DIFFERENT APPROACHES ON THE IMPLEMENTATION OF IMPLICIT POLYNOMIALS IN VISUAL TRACKING

by

Kemal Burak Yöndem

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

Sabancı University
2005

DIFFERENT APPROACHES ON THE IMPLEMENTATION OF IMPLICIT POLYNOMIALS IN VISUAL TRACKING

APPROVED BY:

Prof. Dr. Aytül Erçil …….

(Thesis Supervisor)

Assoc. Prof. Dr. Mustafa Ünel …….

(Thesis Co-supervisor)

Prof. Dr. Lale Akarun …….

Assist. Prof. Dr. Hakan Erdoğan …….

Assist. Prof. Dr. Hasan Ateş …….

ABSTRACT

Visual tracking has emerged as an important component of systems in several application areas including vision-based control, human-computer interfaces, surveillance, agricultural automation, medical imaging and visual reconstruction. The central challenge in visual tracking is to keep track of the pose and location of one or more objects through a sequence of frames.

Implicit algebraic 2D curves and 3D surfaces are among the most powerful representations and have proven very useful in many model-based applications in the past two decades. With this approach, objects in 2D images are described by their silhouettes and then represented by 2D implicit polynomial curves.

In our work, we tried different approaches in order to efficiently apply the powerful implicit algebraic 2D curve representation to the phenomenon of visual tracking. Through the proposed concepts and algorithms, we tried to reduce the computational burden of fitting algorithms. Besides showing the usage of this representation on boundary data simulations, use of the implicit polynomial as a representative of the target region is also experimented on real videos.

ÖZET

Görsel izleme, görmeye dayalı kontrol, insan-makina arayüzü, gözetleme, tarımsal otomasyon, medikal imgeleme gibi birçok uygulama alanındaki sistemlerin önemli bir bileşeni olmaya başlamıştır. Görsel izlemedeki temel sorun çerçeve dizisi boyunca bir veya birden çok nesnenin duruşunu ve yerini izleyebilmektir.

Örtük cebirsel 2 boyutlu eğriler en güçlü şekil temsil yöntemleri arasındadır ve model tabanlı uygulamalardaki faydaları son yirmi yıllık süreçte kanıtlanmıştır. Bu yaklaşımla 2 boyutlu imgelerdeki nesneler silüetleriyle tanımlanıp 2 boyutlu örtük polinom eğrileriyle temsil edilirler.

Bu çalışmada, güçlü 2 boyutlu örtük cebirsel eğriler yöntemini görsel izleme olgusu içerisinde etkin uygulamaya çalışan farklı yaklaşımlar denenmişitr. Önerilen kavramlar ve algoritmalar yoluyla, eğri uydurma algoritmalarının hesaplama karmaşıklığı azaltılmaya çalışılmıştır. Bu yöntemin kullanımı sınır veri benzetimleriyle gösterilmiş ve örtük cebirsel eğrinin hedef bölgenin tanımlanmasında kullanıldığı gerçek video deneyleri de gerçeklenmiştir.

ACKNOWLEDGEMENTS

I would like to gratefully acknowledge my advisor Prof. Aytül ERÇİL for her valuable guidance, encouragement and understanding throughout my studies at Sabancı University. I am also very grateful to my co-advisor Assoc. Prof. Dr. Mustafa ÜNEL for his priceless efforts, inspiring ideas and his unconditional support during the study.

It was a great pleasure and experience for me working with all the members of VPA Laboratory at Sabancı University. I sincerely thank my colleagues for their companionship, cooperation and many fruitful interactions. I also would like to thank all my friends for their support during all my studies.

I am grateful deeply to my family for their endless love and patience that made everything about me possible.

And this thesis is especially devoted to Özgün Doğan my beloved friend. He always gave me joy even he felt himself weary. I am sure he is watching me from heaven now...

TABLE OF CONTENTS

LIST OF FIGURES

1 INTRODUCTION

Implicit algebraic curves have proven very useful in many model-based applications in the past two decades. Implicit models have been widely used for important computer vision tasks such as single computation pose estimation, shape tracking, 3D surface estimation and indexing into large pictorial databases [1-6]. These models become important especially when the objects to be modelled are free-form. When such objects are in motion, the lack of specific features, such as points or lines, that can be identified easily at different times and in different locations can prevent accurate estimations of the rotational and translational velocities of the object. In such cases, sets of boundary data points can be used to construct "implicit polynomial" models of the object in any given position. Such models will then imply non-visual points that can be used for tracking purposes.

Visual tracking can be described as the process of determining the location of a feature in an image sequence over time. Examples include tracking cars in an intersection via a traffic camera, or tracking the head of a computer user with a web-cam. Another possible application is tracking multiple small features of interest, such as corners of an object, in attempt to determine its 3-dimensional geometry.

The target to be tracked might be a complete object (e.g. a person) or a small area on an object (e.g. a corner). In either case, the feature of interest is typically contained within a target region. Ideally, a tracking algorithm would be able to locate the object anywhere within the image at any point in time. However typically only a limited region of the image is searched (usually the same size as the target region). Reasons for this are efficiency (especially necessary for real-time applications) and the fact that there might be many other similar-looking objects in the image.

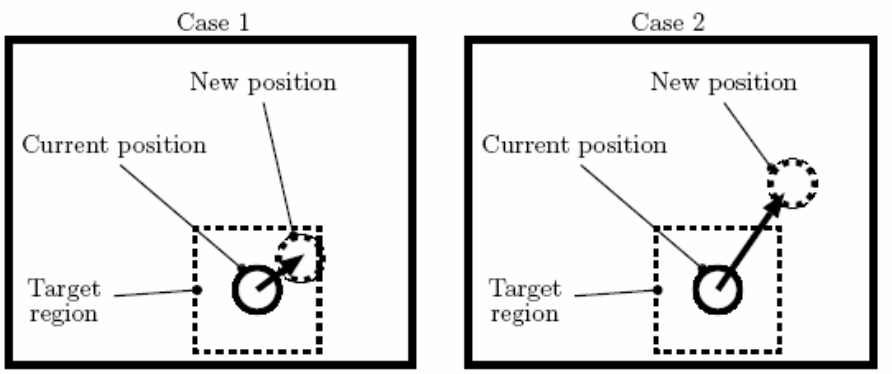


Figure 1-1: Case : Tracking the object without position prediction might be successful, Case 2: Tracking without position prediction will fail

The intuitive approach is to search within a region centered around the last position of the object. But as Figure 1 illustrates, this approach will fail if the object moves outside the target range. There are many possible reasons why the object might not stay within this region:

- The object is moving too fast
- The frame rate is too low
- The searched region is too small

These problems are related to each other and could be avoided by ensuring a high enough frame rate for example. But given other constraints, these problems are often unavoidable.

In addition, even when the target can be located, it seldomly appears the same in all images. The appearance of the same target is continuously affected by changes in orientation, lighting, occlusions, and imperfections in the camera. So essentially, the true location of the target is very difficult to observe accurately under the usual circumstances. In summary, two major problems have been identified:

1. The object can only be tracked if it does not move beyond the searched region.
2. Various factors such as lighting and occlusions can affect the appearance of the target, thus making accurate tracking difficult.

The initial goal of this thesis research was to apply the implicit algebraic 2D curve representation framework to the problem of visual tracking. On this road, we have proposed 3 different approaches, all of which are aiming to reduce the computational burden of fitting algorithms and find solutions to the problems of visual tracking stated above.

2 IMPLICIT POLYNOMIALS

2.1 Literature Overview

Implicit polynomials have found wide application areas in computer vision being among the most effective and leading shape representations for complex free-form object modeling and recognition. Although the underlying theory, ie. *algebraic geometry* has long been around, implicit polynomials could not find effective application areas until 1980's. Since then, independent research in a number of fields, including computer graphics, geometric modeling, and computer vision, has accumulated valuable insights into various properties of implicit polynomials important for solving practical problems. Research in geometric modeling brought to bear parametric to implicit conversion techniques which, in essence, is an elimination theory problem, and relative strengths of parametric and implicit representations were understood. Although computer graphic works proved that implicit polynomials can play an important role in visualization, their strengths, such as their interpolation property for handling missing data, smoothing property against noise and perturbations, Bayesian recognizers, and their algebraic invariants, made them much more suitable for computer vision, where the idea created an object representation and recognition paradigm.

2.2 Geometry of the 2D Plane

Many geometric terms will be used throughout the thesis, and this section is devoted to the explanation of these terms and their geometric interpretations. An

arbitrary coordinate system $\{O, X, Y\}$ is chosen, in order to investigate the properties of a geometric shape (object). A point in 2D space is defined as $r = (x, y)$. Another coordinate system $\{\overline{O}, \overline{X}, \overline{Y}\}$ is related to the original coordinate system by coordinate transformations. As coordinate system and object coordinates are related to each other, a change in objects coordinates can be counted as a coordinate transformation. These coordinate transformations consist of Euclidean, affine, and projective transformations.

At this point, we need to give more formal definitions of the Euclidean, affine, and projective transformations. An Euclidean transformation is a rigid motion of the coordinate frame. Thus it does not allow any scaling or skewing, and only rotation and translation are allowed. For rotation, we need only one parameter which is θ whereas for translation we need two translation parameters t_x and t_y along the x-axis and y-axis, respectively. Under Euclidean transformation the distance of two points and the angle between two lines are preserved. Equation 2.1 defines the Euclidean transformation.

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \overline{x} \\ \overline{y} \\ 1 \end{bmatrix} \qquad (2.1)$$

A slight generalization of the Euclidean transformation is the similarity transformation. Similarity transformations include a uniform scale factor λ, to the Euclidean transformation defined above. Equation 2.1 defines the similarity transformation.

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \lambda\cos\theta & -\lambda\sin\theta & t_x \\ \lambda\sin\theta & \lambda\cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \overline{x} \\ \overline{y} \\ 1 \end{bmatrix} \qquad (2.2)$$

An affine transformation has 6 parameters and is a more general transformation then the Euclidian transformation. An affine transformation does not

preserve angles; therefore the coordinate axes are not necessarily perpendicular. Parallelism and the ratio of the length of a pair of line segments are preserved under affine transformation. Equation 2.3 defines the affine transformation.

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & t_x \\ c & d & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \bar{x} \\ \bar{y} \\ 1 \end{bmatrix} \tag{2.3}$$

Here, it is assumed that the following matrix in Equation 2.3 is non-singular.

$$\begin{vmatrix} a & b \\ c & d \end{vmatrix} \neq 0 \tag{2.4}$$

The most general linear transformation is the projective transformation. The projective plane is constructed from the Cartesian plane by introducing homogeneous coordinates X', Y', Z' for the plane points.

$$x = X'/Z', \quad y = Y'/Z', \quad Z \neq 0$$

where x, y are the Cartesian coordinates of a point. This construction is augmented by adding points at infinity (improper points) for which at least one of X', Y' is non-zero and $Z' = 0$. The plane so obtained is called the projective plane. In this plane, proportional points (X', Y', Z') and (kX', kY', kZ') correspond to the same point, and points which are not proportional are different points. Equation 2.5 defines the projective transformation of the projective plane in three variables.

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & k \end{bmatrix} \begin{bmatrix} X'' \\ Y'' \\ Z'' \end{bmatrix} \tag{2.5}$$

because of the proportional point equivalence, projective transformations have 8 parameters instead of 9.

In modeling image plane correspondences, the coordinate transformations defined above are mostly used. Different camera geometry assumption result modeling the image plane correspondence with different transformations.

2.3 Implicit Polynomial Model

Implicit polynomial (IP) curves and surfaces are mathematical models for the representation of 2D curves and 3D surfaces. An implicit algebraic curve is defined as the zero set of an implicit polynomial in 2 variables x and y. A more formal representation of a 2D IP curve of degree n is illustrated below

$$\begin{aligned} f_n(x,y) &= \sum_{0\le i,j;i+j\le n} a_{ij}x^i y^j \\ &= \underbrace{a_{00}}_{H_0} + \underbrace{a_{10}x + a_{01}y}_{H_1(x,y)} + \underbrace{a_{20}x^2 + a_{11}xy + a_{02}y^2}_{H_2(x,y)} \\ &\quad \ldots + \underbrace{a_{n0}x^n + \ldots + a_{n-1,1}x^{n-1}y + \ldots + a_{0n}y^n}_{H_n(x,y)} \\ &= \sum_{r=0}^{n} H_r(x,y) = 0 \end{aligned} \tag{2.6}$$

Here $H_r(x,y)$ is *a homogenous binary (i.e., two variables) polynomial (form)* of degree *r* in *x* and *y*. An *nth* degree IP surface has *(n+1) (n+2)/2* coefficients. In vector notation the above equation can be represented as:

$$f_n(x,y) = Y^t A, \tag{2.7}$$

where

$$A = \begin{bmatrix} a_{00} & a_{10} & a_{01} & a_{20} & a_{11} & a_{02} & \ldots & a_{0n} \end{bmatrix}^t \tag{2.8}$$

and

$$Y = \begin{bmatrix} 1 & x & y & x^2 & xy & y^2 & x^3 & \ldots & x^n \ldots & xy^{n-1} & y^n \end{bmatrix}^t \tag{2.9}$$

Algebraic curves of degree 1, 2, 3, 4… are called lines, conics, cubics, quartics …etc. Quartics, have drawn much attention and their properties have been extensively studied.

Objects in 2D images are described by their silhouettes and, in the thesis, then represented by implicit polynomial curves. Given $\Gamma_0 = \{(x_m, y_m) | m = 1,...K\}$ a set of data points along an object boundary, an implicit polynomial is said to represent this object boundary if every point Γ_0 is sufficiently close to the zero set $Z_f = \{(x,y) | f(x,y) = 0\}$ of the implicit polynomial. Given such a data set Γ_0, an implicit polynomial representation is obtained by implementation of a fitting algorithm such as 3L fitting or linear programming as discussed in Section 2.1.

2.4 Implicit Polynomial Fitting

The general IP fitting problem can be set up as follows. Given a dataset $\Gamma_0 = \{p_m = (x_m, y_m) | m = 1,...K\}$, find the nth degree implicit polynomial $f_n(x,y)$ that minimizes the average squared distance from the data points to the zero set Z_f of the polynomial. When the geometric distance from a point to the zero set of an implicit polynomial is minimized, an iterative process is needed because there is no explicit expression for this distance. This formulation requires non-linear optimization. Several geometric distance approximations have been used, such as the first order approximation, which speeds up the computation considerably, but iterative non-linear optimization is still required.

The idea behind using implicit polynomials in vision is that boundaries of objects can be approximated as piecewise smooth surfaces and curves in three-dimensional and two-dimensional cases, respectively. The approximation amounts to estimation of polynomial coefficients, which minimize the mean square distance from the data points to the polynomial defined by those coefficients. Bounded and unbounded fitting differ in some respects.

Fitting an unbounded polynomial to a set of data points is in essence an unconstrained optimization problem. The optimization criterion is the minimization of the total distance of the data points to the polynomial. At this point, the following problem arises: “Can the total distance of the set of data point D, to a polynomial p be described by means of a closed form expression?”

As a first approximation, the distance term can be equated to Equation 2.10.

$$\sum_{(x_0,y_0)\in D} p^2(x_0,y_0) \tag{2.10}$$

Yet this is not an adequate measure of the distance. A better measure developed by Taubin [4] is given in Equation 2.11.

$$\sum_{(x_0,y_0)\in D} \frac{p^2(x_0,y_0)}{\nabla^2(x_0,y_0)} \tag{2.11}$$

Where $\nabla^2(x_0,y_0)$ stands for the norm of the gradient squared. Taubin approximates this expression with the following;

$$\frac{\sum_{(x_0,y_0)\in D} p^2(x_0,y_0)}{\sum_{(x_0,y_0)\in D} \nabla^2(x_0,y_0)} \tag{2.12}$$

The objective function is minimized by generalized eigenvector fitting techniques. After this, an iterative algorithm improving the polynomial fit is applied. In bounded fitting, in addition to these, the necessary and sufficient conditions for boundedness are taken into account.

