficients to small changes in the data. However, stability in fitting algorithms have been studied a lot and methods with significantly increased stability exist in the literature [39],[40]. Once the algebraic model for an object boundary is properly obtained, it provides certain advantages [41]. As algebraic models provide simple but powerful model with low computational cost it is easier to work

with them in many applications that require real time performance. Furthermore as being model based, they provide robustness against noise and partial occlusions. In addition, unlike many deformable model based methods, algebraic curve/surface fitting algorithms does not have initialization problems.

Motion estimation is one of the important problems in computer vision. Basically, it is the process of determining motion vectors that describe the transformation from one 2D image to another (generally the consecutive frame). It is used in many visual tracking algorithms such as optical flow [12] or Kalman filter [13, 14]. Also it plays a key role in video coding as it realizes high compression rates, achieved through removal of temporal redundancies between successive images. There are various approaches in this field such as block matching [15, 16, 17, 18] which are based on the regional matchings or other methods based on the matching of image features such as lines, points, etc [19].

In this thesis we are particularly interested in the estimation of motion parameters of a planar algebraic curve which is obtained from the boundary data of a target object. In estimating the motion parameters, one is faced with the problem of modeling planar curves in motion. There has been a steadily growing literature in robotics on the problem of line correspondence for line features moving in $\Re^3$,(see [26]- [30]). For some other older references in the literature on the dynamics of curves, see [31]- [33]. In order to describe dynamics of planar algebraic curves we use a polynomial decomposition method [4, 6] to express curve models as a unique sum of product of line factors. It is shown in [24, 25] that a plane polynomial curve under-

going time invariant Euclidean or affine motion implies Riccati equations in terms of parameters of these line factors. In this thesis we extend this to the time varying case and show that same Riccati equations can be obtained. Using this result, a motion estimation technique which uses line parameters as measurement signals is proposed.

Vision based control of robotic systems has been a steadily improving research area recently. Commercially available cameras provide a cheap and powerful tool for many complex robotic tasks in dynamic environments. Consequently, many researchers from computer vision, robotics and control disciplines have been working on different problems of vision based control. One particular problem in this domain is object alignment. In visual servoing applications, most of the current alignment systems are based on objects with known 3D models such as industrial parts or objects which have good features due to their geometry or texture. Mostly, features which are feasible to extract and track in real time [49] are used in these approaches. Many works are reported in the literature on alignment using points, lines, ellipses, image moments, etc. [50]-[53]. On the contrary, the alignment of smooth free-form planar objects presents a challenge in visually guided alignment tasks since these objects may not provide necessary amount of such features. Instead of using these features, curves can be fitted to these free-form objects [34], [39]. However obtaining features from these curves for visual servoing algorithms is not a trivial task.

We propose to use implicit polynomial representation in aligning planar closed curves by employing calibrated image based visual servoing [50] as a solution for such cases. With the proposed method [54], an implicit polyno-

mial representation of target object boundary is obtained by a curve fitting algorithm. Acquired polynomial is then decomposed as a unique sum of product of line factors [4], [6]. This decomposition is then used to provide features for visual servoing.

1.1 Contribution of the Thesis

- It is shown in [24, 25] that a plane polynomial curve undergoing time invariant Euclidean or affine motion implies Riccati equations in terms of parameters of these line factors. In this thesis we extend this to the time varying motion parameters and show that the same Riccati equations can be obtained.

- A novel motion estimation algorithm for time varying affine motion is proposed. Motion parameters of a planar implicit curve are identified through its line decomposition [4, 6].

- Importance of data normalization in estimation algorithms is investigated. It is shown that data normalization increases the stability of the estimation algorithm.

- A novel visual servoing method [54] is proposed. In this method, real intersection points of complex conjugate lines obtained from decomposition of planar algebraic curves are utilized as visual features.

Chapter 2 presents some background on computer vision techniques that are used. Chapter 3 is on motion estimation of free form planar objects. Visual servoing through the line decomposition of planar algebraic curves is

explained in chapter 4. Chapter 5 presents the experimental results. Finally chapter 6 concludes the thesis with some remarks and future directions.

Chapter 2

Background on Computer Vision Techniques

2.1 Segmentation

Segmentation is one of the important problems of image processing. Basic definition of the problem can be given as follows: "Given an image $I(x, y)$, partition it into meaningful homogeneous regions". With this definition boundary extraction is a subtopic of image segmentation. There are various studies on this field varying from the earlier methods based on gray level intensity discontinuities [43], [42] or thresholding to more advanced algorithms such as histogram based algorithms, model based [59] algorithms or active contours [36, 44, 60, 62, 61] which use partial differential equations. Though the simple gradient based algorithms may result in good results for certain scenarios, in many cases they may be insufficient by outputting edge pixels not only on the boundary of an object but also within the object due to

texture. Thus in the implementation we prefer to use a segmentation which is sufficient for the particular scenario. A brief review for two particular algorithms, Canny Edge detection and level sets are given below.

2.1.1 Canny Edge Detection

Canny edge detection algorithm is one of the most widely used edge detection algorithms based on the gray level intensity discontinuities.It is optimal in a precise, mathematical sense [63]. Its being easy to implement and having low computational cost makes it helpful in simple scenarios. It considers and tries to optimize three criteria : Good detection, good localization and single response constraint. That is to say, it aims to minimize the probability of false edges, tries to detect the edges as close as possible to the true edge and return only one point for each true edge pixel by minimizing the number of local minima around the true edge which occur due to measurement noise.

Canny edge detection algorithm is composed of 3 main steps: Canny enhancer, non-max suppression and hysteresis thresholding. Input to the algorithm is a gray scale image $I(x, y)$. In image enhancement step the input is first convolved with a Gaussian mask which has 0 mean and standard deviation of σ to filter noise.

$$J(x, y) = I(x, y) * G(x, y) \tag{2.1}$$

By computing the image gradients on $J(x, y)$, every pixel is assigned an edge

strength $E_s(x, y)$, and edge normal orientation ($E_o(x, y)$).

$$E_s(x, y) = \sqrt{J_x^2(x, y) + J_y^2(x, y)}$$
$$E_o(x, y) = \arctan \frac{J_x(x, y)}{J_y(x, y)} \tag{2.2}$$

In non-max suppression step edge normal orientation is used to obtain thin edges. For every pixel (x,y) if $E_s(x, y)$ is smaller than any of its neighbors along the direction $E_o(x, y)$, $E_s(x, y)$ is set to 0. Finally a hysteresis thresholding step is applied on E_s. In general, if a single threshold is used its value may significantly affect the output. If a low threshold is chosen, many false edges can be detected due to noise. On the other hand if a high threshold is picked, many true edges may be undetected. Canny algorithm proposes a hysteresis thresholding as an efficient solution to this problem. In hysteresis thresholding a high threshold τ_h, and a low threshold τ_l are used. First an unvisited edge pixel that satisfy $E_s(i, j) > \tau_h$ is detected and starting from that pixel, moving in both directions perpendicular to the edge normal direction, every pixel that satisfy $E_s(p, q) > \tau_l$ is declared as true edge.

Some edge detection results with Canny edge detection algorithm are given in Fig. 2.1. As one can see from these results, as long as the contrast between the background and the object boundary is large compared to the intensity gradients within the object (as in the case of mouse example) Canny edge detector can be used to extract the boundary data of target object. However for more general cases strong intensity changes may be existing within the object due to texture or illumination effects. In such conditions Canny detector may not be sufficient.

Figure 2.1: Input image and output of Canny edge detection for different objects.

2.1.2 Level Sets

Level sets [36] aim to automatically find the contours of objects by using partial differential equations (PDE). Level set approach is based on the following observation: "A planar curve can be considered as the zero-level of a function in 3D". Main idea is to represent a closed evolving curve $C(p,t)$ as the zero level set of an implicit function $\psi(x,y,t) = z$. The level set function $\psi(x,y,t)$ is considered to attain zero on the curve, negative values inside of the curve and positive values outside of the curve.

$$C(p,t) = \{(x,y) : \psi(x,y,t) = 0\} \tag{2.3}$$

$$\psi(C(p,t),t) = 0 \tag{2.4}$$

Consider that level set function $\psi(x,y,t)$ is initialized by the user and the aim is to evolve it in a fashion that $\psi(x,y,t) = 0$ grasps the boundary contours

of objects. If we take the derivative of (2.4)with respect to time:

$$\frac{d}{dt}\psi(C(p,t),t) = \underbrace{\frac{\partial\psi(C(p,t),t)}{\partial C(p,t)}}_{\nabla\psi.N}\underbrace{\frac{\partial C(p,t)}{\partial t}}_{V(x,y)} + \frac{\partial\psi}{\partial t} = V(x,y)(\nabla\psi.N) + \psi_t = 0 \tag{2.5}$$

To do this we need to define a speed function $V(x,y)$ that tells how to move each point on the curve perpendicular to the tangent (in direction N) at that point (note that moving along the tangential direction would have no effect on the evolution of curve). Plugging in the outward normal direction, $N = \frac{\nabla\psi}{||\nabla\psi||}$ to (2.5) we get

$$\begin{aligned} \psi_t + V(x,y)(\nabla\psi.\frac{\nabla\psi}{||\nabla\psi||}) &= 0 \\ \psi_t &= -V(x,y)||\nabla\psi|| \end{aligned} \tag{2.6}$$

Equation above gives the main evolution principle in level set method. All needed to be defined is the velocity term $V(x,y)$. For this purpose properties such as the curvature of of the contour and image gradient magnitudes may be used. For example following equation may be used in the implementation of level set algorithm.

$$\frac{\partial\psi}{\partial t} = -\underbrace{k(x,y)(\beta\kappa+\alpha)}_{V(x,y)}||\nabla\psi|| \tag{2.7}$$

where β, α are weighting parameters, κ is the curvature function and $k(x,y)$ is the speed term which is taken as:

$$k(x,y) = \frac{1}{1+||\nabla((G_\sigma * I)(x,y))||^n}, \quad n \geq 1 \tag{2.8}$$

G_σ given in (2.8) is a Gaussian with standard deviation σ and convolution of image $I(x,y)$ with it provides smoothing and noise filtering. Choice of

$V(x, y)$ in equation above includes both the curvature and an image gradient based term. Note that as the image gradient increases or curvature decreases evolution at that point slows down. Consequently two tasks are achieved with this speed function, both the smoothness of the curve is achieved and the evolution of the curve is forced to slow down or stop at points where magnitude of image gradients are large (such as edges). This algorithm is implemented in Matlab, and its outputs for different objects are presented in Fig. 2.2.

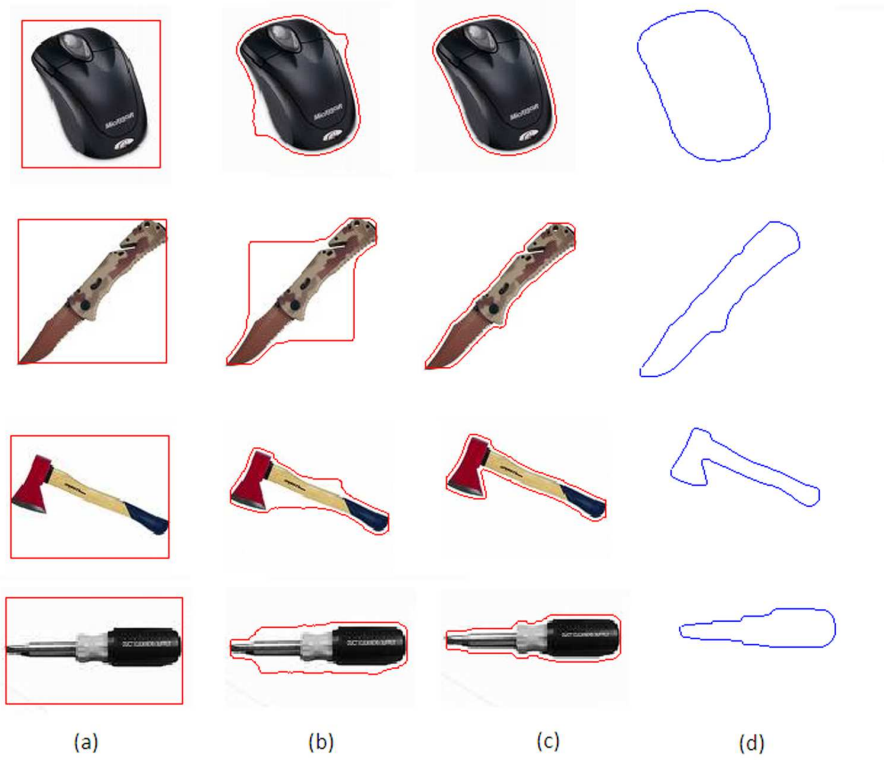

Figure 2.2: Active contour algorithm (a-initial contour, b-after 25 iterations c-after 50 iterations)and extracted boundary data (d) for different objects.

2.2 Object Modelling by Using Algebraic Curves

Once the object boundary data is obtained via visual segmentation step, it can be modeled by algebraic curves.This provides the advantages of model based approaches such as robustness in the presence of noise, clutter, and occlusion. Algebraic curve models are used in various computer vision problems such as pose estimation [71] and object recognition [8].

2.2.1 Algebraic Curves

Algebraic curves are defined by implicit equations of the form $f(x, y) = 0$, where $f(x, y)$ is a polynomial in the variables x, y, i.e. $f(x, y) = \sum_{ij} a_{ij} x^i y^j$ where $0 \leq i + j \leq n$ (n is finite) and the coefficients a_{ij} are real numbers. Alternatively, the intersection of an explicit surface $z = f(x, y)$ with the $z = 0$ plane yields an algebraic curve if $f(x, y)$ is a polynomial. Since the field of real numbers is not algebraically closed, it is often useful and illuminating to extend this definition to the complex field [1].

In general, an algebraic curve of degree n can be defined by the implicit polynomial (IP) equation:

$$f_n(x,y) = \underbrace{a_{00}}_{h_0} + \underbrace{a_{10}x + a_{01}y}_{h_1(x,y)} + \underbrace{a_{20}x^2 + a_{11}xy + a_{02}y^2}_{h_2(x,y)} + \ldots$$

$$+ \underbrace{a_{n0}x^n + a_{n-1,1}x^{n-1}y + \ldots + a_{0n}y^n}_{h_n(x,y)} = \sum_{r=0}^{n} h_r(x,y) = 0, \qquad (2.9)$$

where each binary form $h_r(x, y)$ is a homogeneous polynomial of degree r in the variables x and y. $h_n(x, y)$ is called the *leading form*. The number of terms in each $h_r(x, y)$ is $r + 1$, so that the equation defined by (2.9) has one

constant term, two terms of the first degree, three terms of the second degree, etc., up to and including $n+1$ terms of the (highest) n-th degree, for a total of $(n+1)(n+2)/2$ coefficients. Since the above equation can be multiplied by a non-zero constant without changing the zero set, an algebraic curve defined by $f_n(x,y) = 0$ has $(n+1)(n+2)/2-1 = n(n+3)/2$ independent coefficients or degrees of freedom (DOF). A *monic* algebraic curve $f_n(x,y) = 0$ will be defined by the condition that $a_{n0} = 1$ in (2.9). In the sequel, we will consider monic curves.

Algebraic curves of degree $1, 2, 3, 4, \ldots$ are called *lines, conics, cubics, quartics,* ...etc. Odd degree ($n = 2k+1$) algebraic curves have at least one real asymptote, and therefore they are inherently open. Even degree ($n = 2k$) curves can be either closed-bounded, i.e. no real asymptotes, or open, i.e. with some real asymptotes. Closed-bounded object contours can therefore be represented using only even degree algebraic curves.

2.2.2 Fitting Algebraic Curves to Object Boundary

Implicit polynomial (IP) models have proven to be more suitable than parametric representations for fitting algebraic curves to data with their advantages like global shape representation, smoothing noisy data and robustness against occlusion [68, 69, 70, 10, 4, 5]. Nonlinear or linear optimization methods may be used for IP fitting. However as nonlinear models have high computational costs [69, 70], linear models are preferable for especially real time applications.