Recently linear approaches to curve fitting have started to emerge, which overcomes many drawbacks of the previous algorithms. The objective of all linear IP curve fitting techniques is to approximate a given dataset with a polynomial as closely as possible by minimization of their algebraic distance. The 3L (see section 2.4.1) algorithm provides such a linear solution, however suffers from the global stability

and is not robust to handle fair amount of noise or missing data. A ridge regression regularization method can be used to boost the stability and robustness of 3L fitting algorithm.

The line decomposition detailed in [2], provides an important simple geometric interpretation of the underlying geometry for a given implicit polynomial. Polynomial decomposition expresses the curve as a unique sum of products of (possibly) complex lines. Each real intersection of these lines, i.e. related-points, undergoes the same motion with the curve, leading to a pose estimation algorithm. This representation also allows the formulation of new invariants based on certain covariant characteristics such as conics centers because they map under an affine transformation. More detailed concept of related-points and their application in our work is presented in Chapter 2.4.2.

2.4.1 3L Fitting

As explained in Section 2.1, 3L Fitting is an explicit, linear, least squares fitting depending on whether the points are inside or outside the original data procedure that is implemented by augmenting each data point of a data set by a pair of synthetically generated points at a distance of c to either side of the data point in a direction perpendicular to the data surface or curve. An implicit polynomial is then fit to the entire data set, where the values assigned to the synthetic data points are +c or –c, points, respectively. The original data points are assigned a value of 0. This procedure is illustrated further in the following paragraphs.

The polynomial $f_n(x, y)$ is an explicit function at all values of x, y and usually fitting formulations take into account only Γ_0. We can get fast, stable, repeatable implicit polynomial surface fits by fitting the explicit polynomial $f_n(x, y)$ to a portion of distance transform $d(x, y)$ of Γ_0. 3L fitting, besides the original data set Γ_0 uses a pair of synthetically generated data sets Γ_{c+} and Γ_{c-} consisting of points at a distance c to either side of Γ_0. Note, Γ_{c+} and Γ_{c-} are the level sets of $d(x, y)$ at levels $+c$ and $-c$, respectively (Figure 2.4) [8]. As a result, estimating the vector of

coefficients *A* is the minimization of Γ_0, the Euclidean distance transform determines a point in Γ_{c+} and another one in Γ_{c-} which are at a perpendicular distance *c* to each side of the original curve Γ_0.

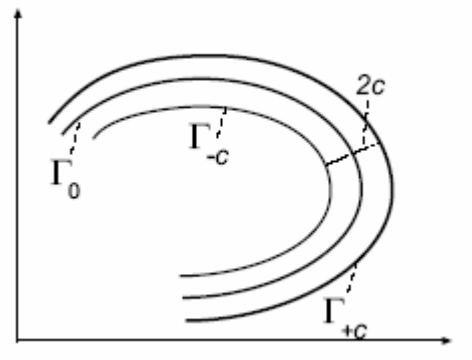


Figure 2-1 Level set geometry in 2D

Then estimating the vector of polynomial coefficients *A* is minimization of $\sum_{m=1}^{3K}(d(x_m, y_m) - Y_m^t A)^2$ or $\|MA\text{-}d\|$ where $M = [Y_1 \;\; Y_2 \ldots Y_{3K}]$, Y_m is Y evaluated at $p_m=(x_m,y_m)$ and d as a vector whose m[th] component is $d=(x_m,y_m)$, the distance of the point p_m to Γ_0 (1). The least squares solution to this problem is:

$$A = (M^T M)^{-1} M^T d \tag{2.13}$$

There are several advantages of introducing the two level sets as additional constraints. First, by fitting a polynomial to more data then just Γ_0, it makes the fitting more stable and consistent with regard to transformations of data sets, and more robust to noisy or missing data. The level sets bring in the additional benefit of forcing singularities away from the vicinity of the data set, as singularities occur at local extreme or saddle points, and use of Γ_{c+} and Γ_{c-} discourage the occurrence of singularities within the synthetic data ribbon. Second, as the fitted polynomial *f*(*x, y*) is an approximation to the distance transform *d*(*x, y*), given a new data point $(\hat{x}, \hat{y}) \mid f(\hat{x}, \hat{y})$ is an approximation to the Euclidean distance from $(\hat{x}, \hat{y})$ to Γ_0. Finally, since the implicit curve representation has been turned into the problem of studying the properties of the explicit polynomial *f*(*x, y*), the full arsenal of linear

vector space theory and algorithms becomes applicable, and implicit polynomial curves can be further studied and manipulated in completely new ways.

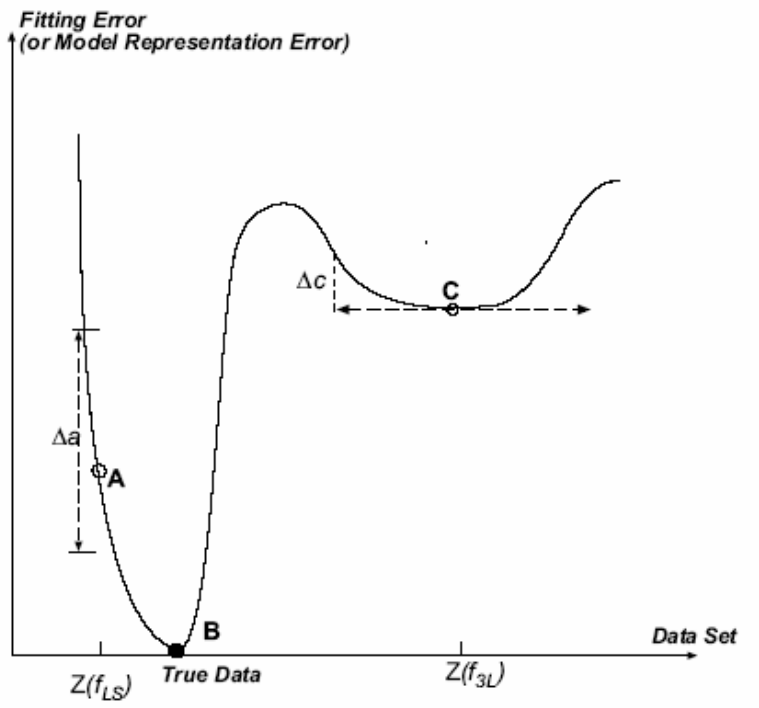


Figure 2-2 Comparison of least squares and the 3L fitting methods for fitting an implicit polynomial model for a given dataset.

In Figure 2.3, the horizontal axis represents the *Data Set* and the vertical axis represents the *IP Model Fitting Error* [8]. In the graph, *B* is the point where the true data set lies. Nonlinear optimization and least square 1L fitting methods trying to reach to this solution, stop at point *A* with a zero set $Z(f_{LS})$, which could be close or far away from the *true data* because no stability or robustness mechanism is employed. *A* is in a region which is very sensitive to the perturbation in the data such as noise, missing data or transformations. A small change in the position of *A* can result in a big change in the fitting error Δa. However, point *C*, which is the solution of 3L fitting, is in a much more stable region. Even though its zero set $Z(f_{3L})$ could be more far away from the *true data set* than other nonlinear or 1L algorithms, it is very robust to changes in *true data*. Δc is very small regardless of the change in the position of *C* [8].

2.4.1.1 Globally Stabilized 3L Fitting

As discussed above, 3L fitting method improves the performance of nonlinear optimization and least square fitting algorithms. However, it can not provide globally stabilized fits for many cases and are not robust against perturbational effects like noise. The reason for this is the M^tM matrix of the products of the monomials to be almost singular with some eigenvalues much smaller than the others. Small eigenvalues do not add to the fit around the dataset and cause extra open branches. A way to overcome these problems is to apply ridge regression regularization to the 3L fitting method as proposed in [9].

Ridge regression is a computationally efficient method for reducing data collinearity and the resulting instability. By improving the condition number of M^tM, this method moves the extra curves to infinity and a stable closed bounded fit is obtained. To apply the method, a κD term is added to equation 2.13 as:

$$A = (M^T M + \kappa D)^{-1} M^T d \qquad (2.14)$$

In equation 2.14, κ is the ridge regression parameter. It should be increased from 0 to higher values until a stable closed bounded curve is obtained. The diagonal D matrix which is the other term added to the original equation has the same number of terms as the coefficient vector A. The entries of D can be obtained by:

$$D_{ii} = \beta_{j+k} \frac{j!k!}{(j+k)!} \qquad (2.15)$$

The index of each diagonal element in D depends on the variation of the degrees of x and y components in equation 2.8 as $i = k + \frac{(j+k)(j+k+1)}{2} + 1$. Also β_{j+k} is chosen to be:

$$\beta_{j+k} = \sum_{r,s\geq 0; r+s=j+k} \frac{(r+s)!}{r!s!} \sum_{l=1}^{N} x_l^{2r} y_l^{2s} \qquad (2.16)$$

When expanded:

$$\beta_0 = \sum_{l=1}^{N} x_l^0 y_l^0 = N$$

$$\beta_1 = \sum_{l=1}^{N} x_l^2 + y_l^2 = \sum_{l} (x_k^2 + y_k^2)^1$$

$$\beta_2 = \frac{2!}{2!0!}\sum_{l} x_l^4 + \frac{2!}{1!1!}\sum_{l} x_l^2 y_l^2 + \frac{2!}{0!2!}\sum_{l} y_l^4 = \sum_{l} (x_l^2 + y_l^2)^2 \qquad (2.17)$$

. . .

. . .

$$\beta_n = \sum_{l} (x_l^2 + y_l^2)^n$$

where $(x_l, y_l)_{l=1}^{N}$ are the elements of the normalized object data and n is the degree of the resulting IP. In our simulations, radial distance normalization[9] is used, which is a linear process. Normalized data points are found by dividing every data point, $(x_l, y_l)_{l=1}^{N}$, with the average radial distance of the set $\frac{1}{N}\sum_{l} (x_l^2 + y_l^2)^{1/2}$ after the center of the data has been shifted to the origin. Inherent Euclidean invariance properties of fitting methods are also preserved by this method.

2.4.2 Decomposed Quartics and Related Points

It has been shown in [2,3,10] that algebraic curves can be decomposed as a unique sum of line factors, the intersection of which are examples of *related-points.* Considering an accordingly decomposed monic quartic curve:

$$f_4(x, y) = \prod\nolimits_4 (x, y) + \gamma_2 \prod\nolimits_2 (x, y) + \gamma_0 \underbrace{\prod\nolimits_0 (x, y)}_{=1}$$

$$= \prod_{i=1}^{4} \underbrace{\begin{bmatrix} 1 & l_{4i} & k_{4i} \end{bmatrix}}_{=L_{4i}^T} \underbrace{\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}}_{=X} + \gamma_2 \prod_{i=1}^{2} \underbrace{\begin{bmatrix} 1 & l_{2i} & k_{2i} \end{bmatrix}}_{=L_{2i}^T} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} + \gamma_0 = 0 \qquad (2.18)$$

The intersection point $d_p = \{x_p, y_p\}$ of any two non-parallel line factors, such as $L_{ij}^T X = x + l_{ij} y + k_{ij}$ and $L_{qr}^T X = x + l_{qr} y + k_{qr}$, can be defined by the matrix/vector relation:

$$\begin{bmatrix} 1 & l_{ij} & k_{ij} \\ 1 & l_{qr} & k_{qr} \end{bmatrix} \begin{bmatrix} x_p \\ y_p \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Rightarrow \begin{bmatrix} x_p \\ y_p \end{bmatrix} = \begin{bmatrix} l_{ij} k_{qr} - l_{qr} k_{ij} \\ k_{ij} - k_{qr} \end{bmatrix} \div (l_{qr} - l_{ij}) \qquad (2.19)$$

In general, any two 4th degree curves defined by a monic $f_4(x, y) = 0$ and a monic $\bar{f}_4(\bar{x}, \bar{y}) = 0$ will be affine equivalent if for some scalar s_4,

$$f_4(x, y) = 0 \xrightarrow{A} f_4(a\bar{x} + b\bar{y} + t_x, c\bar{x} + d\bar{y} + t_y) \overset{def}{=} s_4 \bar{f}_4(\bar{x}, \bar{y}) = 0$$

Where A represents the affine transformation. Two corresponding related-points of the affine equivalent curves defined by $f_4(x, y) = 0$ and $\bar{f}_4(\bar{x}, \bar{y}) = 0$, such as $\{x_i, y_i\}$ and $\{\bar{x}_i, \bar{y}_i\}$, respectively, will be defined by the condition that

$$\begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} a & b & t_x \\ c & d & t_y \\ 0 & 0 & 1 \end{bmatrix}}_{A} \begin{bmatrix} \bar{x}_i \\ \bar{y}_i \\ 1 \end{bmatrix} \qquad (2.20)$$

Any two corresponding related-points will satisfy the following relation according to the equations 2.18 and 2.19.

$$f_4(x_i, y_i) \overset{def}{=} z_i = s_4 \bar{f}_4(\bar{x}_i, \bar{y}_i) = s_4 \bar{z}_i \qquad (2.21)$$

Under the affine transformation A, every $\prod_q(x, y)$ in equation (2.17) is equal to

$$\prod_q(x, y) = \prod_{i=1}^{q} L_{qi}^T X \xrightarrow{A} \prod_{i=1}^{q} L_{qi}^T A \bar{X} = \prod_{i=1}^{q} \underbrace{(m_1 + l_{qi} m_3)}_{\overset{def}{=} s_{qi}} \bar{L}_{qi}^T \bar{X} = s_q \underbrace{\prod_{i=1}^{q} \bar{L}_{qi}^T \bar{X}}_{\overset{def}{=} \prod_q(\bar{x}, \bar{y})} \qquad (2.22)$$

for a real scalar $s_q = \prod_{i=1}^{q} s_{qi}$ and q monic line factors $\bar{L}_{qi}^T \bar{X}$, with $q = 4$ or 2.

Hence, under an affine transformation A, the implicit polynomial defined by equation 2.17 will imply

$$\bar{f}_4(x,y) \overset{A}{\mapsto} s_4 \bar{f}_4(\bar{x},\bar{y}) = s_4 \prod_4(\bar{x},\bar{y}) + \gamma_2 \left[s_2 \prod_2(\bar{x},\bar{y}) + \gamma_0 s_0 \right] \tag{2.23}$$

A unique monic polynomial that is affine equivalent to $f_4(x,y)$, is equal to

$$\bar{f}_4(\bar{x},\bar{y}) = \prod_4(\bar{x},\bar{y}) + \underbrace{\frac{\gamma_2}{s_4} s_2}_{\bar{\gamma}_2} \left[\prod_2(\bar{x},\bar{y}) + \underbrace{\frac{\gamma_0}{s_0} s_2}_{\bar{\gamma}_0} \right] \tag{2.24}$$

Each $\prod_q(x,y)$ of $f_4(x,y)$, and each corresponding $\prod_q(\bar{x},\bar{y})$ of an affine equivalent $\bar{f}_4(\bar{x},\bar{y})$, will have the same number line factors. Moreover, according to equation 2.21, all of these factors will map to one another under affine transformations. Hence, $f_4(x,y)$ and $\bar{f}_4(\bar{x},\bar{y})$ will have the same number of corresponding related-points, as defined by the intersections of their corresponding line factors. These related points can be determined from the IP equation and are suitable for the analysis of affine and rigid curve motion.

3 FILTERING TECHNIQUES

3.1 Kalman Filtering

In 1960, R.E. Kalman published his famous paper describing a recursive solution to the discrete-data linear filtering problem [11]. Since that time, due in large part to advances in digital computing, the Kalman filter has been the subject of

extensive research and application, particularly in the area of autonomous or assisted navigation.