One problem of the linear fitting methods is providing globally stabilized fits and being robust versus perturbational effects. Linear methods such as

3L fitting [34] address such problems. Furthermore it has been shown in [39] that ridge regression regularization noticeably increases the global stability of 3L fitting. In our work, we use this regularized 3L method for IP fitting purposes.

The main objective of all ideal IP curve fitting techniques is to approximate a given data set with a polynomial as closely as possible. This aim is achieved through minimization of the algebraic distance between the fitted curve and the input data. Generally IP's should have zero values at the data points (object boundary), negative values for inside points and positive values for outside points, or vice versa.

In 3L algorithm, data set to be curve fitted is first integrated with two more data sets with points at a distance, ϵ, inside and outside the original data Accordingly the IP function is forced to take $+1$ value at the outer layer, -1 at the inner level, and 0 at the intermediate layer. To express this relation in matrix form one can define a b vector, a coefficient vector a and the matrix of 3 layers of data, M, such that these matrices and vectors satisfy $Ma = b$. M, a and b can be given as follows:

$$M = \begin{bmatrix} M_{+\epsilon} \\ M_0 \\ M_{-\epsilon} \end{bmatrix} = \begin{bmatrix} Y_1^T \\ Y_2^T \\ ... \\ Y_{3N}^T \end{bmatrix}_{3N \times c}$$

$$a = \begin{bmatrix} a_{n,0} & a_{n-1,1} & ... & a_{0,n} & ... & a_{1,0} & a_{0,1} & a_{0,0} \end{bmatrix}_{c \times 1}^T$$

$$b = \begin{bmatrix} +1 & ... & +1 & 0 & ... & 0 & -1 & .. & -1 \end{bmatrix}_{3N \times 1}^T \tag{2.10}$$

where N is the number of data points, n is the degree of polynomial and

$c = (n + l)(n + 2)/2$ is the number of the coefficients of the IP curve and Y_i are the vectors of monomials for the 3 layers of data which can be shown as below:

$$Y_i = \begin{bmatrix} x_i^n & x_i^{n-1}y_i ... & x_i y_i^{n-1} & y_i^n ... & x_i^2 & x_i y_i & y_i^2 & x_i & y_i & 1 \end{bmatrix}^T \tag{2.11}$$

In this matrix form, the curve coefficient vector, a, can be obtained from M and b by:

$$a = M^{\dagger} b \tag{2.12}$$

where $M^{\dagger} = (M^T M)^{-1} M^T$ is the pseudo-inverse matrix for M. This method is invariant under Euclidean transformations, as the two synthetic layers is formed by using the distance measure ϵ.

As generalized inverse solutions are usually sensitive to perturbations and noise, regularization methods such as Tikhonov regularization or ridge regression helps in improving their performance and stability. In ridge regression regularized algorithm, coefficient matrix is obtained as:

$$a = (M^T M + \kappa D)^{-1} M^T b \tag{2.13}$$

where κ is the ridge regression parameter, and D is a diagonal matrix. Some example objects are presented in Fig. 2.3, which also depict their outline quartic curves obtained by a fitting technique [39].

2.2.3 Degree of the Implicit Polynomial

Selecting the degree of the implicit polynomial that can represent the shape of a boundary data is an interesting issue. What is the minimum degree for a polynomial that can fit the available data and what happens as

Figure 2.3: Images of various objects and their outline quartic curve models

the degree of the polynomial is increased? By intuition, one may expect that as the object shape gets more "complex", its IP representation should be of higher degree and as the degree of IP is increased it represents the boundary data better. Though such an interpretation seems reasonable, increasing the degree of IP in fitting procedures also can worsen the performance.

One of the main reasons is the noise that perturbs the boundary data. In general polynomials of high degree are more sensitive to such perturbations. Wilkinson [45] has explained this condition and presented interesting

examples where he shows that for 1D polynomials, a very tiny change the coefficient of high order terms can dramatically change the roots. Though his examples present some high degree polynomials can be quite ill-conditioned not all are so. Stability of a root is determined by its absolute value and Euclidean distance between it and other roots. Generally a root is more stable if its absolute value is small compared to its distance to other roots. It should be stated that many fitting algorithms are focused on the stability to avoid unstable polynomials. In this aspect fitting algorithms try to optimize the tradeoff between capturing the shape and obtaining stable fits. However, one should keep in mind that increasing the degree may result in unnecessary or undesired conditions such as fitting the noise or increasing the sensitivity of the polynomial against data perturbations. Thus, for practical issues, it is more reasonable to represent the shape of an object with an IP of degree as minimum as possible. Bezout's theorem [1] provides an easy and powerful procedure to achieve this goal.

Theorem 2.2.1. *Suppose that f and g are two plane projective curves that do not have a common component and defined over a field F. Then the total number of intersection points (counted with their multiplicities) of f and g with coordinates in an algebraically closed field E which contains F, is equal to the product of the degrees of f and g.*

Corollary 2.2.1.1. *Suppose that $f(x, y)$ is an algebraic curve defined over $\Re^2$ that represents the boundary data of an object and let $l(x, y)$ be a line defined over $\Re^2$ which has maximum n real intersection points with $f(x, y)$. Then $f(x, y)$ has a degree m, where $m \geqq n$.*

The corollary above is quite helpful in choosing the degree of IP to be

fitted. The procedure is simple. Boundary data of an object is inspected and an arbitrary line that intersects the curve at maximum number of points, n, is found. IP that will capture the shape of the object sufficiently should have degree m, where $m \geqq n$. This procedure is tested on different objects and fitted IPs of different degrees are shown in Fig. 2.4. It is seen that increase in the degree of IP improves the fit dramatically until the observation based on Bezout theorem is satisfied. Note that this observation does not guarantee that an IP with degree m will grasp the object shape as desired but it guarantees that a polynomial with degree less than m will not achieve a satisfactory performance. As it is desired to represent the curve with a minimum degree algebraic curve, Bezout theorem provides a very good start point in determining the degree of IP to be fitted.

2.2.4 Data Normalization

Data normalization is a crucial step in linear data processing procedures, especially when numeric inputs have significant scale differences. Under such conditions it is well known that normalization significantly increase the stability of algorithms as the condition number of the measurement matrix is an important factor in the analysis of the stability of linear problems. In his famous work, Hartley showed that the main reason for the poor conditioning of the measurement matrix is lack of comparability in the scales of data coordinates [38]. By pre-normalizing the data, the condition number of the measurement matrix can be decreased resulting in more robust estimations.

In our work data normalization is necessary for various reasons. First of all, it is important for polynomial fitting step to reduce the pathological

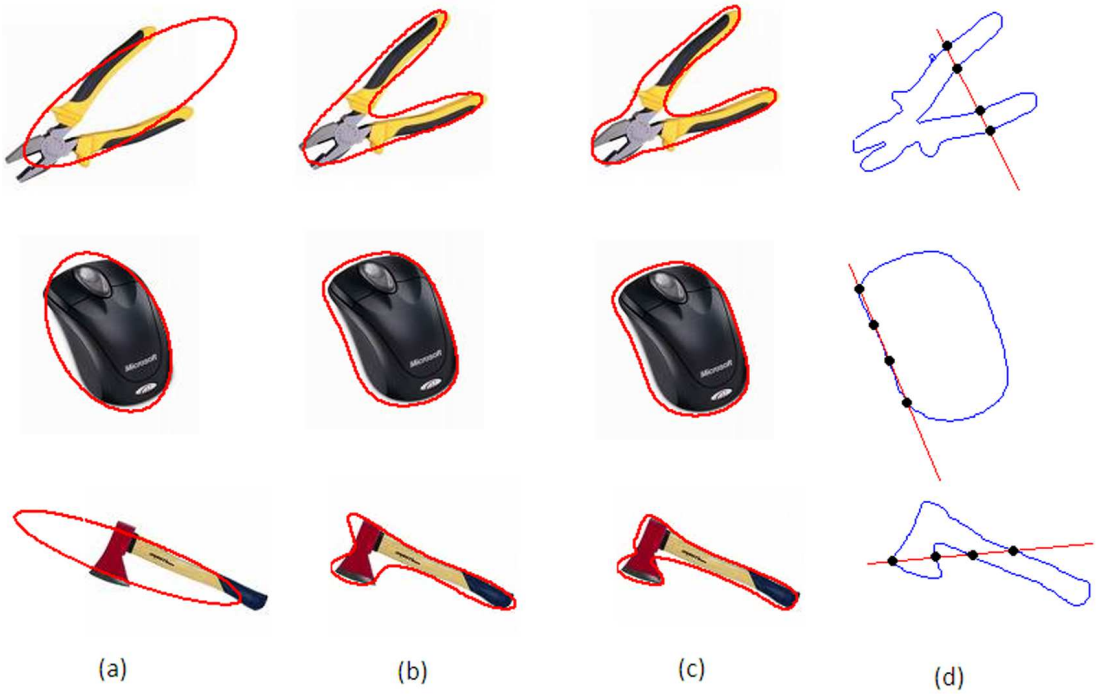

Figure 2.4: Boundary data of objects modeled with IPs of degree 2 (a), 4 (b) and 8 (c) along with the boundary curve intersected by a real line (d) for Bezout's theorem.

conditions in the resulting IPs. Such issues occur mostly due to high degree terms of the implicit polynomial when they are taking large values. It has been observed that normalization is necessary to obtain better results [39]. Furthermore, when the fitted IP is used to obtain the necessary features for motion estimation or visual servoing algorithms, the comparability in the numeric scale of these features dramatically change as the measurement data get higher values. As these features will be used as measurements, stability

of the considered algorithms should be increased by using normalized data.

There are different data normalization techniques in the literature. Among available techniques, we prefer to use a linear scaling technique, namely whitening [37]. One significant advantage of using this normalization procedure is due to the nature of used polynomial fitting algorithm. In [10] it is shown that whitening of two affine equivalent curves lead to normalized rotational equivalent curves and this is crucial since the used fitting algorithm is not affine invariant but Euclidean invariant. Two affine equivalent images of an object and corresponding rotationally equivalent normalized data are shown in Fig. 2.5.

Consider a set S of N data points $P_i = [x_i, y_i]^T$ which outline the boundary of a 2D curve. The center C and the covariance matrix Σ of S are defined as

$$C = \frac{1}{N}\sum_{i=1}^{N} P_i$$

$$\Sigma = \frac{1}{N-1}\sum_{i=1}^{N}(P_i - C)(P_i - C)^T \tag{2.14}$$

Since the covariance matrix Σ is symmetric, it can be diagonalized by an orthogonal matrix U composed of the eigenvectors of Σ, so that

$$\Lambda = U^T \Sigma U$$

where Λ is a diagonal matrix composed of the eigenvalues of Σ. In whitening normalization, normalized data set, $\hat{S}$, is obtained by applying the following transformation to the data points:

$$\hat{P}_i = \Lambda^{-1/2} U^T (P_i - C) \tag{2.15}$$

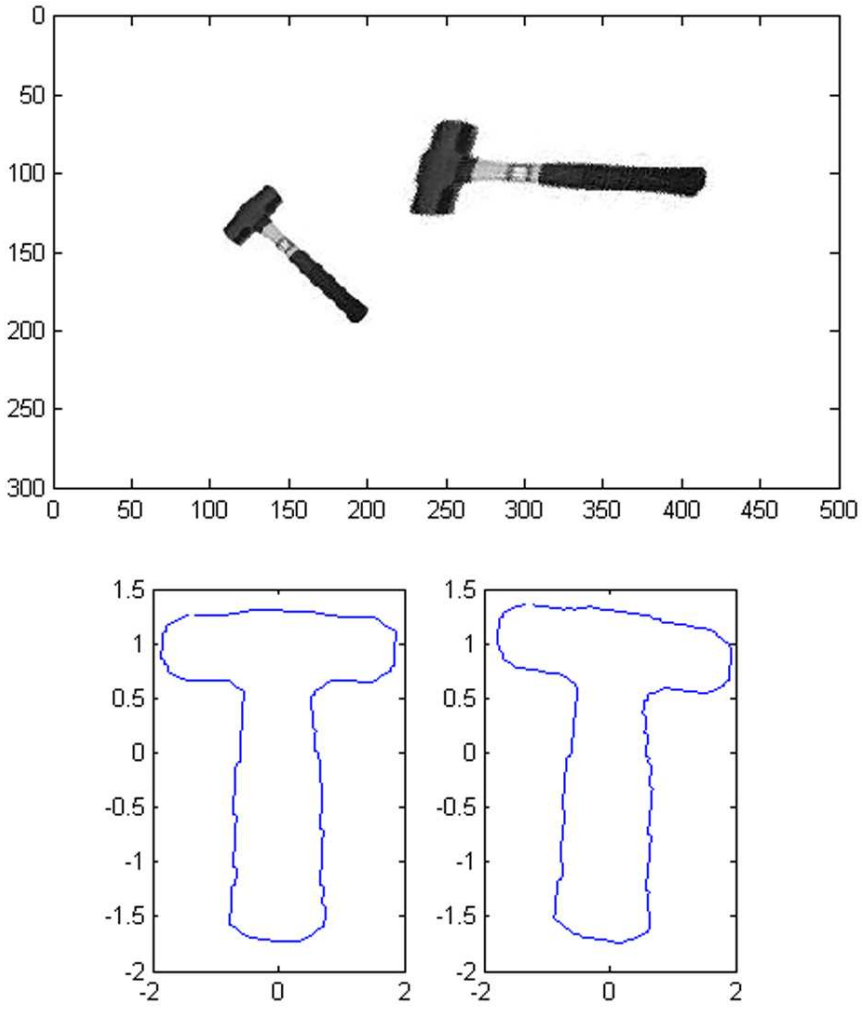

Figure 2.5: Affine equivalent images of an hammer and rotationally equivalent normalized boundary data obtained via whitening.

2.3 Optical Flow

As the visual information will be used for motion estimation and visual servoing purposes, it is important to know a motion in 3D world corresponds

to the motion on 2D image plane. This would enable a deeper understanding of both problems. Also the *interaction matrix* in visual servoing algorithms, as being relating the change of visual features with respect to the motion in 3D (V_c) are developed on the basis of this understanding. Developments in this part will be based on the *pinhole camera model* [63].

In pinhole camera model a point p with 3D coordinates $[X, Y, Z]^T$ in camera frame are projected onto the 2D metric normalized image plane coordinates $[x, y]^T$ with the following equations:

$$\begin{aligned} x &= \frac{X}{Z} \\ y &= \frac{Y}{Z} \end{aligned} \tag{2.16}$$

Taking the derivative of above equations with respect to time we get:

$$\begin{aligned} \dot{x} &= \frac{\dot{X}Z - X\dot{Z}}{Z^2} = \frac{\dot{X}}{Z} - \frac{\dot{Z}x}{Z} \\ \dot{y} &= \frac{\dot{Y}Z - Y\dot{Z}}{Z^2} = \frac{\dot{Y}}{Z} - \frac{\dot{Z}y}{Z} \end{aligned} \tag{2.17}$$

Now suppose that the point p is going through a rigid body motion in space with the translational (V_x, V_y, V_z) and rotational velocities (w_x, w_y, w_z) defined in the camera frame. Its instantaneous velocity satisfies the following equation:

$$\frac{d}{dt}\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0 & -w_z & w_y \\ w_z & 0 & -w_x \\ -w_y & w_x & 0 \end{bmatrix}\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} V_x \\ V_y \\ V_z \end{bmatrix} \tag{2.18}$$

Plugging (2.18) into (2.17)

$$\begin{aligned}\dot{x} &= \frac{-w_z Y + w_y Z + V_x}{Z} - \frac{(-w_y X + w_x Y + V_z)x}{Z} \\ \dot{y} &= \frac{w_z X - w_x Z + V_y}{Z} - \frac{(-w_y X + w_x Y + V_z)y}{Z}\end{aligned} \tag{2.19}$$

Rearranging (2.19) in matrix form we get,

$$\frac{d}{dt}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \frac{V_x}{Z} + w_y \\ \frac{V_y}{Z} - w_x \end{bmatrix} + \begin{bmatrix} -\frac{V_z}{Z} & -w_z \\ w_z & -\frac{V_z}{Z} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} w_y x^2 - w_x xy \\ -w_x y^2 + w_y xy \end{bmatrix} \tag{2.20}$$

Equation given above is quite helpful in studying the motion estimation and visual servoing problems. For motion estimation analysis, when certain motion models (such as rigid body or affine) are assumed for the data in image plane this equation will point the underlying motion assumption in 3D. For the visual servoing applications, this derivation provides an example for the analytical derivation of interaction matrix. Actually when the image features used in a visual servoing task are point coordinates in image plane, one can easily see that interaction matrix can be directly obtained from this equation.