Initially, the filter was designed for the application in spacecraft navigation, but its general nature allows it to be applied to many fields. The introduction to Kalman filtering by Simon [12] also mentions applications in instrumentation, demographic modeling, manufacturing, fuzzy logic, and neural network training.

An example of a study of the applicability of Kalman filtering methods to navigation systems is provided by Brock & Schmidt [13]. They point out some of the advantages and problems associated with using this type of filtering for navigation. These also apply to visual tracking.

The main problem with Kalman filtering that Brock & Schmidt identify, is that statistical models are required for the system and the measurement instruments. Unfortunately, they are typically not available, or difficult to obtain. The need for statistical models is also pointed out as a problem in many other papers [12, 13, 14]. From these papers, the two most commonly recommended methods of approaching this problem are:

- Employ an adaptive algorithm which adjusts these unknown parameters (such as the measurement noise variance) after each time step based on the observed measurements. This also accounts for processes with changing parameters.

- Perform an offline analysis of the system and measurement instruments prior to running the process (system identification).

It should be noted however that the second approach will not always be applicable if the process can not be observed directly. In other words, if the measurements in the off-line analysis also contain errors, the process can not be accurately profiled.

It was recognized early on, that Kalman filtering does not stand alone as a unique technique for prediction. For example the work of Sorenson [15] points out the similarities between the Kalman filter, Bayesian and maximum likelihood estimation.

3.1.1 Theory

The idea behind Kalman filtering is modelling sequential data. In the context of visual tracking, images taken at discrete time steps form a sequence. The relationship between each of the images is based on a physical model of the scene.

Figure 3.1 gives a typical representation of a state-space model. The y_t nodes, for $t = 0,.....,T$ represent the observable output data (or measurements), and x_t are the hidden nodes which are states of the dynamical system. The assumption is that the nodes are vectors of real values and the probability model is Gaussian.

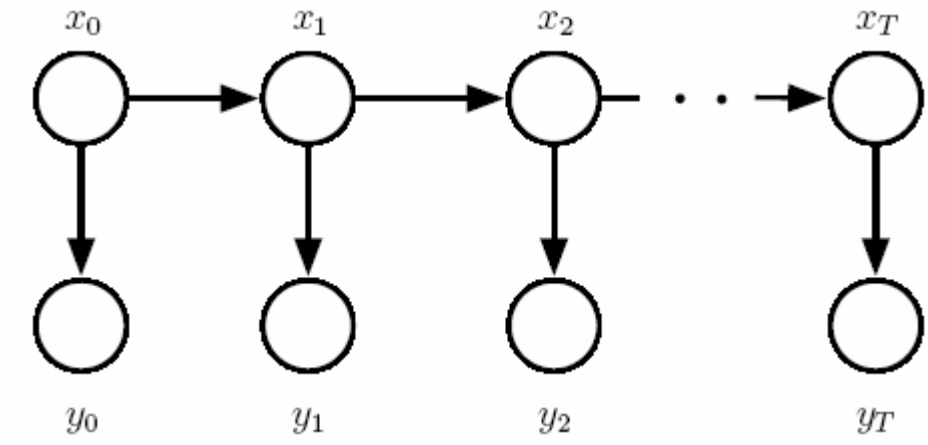


Figure 3-1 A representation of a state-space model

Each x_t contains an $m \times 1$ mean vector $\hat{x}$ and an $m \times m$ covariance matrix P, where m is the number of parameters that describe the state. A simple example of the parameters necessary for tracking are the x and y coordinates as well as the u and v velocity components. The y_t nodes are represented by an $n \times 1$ vector which is nothing but the observed position of the target in the context of visual tracking. This method of describing the state through a finite set of parameters is known as the state-space model (SSM).

This state space approach yields recursive formulas which describe each current estimate as a function only of the previous estimate and the new data sample. Thus only the last estimate must be stored. In addition to eliminating the need for extensive data storage, the Kalman filter is computationally more efficient than non-recursive techniques (e.g. Weiner filter) which require inversion of large matrices.

As mentioned earlier, the state nodes are related to each other through the physics underlying object motion. The transition from one state to the next could be described in many ways. These different alternatives can be grouped into linear and non-linear functions describing the state transition. Although it is possible to handle either of these transition types, the standard Kalman filter employs a linear transition function. The extended Kalman filter (EKF) allows a non-linear transition, together with a non-linear measurement relationship. For the standard Kalman filter, the state transition from t to $t+1$ can be expressed with the equation

$$x_{t+1} = Ax_t + w_t \tag{3.1}$$

where A is referred to as the state transition matrix and w_t is a noise term. This noise term is a Gaussian random variable with zero mean and a covariance matrix Q, so its probability distribution is

$$p(w) \sim N(0, Q) \tag{3.2}$$

The covariance matrix Q will be referred to as the process noise covariance matrix in the remainder of the thesis. It accounts for possible changes in the process between t and $t+1$ that are not already accounted for in the state transition matrix. Another assumed property of w_t is that it is independent of the state x_t.

It is also necessary to model the measurement process, or the relationship between the state and the measurement. In a general sense, it is not always possible to observe the process directly (i.e. all the state parameters are observable without error). Some of the parameters describing the state may not be observable at all, measurements might be scaled parameters, or possibly a combination of multiple

parameters. Again, the assumption is made that the relationship is linear. So the measurement y_t can be expressed in terms of the state x_t with

$$y_t = Hx_t + v_t \tag{3.3}$$

where H is the $m \times n$ observation matrix which relates the states to the measurements. Much like w_t for the process, v_t is the noise of the measurement. It is also assumed to have a normal distribution expressed by

$$p(v) \sim N(0, R) \tag{3.4}$$

where R is the covariance matrix referred to as measurement noise covariance matrix.

3.1.2 Computational Origins of the Filter

A priori state estimate given knowledge of the process prior to step k, and a posteriori state estimate given measurement z_k at step k is defined as $\hat{x}_k^-$ and $\hat{x}_k$, respectively. Then a priori and posteriori estimate errors are defined as

$$e_k^- \equiv x_k - \hat{x}_k^- \tag{3.5}$$

$$e_k \equiv x_k - \hat{x}_k \tag{3.6}$$

The a priori estimate error covariance is then

$$P_k^- = E\left[e_k^- e_k^{-T}\right] \tag{3.7}$$

And the a posteriori estimate error covariance is

$$P_k = E\left[e_k e_k^T\right] \tag{3.8}$$

Writing the a posteriori state estimate $\hat{x}_k$ as a linear combination of the a priori estimate $\hat{x}_k^-$ and a weighted difference between an actual measurement z_k and

a measurement prediction $H\hat{x}_k$ is the main block of Kalman filtering equations. The resulting equation is given below.

$$\hat{x}_k = \hat{x}_k^- + K(\underbrace{z_k - H\hat{x}_k^-}_{residual}) \tag{3.9}$$

The residual part of the equation 3.9 is also called the measurement innovation. The residual is a kind of measure of disagreement between the predicted measurement $H\hat{x}_k^-$ and the actual measurement z_k. If these two components are in full agreement then the residual becomes zero.

The matrix K is called the Kalman gain or blending factor and has the role of modulating the update of the state vector $\hat{x}_k^-$ into $\hat{x}_k$ by appropriately weighting the measurement vector v_k. Kalman gain tries to minimize the a posteriori error covariance (equation 3.8). One form of the resulting K that minimizes equation 3.8 is given by

$$\begin{aligned} K_k &= P_k^- H^T (HP_k^- H^T + R)^{-1} \\ &= \frac{P_k^- H^T}{HP_k^- H^T + R} \end{aligned} \tag{3.10}$$

According to equation 3.10, the Kalman gain K weights the residual more heavily as the measurement error covariance R approaches zero.

$$\lim_{R_k \to 0} K_k = H^{-1} \tag{3.11}$$

The Kalman gain weights the residual less heavily, as the a priori estimate error covariance P_k^- approaches zero.

$$\lim_{P_k^- \to 0} K_k = 0 \tag{3.12}$$

Another interpretation of the equation 3.9 is that, the actual measurement z_k is trusted more while the predicted measurement $H\hat{x}_k^-$ is trusted less as the measurement error covariance R approaches zero. On the other side, the actual measurement z_k is trusted less while the predicted measurement $H\hat{x}_k^-$ is trusted more as the a priori estimate error covariance P_k^- approaches zero.

Kalman filter keeps the first two moments of the state distribution, hence

$$E[x_k] = \hat{x}_k \tag{3.13}$$

$$E[(x_k - \hat{x}_k)(x_k - \hat{x}_k)^T] = P_k \tag{3.14}$$

The state distribution is a normally distributed Gaussian with the mean of a posteriori state estimate and the variance of the a posteriori estimate error covariance

$$\begin{aligned} p(x_k \mid z_k) &\sim N(E[x_k], E[(x_k - \hat{x}_k)(x_k - \hat{x}_k)^T] \\ &= N(\hat{x}_k, P_k) \end{aligned} \tag{3.15}$$

3.1.3 Discrete Kalman Filter Algorithm

There are two types of equations in the Kalman filter algorithm: time update equations and measurement update equations. The time update equations can be thought of as predictor equations which produce a priori estimates of state and error covariance for the next time step. Measurement update equations, however is responsible from the correction of the priori estimates to obtain an improved a posteriori estimate by using the new measurement [16].

Figure 3.2 shows a roughly representation of the general algorithm of Kalman filter. The equations involved in these two stages will be explained below.

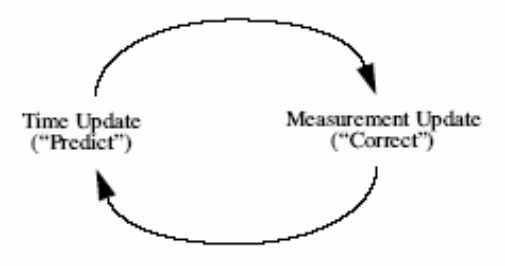


Figure 3-2 The general Kalman filter cycle

The time update stage has two distinct equations, which are projecting the state and covariance estimates forward from time step $k-1$ to k. Those two equations are as follows

$$\hat{x}_k^- = A\hat{x}_{k-1} \tag{3.16}$$

$$P_k^- = AP_{k-1}A^T + Q \tag{3.17}$$

Measurement update equations, on the other hand have three distinct equations, in which the computation of the Kalman gain, K_k comes first. After the measurement is taken at time step k a posteriori state estimate is calculated according to the new measurement. Finally, a posteriori error covariance estimate is obtained again by using the Kalman gain K_k calculated at the first step. Three equations of the measurement update stage is as follows

$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1} \tag{3.18}$$

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-) \tag{3.19}$$

$$P_k = (I - K_k H)P_k^- \tag{3.20}$$

The process is repeated after each time and measurement update pair recursively. The complete algorithm of the Kalman filter is shown in figure 3.3.

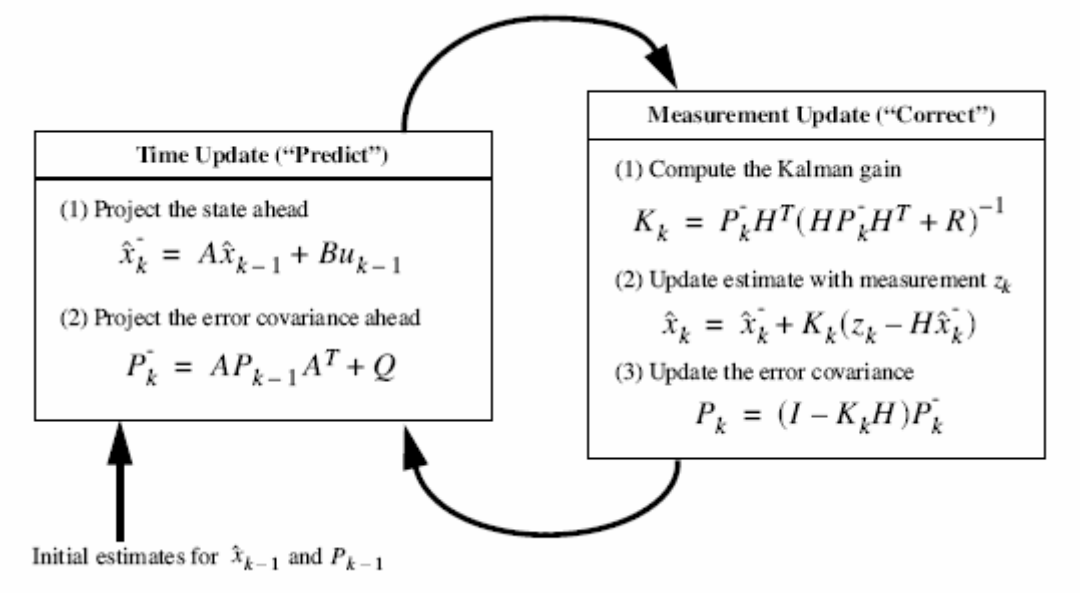


Figure 3-3 The complete Kalman filter cycle with all the equations

3.2 Particle Filters

Object tracking in our context is viewed as an optimal filtering problem of the state equations under a Bayesian framework. If the state equations are linear and the posterior density is Gaussian, the Kalman filter which we have used in previous sections provides an optimal solution [18]. However, where these assumptions do not hold, there exist no analytical solution and approximations have to be made. One example is the Extended Kalman filter (EKF) which assumes a Gaussian posterior density and adopts a first order Taylor expansion to provide local approximation within the current state. In practice where the state equations are highly non-linear and the posterior density is non-Gaussian, the EKF may give a large estimation error. An alternative algorithm is the approximate grid-based filter. The computational cost of the grid-based filter increases exponentially with the state dimension, thus limiting its widespread application.

Particle filters have become a useful and important tool for the task of object tracking due to the applicability to a wide range of cases. The basic idea behind particle filters is to approximate posterior probability of the states using a large number of samples (particles) with associated weights. These particles and weights are then updated sequentially along with the state evolution when new observations

become available. They make no assumption of Gaussian or linear behavior and are therefore more practical than its alternatives.

3.2.1 Theory

A state space model is defined by the following state and measurement functions

$$x_k = f_k(x_{k-1}, w_{k-1}) \tag{3.21}$$

$$z_k = h_k(x_k, v_k) \tag{3.22}$$

Where k is the time index and f_k is a non-linear function describing the evolution of the state with independent and identically distributed process noise, w_{k-1}. h_k is a non-linear function mapping the state space to the measurements with independent and identically distributed noise, v_k. Letting $z_{1:k} = (z_i, i = 1,.....k)$ be the measurement sequence and assuming the prior distribution $p(x_0)$ is known, the posterior probability can be obtained sequentially by prediction and updated as follows:

$$p(x_k \mid z_{1:k-1}) = \int p(x_k \mid x_{k-1}) p(x_{k-1} \mid z_{1:k-1}) dx_{k-1} \tag{3.23}$$

$$p(x_k \mid z_{1:k}) = \frac{p(z_{1:k} \mid x_k) p(x_k \mid x_{k-1})}{p(z_k \mid z_{1:k-1})} \tag{3.24}$$

The above equations are the optimal solution from a Bayesian perspective to the non-linear state estimation problem. One limitation is that the evolution of the posterior density can not in general be determined analytically. Thus, some approximation must be made. Particle filters approximate the posterior probability by a set of support points (particles) $x_k^i, i = 1,.....N$ with associated weights w_k^i:

$$p(x_k \mid z_{1:k}) \approx \sum_{i=1}^{N} w_k^i \delta(x_k - x_k^i) \tag{3.25}$$

Where $\delta(.)$ is an indicator function. The filtered state is taken as the mean of the posterior density. The weights are decided using importance sampling [21]. An importance density $q(x_k \mid z_{1:k})$ is identified from which samples are drawn. The weights are then defined as being proportional to the ratio of $p(x_k \mid z_{1:k})$ to $q(x_k \mid z_{1:k})$. It has been shown [22] that if the importance density is selected appropriately and is only dependent on the current observation, z_k, and the past state, x_{k-1}, the weights can be updated as follows:

$$w_k^i \propto w_{k-1}^i \frac{p(z_k \mid x_k^i)\, p(x_k^i \mid x_{k-1}^i)}{q(x_k^i \mid x_{k-1}^i, z_k)} \tag{3.26}$$

Two implementation issues should be considered. The first is that of degeneracy, where after some time steps, only one particle has significant weight. Thus considerable computational effort will have been spent updating particles whose contribution to the approximation of $p(x_k \mid z_{1:k})$ is negligible. Re-sampling can be used to eliminate those particles with small weights thereby focusing on particles with large weights. Re-sampling generates a new particle set by sampling with replacement from the original set $\{x_k^i, i = 1,.....N\}$ with $\Pr(x_k^j = x_k^i) = w_k^i$. Here j is the particle index after re-sampling. The parent relationship is denoted, $parent(j) = i$. The weights are reset to $1/N$ as the samples are independent and identically distributed and are drawn from a discrete density function.