Chapter 3

Motion Estimation of Freeform Planar Objects

Motion estimation is the process of determining the parameters that describe the transformation from one 2D image to another. Usually two consecutive frames in a video are considered for this task. It is an ill-posed problem due to the loss of dimension in image acquisition (from 3D scene onto 2D image plane). Generally a motion model is assumed for the data and parameters of that model are estimated. The motion model may be simple translational model, affine model or other models that leads to a successful approximation. Usually simple translational model leads to accumulation of errors and give sufficient results only for a small period of time. The affine motion model on the other hand provides good approximation for the induced image motion as long as the distance of the scene to the camera is large. In general image motion of an arbitrary planar surface between two frames is described by a projective transformation (*homography*).

Depending on the purpose of motion estimation algorithm motion vectors may be estimated for the whole image, which may be helpful for tasks such as video compression, or may be focused on a certain image region or object, which is preferred for tasks such as visual tracking. It is possible to classify motion estimation algorithms in two groups, namely *direct methods* [20] and *feature based methods* [19]. Direct methods claim image intensity is invariant to motion (i.e. it is constant throughout the motion) and they make use of regional intensity matching to estimate motion parameters. Block matching methods [15, 16], phase correlation methods [21], optical flow methods [12] are some examples of direct methods. Feature based methods on the other hand rely on the correspondence of a set of highly reliable image features. Lines [26, 27, 28, 29], Harris corners, color moments [22] or SIFT [23] are some of the features which can be used for motion estimation. Motion estimation method discussed in this work is also a feature based method as an IP representation of an object boundary will be used to obtain certain features that can be used to estimate the motion parameters of the curve.

3.1 Motion Estimation Using IP representation of A Planar Curve

In this section we are interested in exploring the dynamics of a planar algebraic curve and estimating its motion parameters. As an application to computer vision let us define the problem as follows.

Assume that you are given the boundary data of a freeform planar object through a sequence of images (possibly provided by a visual tracking al-

gorithm) and this boundary data satisfies an affine motion as provided in equation below. Estimate the motion parameters by using the IP model of available boundary data.

$$\frac{d}{dt}\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} a_1 & a_2 & b_1 \\ a_3 & a_4 & b_2 \\ 0 & 0 & 0 \end{bmatrix}}_{A} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{3.1}$$

As stated above we will be focusing on an affine motion model. In general, given two views of a scene, there is a linear projective transformation (*homography*) H, relating the projection of a point of a plane in the first view to its projection in the second view. This H is a three by three invertible matrix and need not be an affine matrix. However our affine motion model is quite sufficient for certain scenarios. This can be observed in (2.20). As long as the target planar curve has very small (ideally zero) rotational velocities around the axes perpendicular to the optical axis (w_x and w_y) motion of that planar curve in 3D space induces an affine motion on image plane. Furthermore, if its translational velocity is very small compared to the average depth from the camera ($V_z \ll Z$) then this motion corresponds to a rigid body transformation in image plane.

As we have seen in previous sections an IP model of this boundary data can be obtained through fitting algorithms. The problem will then be estimating the motion parameters of this curve from the fitted IP model. For this purpose first the dynamics of an algebraic curve should be represented. In order to describe dynamics of planar algebraic curves we use a polynomial decomposition method [6] to express curve models as a unique sum of

product of line factors.

Theorem 3.1.1. *Any non-degenerate monic polynomial, $f_n(x, y)$, can be uniquely decomposed as sum of product of line factors [4, 6] as shown in (3.2).*

$$f_n(x, y) = \Pi_n(x, y) + \alpha_{n-2}[\Pi_{n-2}(x, y) + \alpha_{n-4}[\Pi_{n-4}(x, y) + ...]] \tag{3.2}$$

where α_j's are scalar and $\Pi_j(x, y)$'s are the product of j line factors as given below.

$$\Pi_j(x, y) = \prod_{i=1}^{j}[x + l_{j,i}y + k_{j,i}] \tag{3.3}$$

For example, (monic) conic, cubic and quartic curves can be (line) decomposed as

$$f_2(x, y) = L_1(x, y)L_2(x, y) + \alpha_0 = 0,$$

$$f_3(x, y) = L_1(x, y)L_2(x, y)L_3(x, y) + \alpha_1 L_4(x, y) = 0,$$

and

$$f_4(x, y) = L_1(x, y)L_2(x, y)L_3(x, y)L_4(x, y) + \alpha_2 L_5(x, y)L_6(x, y) + \alpha_0 = 0, \tag{3.4}$$

respectively, where α_2, α_1 and α_0 are real scalars. Example of a quartic decomposition in terms of 6 complex lines is geometrically shown in Fig. 3.1.

For non-degenerate implicit polynomials this decomposition is unique. Dynamics of the parameters of line factors can be used to identify the dynamics of the curve. For example, $f_4(x, y)$ is completely described by the

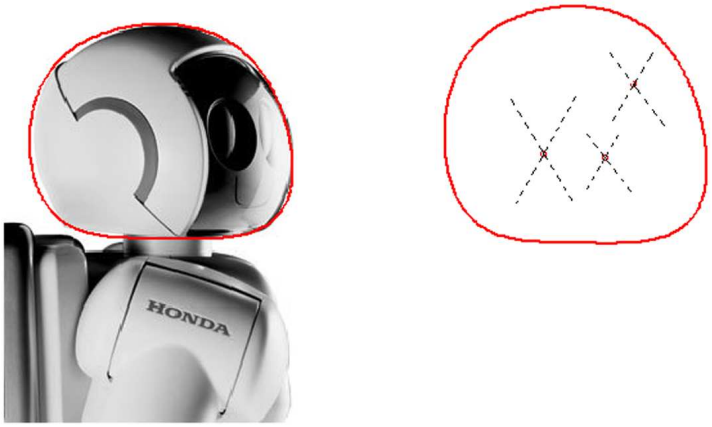

Figure 3.1: Head of humanoid robot Asimo by Honda along with fitted quartic polynomial and its complex line factors (dashed).

dynamics of the six line factors $L_i(x, y), i = 1, 2, ..., 6$ and two scalar parameters α_2 and α_0.

3.2 Homogeneous Representations for Even Degree Curves

It is shown in [24] that time invariant affine motion of a quartic curve induces Riccati equations in terms of parameters of these line factors. Further in [25] an adaptive identification is used to estimate time invariant parameters of a rigid body motion through line decomposition of planar algebraic curves.

Here we will consider a more general case. First we will show that same Riccati equations hold for time varying affine motion of an even degree algebraic curve. To this end we will use a homogeneous representation of even

degree curves similar to the development in [24].

Consider a line decomposed planar curve of degree n where $n = 2r$ is an even number.

$$f_n(x,y) = \prod_{i=1}^{n} \begin{pmatrix} 1 & l_{n,i} & k_{n,i} \end{pmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} + \alpha_{n-2} \prod_{i=1}^{n-2} \begin{pmatrix} 1 & l_{n-2,i} & k_{n-2,i} \end{pmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} + ...$$

$$+\alpha_2 \prod_{i=1}^{2} \begin{pmatrix} 1 & l_{2,i} & k_{2,i} \end{pmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} + \alpha_0 = 0 \tag{3.5}$$

Following substitutions are used to homogenize lines in the decomposition

$$x = \frac{\bar{x}}{\bar{w}}, \;\; y = \frac{\bar{y}}{\bar{w}}, \; l_{m,i} = \frac{\bar{l}_{m,i}}{\bar{p}_{m,i}}, \;\; k_{m,i} = \frac{\bar{k}_{m,i}}{\bar{p}_{m,i}}, \;\; m = 2, 4, ..., n \tag{3.6}$$

and obtain a homogeneous representation for the original curve as

$$f_n(\bar{x},\bar{y},\bar{w}) = \textstyle\prod_{i=1}^{n} \begin{pmatrix} \bar{p}_{n,i} & \bar{l}_{n,i} & \bar{k}_{n,i} \end{pmatrix} \begin{bmatrix} \bar{x} \\ \bar{y} \\ \bar{w} \end{bmatrix} +$$

$$\alpha_{n-2}(\tfrac{\prod_{i=1}^{n} \bar{p}_{n,i}}{\prod_{i=1}^{n-2} \bar{p}_{n-2,i}}) \bar{w}^2 \textstyle\prod_{i=1}^{n-2} \begin{pmatrix} \bar{p}_{n-2,i} & \bar{l}_{n-2,i} & \bar{k}_{n-2,i} \end{pmatrix} \begin{bmatrix} \bar{x} \\ \bar{y} \\ \bar{w} \end{bmatrix} + ... +$$

$$\alpha_2(\tfrac{\prod_{i=1}^{n} \bar{p}_{n,i}}{\prod_{i=1}^{2} \bar{p}_{2,i}}) \bar{w}^{n-2} \textstyle\prod_{i=1}^{2} \begin{pmatrix} \bar{p}_{2,i} & \bar{l}_{2,i} & \bar{k}_{2,i} \end{pmatrix} \begin{bmatrix} \bar{x} \\ \bar{y} \\ \bar{w} \end{bmatrix} + \alpha_0(\textstyle\prod_{i=1}^{n} \bar{p}_{n,i}) \bar{w}^n = 0 \tag{3.7}$$

3.3 Line Dynamics

For the affine motion of data coordinates, (3.1) represents the dynamics. Dynamics in (3.1) and homogezitation of data coordinates presented in (3.6) together imply,

$$\frac{d}{dt}\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} a_1 & a_2 & b_1 \\ a_3 & a_4 & b_2 \\ 0 & 0 & 0 \end{bmatrix}}_{A} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad \begin{matrix} x = \frac{\bar{x}}{\bar{w}},\ y = \frac{\bar{y}}{\bar{w}} \\ \longrightarrow \end{matrix} \quad \frac{d}{dt}\begin{bmatrix} \bar{x} \\ \bar{y} \\ \bar{w} \end{bmatrix} = \underbrace{\begin{bmatrix} a_1 & a_2 & b_1 \\ a_3 & a_4 & b_2 \\ 0 & 0 & 0 \end{bmatrix}}_{A} \begin{bmatrix} \bar{x} \\ \bar{y} \\ \bar{w} \end{bmatrix} \tag{3.8}$$

Let us consider any line, $\begin{bmatrix} \bar{p}_{m,i}(t) & \bar{l}_{m,i}(t) & \bar{k}_{m,i}(t) \end{bmatrix}^T$, that is obtained through the decomposition. For each point, $\begin{bmatrix} \bar{x}_l(t) & \bar{y}_l(t) & \bar{w}_l(t) \end{bmatrix}^T$, on that line, following equation is satisfied.

$$\begin{bmatrix} \bar{p}_{m,i}(t) & \bar{l}_{m,i}(t) & \bar{k}_{m,i}(t) \end{bmatrix} \begin{bmatrix} \bar{x}_l(t) \\ \bar{y}_l(t) \\ \bar{w}_l(t) \end{bmatrix} = 0 \tag{3.9}$$

Taking derivative of (3.9) with respect to time we get,

$$\begin{bmatrix} \dot{\bar{p}}_{m,i}(t) \\ \dot{\bar{l}}_{m,i}(t) \\ \dot{\bar{k}}_{m,i}(t) \end{bmatrix}^T \begin{bmatrix} \bar{x}_l(t) \\ \bar{y}_l(t) \\ \bar{w}_l(t) \end{bmatrix} + \begin{bmatrix} \bar{p}_{m,i}(t) \\ \bar{l}_{m,i}(t) \\ \bar{k}_{m,i}(t) \end{bmatrix}^T \begin{bmatrix} \dot{\bar{x}}_l(t) \\ \dot{\bar{y}}_l(t) \\ \dot{\bar{w}}_l(t) \end{bmatrix} = 0 \tag{3.10}$$

If we plug the dynamics in (3.8) into (3.10) we obtain,

$$\begin{bmatrix} \dot{\bar{p}}_{m,i}(t) \\ \dot{\bar{l}}_{m,i}(t) \\ \dot{\bar{k}}_{m,i}(t) \end{bmatrix}^T \begin{bmatrix} \bar{x}_l(t) \\ \bar{y}_l(t) \\ \bar{w}_l(t) \end{bmatrix} + \begin{bmatrix} \bar{p}_{m,i}(t) \\ \bar{l}_{m,i}(t) \\ \bar{k}_{m,i}(t) \end{bmatrix}^T A \begin{bmatrix} \bar{x}_l(t) \\ \bar{y}_l(t) \\ \bar{w}_l(t) \end{bmatrix} = 0$$

$$\left(\begin{bmatrix} \dot{\bar{p}}_{m,i}(t) \\ \dot{\bar{l}}_{m,i}(t) \\ \dot{\bar{k}}_{m,i}(t) \end{bmatrix}^T + \begin{bmatrix} \bar{p}_{m,i}(t) \\ \bar{l}_{m,i}(t) \\ \bar{k}_{m,i}(t) \end{bmatrix}^T A \right) \begin{bmatrix} \bar{x}_l(t) \\ \bar{y}_l(t) \\ \bar{w}_l(t) \end{bmatrix} = 0 \tag{3.11}$$

In light of (3.9) it follows that

$$\frac{d}{dt} \begin{bmatrix} \bar{p}_{m,i}(t) \\ \bar{l}_{m,i}(t) \\ \bar{k}_{m,i}(t) \end{bmatrix} + A^T \begin{bmatrix} \bar{p}_{m,i}(t) \\ \bar{l}_{m,i}(t) \\ \bar{k}_{m,i}(t) \end{bmatrix} = \lambda \begin{bmatrix} \bar{p}_{m,i}(t) \\ \bar{l}_{m,i}(t) \\ \bar{k}_{m,i}(t) \end{bmatrix} \tag{3.12}$$

where λ is an unknown scalar. Note that this equality implies the following dynamics for the homogenized line parameters:

$$\frac{d}{dt} \begin{bmatrix} \bar{p}_{m,i}(t) \\ \bar{l}_{m,i}(t) \\ \bar{k}_{m,i}(t) \end{bmatrix} = \begin{bmatrix} \lambda - a_1 & -a_3 & 0 \\ -a_2 & \lambda - a_4 & 0 \\ -b_1 & -b_2 & \lambda \end{bmatrix} \begin{bmatrix} \bar{p}_{m,i}(t) \\ \bar{l}_{m,i}(t) \\ \bar{k}_{m,i}(t) \end{bmatrix} \quad i = 1, 2, ..., m \tag{3.13}$$

Notice that A can be any affine matrix with possibly time varying entries.