The second issue is how to choose the importance density. A convenient choose is to use the prior, $p(x_k \mid x_{k-1}^i)$. A particle filter with this importance density and re-sampling step is called a sequential importance re-sampling (SIR) filter [23]. However, as the importance density is independent of the current measurement, the state space is explored without knowledge of the observations, which makes the SIR filter sensitive to outliers. Recently the Auxiliary SIR (ASIR) filter [24] was proposed to obtain a more reliable importance density. It is defined as:

$$q(x_k^i \mid z_k) \propto p(z_k \mid \mu_k^i)\, p(x_k \mid x_{k-1}^i) w_{k-1}^i \tag{3.27}$$

Where μ_k^i can be the mean of x_k conditional on x_{k-1}^i or an associated sample. In addition Bayes's rule shows that:

$$p(x_k \mid z_k) \propto p(z_k \mid x_k)\, p(x_k \mid x_{k-1}^i) w_{k-1}^i \tag{3.28}$$

Considering the re-sampling step presented earlier, the particle x_k^j is assigned a weight a proportional to the ratio of the right-hand side of equation 3.28 to equation 3.27 as:

$$w_k^j \propto \frac{p(z_k \mid x_k^j)}{p(z_k \mid \mu_k^{parent(j)})} \tag{3.29}$$

The ASIR filter generates particles from the sample at time step $k-1$ conditional on the current observation, which can be closer to the true state compared with those obtained using the SIR filter. If the noise of state evolution is low, even with relatively high measurement noise, ASIR is less likely to be sensitive to outliers as $p(x_k \mid x_{k-1}^i)$ is well characterized by μ_k^i.

4 FILTER PARAMETER EXPERIMENTS

4.1 Determination of Kalman Filter Parameters

One problem of predictive filter identified in the literature is the requirement for prior knowledge about the process and the measurement procedure. Specifically, the values of the process and the measurement error covariance matrices are needed. Before going into the details of how those parameters are determined, the measurement procedures for center of mass coordinates and orientation of the target is presented.

4.1.1 Orientation Measurement

In the case of closed-bounded quartics, we have two pairs of complex-conjugate lines, i.e. $L_{42} = L_{41}^{*}$ $L_{44} = L_{43}^{*}$, the intersection points of which are real. For tracking, we will be using the centroid of the bounding curve and these two related points. For the robust calculation of the orientation of the free-form curve, we follow [17] and form two vectors originating from the center of mass to the two related points. The sum of these two vectors is a new vector that is quite robust against noise throughout the whole trajectory. The angle between this sum vector and the positive x-axis is defined to be the orientation of the curve.

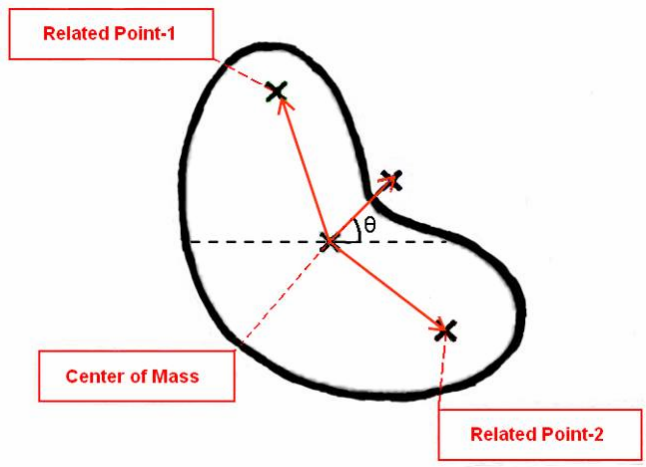


Figure 4-1 Object with the center of mass, two related points and the corresponding point used for orientation

4.1.2 Measurement for Center of Mass

Fourth degree implicit polynomials fitted to the boundary data is used to find the coordinates of the target's center of mass directly. The points on the implicit polynomial curve are averaged to calculate the x and y coordinates of the center of mass.

Definitely, the method used to calculate the orientation of the object contains noise creative procedures. The fitting process to the available boundary data points

can be one cause of this type of noise. Another noise term can be added from the calculation of related points. Although, the boundary data points are assumed to be reliable in the simulations here, in real situations they can be disturbed with noise coming from the boundary detection algorithm possibly. All of the possible noise creative terms are considered while calculating the measurement covariance matrices.

4.1.3 Experiments for Measurement Errors

In order to find meaningful values for measurement errors to set in the Kalman filter formulation, we have performed some experiments. Missing data condition, which is the representation of occlusion in real applications are experimented independently. Then, combination of this factor with perturbations is combined in order to obtain reasonable values for orientation measurement procedure. As can be seen from the performance results, the interpolation property of implicit polynomial representation and the method used for calculating the orientation is robust enough to be used inside such a prediction filter for measurement purposes.

4.1.3.1 Orientation measurement under missing data and perturbation around data points

8 arbitrary shapes shown in Figure 4.2 were used for the orientation measurement experiments. The performance of the method for orientation measurement under 5% and 10% missing data of the object boundaries is intended.

For systematic missing data analysis, approach proposed in Civi's work[8] is adopted. The object boundaries were divided into 50 equal pieces and the boundary was traced clockwise each time starting from the beginning of the next piece and occluding some percentage of the whole data. Figure 4.3 shows examples of 5% and 10% missing data.

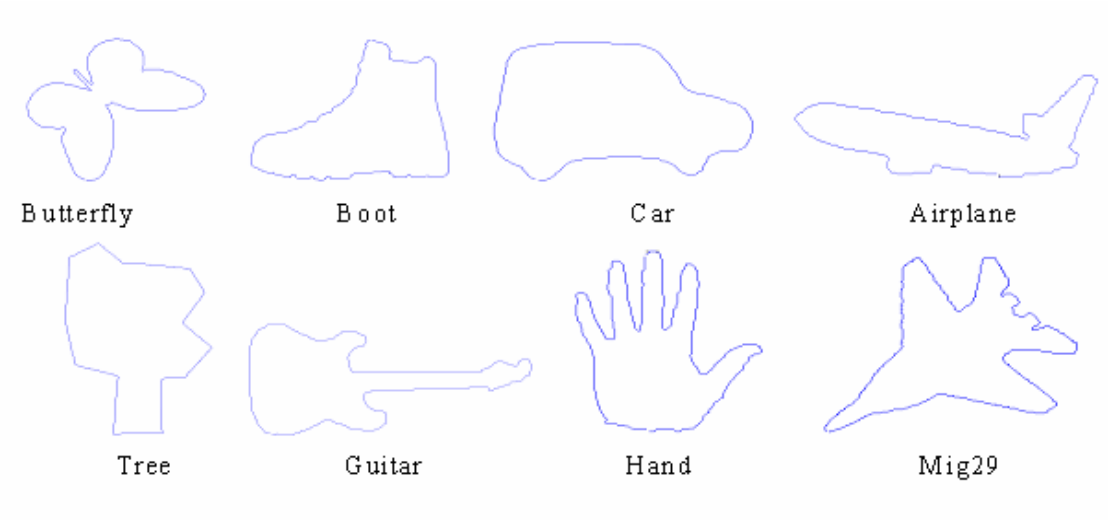


Figure 4-2 8 objects used in the orientation measurement experiments

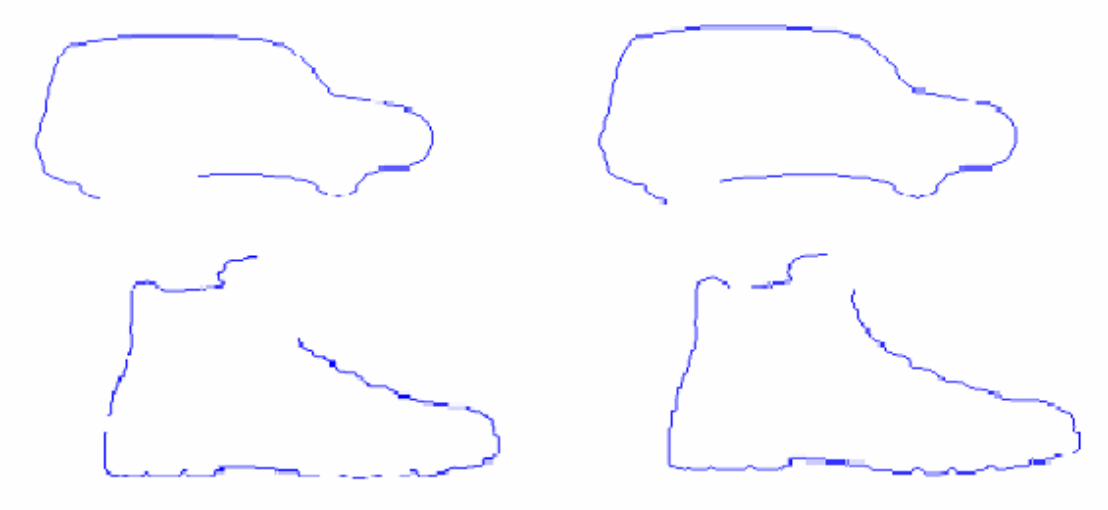


Figure 4-3 %5 and %10 occluded versions of the car and boot shapes

The performance of the orientation measurement algorithm for the car and boot shapes with 5% and 10% missing data under 80 degree orientation is presented in the following figures.

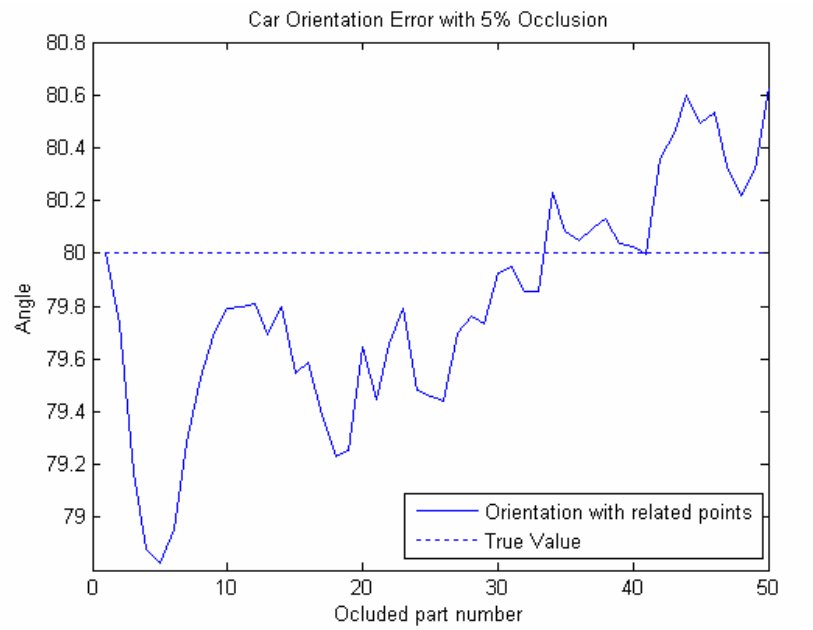


Figure 4-4 Orientation Measurement results for car shape with 5% missing data under 80 degree orientation

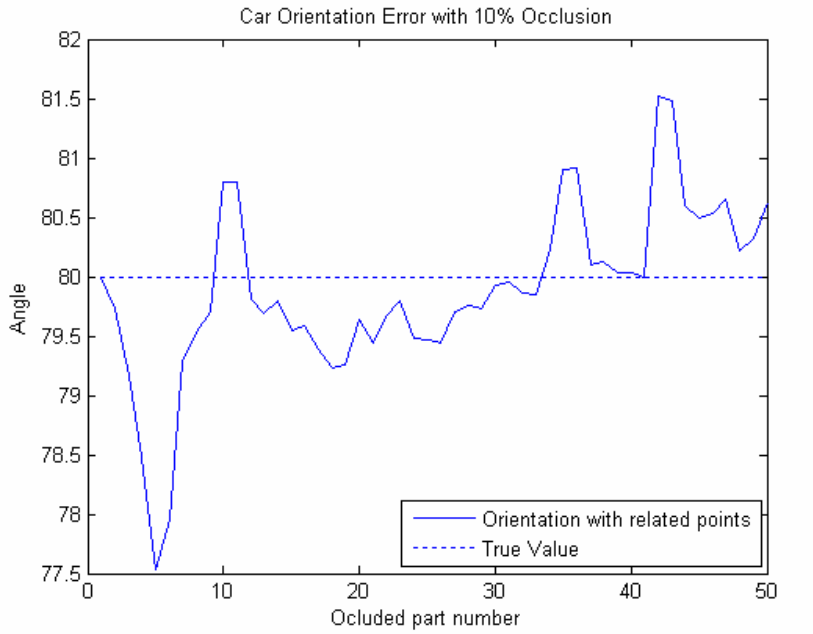


Figure 4-5 Orientation Measurement results for car shape with 10% missing data under 80 degree orientation

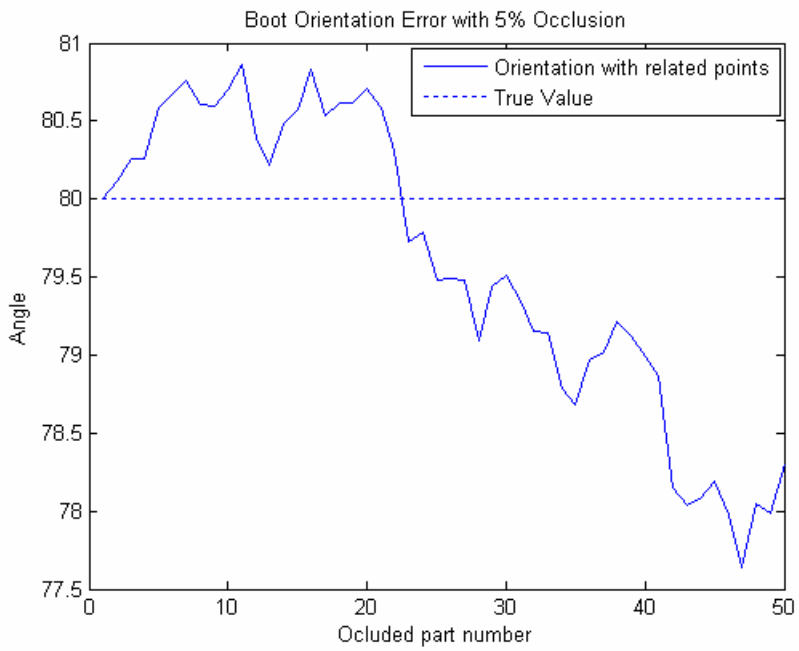


Figure 4-6 Orientation Measurement results for boot shape with 5% missing data under 80 degree orientation

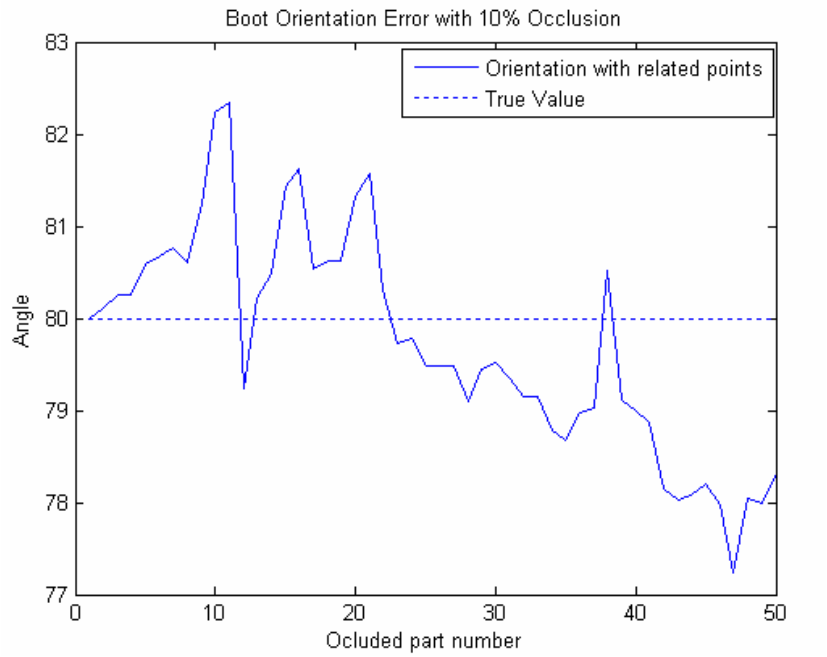


Figure 4-7 Orientation Measurement results for boot shape with 10% missing data under 80 degree orientation

In order to obtain reasonable values for the measurement error covariance matrix, performances of the measurement algorithm for all 8 objects are obtained and summarized in Table 4.1 and 4.2.

Object	Orientation Error
Butterfly	0.585
Boot	0.984
Car	0.843
Airplane	0.452
Tree	0.823
Guitar	1.287
Hand	1.957
Mig29	1.235

Table 4.1: Average measurement error for 8 objects with 5% missing data

Object	Orientation Error
Butterfly	1.234
Boot	1.980
Car	1.845
Airplane	0.762
Tree	1.832
Guitar	1.545
Hand	3.854
Mig29	2.745

Table 4.2: Average measurement error for 8 objects with 10% missing data

For representing the problems in real computer vision and image processing applications, missing data and perturbations of data analysis are combined and a new set of errors are obtained. For the perturbation of the data each data point is perturbed

by a Gaussian noise with a standard deviation of 2 pixels. The results are shown below in Tables 4.3 and 4.4.

Object	**Orientation Error**
Butterfly	0.620
Boot	1.154
Car	0.948
Airplane	0.482
Tree	0.915
Guitar	1.313
Hand	2.357
Mig29	1.432

Table 4.3: Average measurement error for 8 objects with 5% missing data + perturbation to data points with Gaussian noise

Object	**Orientation Error**
Butterfly	1.312
Boot	2.052
Car	1.921
Airplane	0.873
Tree	1.987
Guitar	1.712
Hand	4.428
Mig29	2.981

Table 4.4: Average measurement error for 8 objects with 10% missing data + perturbation to data points with Gaussian noise

4.1.3.2 Center of mass measurement under missing data and perturbation around data points

The points on the implicit polynomial curve are averaged to calculate the x and y coordinates of the center of mass in the experiments. The translation measurement results are compared with the center of mass directly calculated with the data points available. Since implicit polynomials have interpolation property, the resulting center of mass calculated from the implicit function is less boundary data point dependent compared to other classical method.

Figure 4-8 Interpolation property of implicit polynomial representation.
a) Polynomial fit of butterfly with 10% missing data b) Polynomial fit of butterfly with whole data

The center of mass experiments are performed with combination of boundary point perturbations with Gaussian noise and missing data. Again, the systematic approach of dividing the data points into 50 pieces is used here. The error values are averaged to obtain reasonable values for error terms. The following tables show the performance results of 8 objects with 5% and 10% missing data and data point perturbations with Gaussian noise (standard deviation of 2 pixels).

Object	Center of Mass Error (pixel)
Butterfly	1.210
Boot	0.454
Car	0.231
Airplane	0.523
Tree	0.453
Guitar	0.973
Hand	1.249
Mig29	1.102

Table 4.3: Average measurement error for 8 objects with 5% missing data + perturbation to data points with Gaussian noise

Object	Center of Mass Error (pixel)
Butterfly	1.420
Boot	0.671
Car	0.452
Airplane	0.485
Tree	0.529
Guitar	1.108
Hand	1.467
Mig29	1.355

Table 4.4: Average measurement error for 8 objects with 10% missing data + perturbation to data points with Gaussian noise

5 TRACKING EXPERIMENTS

5.1 Introduction

In our work, we tried different approaches in order to efficiently apply the powerful implicit algebraic 2D curve representation to the phenomenon of visual tracking. Different types of prediction methods were used in order to provide the trajectory and orientation predictions. Through the proposed concepts and algorithms, we tried to reduce the computational burden of fitting algorithms.

The first approach was to do fitting only for certain frames in an image sequence and fill in the missing ones using Kalman filtering technique. A discrete steady-state Kalman filter is used to estimate the future positions and orientation of the target object. In second proposed method, the fitting is done once offline and an algebraic curve space is calculated. Then, in every frame, one curve from the search region of curve space that has the smallest error according to some error metric is chosen to be the best fit for that frame. The third approach was to apply a region-based method using using appearance-adaptive methods models and particle filters which make complete use of all available image insensity information. By employing implicit algebraic curves, the boundary of the target can be tracked and also the model can be adapted to fit inside an implicit curve rather than a rectangle or an apriori known geometric shape.

5.2 Target Model

In order to create a Kalman filter, an appropriate linear model of the target must be created. The model must describe the x and y coordinates of the target centroid and the orientation of the target. All three parameters are independent of each other. The x and y models are the same and based on Newton's second law. The orientation is based on a moment equation. As it turns out this description simplifies to a model identical to the position representations.

Each position coordinate is assumed to be linearly independent, hence according to Newton's second law they can be partitioned as:

$$F_x = ma_x \tag{5.1}$$

$$F_y = ma_y \tag{5.2}$$

$$F_z = ma_z \tag{5.3}$$

The image only moves in two dimensions so the z component can be ignored. The masses of an object, in an image, is the sum of all its pixels. Since this model describes the centroid then its mass is unity. Therefore, the system reduces to:

$$F_x = a_x \tag{5.4}$$

$$F_y = a_y \tag{5.5}$$

It is sufficient to examine a generalized system, $F = a$ since the two systems above are identical to each other. In this form, this system is second order with respect to position. However, when the force and accelaration vary with time the system is defined as $F(t) = a(t)$. After taking the derivative of this equation the result is a third order equation that describes the jerk of the object. The state space represenation of this model is of the form:

$$\begin{bmatrix} \dot{x} \\ \dot{v} \\ \dot{a} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ v \\ a \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \dot{F} \tag{5.6}$$

We can discretize this model into the following state space representation

$$\begin{bmatrix} x_{k+1} \\ v_{k+1} \\ a_{k+1} \end{bmatrix} = \begin{bmatrix} 1 & T & T^2 \\ 0 & 1 & T \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_k \\ v_k \\ a_k \end{bmatrix} + \begin{bmatrix} T^3/6 \\ T^2/2 \\ T \end{bmatrix} J_k \tag{5.7}$$

where T is the sampling period of the discretized system . Since the problem is to track a target with an unknown trajectory, the object moves solely due to disturbances. The final model for the x and y coordinates is

$$\begin{bmatrix} x_{k+1} \\ v_{k+1} \\ a_{k+1} \end{bmatrix} = \begin{bmatrix} 1 & T & T^2 \\ 0 & 1 & T \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_k \\ v_k \\ a_k \end{bmatrix} + w_k \tag{5.8}$$

where w_k is the disturbance applied to the object. The model for the orientation is also a third order equation describing the angular jerk of the target.

$$\dot{M}(t) = I_{object}\dot{\alpha} \tag{5.9}$$

We can expand this into the form

$$\begin{bmatrix} \dot{\theta} \\ \dot{\omega} \\ \dot{\alpha} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \theta \\ \omega \\ \alpha \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} I^{-1}\dot{M} \tag{5.10}$$

After discretizing into the state space representation

$$\begin{bmatrix} \theta_{k+1} \\ \omega_{k+1} \\ \alpha_{k+1} \end{bmatrix} = \begin{bmatrix} 1 & T & T^2 \\ 0 & 1 & T \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \theta_k \\ \omega_k \\ \alpha_k \end{bmatrix} + \begin{bmatrix} T^3/6 \\ T^2/2 \\ T \end{bmatrix} I^{-1}M_k \tag{5.11}$$

Since there are no known input, only disturbances the orientation model reduces to:

$$\begin{bmatrix} \theta_{k+1} \\ \omega_{k+1} \\ \alpha_{k+1} \end{bmatrix} = \begin{bmatrix} 1 & T & T^2 \\ 0 & 1 & T \\ 0 & 0 & 1 \end{bmatrix} + \begin{bmatrix} \theta_k \\ \omega_k \\ \alpha_k \end{bmatrix} + w_k \tag{5.12}$$

5.3 Method of Fitting Only at Certain Frames

In this work, we used the steady-state form of the Kalman filter in order to estimate the position and orientation of the object between measurement frames. Since the system is steady-state a single equation is used to determine the filter.

$$x_{k+1} = [A - KH]x_k + Kz_k \tag{5.13}$$

$$x_{k+1} = \overline{A}x_k + Kz_k \tag{5.14}$$

Where $\overline{A} = A - KH$. From now on the bars will be ignored so that A corresponds to $\overline{A}$. Let us compute x_{k+2}:

$$x_{k+2} = Ax_{k+1} + Kz_{k+1} \tag{5.15}$$

$$= A[Ax_k + Kz_k] + Kz_{k+1} \tag{5.16}$$

$$= A^2x_k + AKz_k + Kz_{k+1} \tag{5.17}$$

Recall that for the filter, the measurement is constant until the feature extractor's next time step. So $z_k = z_{k+1} = z_{k+2} = ... = z_{k+n-1}$ Therefore

$$x_{k+2} = A^2x_k + [A + I]Kz_k \tag{5.18}$$

If we consider x_{k+3}

$$x_{k+3} = Ax_{k+2} + Kz_{k+2} \tag{5.19}$$

$$= A^2x_k + [A[A + I]]Kz_k + Kz_k \tag{5.20}$$

$$= A^3x_k + [A^2 + A + I]Kz_k \tag{5.21}$$

Hence,

$$x_{k+n} = A^n x_k + \left[\sum_{i=0}^{n-1} A^i\right] K z_k \qquad (5.22)$$

The feature exraction algorithm, once every sample period, sends the measured values, which describe the target position and orientation to the filter. The filter will hold this value and use it as a measurement until it is updated by the feature extraction algorithm. The process of holding the measurement value has the effect of creating another input trajectory that operates on a higher frequency.

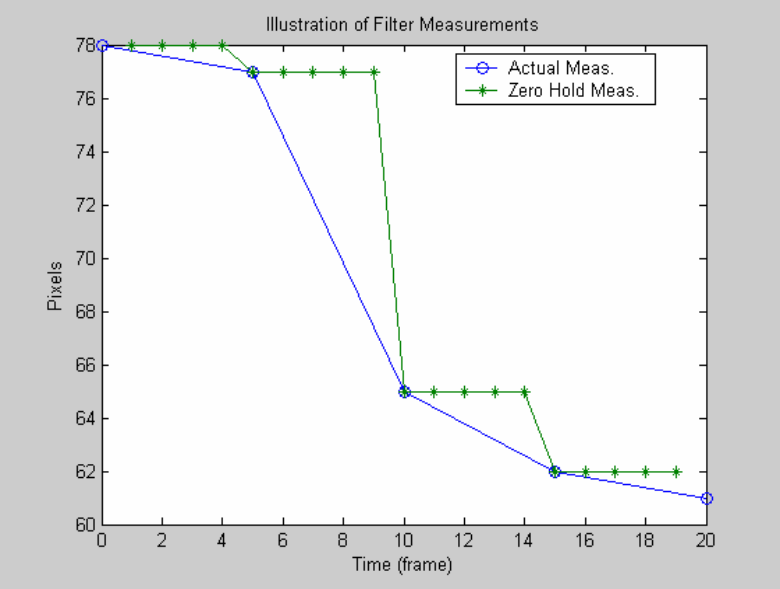


Figure 5-1 An example of a new trajectory

The filter uses the new input measurement function to determine the incremental estimates of the object.

5.3.1 Experimental Results

For our experiments we used a Boomerang shaped object undergoing a rigid motion with a relatively complex trajectory. Object boundaries have been modeled by quartic curves. The related points of these curves are obtained from the decomposition of the curve.

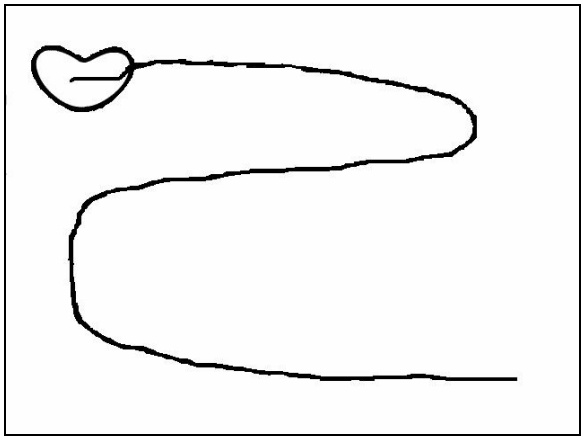


Figure 5-2 Trajectory of the target

Figure 5.3 shows the ability of the filter to predict the x-coordinate of the target's centroid. There is an overshoot when tracking fast changes in the x direction. A clearer illustration of the performance can be seen in an error comparison. The objective is to track the measured signal, so it is assumed that the measure is the true coordinate position. So, the error is the difference between the prediction and measured value. The error values are low and within a band of +-3 pixels when the target performs relatively uniform motion. When the target makes a maneuver, error values shows rapid increases, however the values converge to normal error values when the manevuer finishes.