3.4 Riccati Equations

Following the development in [24] we show that line parameters $l_{m,i}, k_{m,i}$ in the decomposition of the original curve satisfy coupled Riccati equations. Also we will see in the derivations that uncertainty in (3.13) due to unknown

scalar λ disappears in these Ricatti equations. To this end, first we differentiate (3.6) with respect to time

$$l_{m,i} = \frac{\bar{l}_{m,i}}{\bar{p}_{m,i}}$$

$$\dot{l}_{m,i} = \frac{\dot{\bar{l}}_{m,i}\bar{p}_{m,i} - \bar{l}_{m,i}\dot{\bar{p}}_{m,i}}{\bar{p}_{m,i}^2} \tag{3.14}$$

Using (3.13) we can get

$$\dot{l}_{m,i} = \frac{(-a_2\bar{p}_{m,i} + (\lambda - a_4)\bar{l}_{m,i})\bar{p}_{m,i} - \bar{l}_{m,i}((\lambda - a_1)\bar{p}_{m,i} - a_3\bar{l}_{m,i})}{\bar{p}_{m,i}^2}$$

$$\dot{l}_{m,i} = -a_2 + (\lambda - a_4)\underbrace{\frac{\bar{l}_{m,i}}{\bar{p}_{m,i}}}_{l_{m,i}} + (a_1 - \lambda)\underbrace{\frac{\bar{l}_{m,i}}{\bar{p}_{m,i}}}_{l_{m,i}} + a_3\underbrace{\frac{\bar{l}_{m,i}^2}{\bar{p}_{m,i}^2}}_{l_{m,i}^2}$$

$$\dot{l}_{m,i} = -a_2 + (a_1 - a_4)l_{m,i} + a_3 l_{m,i}^2, \quad i = 1, \ldots, m, \quad m = 2, 4, \ldots, n \tag{3.15}$$

With a similar development, equation for $k_{m,i}$ is obtained as:

$$\dot{k}_{m,i} = -b_1 - b_2 l_{m,i} + a_1 k_{m,i} + a_3 l_{m,i} k_{m,i}, \quad i = 1, \ldots, m, \quad m = 2, 4, \ldots, n \tag{3.16}$$

Note that the line parameters, i.e. slope and intercept, satisfy coupled Riccati equations where the uncertainty due to unknown scalar λ disappears and parameters in the equation depend on the motion of the curve. Note also that each of the lines satisfies the same Riccati equation initialized at different points on the state space.

As remarked earlier, closed-bounded curves have no real asymptotes and therefore the first n lines in the decomposition of such curves are all complex. In light of this observation, $l_{n,i}$ and $k_{n,i}$ are complex numbers. However, $l_{m,i}$

and $k_{m,i}$ where $m \neq n$, may or may not be complex. Since the coefficients of a curve are real, if there exists a complex parameter its conjugate must also exist.

3.4.1 Riccati Equations in Real Variables

In [25], Riccati equations in real variables were derived with constant motion parameters. We extend that work to time varying parameters. Since the first n lines in the decomposition of a curve of degree n are complex, their parameters can be written as:

$$l_{n,i} = \eta_{1i} + j\eta_{2i}, \quad k_{n,i} = \eta_{3i} + j\eta_{4i}, \quad i = 1, 3, \ldots, n-1 \tag{3.17}$$

where $\eta_{1i} = Re(l_{n,i})$, $\eta_{2i} = Im(l_{n,i})$, $\eta_{3i} = Re(k_{n,i})$, $\eta_{n,i} = Im(k_{n,i})$, and

$$l_{n,i} = \eta'_{1i} + j\eta'_{2i}, \quad k_{n,i} = \eta'_{3i} + j\eta'_{4i}, \quad i = 2, 4, \ldots, n \tag{3.18}$$

where $\eta'_{1i} = \eta_{1i}$, $\eta'_{2i} = -\eta_{2i}$, $\eta'_{3i} = \eta_{3i}$, $\eta'_{4i} = -\eta_{4i}$.

Substituting these into (3.15) and (3.16), and equating real and imaginary parts, we get the following Riccati equations in real variables:

$$\dot{\eta}_{1i} = -a_2 + (a_1 - a_4)\eta_{1i} + a_3({\eta_{1i}}^2 - {\eta_{2i}}^2) \tag{3.19}$$

$$\dot{\eta}_{2i} = (a_1 - a_4)\eta_{2i} + 2a_3\eta_{1i}\eta_{2i} \tag{3.20}$$

$$\dot{\eta}_{3i} = -b_1 - b_2\eta_{1i} + a_1\eta_{3i} + a_3(\eta_{1i}\eta_{3i} - \eta_{2i}\eta_{4i}) \tag{3.21}$$

$$\dot{\eta}_{4i} = -b_2\eta_{2i} + a_1\eta_{4i} + a_3(\eta_{1i}\eta_{4i} + \eta_{2i}\eta_{3i}) \tag{3.22}$$

and similarly for the conjugate variables η'_{1i} to η'_{4i} as:

$$\dot{\eta}'_{1i} = -a_2 + (a_1 - a_4)\eta'_{1i} + a_3({\eta'_{1i}}^2 - {\eta'_{2i}}^2) \tag{3.23}$$

$$\dot{\eta}'_{2i} = (a_1 - a_4)\eta'_{2i} + 2a_3\eta'_{1i}\eta'_{2i} \tag{3.24}$$

$$\dot{\eta}'_{3i} = -b_1 - b_2\eta'_{1i} + a_1\eta'_{3i} + a_3(\eta'_{1i}\eta'_{3i} - \eta'_{2i}\eta'_{4i}) \tag{3.25}$$

$$\dot{\eta}'_{4i} = -b_2\eta'_{2i} + a_1\eta'_{4i} + a_3(\eta'_{1i}\eta'_{4i} + \eta'_{2i}\eta'_{3i}) \tag{3.26}$$

3.5 Identification of Motion Parameters

Using vector-matrix notation and dropping the subscript i, equations (3.19) to (3.22), or alternatively (3.23) to (3.26), for a specific complex conjugate line pair can be recast as

$$\begin{bmatrix} \dot{\eta}_1 \\ \dot{\eta}_2 \\ \dot{\eta}_3 \\ \dot{\eta}_4 \end{bmatrix} = \begin{bmatrix} 0 & 0 & -\eta_1 & -1 & \eta_1^2 - \eta_2^2 & -\eta_1 \\ 0 & 0 & \eta_2 & 0 & 2\eta_1\eta_2 & -\eta_2 \\ -1 & -\eta_1 & \eta_3 & 0 & \eta_1\eta_3 - \eta_2\eta_4 & 0 \\ 0 & -\eta_2 & \eta_4 & 0 & \eta_1\eta_4 + \eta_2\eta_3 & 0 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix} \tag{3.27}$$

Note that in this form all the unknown motion parameters are stacked in a vector $\begin{bmatrix} b_1 & b_2 & a_1 & a_2 & a_3 & a_4^T \end{bmatrix}^T$ and we want to estimate them through our measurements $\begin{bmatrix} \eta_1 & \eta_2 & \eta_3 & \eta_4 \end{bmatrix}^T$. From (3.27) it is clear that with 6 unknowns and 4 equations we do not have a unique solution. Consequently we need to consider parameters for at least 2 lines. For the additional line it is better to choose not the complex conjugate of the first line but one from another pair as parameters of complex conjugate lines do not provide fully independent information. By using the parameters for two complex lines we

obtain the following equation.

$$\underbrace{\begin{bmatrix}\dot{\eta}_1\\ \dot{\eta}_2\\ \dot{\eta}_3\\ \dot{\eta}_4\\ \dot{\eta}_5\\ \dot{\eta}_6\\ \dot{\eta}_7\\ \dot{\eta}_8\end{bmatrix}}_{\dot{X}_p} = \underbrace{\begin{bmatrix} 0 & 0 & -\eta_1 & -1 & \eta_1^2-\eta_2^2 & -\eta_1\\ 0 & 0 & \eta_2 & 0 & 2\eta_1\eta_2 & -\eta_2\\ -1 & -\eta_1 & \eta_3 & 0 & \eta_1\eta_3-\eta_2\eta_4 & 0\\ 0 & -\eta_2 & \eta_4 & 0 & \eta_1\eta_4+\eta_2\eta_3 & 0\\ 0 & 0 & -\eta_5 & -1 & \eta_5^2-\eta_6^2 & -\eta_5\\ 0 & 0 & \eta_6 & 0 & 2\eta_5\eta_6 & -\eta_6\\ -1 & -\eta_5 & \eta_7 & 0 & \eta_5\eta_7-\eta_6\eta_7 & 0\\ 0 & -\eta_6 & \eta_8 & 0 & \eta_5\eta_8+\eta_6\eta_7 & 0\end{bmatrix}}_{F} \underbrace{\begin{bmatrix}b_1\\ b_2\\ a_1\\ a_2\\ a_3\\ a_4\end{bmatrix}}_{\varphi} \tag{3.28}$$

This system can be constructed with more lines, however in simulations and experiments it is observed that with two independent lines matrix F attains rank of 6 which is enough for solving the system. Adding more lines may help in increasing robustness but not necessary. As the line parameters are measurable from acquired images we can obtain the estimates for motion parameters as.

$$\varphi = F^{\dagger}\dot{X}_p \tag{3.29}$$

where $F^{\dagger} = (F^T F)^{-1}F^T$ is the pseudo-inverse of F. Equation (3.29) provides the solution for a continuous system. On the contrary when using this method for computer vision applications discrete measurements are provided. In estimating the motion parameters for a boundary data, line parameters are extracted from each acquired image and the motion estimation problem is formulated in a discrete fashion. In this discrete form, we can obtain $\dot{X}_p$ with a backward Euler approximation, namely

$$\dot{X}_p[k] \cong \frac{X_p[k] - X_p[k-1]}{T} \tag{3.30}$$

where T is the time interval between two acquired images. T can be assumed to be unity in applications and in that case estimated motion parameters will just be normalized by T. Using the backward Euler approximation for $\dot{X}_p[k]$ and forming $F[k]$ from the measured line parameters $X_p[k]$ we can obtain the motion estimation at instant k as:

$$\varphi[k] = F^{\dagger}[k]\dot{X}_p[k] \tag{3.31}$$

3.5.1 Data Normalization in Motion Estimation Algorithm

$F[k]$ in 3.31 is computed in each iteration and entries of this matrix involves the terms η_1,...,η_8 which are real and complex terms of slope (l) and intercept (k) of two decomposed lines in the form of $x + ly + k = 0$. In the decomposed lines l terms are independent of the translations of data, whereas k terms can be greatly affected as the set of data points get farther from the origin and get larger values. Note that the leading form of an implicit polynomial is invariant of translation and l term is obtained from the decomposition of leading form whereas k terms comes from the lower degree coefficients. To illustrate this condition let us consider a simple scenario where the data of an object boundary satisfies $x^2 + y^2 - 1 = 0$, which is a unit circle at the origin. Decomposition of this polynomial would result in

$$x^2 + y^2 - 1 = (x + jy)(x - jy) - 1 = 0$$

Now let us consider that the represented data is taken translated with an amount of $[20, 20]^T$, which results in a new polynomial

$$(x - 20)^2 + (y - 20)^2 - 1 = x^2 + y^2 - 40x - 40y + 799 = 0$$

Decomposing this polynomial we get

$$x^2 + y^2 - 40x - 40y + 799 = (x + jy + 20 - 20j)(x - jy + 20 + 20j) - 1$$

Note that with that amount of translation of the data points l terms remain constant as j and $-j$ whereas k terms change from 0 to $20 - 20j$ and $20 + 20j$. As it can be seen, although the data coordinates coming from a unit circle remain comparable as data is translated, k terms of decomposed lines can increase or decrease significantly. Consequently as the data gets farther from the origin, k terms increase significantly in magnitude leading to very large η_3, η_4, η_7 and η_8 which dramatically affect the condition number of matrix $F[k]$ in 3.31 and stability of motion estimation algorithm. Thus, data normalization is necessary in this algorithm. First the motion parameters for the normalized data can be robustly estimated with this algorithm and the motion parameters for original data can be extracted from this estimation. There are different data normalization techniques in the literature. Among available techniques, we prefer to use a linear scaling technique, namely whitening [37]. One significant advantage of using this normalization procedure is due to the nature of used polynomial fitting algorithm. As shown in [10], whitening of two affine equivalent curves lead to normalized rotational equivalent curves and this is crucial since the used fitting algorithm is not affine invariant but Euclidean invariant.

3.5.2 Extraction of Motion Parameters from the Normalized Data

Let S_h represent the homogeneous coordinates of the original boundary data and $\hat{S}_h$ be the normalized data.

$$S_h = \begin{bmatrix} x_1 & x_2 & ... & x_N \\ y_1 & y_2 & ... & y_N \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Since the whitening normalization is a linear process there exists a matrix B such that,

$$\hat{S}_h = BS_h \tag{3.32}$$

In light of (2.15) B is defined as

$$B = \begin{bmatrix} \Lambda^{-1/2}U^T & -\Lambda^{-1/2}U^TC \\ 0_{1\times 2} & 1 \end{bmatrix} \tag{3.33}$$

Once the normalized data is obtained, its motion parameters can be estimated by using (3.31). As the whitening of two affine equivalent curves give rotationally equivalent curves, the estimated parameters will correspond to a pure rotation. In general dynamics of the normalized data and original data can be represented in matrix form as:

$$\dot{\hat{S}}_h = \hat{A}\hat{S}_h \tag{3.34}$$

$$\dot{S}_h = AS_h \tag{3.35}$$

Note that $\hat{A}$ is obtained in motion estimation algorithm and if one can relate A and $\hat{A}$ properly, motion estimation task can be completed. To achieve this

goal, let us consider the time derivative of (3.32)

$$\dot{\hat{S}}_h = \dot{B}S_h + B\dot{S}_h \tag{3.36}$$

Using the relations in (3.32) and (3.34) one can obtain

$$\dot{\hat{S}}_h = \hat{A}\hat{S}_h = \hat{A}BS_h \tag{3.37}$$

From the equality of (3.36) and (3.37) one can obtain

$$\dot{B}S_h + B\dot{S}_h = \hat{A}BS_h$$

$$B\dot{S}_h = (\hat{A}B - \dot{B})S_h$$

since B is always invertible,

$$\dot{S}_h = B^{-1}(\hat{A}B - \dot{B})S_h \tag{3.38}$$

Using the equality of (3.35) and (3.38) one can obtain the matrix A as

$$A = B^{-1}(\hat{A}B - \dot{B}) \tag{3.39}$$

By using (3.39) motion estimation for the target object is recovered from the motion estimation of normalized data.

3.6 Simulation Results

Simulations are performed in the Matlab and Simulink environment. Boundary data of different objects are extracted via level set method. Different affine motions are applied to these data. Sample affine motion for the screwdriver data is shown in Fig. 3.2 whereas Fig. 3.3 shows a sample rigid body motion for a machine part data.

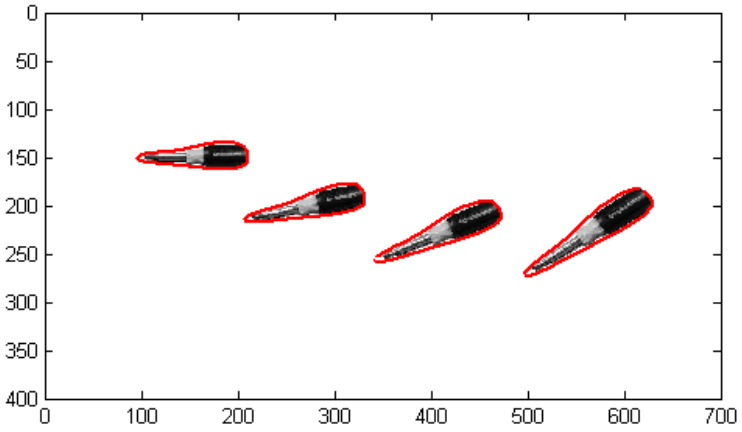

Figure 3.2: Affine motion of a screwdriver with superimposed quartic curves

To simulate the behavior of the camera, we constructed motion dynamics subsystem in Simulink, which generates state values, i.e. image coordinates of the object boundary, at prescribed frame rates. Boundary data is normalized by means of whitening in the curve fitting step. Closed-bounded quartic curves are fitted to the object boundaries each sampling instant using the IP fitting procedure. Fitted polynomials are decomposed into line factors and 1 line from each of the first 2 complex conjugate couples are picked for motion estimation. Motion parameters of the normalized data is estimated and motion parameters for the original data is extracted from that estimation. In simulations we considered a low frame rate (30 fps) camera. This is achieved using zero-order hold at the output port of the motion dynamics subsystem with a sample time equal to 1/30. Sample time of the simulations is 0.0001 s. Run time is 3 s.