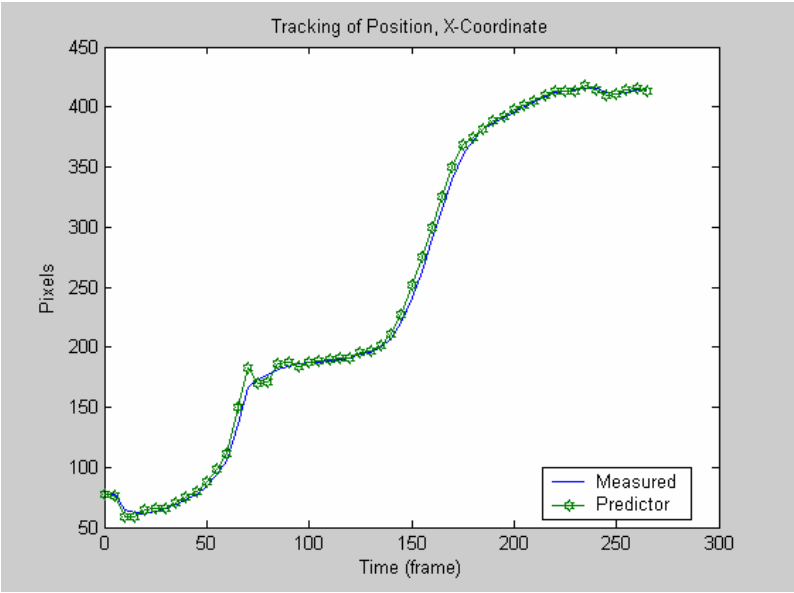


Figure 5-3 X-Coordinate Tracking of the Target

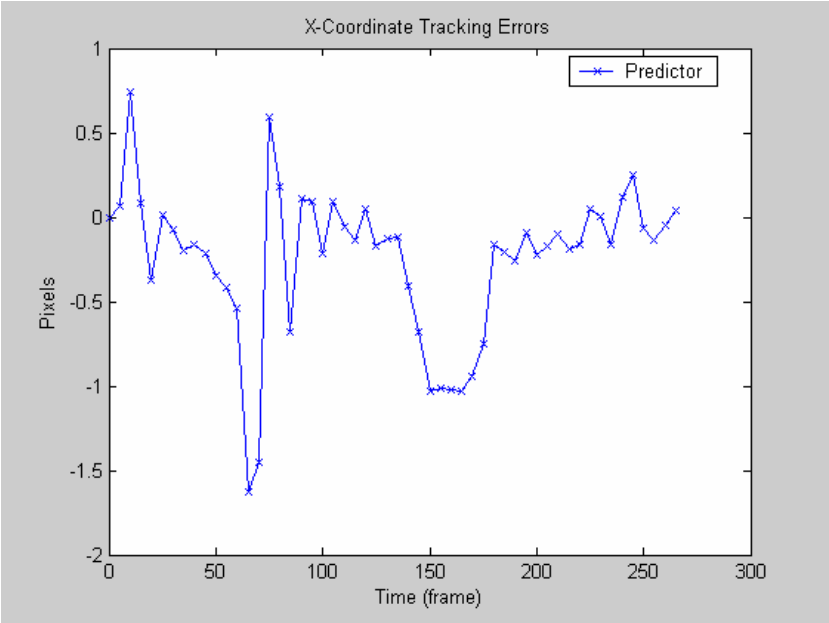


Figure 5-4 X-Coordinate Tracking Error of the Filter

The y-coordinate, on the other hand was exposed to higher speeds and sharp manevuers. Figure 5.5 shows the y coordinate position as a function of time, and it shows the predictor's ability to track this trajectory. Figure 5.6 illustrates the y coordinate tracking performance.

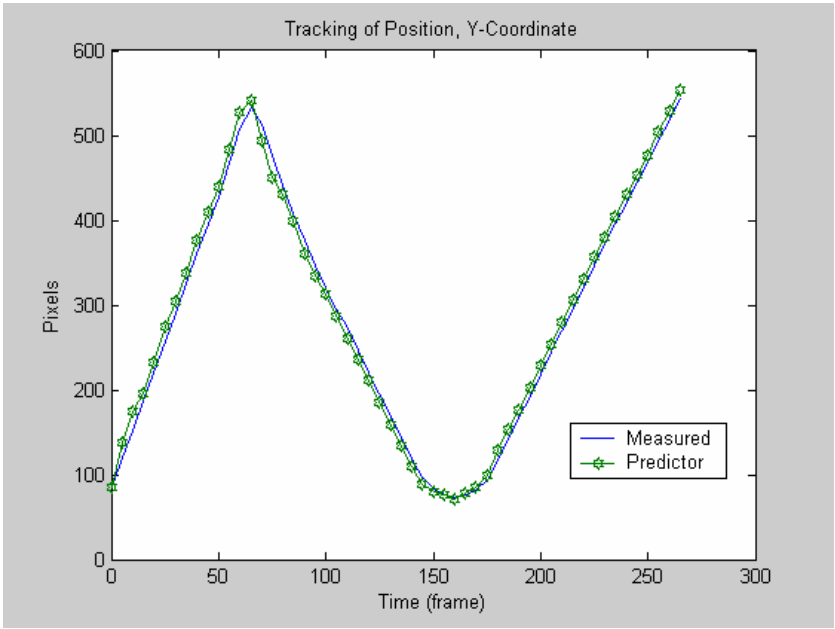


Figure 5-5 Y-Coordinate Tracking of the Target

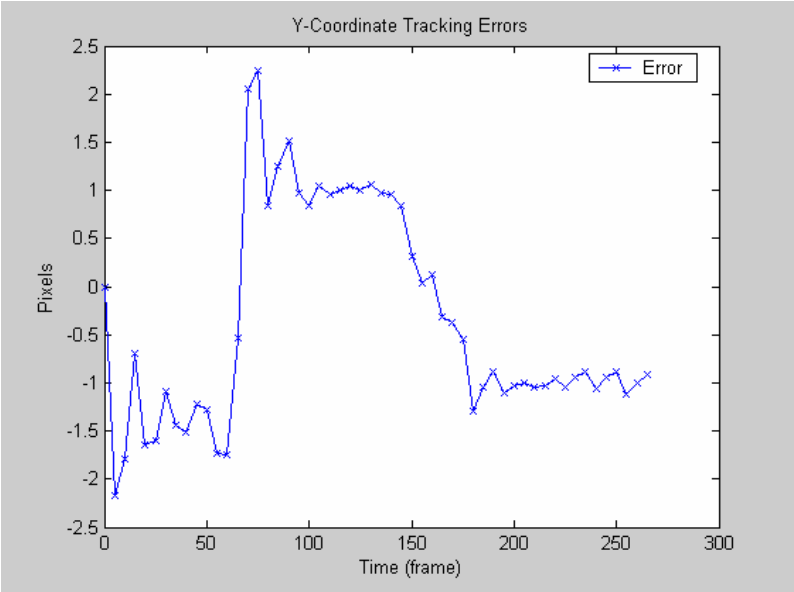


Figure 5-6 Y-Coordinate Tracking Error of the Filter

Purposely, the trajectory under consideration implies large angle variations to test robustness of the proposed orientation estimation method. As can be seen from Figure 5.7, orientation tracking error is within reasonable bounds.

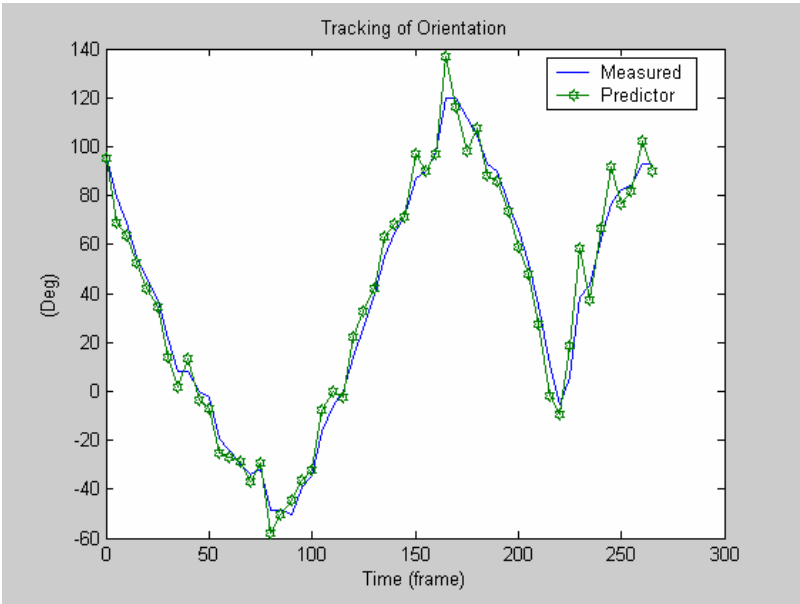


Figure 5-7 Orientation Tracking of the Filter

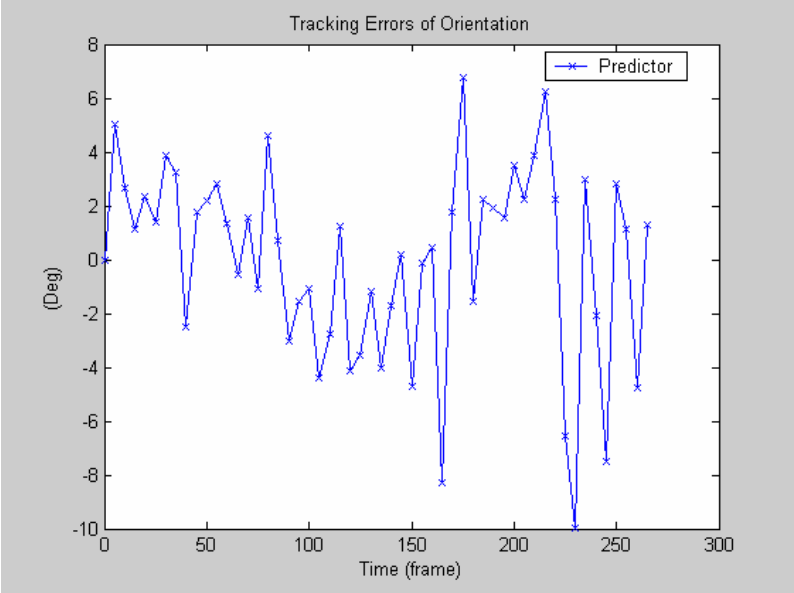


Figure 5-8 Orientation Tracking Error of the Filter

In order to test the robustness of the proposed algorithm against occlusion and noisy boundary data conditions, butterfly object is experimented in different trajectory than the boomerang-shaped object. Butterfly object without any noise and occlusion, with %5 occlusion and %10 occlusion and boundary data noise is experimented and the prediction error of each parameter is compared.

The following figures show the trajectory of x and y coordinates of the object's centroid and the orientation of the object for the butterfly object during 250 frame long video sequence. Those figures are indented to indicate the frames where the object makes a maneuver which are the times that the prediction errors are expected to increase suddenly. Orientation measures are between ± 180 degrees, however for the purpose of presentation under -180 degrees are not rounded to +180.

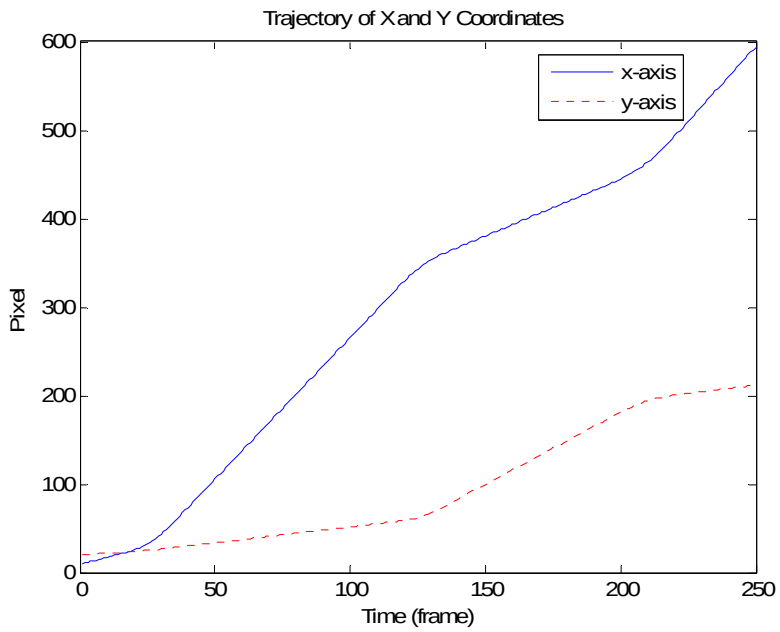


Figure 5-9 Centroid Trajectory of Butterfly Object

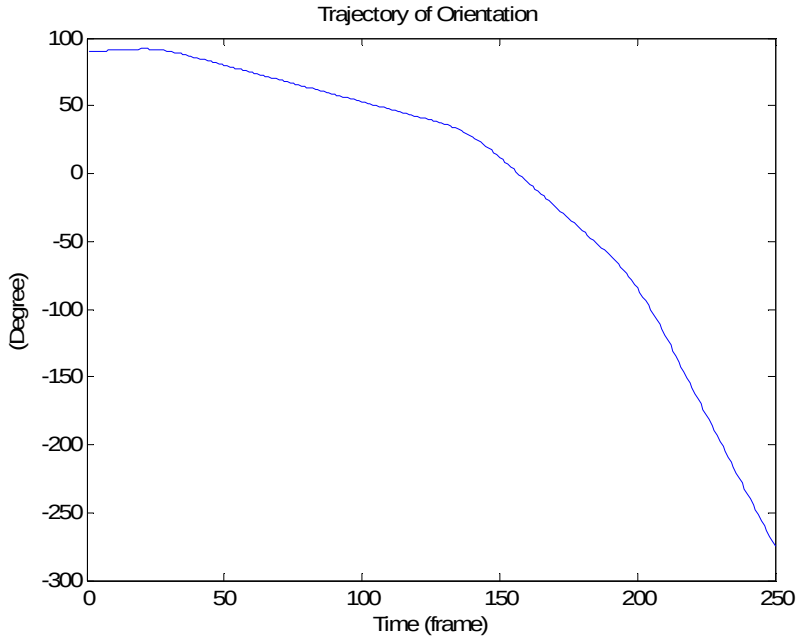


Figure 5-10 Orientation Trajectory of Butterfly Object

As can be seen from the following figure the increase in prediction errors of centroid coordinates is among 0.5 pixels when a fair amount of occlusion (%5) and boundary data noise (Gaussian noise with std.: 2 pixels) is introduced. When %10 of the butterfly object is occluded and a Gaussian noise with standard deviation of 2

pixels is introduced into the boundary data set, errors can reach upto 2.5 pixels at maneuvering times.

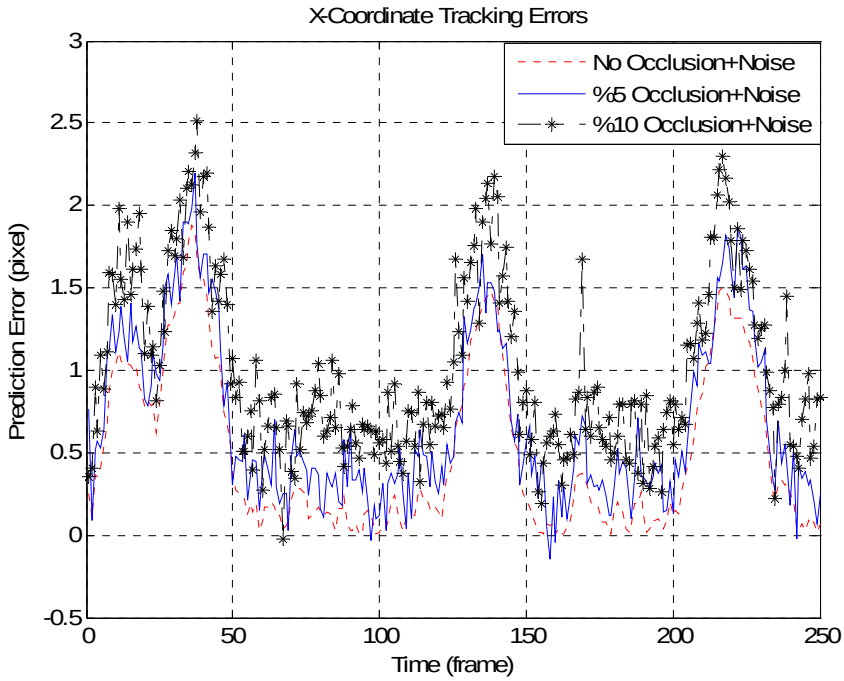


Figure 5-11 X-Coordinate Tracking Error for Different Situations

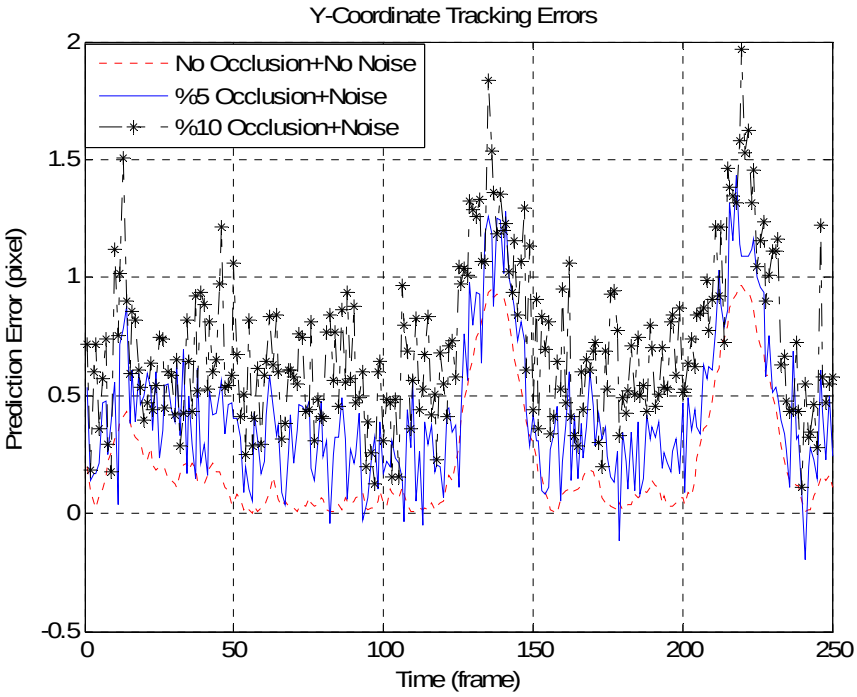


Figure 5-12 Y-Coordinate Tracking Error for Different Situations

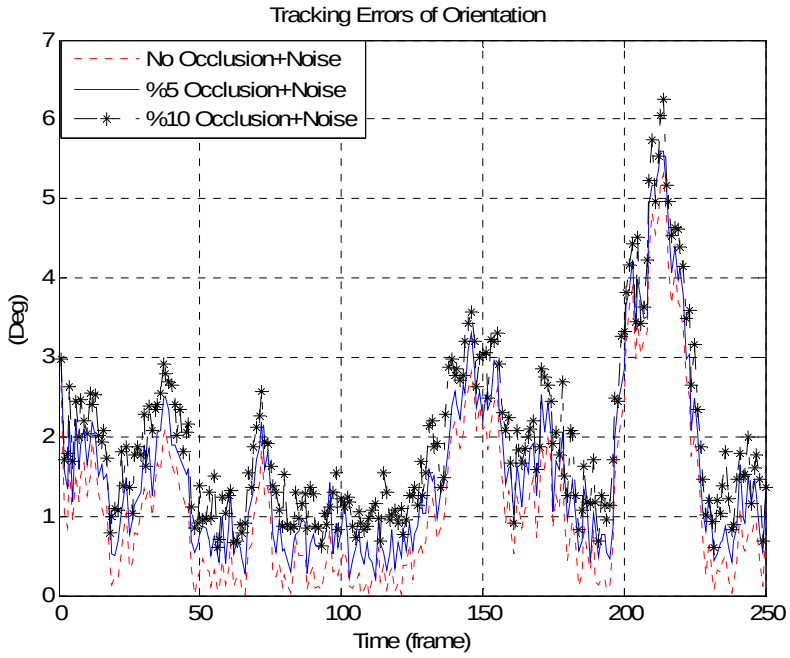


Figure 5-13 Y-Coordinate Tracking Error for Different Situations

The orientation measurement method together with the interpolation property of implicit polynomials provides the robustness of predictions against occlusion and noisy boundary data.

5.3.2 Assessment of Results

A new method is proposed for tracking the position and the orientation of 2D free-form objects undergoing rigid motion. By using the fact that the related points undergo the same motion with the curve, we have employed a robust orientation measure for the curve. Tracking approach was aiming to reduce the number of computations and was quite successful.

5.4 Fitting Only at First Frame & Using Algebraic Curve Spaces

In this part, a new method is proposed to do fitting once offline and calculate an algebraic curve space. Then, in every frame, algebraic curves from the search region of curve space are evaluated with the extracted edge points. The curve that has the smallest error according to some error metric is chosen to be the fit for that frame. The algorithm presented is for tracking a free-form shaped object, moving along an unknown trajectory, within the camera's field of view (FOV). A discrete steady-state Kalman filter estimates the future position and orientation of the target object and provides the search area of curve space for the next frame. For initialization of the Kalman filter we used the "related points" extracted from the decomposition of algebraic curves which represent the target's boundary and measured position of target's centroid. Related points undergo the same motion with the curve, hence can be used to initialize the orientation of the target. Proposed algorithm is verified with simulations. The steps involved in the proposed scheme are summarized by the block diagram of Figure 5-9.

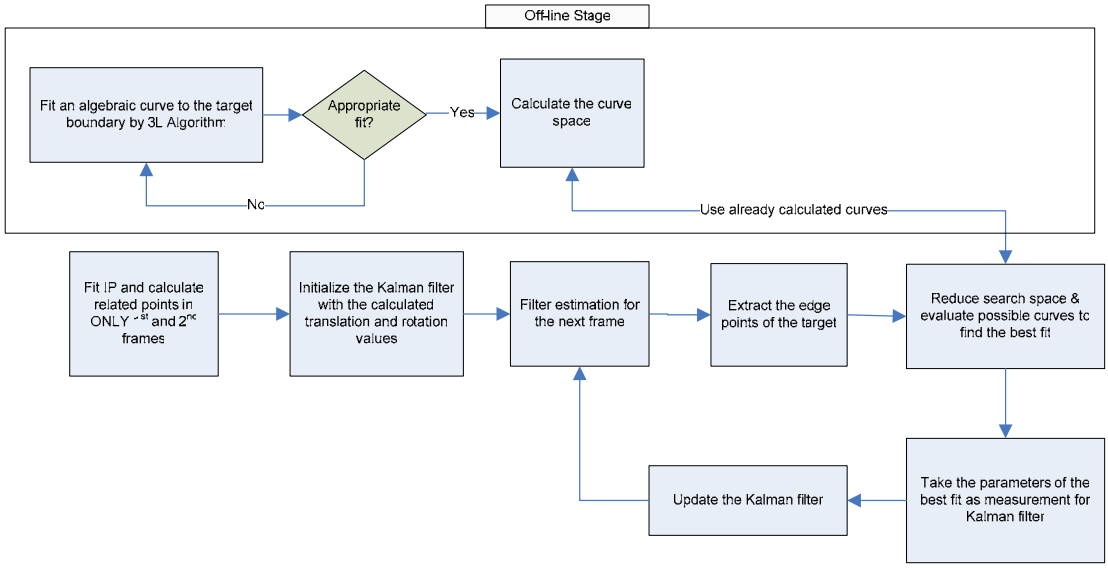


Figure 5-9 Complete diagram for the proposed algorithm

5.4.1 Algebraic Curve Spaces

Instead of fitting an algebraic curve to the boundary of the target at each frame, selecting a suitable curve among possible candidates decreases the computational complexity drastically. A curve space for the given target can found by first fitting an algebraic curve to the target's boundary offline and computing all possible Euclidean transformations, rotations and translations, of that polynomial. Polynomials in the curve space is calculated using the Euclidean mappings of the first implicit polynomial fitted to the boundary data of the target.

5.4.1.1 Euclidean Mappings of Algebraic Curve

A Euclidean transformation, E is defined by both a rotation R and a linear translation T; i.e.

$$\begin{bmatrix} x \\ y \end{bmatrix} = \underbrace{\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}}_{R} \begin{bmatrix} \bar{x} \\ \bar{y} \end{bmatrix} + \underbrace{\begin{bmatrix} t_x \\ t_y \end{bmatrix}}_{T} \Rightarrow \underbrace{\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}}_{X} = \underbrace{\begin{bmatrix} \cos\theta & -\sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix}}_{E} \underbrace{\begin{bmatrix} \bar{x} \\ \bar{y} \\ 1 \end{bmatrix}}_{\bar{X}} \quad (5.23)$$

The mathematical relationship defined by (5.11) will be abbreviated as $X \xrightarrow{E} E\bar{X}$, where R is an orthogonal(rotation) matrix, so that $R^T R = RR^T = I$. In general, any two n-th degree curves, defined by a monic $f_n(x,y) = 0$ and a monic $\bar{f}_n(\bar{x},\bar{y}) = 0$, which outline the boundary of the same object in two different configurations will be Euclidean equivalent if:

$$f_n(x,y) = 0 \xrightarrow{E} f_n(\cos\theta\,\bar{x} - \sin\theta\,\bar{y} + p_x, \sin\theta\,\bar{x} + \cos\theta\,\bar{y} + p_y) = s_n \bar{f}_n(\bar{x},\bar{y}) = 0 \quad (5.24)$$

Instead of using the whole curve space as the search region for the next frame, estimated translation and orientation values from the Kalman filter is used to reduce the search region.

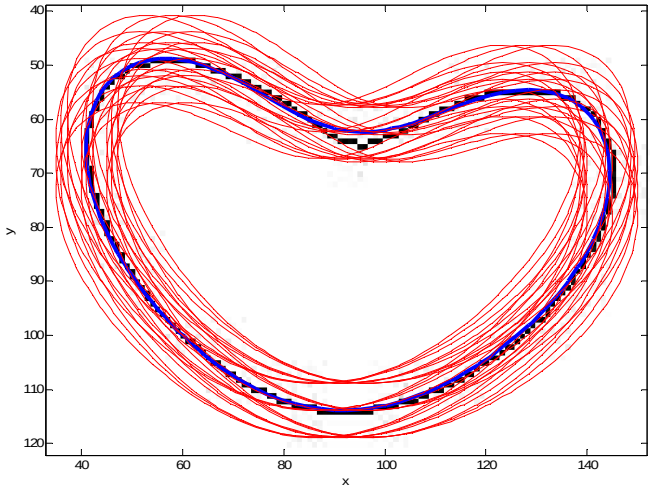


Figure 5-10 Algebraic curves in the search region for frame 50 and the selected curve with smallest error term

5.4.2 Error Metrics

For the evaluation of algebraic curves within the search region of our curve space, we use the sum of squared distances from the data points to the algebraic curves. For a collection of data points Γ, error is calculated as

$$\sum_{(x,y)\in\Gamma} f^2(x,y) \tag{5.25}$$

Other distance measures can also be employed for the evaluation of curves. For example, the sum of absolute distances from data points to the implicit curve is given as

$$\sum_{(x,y)\in\Gamma} |f(x,y)| \tag{5.26}$$

Yet another distance measure which approximates the true geometric distance can be calculated as follows:

$$\sum_{(x,y)\in\Gamma} \frac{|f(x,y)|}{\|\nabla f(x,y)\|} \tag{5.27}$$

5.4.3 Experimental Results

Again we have used the Boomerang shaped object undergoing a rigid motion with same trajectory for comparison purposes. Object boundaries have been modelled by quartic curves. For the initialization of the Kalman filter, related points of curves are obtained from the decomposition of the curve. X-coordinate of the target centroid is predicted with filter and the illustration of the trajectory of coordinates and the predicted values are shown in Figure 5.10. There is again an overshoot when tracking fast changes in the x direction. A better illustration of the performance can be seen in an error comparison. The objective is to track the measured signal, so it is assumed that the measure is the true coordinate position. So, the error is the difference between the predicted and measured value. The error values are low and within a band of (-/+)0.5 pixels when the target performs relatively uniform motion. Maneuver is again a problem with prediction filter. It takes some time for the error values to converge to normal values after the target makes a maneuver.

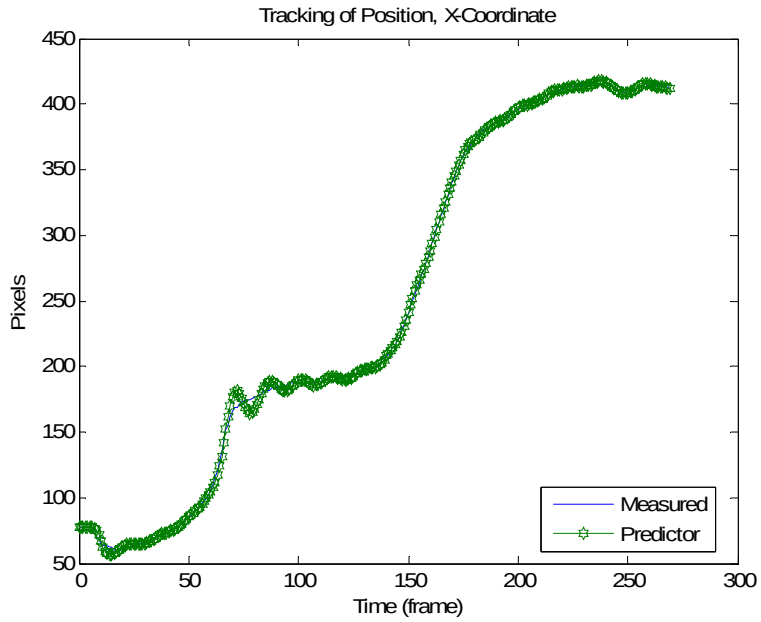


Figure 5-11 X-Coordinate Tracking of the Target

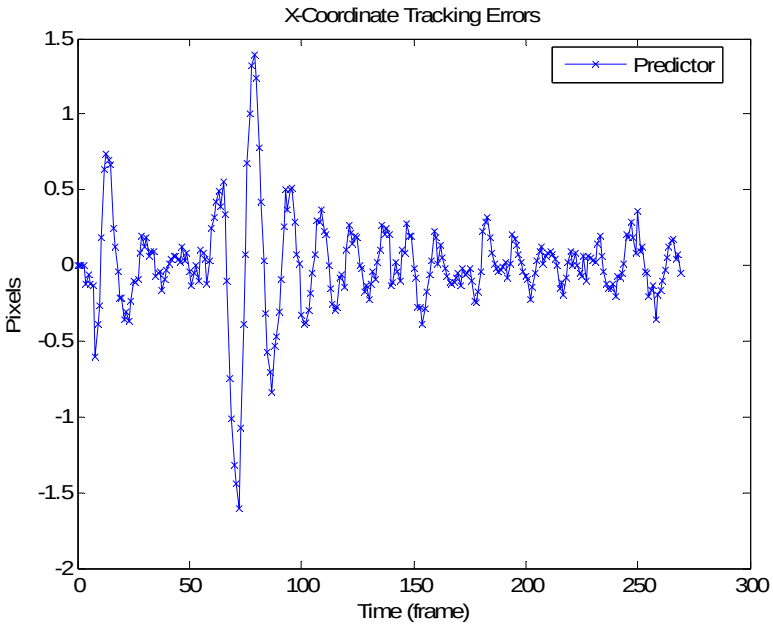


Figure 5-12 X-Coordinate Tracking Error of the Filter

The y-coordinate performance measure are shown in Figures 5.12 and 5.13. In this case errors may reach values of 2 pixels due to the sharp maneuvers that the object performs in y direction. However, as in the case of x-coordinate when the manevuer is over error values turn to their normal values.