In the first simulation a time varying rigid body motion is applied to the

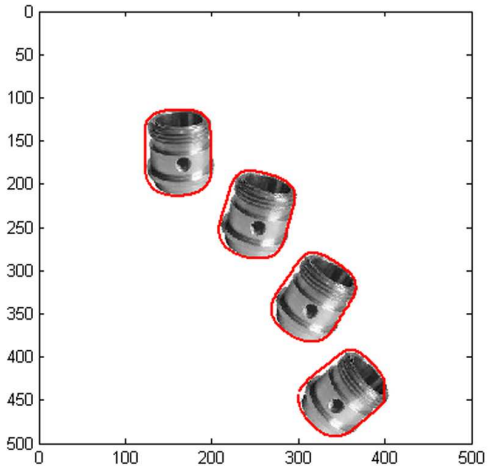

Figure 3.3: Rigid motion of a machine part with superimposed quartic curves

boundary of a machine part data. Actual and estimated parameters for this simulation are given in Fig. 3.4. It is seen that the estimated parameters track the actual parameters very closely. In the second simulation a time varying affine motion is applied to the boundary data of a screwdriver. Actual and estimated parameters for this simulation are given in Fig. 3.5. In both simulations rigid and affine motions with arbitrary time varying parameters are applied to the data and it is seen that proposed estimation method works quite well.

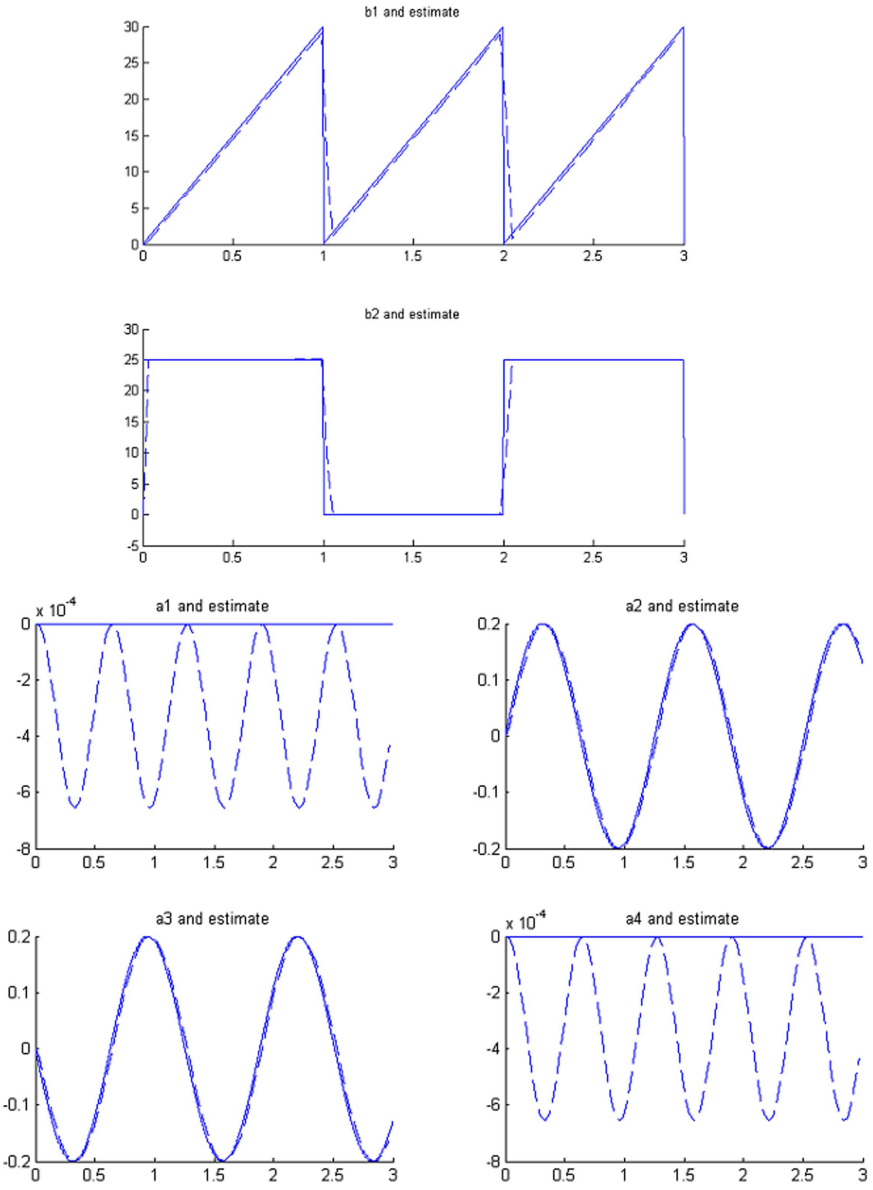

Figure 3.4: Actual (solid) and estimated (dashed) motion parameters for rigid body motion.

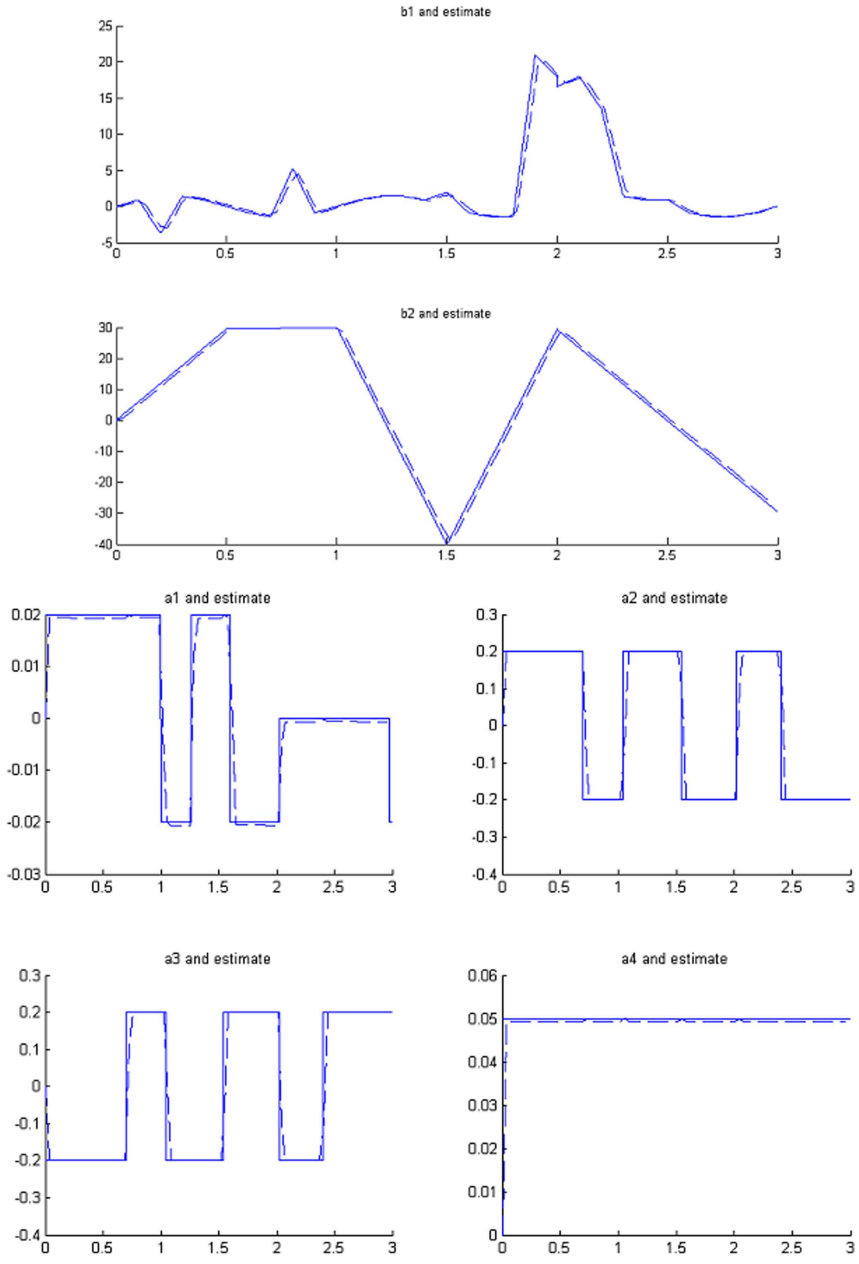

Figure 3.5: Actual (solid) and estimated (dashed) motion parameters for affine motion.

Chapter 4

Visual Servoing Using Planar Algebraic Curve Features

4.1 Visual Servoing

Visual servo control refers to the use of computer vision data to control the motion of a robot [51]. Visual data is gathered through a camera and computer vision algorithms are used to obtain certain features that will be used to control the position of the robot. Various camera configurations can be used for such approaches. For instance, a single camera can be mounted on the robot (eye-in-hand) where it moves with the robot, or the camera can be fixed in the workspace and observe the robot motion from a stationary pose (eye-to-hand). Different configurations can also be obtained by using multiple cameras which can provide the advantages of stereo vision. Though certain changes occur in the mathematical derivations, similar principles apply in all cases.

Just in any control task, the aim of visual servoing is to minimize a defined error vector, $e(t) = s(p(t), a) - s^*$, where s^* defines the value of the feature vector at the desired pose and $s(p(t), a)$ is the current feature vector which depends on relative pose of the target object with respect to the camera, $p(t)$, and a, a set of parameters that represent potential additional knowledge about the system such as camera intrinsic parameters or 3D models of objects. In many applications s^* is constant, meaning that we have a constant reference pose of the camera with respect to the target object. This assumption is acceptable for many applications such as pick and place or robot navigation task where the robot is desired to keep or achieve a particular pose with respect to an other robot or object.

Depending on the choice of feature vector, $s(p(t), a)$, different approaches exist. In image based visual servoing [46, 51], visual measurements are directly used for the control purpose. These measurements can be point coordinates, lines, ellipses, visual moments [50, 51, 52, 53] or other visual features which can be gathered and tracked in real time [49]. On the other hand in position based visual servoing approaches [47], the visual measurements are not directly used in the control loop, but they are used to recover the relative pose of the camera between the reference and current poses and this relative pose is used as the error vector. Both approaches has certain advantages and disadvantages. To make use of advantages of both algorithms, an alternative approach, named as "$2\frac{1}{2}$D Visual Servoing" [48], was proposed. In this thesis we will focus on image based visual servoing.

For an image based visual servoing task, the problem is designing a control signal (V_c - velocity screw for the target object in camera frame) that will min-

imize the error, $e(t)$. In a classical image based visual servoing the additional parameters, a, in $s(p(t), a)$ are the camera intrinsic parameters (coordinates of the principal point and effective focal lengths in x and y directions) which can be assumed to be constant throughout the control action. To obtain a relation for V_c, let us first take the derivative of $e(t) = s(p(t), a) - s^*$ with respect to time:

$$\frac{de(t)}{dt} = \frac{ds(p(t), a)}{dt} - \frac{ds^*}{dt} \tag{4.1}$$

Considering s^* and camera intrinsic parameters to be constant we get:

$$\frac{de(t)}{dt} = \underbrace{\frac{\partial s(p(t), a)}{\partial p(t)}}_{L_e} \underbrace{\frac{\partial p(t)}{\partial t}}_{V_c} \tag{4.2}$$

where L_e is named as *interaction matrix* (or *image Jacobian*). For a control task, a simple approach may be trying to enforce a decoupled exponential decrease in error. That can be achieved by following equation:

$$\frac{de(t)}{dt} = -\lambda e(t) \tag{4.3}$$

where λ is a diagonal positive definite gain matrix. Setting (4.2) and (4.3) to be equal we get the following relation .

$$L_e V_c = -\lambda e(t) \tag{4.4}$$

From (4.4) one can obtain the relation for the control signal as

$$V_c = -\lambda L_e^{\dagger} e(t) \tag{4.5}$$

where $L_e^{\dagger} = (L_e^T L_e)^{-1} L_e^T$ is the pseudo-inverse of image Jacobian, L_e. Usually L_e contains terms such as depth, intrinsic parameters, etc. which are unknown but can be estimated. Hence this is more properly shown as:

$$V_c = -\lambda \widehat{L_e^{\dagger}} e(t) \tag{4.6}$$

where $\widehat{L_e^{\dagger}}$ is an estimate for the pseudo-inverse of interaction matrix.

This basic mathematical development is used in many visual servoing applications. Remaining issues for any visual servoing algorithm is choosing visual features to be used, obtaining an analytical interaction matrix corresponding to the chosen feature vector, propose how certain unknowns in interaction matrix can be estimated and analyzing its performance in a closed loop system. By considering such issues many different methods can be proposed and used for visual servoing tasks.

In this section we consider the use of features extracted from the IP representation of planar algebraic curves in aligning planar closed curves by employing calibrated image based visual servoing [51]. An implicit polynomial representation of target object boundary is obtained by a curve fitting algorithm. Acquired polynomial is then decomposed as a unique sum of product of line factors. As the line decomposition is unique for a non-degenerate implicit polynomial in variables x and y, it can be used to extract certain robust features that represent the curve and we propose that such features can be used for visual servoing purposes. In this work we propose to use intersection of complex conjugate line pairs. Since the first n lines in the decomposition of an IP with degree n are complex conjugate pairs, their intersections are real points on the image plane and can be used as point features.

It should be stated that under 6 DOF motion, reference and current boundary data are related by a projective transformation. Since we treat the extracted points as visual features, they should correspond to the same points with respect to the curve under projective transformations. Such a correspondence depends on the invariance of curve fitting. IP fitting method

used in this work is Euclidean invariant and we achieve affine invariance through the whitening normalization of boundary data. If two boundary curves are affine equivalent, their whitening normalization provide rotationally equivalent curves. Consequently with our method correspondence of extracted features under affine transformations is achieved. As long as the average depth of the object from the camera is large, or rotations about the X and Y axis of the camera are small object boundary in two different images will be related by an affine transformation and our method would work properly. However, even in the deviations from affine relation the closed loop control helps in handling this problem. As the closed loop control forces the end effector to the reference pose it also forces the relation between the current and reference boundary data to be affine. In the 6 DOF simulations our results support this claim, however in very large deviations from the affine model this method may not be applicable due the lack of perspective invariance in fitting method.

4.2 Visual Servoing by Using the Intersection of Complex Line Pairs

In this section we treat the real intersection of complex lines as real point features. Let $s \in \Re^k$ and $r \in \Re^6$, denote the vectors of image features obtained from visual system and the pose of the end effector of the robot, respectively. The vector s is a function of r, and their time derivatives are related with the image Jacobian $J_I(r) = \partial s/\partial r \in \Re^{kx6}$ as,

$$\dot{s} = J_I(r)\dot{r} \tag{4.7}$$

For a fixed camera system the image Jacobian of a single point feature vector $s = [x, y]^T$ is given as [51]:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \underbrace{\begin{bmatrix} 1/Z & 0 & -x/Z & -xy & (1+x^2) & -y \\ 0 & 1/Z & -y/Z & -(1+y^2) & xy & x \end{bmatrix}}_{J_{xy}} V_c \tag{4.8}$$

where

$$x = \frac{x_p - x_c}{f_x},\ y = \frac{y_p - y_c}{f_y} \tag{4.9}$$

and (x_p, y_p) are pixel coordinates of the image point, (x_c, y_c) are the coordinates of the principle point, and (f_x, f_y) are effective focal lengths of the camera, respectively. By rearranging and differentiating (4.9), and writing in matrix form, the following expression can be obtained.

$$\begin{bmatrix} \dot{x_p} \\ \dot{y_p} \end{bmatrix} = \begin{bmatrix} f_x & 0 \\ 0 & f_y \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \underbrace{\begin{bmatrix} f_x & 0 \\ 0 & f_y \end{bmatrix} J_{xy}}_{J_I} V_c \tag{4.10}$$

where J_I is the pixel-image Jacobian. In (4.7), $\dot{r} = V_c$ is also called the end effector velocity screw in camera frame. This velocity screw is defined in the camera frame, and should be mapped onto the robot control frame. Denoting V_R as the end effector velocity screw in robot base frame, the mapping can be written as

$$V_c = TV_R \tag{4.11}$$

The robot-to-camera velocity transformation matrix $T \in \Re^{6x6}$ is defined as below

$$T = \begin{bmatrix} R & 0_3 \\ 0_3 & R \end{bmatrix} \tag{4.12}$$

where R is the rotation matrix that map camera frame onto robot base frame. In light of equation (4.12), (4.7) can be rewritten as,

$$\dot{s} = \underbrace{J_I T}_{\triangleq \bar{J}_I} V_R = \bar{J}_I V_R \tag{4.13}$$

The new image Jacobian matrix $\bar{J}_I$ defines the relation between the changes of image features and end effector velocity in robot control frame. Considering p point features e.g. $s = \begin{bmatrix} x_1 & y_1 & ... & x_p & y_p \end{bmatrix}^T$, the Jacobian matrices corresponding to each point should be stacked as below.

$$\bar{J}_I = \begin{bmatrix} \bar{J}_I^1 \\ . \\ . \\ . \\ \bar{J}_I^p \end{bmatrix} \tag{4.14}$$

Let s^* be the constant reference feature vector and $e = s - s^*$ define the error. The visual servoing problem is designing an end-effector velocity screw V_R in such a way that the error decays to zero, i.e. $e \longrightarrow 0$. By imposing $\dot{e} = -\Lambda e$, where Λ is a positive definite gain matrix, an exponential decrease of the error function is realized. Consequently, if a diagonal gain matrix is used, the velocity screw is derived as:

$$V_R = -\Lambda \bar{J}_I{}^{\dagger}(s - s^*) \tag{4.15}$$

where $\bar{J}_I{}^{\dagger}$ is the pseudo-inverse of the image Jacobian and V_R is given as:

$$V_R = \begin{bmatrix} V_x & V_y & V_z & \omega_x & \omega_y & \omega_z \end{bmatrix}^T.$$

4.3 Simulation Results

Proposed method is simulated on a six DOF Puma 560 robot in eye-in-hand configuration as shown in Fig. 4.1.

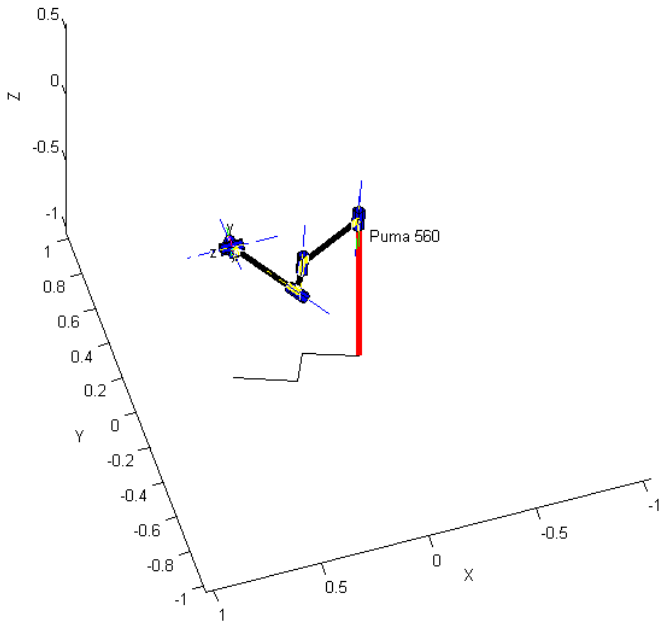

Figure 4.1: Puma 560 robot in Matlab Robotic Toolbox.

In simulations, Matlab Robotics Toolbox [72] is used. A planar object is initialized in the field of view of the camera. To evaluate the performance of the method in applications that require six DOF motion, a combination of translations and rotations in x, y and z directions are introduced between reference and initial positions. Homogeneous transformation between the

reference and initial poses are given in the world coordinate frame as

$$H = \begin{bmatrix} 0.975 & -0.037 & 0.218 & 0.1 \\ 0.097 & 0.956 & -0.275 & 0.05 \\ -0.198 & 0.289 & 0.936 & -0.1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.16}$$

Camera is placed at the end effector of the robot and it is assumed that origin and axis of the camera frame coincide with the end effector frame. Pinhole camera model is used to model the camera with the following intrinsic parameters matrix K,

$$K = \begin{bmatrix} 8000 & 0 & 320 \\ 0 & 8000 & 240 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.17}$$

Reference and initial position of the object boundary in image and the trajectories of the points which are extracted from the decomposition are given in Fig. 4.2. A diagonal gain matrix as given in (4.18) is used in the simulation.

$$\Lambda = \begin{bmatrix} 0.3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 \end{bmatrix} \tag{4.18}$$

Control signals (V_R) and feature errors are presented in Figures 4.3 and 4.4 respectively. An exponential decrease in errors on x coordinates is achieved in these simulations but error on y coordinates do not converge exponentially

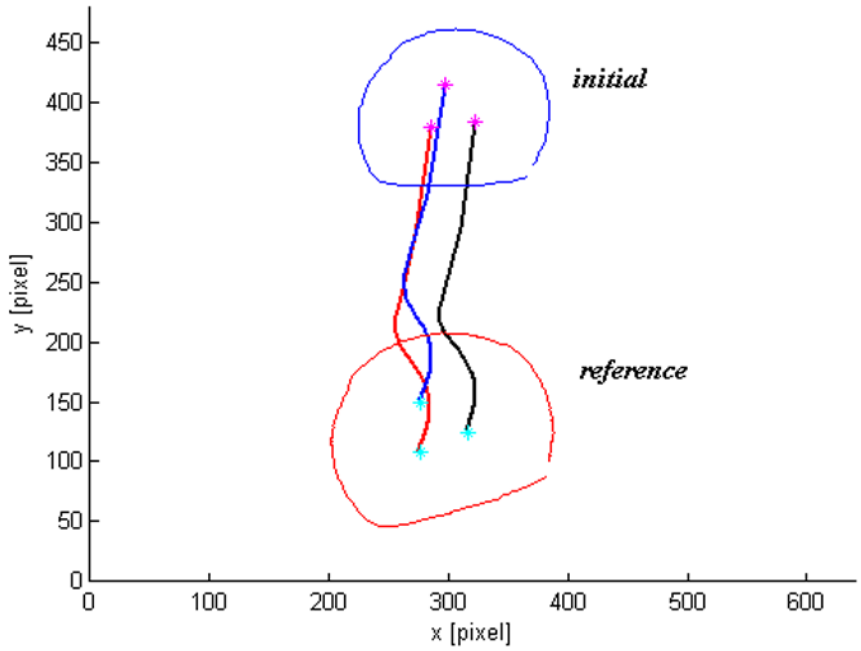

Figure 4.2: Induced trajectories of feature points from the initial to the reference view of object in image space

to zero but makes a zero crossing and then decays. This behavior is expected for image based visual servoing when points are used as features. Reason is the couplings that can be seen in the interaction matrix given in (4.8). As it can be seen in the results of the simulation, proposed method successfully position the end effector and errors on pixel coordinates of intersection points converges to zero in a short period of time.

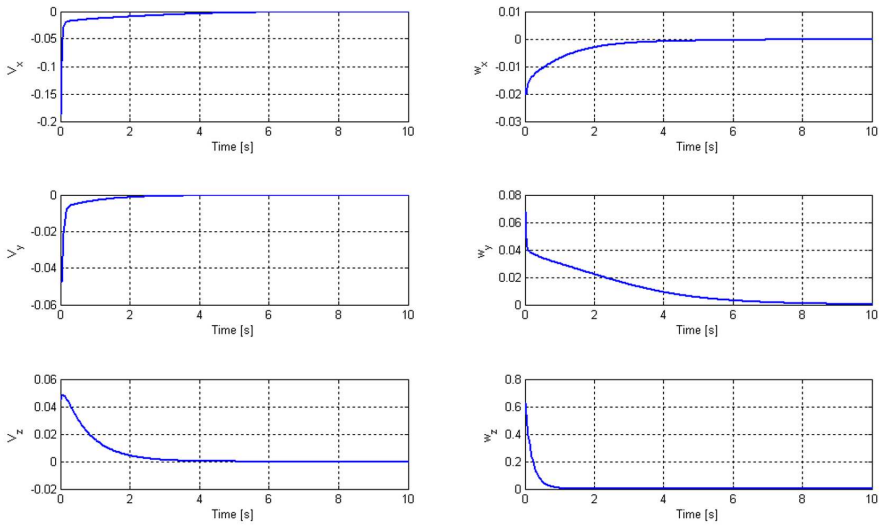

Figure 4.3: Control signals.

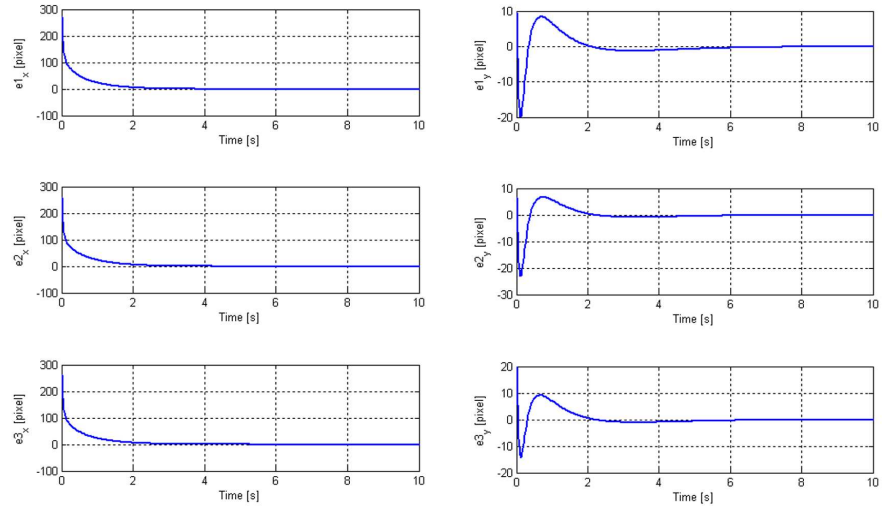

Figure 4.4: Errors on x and y coordinates of points.

Chapter 5

Experimental Results

5.1 Motion Estimation Experiments

Motion estimation experiments are conducted by using a Fire-i400 digital camera and an arbitrary shape which is printed on a planar board. The curve is moved randomly and object is tracked via ESM [73] algorithm. From the tracked boundary data motion of the curve is estimated with the proposed algorithm. Visual algorithms are implemented in Visual C++. To evaluate the performance of the algorithm by comparing with a ground truth, at each iteration extracted motion parameters are applied to the boundary data and resulting coordinates are marked with blue.

Algorithm starts with a user input where user sets the upper left and lower right corner for a window around target object. This window is used for ESM tracking algorithm. At each iteration motion parameters are estimated using the method presented in motion estimation section. At each iteration

starting from the initial image, edge data is stored and updated as

$$\begin{bmatrix} x[k] \\ y[k] \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} \widehat{a_{11}}[k] & \widehat{a_{12}}[k] & \widehat{b_1}[k] \\ \widehat{a_{21}}[k] & \widehat{a_{22}}[k] & \widehat{b_2}[k] \\ 0 & 0 & 0 \end{bmatrix}}_{\widehat{A}} \begin{bmatrix} x[k-1] \\ y[k-1] \\ 1 \end{bmatrix} + \begin{bmatrix} x[k-1] \\ y[k-1] \\ 1 \end{bmatrix} \tag{5.1}$$

At each frame updated edge data is marked on the acquired image in blue. Match of this data with the boundary of the curve displays the performance of the algorithm. We present 2 experiments in this section. In these experiments we want to present the performance of the algorithm and exper-

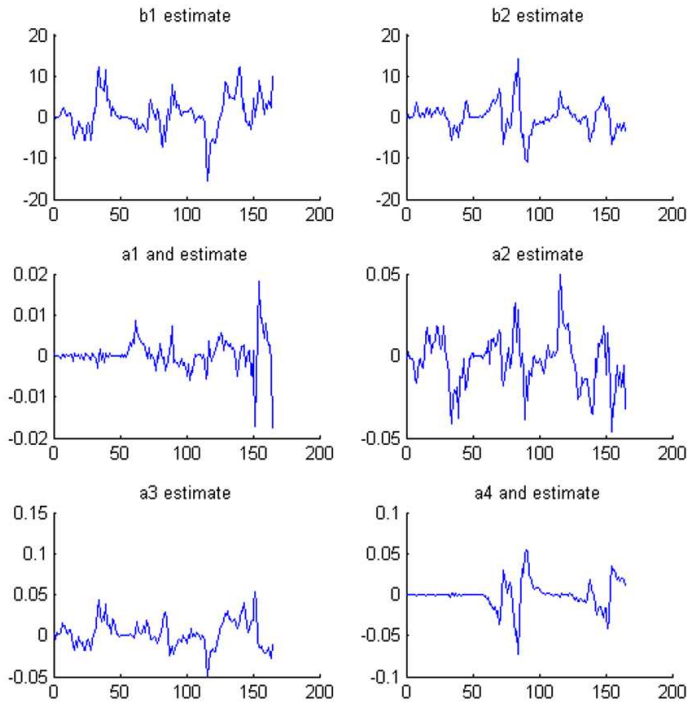

Figure 5.1: Estimated motion parameters for the first experiment.

imentally support the importance of data normalization in the algorithm. Camera is fixed above and planar object is moved in its field of view. In the first experiment motion parameters are estimated through normalized data. Video recorded for the first experiment has 165 frames taken at 25 fps. Estimated parameters for the first experiment are shown in Fig. 5.1 whereas some frames from the recorded video are shown in Fig. 5.2. In the second

Figure 5.2: Performance of motion estimation algorithm by using normalization. Frames 1, 30, 60, 90, 120, 150 are presented.

experiment same configuration is used. Object is randomly moved in the field of view of the camera. Normalization is only used for IP fitting but it is not used in the motion estimation. Video recorded for the second experiment has 128 frames taken at 25 fps. Estimated parameters for the second experiment are shown in Fig. 5.3 whereas some frames from the recorded video are shown in Fig.5.4.

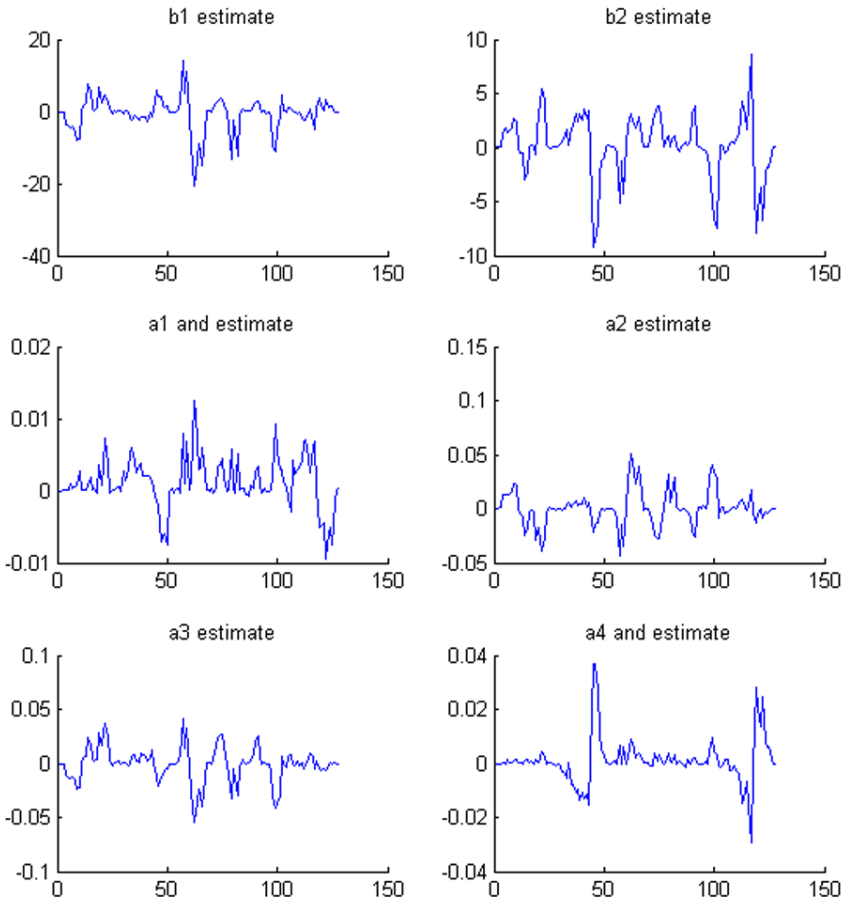

Figure 5.3: Estimated motion parameters for the second experiment.

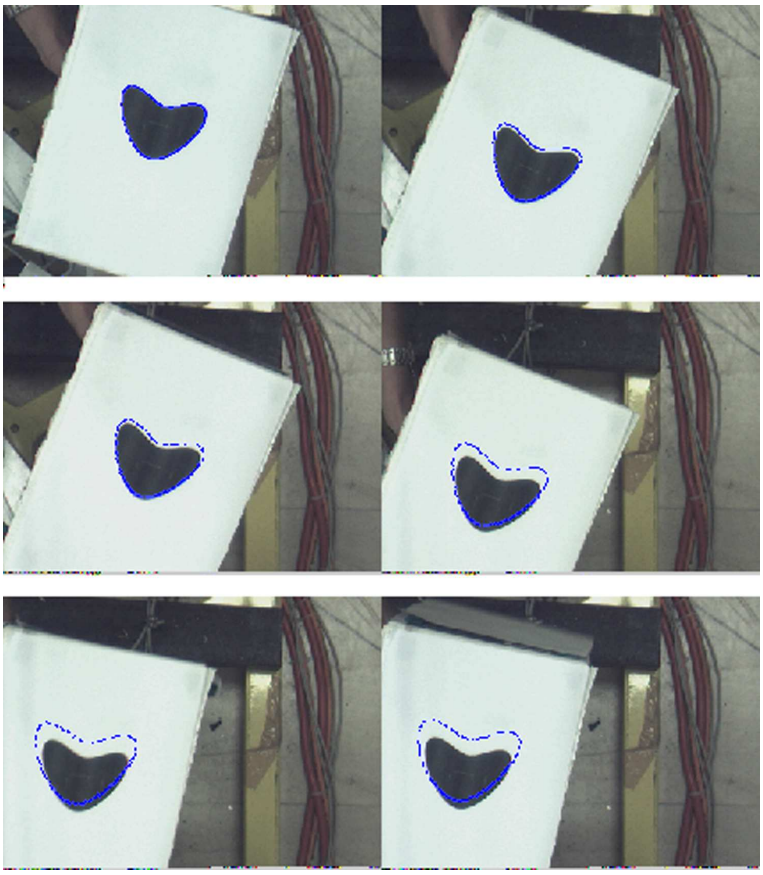

Figure 5.4: Performance of motion estimation algorithm without using normalization. Frames 1, 25, 50, 75, 100, 125 are presented.

5.2 Visual Servoing Experiments

The experimental verification of the proposed visual servoing methods are presented in this section. The experiments are conducted with a 2 DOF direct drive SCARA robot and a Fire-i400 digital camera in an eye to hand configuration. A planar free-form object is placed on the tool tip of the robot and camera is placed and fixed above the robot as it can be seen in

Figure 5.5. The robot is controlled with a dSPACE 1102 controller card. The programming language of the card is Visual C.

Figure 5.5: Experimental Setup

The control loop is made up of one inner and one outer loops. The outer loop is run via vision system. It uses the extracted features to generate velocity references to the inner loop. These references are used by the inner loop to position the robot. Sampling time of the inner control loop is 1 ms. The frame rate of the camera is 30 fps.

In the experiments, object boundary is extracted by using Canny edge

detection algorithm. From these edges, we obtain a fourth degree implicit polynomial by using the regularized 3L fitting algorithm. The implicit curve is then decomposed into line factors.

For point feature method, two point features are obtained from the intersection of the first 4 complex-conjugate lines. These points are then used as point features in visual servoing. A diagonal gain matrix of

$$\Lambda = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix} \tag{5.2}$$

is used in computing the velocity screw of the end effector. According to calibration results, effective focal lengths of the camera in x and y directions are measured as $f_x = f_y = 970$, and image center coordinates $(x_c, y_c) = (160, 120)$.

Two experiments are presented for point based approach. In the first experiment object plane is parallel to the image plane and motion of the end effector results in Euclidean transformation for the object boundary. Significant rotation and translation exist between the reference and initial poses. The reference and initial positions are as in Figure 5.6. Trajectories of the point features can be seen in Figure 5.7. The error plots are given in Figures 5.8 and 5.9. Control signals are presented in Figure 5.15.

In the second experiment, the case when the image plane is not parallel to the object plane is examined. In this case motion of the end effector induce affine motion on the object boundary data. Significant translation and rotation are introduced between reference and initial pose. Reference and initial poses can be seen in Figures 5.11 and 5.12 respectively. Pixel errors in x direction, pixel errors in y direction and control efforts can be

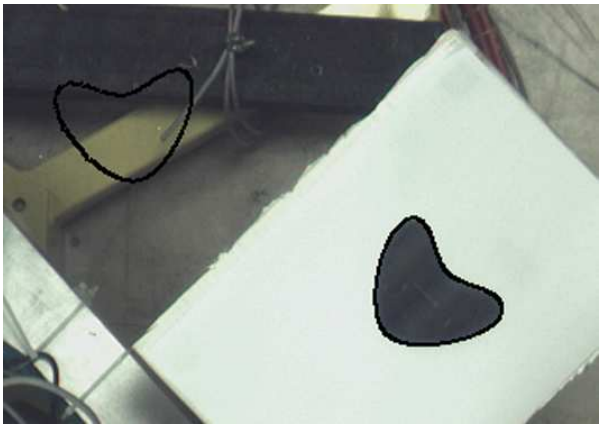

Figure 5.6: Reference and initial positions

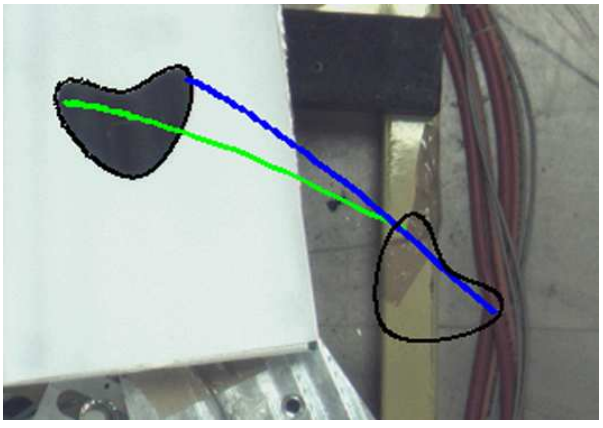

Figure 5.7: Trajectory of point features

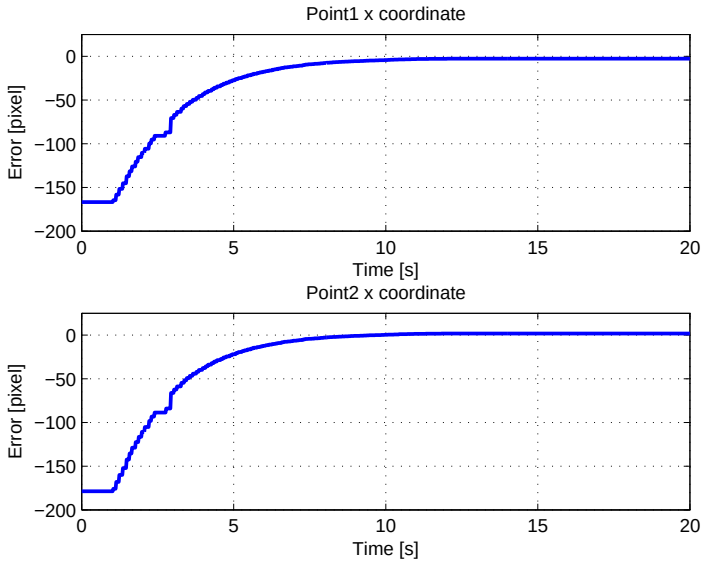

Figure 5.8: Pixel errors in x direction of the image plane

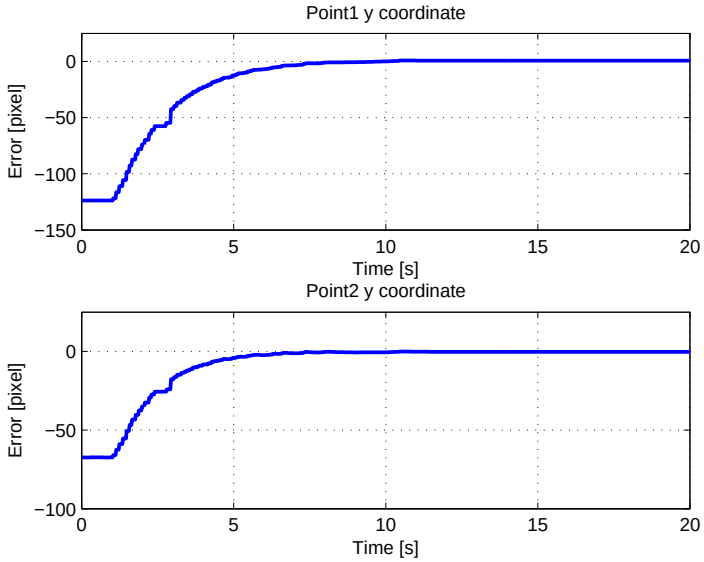

Figure 5.9: Pixel errors in y direction of the image plane

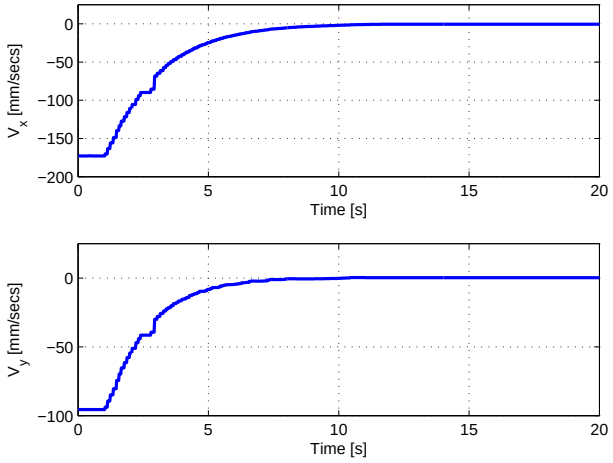

Figure 5.10: Control efforts

seen in Figures 5.13, 5.14 and 5.15.

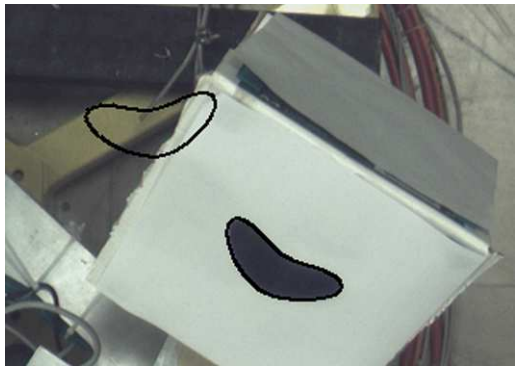

Figure 5.11: Reference and initial positions

With the proposed control method errors decrease exponentially and a proper position of the robot is achieved in both experiments. Note that in these experiments we have only 2 DOF and resulting part of the image Jacobian is decoupled. Dues to this fact, we observe a decoupled exponential

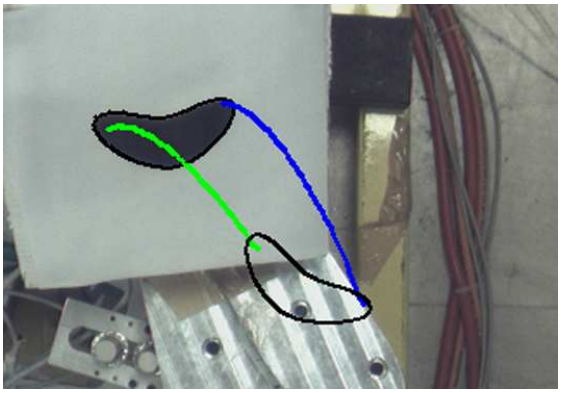

Figure 5.12: Trajectories of point features

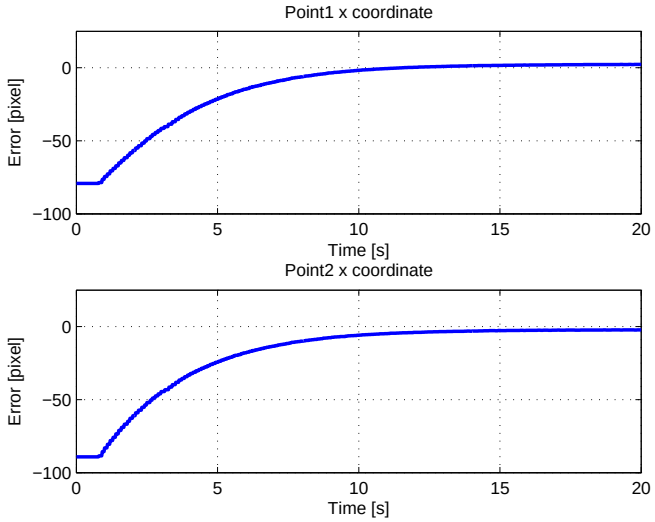

Figure 5.13: Pixel errors in x direction of the image plane

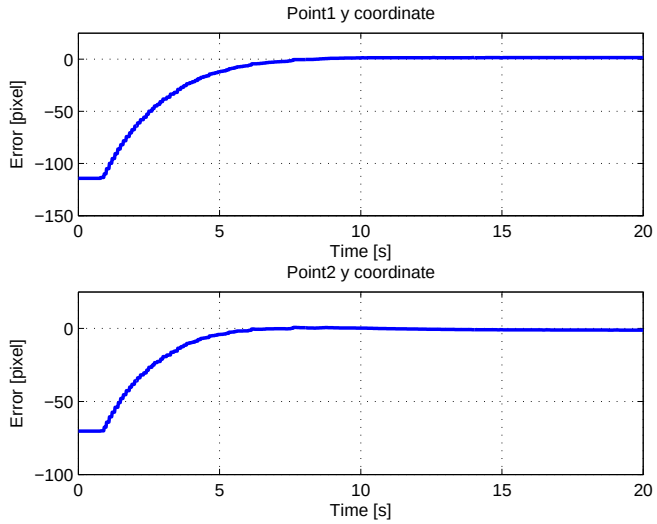

Figure 5.14: Pixel errors in y direction of the image plane

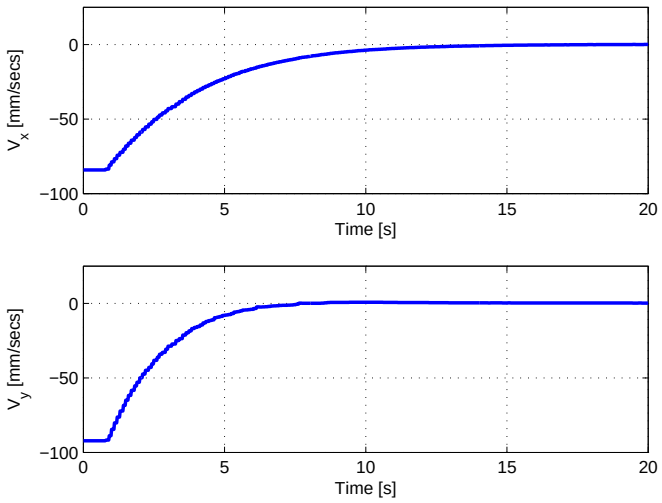

Figure 5.15: Control efforts

decrease in the errors. For a general 6 DOF application such a decoupled decrease may not be obtained but the convergence to a very small (possibly zero) steady state error will be observed. In the experiments less than 2 pixel steady state error on x and y coordinates of extracted points exist in the final pose. These results display the success of the algorithm. It is experimentally verified that as long as the IP fitting invariance is not disturbed, intersection of complex conjugate line pairs can be treated as point features that are rigidly attached to target object and can be used in visual servoing.

Chapter 6

Conclusion and Future Works

6.1 Conclusion

Two novel methods that use implicit representation of planar algebraic curves in motion estimation and visual servoing were proposed in this thesis. These methods are based on the line decomposition of implicit polynomials [4, 6]. For the motion estimation algorithm, parameters of the complex line factors are used as features. It is shown that in the case of time varying affine motion these lines satisfy the Riccati equations that were previously obtained for time invariant motion parameters in [24, 25]. An estimation algorithm is proposed and verified by simulations and experiments. It is shown that normalization is important in the stability of this algorithm and extraction of motion parameters from the motion parameters of normalized data is proposed for this purpose. Experimental results support that this approach increases the performance of the algorithm significantly.

Same decomposition is used to extract features for visual servoing. Object

boundaries are modeled with implicit polynomials of even degree, $n = 2r$. As the first n lines in this decomposition are complex conjugate pairs their pairwise intersections give rise to $n/2$ real points on image plane. We propose that these intersections can be used as point features that are extracted from the implicit representation of the planar algebraic curve. Simulations with a 6 DOF Puma 560 and experimental results conducted on a 2 DOF SCARA robot are presented for the verification of this method.

6.2 Future Works

In this work we were interested in the motion estimation through the implicit representation of planar algebraic curves. Our simulations and experiments are quite promising. However, in this work we were tracking target object with ESM algorithm. As we have shown the promising performance of the motion estimation algorithm, slight improvements may provide a visual tracking method that uses the implicit model of target object boundary.

Visual servoing method presented in this paper is one option for using implicit form of closed curves in these applications. Other than using the intersection of complex conjugate line factors, we believe that the parameters of those lines can also be used in visual servoing. This can be achieved by deriving the analytical image Jacobian corresponding to the parameters of extracted line factors. In our future research we are planing to expand our work in that way.

Bibliography

[1] C.G. Gibson, **Elementary geometry of algebraic curves**, Cambridge University Press, Cambridge, UK, 1998.

[2] J. Bloomenthal, "Introduction to implicit surfaces," Kaufmann, Los Altos, CA, 1997.

[3] D. Keren, C. Gotsman, "Fitting curves and surfaces to data using constrained implicit polynomials," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 23, No. 1, January 1999.

[4] M. Unel, W. A. Wolovich, "On the construction of complete sets of geometric invariants for algebraic curves," Advances in Applied Mathematics Vol. 24, No. 1, pp. 65-87, January 2000.

[5] M. Unel, W. A. Wolovich, "A new representation for quartic curves and complete sets of geometric invariants," International Journal of Pattern Recognition and Artificial Intelligence, December 1999.

[6] M. Unel, "Polynomial decompositions for shape modeling, object recognition and alignment," PhD Thesis, Brown University, Providence, 1999.

[7] J. L. Mundy, Andrew Zisserman, **Geometric invariance in computer vision**, The MIT Press, 1992.

[8] G. Taubin, D. B. Cooper, "2D and 3D object recognition and positioning with algebraic invariants and covariants," Chapter 6 of **Symbolic and Numerical Computation for Artificial Intelligence**, Academic Press, 1992.

[9] G. Taubin, F. Cukierman, S. Sullivan, J. Ponce and D.J. Kriegman, "Parameterized families of polynomials for bounded algebraic curve and surface fitting," IEEE PAMI, March, 1994.

[10] W. A. Wolovich, Mustafa Unel, "The determination of implicit polynomial canonical curves," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 20, No. 10, pp. 1080-1089, October 1998.

[11] W. A. Wolovich, Mustafa Unel, "Vision based system identification and state estimation," The Confluence of Vision and Control, Springer Lecture Notes in Control and Information Sciences, No. 237, pp. 171-182, 1998.

[12] J. Shi and C. Tomasi, "Good Features to Track", CVPR 1994, pp 593-600.

[13] R.E Kalman, "A new approach to linear filtering and prediction problems", Transactions of the ASME, Ser. D., Journal of Basic Engineering, 82, pp. 34-45, 1960.

[14] E. Cuevas, D. Zaldivar, R. Rojas, "Kalman filter for vision tracking" (technical report), August 2005.

[15] J. M. Jou, P. Y. Chen, and J. M. Sun, "The gray prediction search algorithm for block motion estimation," IEEE Trans. Circuits Syst. Video Technol., vol. 9, no. 6, pp. 843848, Sep. 1999.

[16] V. G. Moshnyaga, "A new computationally adaptive formulation of block-matching motion estimation," IEEE Trans. Circuits Syst. Video Technol., vol. 11, no. 1, pp. 118124, Jan. 2001.

[17] M. Mattavelli and G. Zoia, "Vector-tracing algorithm for motion estimation in large search windows," IEEE Trans. Circuits Syst. Video Technol., vol. 10, no. 12, pp. 14261437, Dec. 2000.

[18] C. J. Kuo, C. H. Yeh, and S. F. Odeh, "Polynomial search algorithm for motion estimation," IEEE Trans. Circuits Syst. Video Technol., vol. 10, no. 5, pp. 813818, Aug. 2000.

[19] P. H. S. Torr, A. Zisserman, "Feature Based Methods for Structure and Motion Estimation", ICCV Workshop on Vision Algorithms, pp. 278-294, 1999.

[20] M. Irani, P. Anandan, "About Direct Methods", ICCV Workshop on Vision Algorithms, pp. 267-277, 1999.

[21] H. Foroosh, J. Zerubia and M. Berthod, "Extension of phase correlation to sub-pixel registration" , IEEE Trans. Image Processing, vol. 11, no. 3, pp. 188-200, 2002.

[22] F. Mindru, T. Moons, L. Van Gool, "Color-based moment invariants for the viewpoint and illumination independent recognition of planar color patterns", ICAPR'98, pp. 113-122, 1998.

[23] D. Lowe, "Distinctive image features from scale-invariant key points", International Journal of Computer Vision, vol. 60, no. 2, pp. 91-110, 2004.

[24] M. Unel, B. K. Ghosh, "Dynamic Models of Planar Algebraic Curves", Proceedings of IEEE CDC 2001, vol. 2, pp. 1304-1309, 2001.

[25] M. Unel, B. K. Ghosh, "Motion Estimation of Plane Polynomial Curves", Proceedings of ACC 2005, pp. 1289-1294, 2005.

[26] A. Mitiche, S. Seida and J. K. Aggarwal, "Line based computation of structure and motion using angular invariance," Proc. Workshop on Motion: Representation and Analysis (IEEE Computer Society Press, Silver Spring, MD, 1986).

[27] Y. Liu and T. S. Huang, "A linear algorithm for motion estimation using straight line correspondences," Comput. Vision Graphics Image Process, 44 (1988).

[28] M. E. Spetsakis and J. Aloimonos, "Structure from motion using line correspondences," Internat. J. Comput. Vision, 4 (1990), pp. 171-183.

[29] M. Jankovic and B. K. Ghosh, "Visually guided ranging from observations of points, lines and curves via an identifier based nonlinear observer, Systems and Control Letters, 25, pp. 63-73, 1995.

[30] N. Andreff, B. Espiau and R. Horaud, "Visual servoing from lines," In Proc. of the 2000 IEEE International Conference on Robotics and Automation, pp. 2070-2075, San Francisco, CA, April 2000.

[31] R. Cipolla and A. Blake, "Surface shape from the deformation of apparent contours," Internat. J. Comput. Vision, vol. 9, no. 2, 1992, pp. 83-112.

[32] O. D. Faugeras, "On the motion of 3-D curves and its relationship to optical flow," in: O.D.Faugeras, ed., Proc. 1st ECCV (Springer, Berlin, 1990), pp. 107-117.

[33] O. D. Faugeras and T. Papadopoulo, "A theory of the motion fields of curves," Internat. J. Comput. Vision, vol. 10, no. 2, pp. 125-156, 1993.

[34] M. M. Blane, Z. Lei, H. Civi and D. B. Cooper, "The 3L Algorithm for Fitting Implicit Polynomial Curves and Surfaces to Data", *IEEE Trans. on Pattern Anlaysis and Machine Intelligence*, Vol. 22, No. 3, pp. 298-313, 2000.

[35] B. K. Ghosh, H. Inaba and S. Takahashi, "Identification of Riccati dynamics under perspective and orthographic observations," IEEE Trans. on Aut. Contr., vol. 45, no. 7, July 2000, pp. 1267-1278.

[36] R. Malladi, J. A. Sethian, and B. C. Vemuri, "Shape Modeling with Front Propagation: A Level Set Approach," IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 17, pp. 158-175, 1995.

[37] K. Fukunaga, **Introduction to Statistical Pattern Recognition**, 2nd ed. New York: Academic Press, 1990.

[38] R. I. Hartley, "In Defense of the Eight-Point Algorithm,". IEEE Trans. Pattern Analaysis and Machine Intelligence, vol. 19, no. 6, pp. 580-593, 1997.

[39] T. Sahin, M. Unel, "Globally Stabilized 3L Curve Fitting," Lecture Notes in Computer Science (LNCS-3211), pp. 495-502, Springer-Verlag, 2004

[40] T. Sahin, M. Unel, "Stable Algebraic Surfaces for 3D Object Representation" Journal of Mathermatical Imaging and Vision, vol. 32, pp. 127-137, 2008

[41] B. Yondem, M. Unel and A. Ercil, "2D shape tracking using algebraic curve spaces", Lecture Notes in Computer Science, vol.3733, pp. 698-707, 2005

[42] J. Canny, "A Computational Approach To Edge Detection," IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 8, pp. 679-714, 1986.

[43] J. Prewitt, "Object enhancement and extraction," Picture Processing and Psychopictorics, B. Lipkin and A, Rosenfeld, Eds. , pp. 75-149, New York: Academic, 1979.

[44] X. Han, C. Xu, and J. L. Prince, "A Topology Preserving Level Set Method for Gemetric Deformable Models," IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol 25, No 6, pp. 755-768, June 2003.

[45] J. H. Wilkinson, "The perfidious polynomial," Studies in Numerical Analysis, ed. by G. H. Golub, pp. 128, Washington, D.C.: Mathematical Association of America, 1984.

[46] A. C. Sanderson, L. E. Weiss, Image Based Visual Servo Control Using Relational Graph Error Signals, Proc. of IEEE International Conference on Automation and Robotics, pp. 1074-1077, 1980.

[47] W. Wilson, C. Hulls, and G. Bell, Relative End-Effector Control Using Cartesian Position Based Visual Servoing, IEEE Trans. on Robotics, Vol. 12, pp. 684-696, 1996.

[48] E. Malis, F. Chaumette, S. Boudet, 2 1/2 D Visual Servoing, IEEE Trans. on Robotics and Automation, Vol. 15, pp. 238-250, 1999.

[49] G. D. Hager, "A modular system for robust positioning using feedback from stereo vision", *IEEE Trans. on Robotics and Automation*, Vol. 13, No. 4, pp. 582-595, 1997.

[50] S. Hutchinson, G. D. Hager and P. I. Corke, "A tutorial on visual servo control", *IEEE Trans. on Robotics and Automation*, Vol. 12, No. 5, pp. 651-670, 1996.

[51] F. Chaumette, S. Hutchinson, "Visual Servo Control, Part I: Basic Approaches and Part II: Advanced Approaches", *IEEE Robotics and Automation Magazine*, Vol. 13, No. 4, pp. 82-90, 2006.

[52] B. Espiau, F. Chaumette, P. Rives, "A New Approach to Visual Servoing in Robotics", *IEEE Trans. on Robotics and Automation*, Vol. 8, No. 3, pp. 313-326, 1992.

[53] O. Tahri, F. Chaumette, "Point Based and Region Based Image Moments for Visual Servoing of Planar Objects", *IEEE Trans. on Robotics and Automation*, Vol. 21, No. 6, pp. 1116-1127, 2005.

[54] A. Y. Yazicioglu, B. Calli, M. Unel, "Image Based Visual Servoing Using Algebraic Curves Applied to Shape Alignment", will appear in Proc. of IEEE/RJS International Conference on Intelligent Robots and Systems, 2009.

[55] D. L. Pham, C. Xu, J. L. Prince, "Current Methods in Medical Image Segmentation", Annual Review of Biomedical Engineering, vol. 2, pp 315-337, 2000.

[56] M. Pathegama, Ö. Göl, "Edge-end pixel extraction for edge-based image segmentation", Transactions on Engineering, Computing and Technology, vol. 2, pp 213-216, ISSN 1305-5313, 2004.

[57] S. Osher, N. Paragios, "Geometric Level Set Methods in Imaging Vision and Graphics", Springer Verlag, ISBN 0387954880, 2003.

[58] J. Shi, J. Malik, "Normalized Cuts and Image Segmentation", IEEE Conference on Computer Vision and Pattern Recognition, pp. 731-737, 1997.

[59] G. Szekely and G. Gerig, "Model-based Segmentation of Radiological Images", Kunstliche Intelligenz, (3): pp. 1823, 2000.

[60] C. Xu and J. Prince, "Snakes, shapes and gradient vector flow", IEEE Transactions on Image Processing, 7(3), pp. 359-369, 1998.

[61] M. Kass, A. Witkin, and D. Terzopoulos,"Snakes: active contour models, Int. Journal of Computer Vision, vol. 1, pp. 321-331, 1987.

[62] D. Mumford and J. Shah, "Optimal approximations by piecewise smooth functions and associated variational problems, Comm. Pure and Appl. Math., vol. 42, 1989.

[63] Emanuele Trucco, Alessandro Verri, **Introductory Techniques for 3-D Computer Vision**, Prentice Hall, 1998.

[64] S. Osher, J. A. Sethian, "Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations", Journal of Comput. Phys., vol. 79, pp. 1249, 1988.

[65] S. Osher, J. Fedkiw, P. Ronald, "Level Set Methods and Dynamic Implicit Surfaces", Springer-Verlag. ISBN 0-387-95482-1,2002.

[66] J. A. Sethian, A. James, "Level Set Methods and Fast Marching Methods : Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science", Cambridge University Press. ISBN 0-521-64557-3, 1999.

[67] D. Enright, R. P. Fedkiw, J. H. Ferziger, "A hybrid particle level set method for improved interface capturing", Journal of Comput. Phys., vol.183, pp. 83116, 2002.

[68] M. Pilu, A. Fitzgibbon and R. Fisher, "Ellipse Specific Direct Least Squares Fitting," Proc. IEEE, International Conference on Image Processing, Lausanne, Switzerland, September 1996.

[69] G. Taubin, "Parametrized Families of Polynomials fo Bounded Algebraic Curve and Surface Fitting," IEEE Transactions on Pattem Analysis and Machine Vision, 16(3):287-303, March 1994.

[70] D. Keren, D. Cooper and J. Subrahmonia, "Describing Complicated Objects by Implicit Polynomials," IEEE Transactions on Pattem Analysis and Machme Vision, vol. 16, pp. 38-53, 1994.

[71] B. Zheng, R. Ishikawa, T. Oishi, J. Takamatsu, K. Ikeuchi, "6-DOF pose estimation from single Ultrasound image using 3D IP models", CVPR Workshops 2008, pp. 1 - 8, 2008.

[72] P. I. Corke, "A Robotics Toolbox for MATLAB," IEEE Robotics and Automation Magazine, Vol.3, No.1, pp. 24-32, 1996.

[73] S. Benhimane, E. Malis, "Real-time image-based tracking of planes using efficient second-order minimization", IEEE/RSJ International Conference on Intelligent Robots Systems, pp. 943948, 2004.