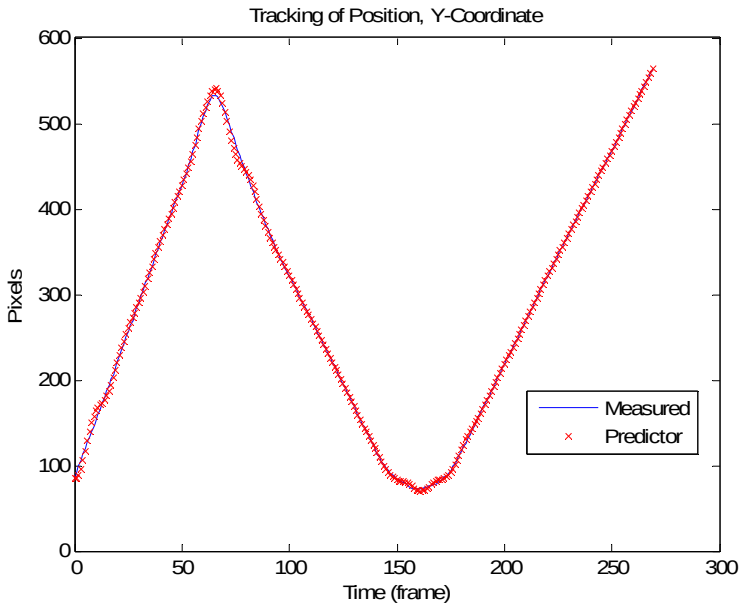


Figure 5-13 Y-Coordinate Tracking of the Target

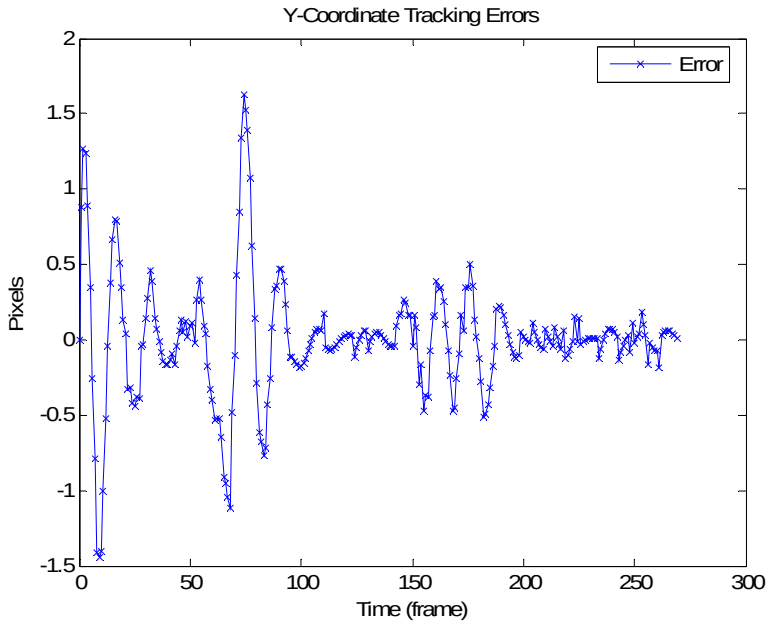


Figure 5-14 X-Coordinate Tracking Error of the Filter

Figure 5.12 shows the predictors ability to predict the orientation. Orientation prediction results are better in this method since we have eliminated the steps of orientation measurement which is noise producing by calculating all the possible curves offline.

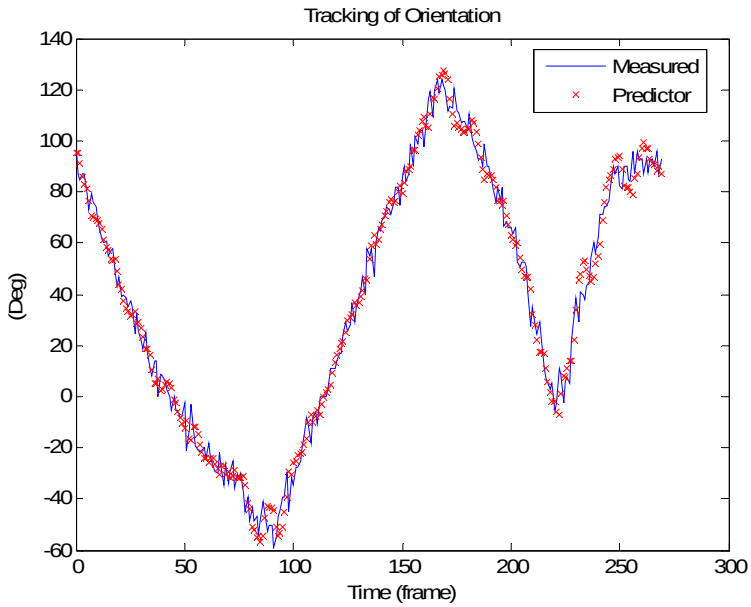


Figure 5-15 Orientation Tracking of the Filter

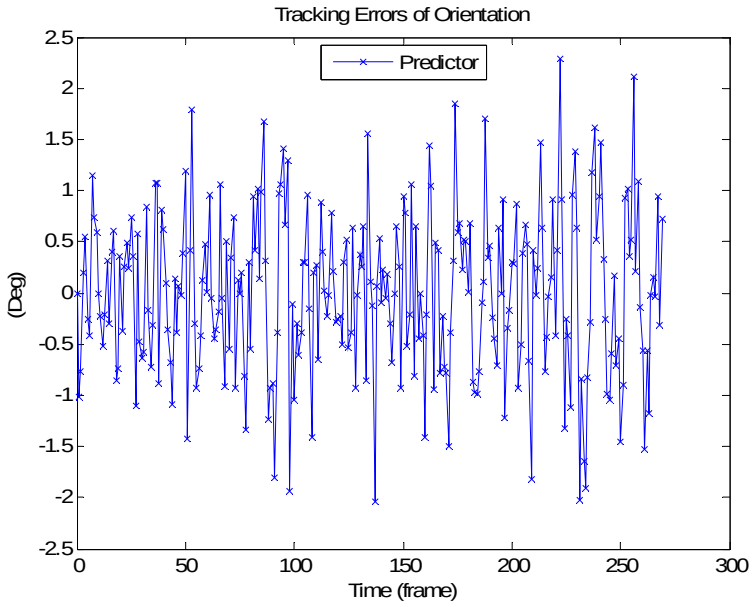


Figure 5-16 Orientation Tracking Error of the Filter

5.4.4 Assessment of Results

A new method is proposed for tracking the position and the orientation of 2D free-form objects undergoing rigid motion. By using the fact that the related points undergo the same motion with the curve, we have employed a robust initialization for the Kalman filter. Reducing the curve-space into a lower dimensional one and calculating the possible curves offline eliminate the computational burden of fitting algorithms. Results are promising and computational burden of the fitting algorithms is reduced more than the previous method since the fitting is done only once.

5.5 Timing Considerations

The main objective of the two proposed methods above was eliminating the need for computationally expensive fitting algorithm to be applied at each frame of the video sequence. While trying to decrease the number of times the fitting algorithms applied, the average time spend on each frame is decreased. For comparison purposes, the typical time requirements for each phase of the tracking procedure are presented in the following tables.

FUNCTION	TIME (MSEC)
Grab the Frame	121
Extracting the Boundary Data	40
3L Fitting to Data	8260
Parameters (orientation & cenroid) + Related Point Calculation	102
TOTAL	**8523**

Table 5.1: Timing Requirements for Fitting in Every Frame Method

FUNCTION	TIME (MSEC)
Grab the Frame	121
Extracting the Boundary Data	40
3L Fitting to Data (frame average)	1652
Parameters (orientation & cenroid) + Related Point Calculation	102
Prediction (Kalman Filter)	52
TOTAL	**1967**

Table 5.2: Timing Requirements for Fitting only in Certain Frames Method

The first table represents the computational burden of the fitting algorithm if it is applied in every frame of the sequence for the calculation of orientation and center of mass for the target. Table 5.2 indicates a decrease in the computational complexity by introducing a less time consuming prediction procedure, namely Kalman filtering. Average time spend for 3L fitting is presented since between the certain frames where measurement is taken fitting algorithm does not run. Average time spend for each frame decrease more when the method presenting the algebraic curve spaces are introduced. Time spend on fitting, related point and parameter calculations does not affect the computational complexity. Yet, those phases of operation are done only at first and the second frames of the whole sequence. The only extra term added in terms of complexity is the evaluation of the polynomials in the search region with the

boundary data points. However, it is nearly %10 of the fitting algorithm comparing the timing results, decreasing the overall time drastically.

FUNCTION	TIME (MSEC)
Grab the Frame	121
Extracting the Boundary Data	40
3L Fitting to Data (only 1St & 2nd frames)	---
Parameters (orientation & cenroid) + Related Point Calculation	---
Evaluation of Boundary Data in the Search Region of Curve Space (avg. 100 polynomials)	850
Prediction (Kalman Filter)	52
TOTAL	**1063**

Table 5.3: Timing Requirements for Fitting only for initialization of the Kalman filter

5.6 *Implicit Polynomials Used With Particle Filters & Online Appearance Model*

In [19], Zhou et. Al. proposed a robust visual tracking algorithm that incorporates appearance-adaptive models in a particle filter. Online appearance model(OAM) proposed in [20] as an appearance adaptive model. The original algorithm in [19] is modified and the implicit polynomial representation is embedded as a representative of the target region. By this approach, boundary curve of the target region is tracked with the help of fitted implicit polynomial. Moreover, the points inside the polynomial are used to define the target model in order to obtain an exact representation of the region to be tracked.

5.6.1 Proposed Algorithm

The approach we adopt here uses an appearance model, A_k. The observation equation used is as follows

$$z_k = T\{Y_k; x_k\} = A_k + v_k \tag{5.28}$$

Where z_k is the image patch of interest in the video frame Y_k, parameterized by x_k. Two extreme cases can be adopted with the template A_k. One can choose a fixed template $A_k = A_0$ and try to minimize a cost function in the form of sum of squared distance by comparing the fixed target with observations. On the other hand, the best patch of the previous frame can be used as a new template, i.e. $A_k = \hat{z}_{t-1}$, which becomes a rapidly changing model. While, a rapidly changing model is at the risk of drifting, a fixed template can not handle appearance changes in the video. Hence, there is a need for a template which can adapt itself to dynamic situations.

5.6.2 Adaptive Observation Model

The original OAM uses a mixture density of components for representation of a model. Three components are used, which are S-component, W-component and L-component. S is the component for stable model, which is intended to capture the behavior of temporally stable image observations when and where they occur. W-component, called the wandering component characterizes two-frame variations. L-component referred as lost component accounts for data outliers, which are expected to arise due to occlusion. In stead of L-component, F-component which is a fixed template that is expected to observe most often is used.

The appearance model at time k, $A_k = \{W_k, S_k, F_k\}$ is a time-varying one that models the appearances seen in all observations up to that time. The appearance is a mixture of Gaussians, with W_k, S_k, F_k as mixture centers $\{\mu_{i,k}; i = w, s, f\}$ and the corresponding variances $\{\sigma^2_{i,k};\ i = w, s, f\}$ and mixing probabilities $\{m_{i,k}; i = w, s, f\}$. The components of the appearance model are images consisting of d pixels and can

be represented with their mean, variance and mixing probabilities as $\{m_{i,k}, \mu_{i,k}, \sigma_{i,k}^2;\ i = w, s, f\}$. The observation likelihood is given as

$$p(Y_k \mid x_k) = p(z_k \mid x_k) = \prod_{j=1}^{d} \{ \sum_{i=w,s,f} m_{i,k}(j) N(z_k(j); \mu_{i,k}(j), \sigma_{i,k}^2(j)) \} \qquad (5.29)$$

Where $N(z; \mu, \sigma^2)$ represents a normal density distribution. The important issue in OAM is the model update stage. New mixing probabilities, mixture centers and variances for time $k+1$, $\{m_{i,k+1}, \mu_{i,k+1}, \sigma_{i,k+1}^2;\ i = w, s, f\}$ need to be computed in order to update the current appearance model A_k to A_{k+1} after the observation $\hat{z}_k$ becomes available. Current appearance model is not updated according to all the past observations. The past observations are exponentially forgotten. Exponential envelope is denoted by $\mathrm{E}_k(m) = \alpha \exp(-\tau^{-1}(k-m))$ for $m \le t$, where $\tau = n_h / \log 2$, n_h is the half-life of the envelope in frames, and $\alpha = 1 - \exp(-\tau^{-1})$.

Each pixel is treated separately in this model, since they are assumed to be independent of each other. The computations for updating the appearance model are as follows:

Posterior responsibility probabilities are calculated as

$$o_{i,k}(j) \propto m_{i,k}(j) N(\hat{z}_k(j); \mu_{i,k}(j), \sigma_{i,k}^2(j));\ i = w, s, f,\ \&\ \sum_{i=w,s,f} o_{i,k}(j) = 1 \qquad (5.30)$$

Where $j = 1, 2, ..., d$ and d is the number of pixels in the appearance model. The mixing probabilities are updated as

$$m_{i,k+1}(j) = \alpha\, o_{i,k}(j) + (1-\alpha)\, m_{i,k}(j);\ i = w, s, f \qquad (5.31)$$

And the first and second moment images $\{M_{p,k+1};\ p = 1,2\}$ are evaluated as

$$M_{p,k+1}(j) = \alpha\, \hat{z}_k^p(j) o_{s,k}(j) + (1-\alpha)\, M_{p,k}(j);\quad p = 1, 2 \qquad (5.32)$$

Finally, the mixture centers and the variances are updated as:

$$S_{k+1}(j) = \mu_{s,k+1}(j) = \frac{M_{1,k+1}(j)}{m_{s,k+1}(j)}, \quad \sigma^2_{s,k+1}(j) = \frac{M_{2,k+1}(j)}{m_{s,k+1}(j)} - \mu^2_{s,k+1}(j) \qquad (5.33)$$

$$W_{k+1}(j) = \mu_{w,k+1}(j) = \hat{z}_k(j), \ \sigma^2_{w,k+1}(j) = \sigma^2_{w,1}(j), \qquad (5.34)$$

$$F_{k+1}(j) = \mu_{f,k+1}(j) = F_1(j), \ \sigma^2_{f,k+1}(j) = \sigma^2_{f,1}(j) \qquad (5.35)$$

The initialization is done manually. User select a number of points on the boundary of the target and an implicit polynomial is fit at the first frame. A_1 is initialized with T_o and $W_1 = S_1 = F_1 = T_0$, $M_{1,1} = m_{s,1}T_{0,}$ $M_{2,1} = m_{s,1}\sigma^2_{s,1} + T_0^2$.

Besides the adaptive appearance model, noise terms and the number of particles are adaptive parameters in the proposed algorithm. Noise term is changed adaptively according to the quality of prediction. Process noise w_k is used in the form of $w_k = r_k * w_0$ where r_k is a function of error term ε_k.

$$\varepsilon_k = \phi(\tilde{z}_k, A_k) = \frac{2}{d}\sum_{j=1}^{d}\{\sum_{i=w,s,f} m_{i,k}(j)\rho(\frac{\tilde{z}_k(j) - \mu_{i,k}(j)}{\sigma_{i,k}(j)})\} \qquad (5.36)$$

The quality of prediction can be represented by the value of ε_k. If the prediction is successful which means the value of ε_k is small, time-update equation 6.1 needs a noise term with small variance. On the other hand, if the prediction is unsuccessful implying a large ε_k then a noise term with large variance is needed in order to compensate possible jumps in the motion state. Accordingly, r_k is calculated at each time step as follows

$$r_k = \max(\min(r_0\sqrt{\varepsilon_k}, r_{\max}), r_{\min}) \qquad (5.37)$$

Where $r_{\min}$ is aimed to give a lower bound for the sample coverage and $r_{\max}$ is aiming to limit the computational load as an upper bound.

The number of particles are also adaptively changed according to r_k. More particles need to be used when r_k is large. On the other hand, when noise variance is small less particles will be enough. Hence, the number of particles J_k is calculated each time step according to

$$J_k = (J_k r_k)/r_0 \qquad (5.38)$$

5.6.3 Experimental Results

Motion in the experiments is represented with state vector $x = (a, b, c, d, t_x, t_y)$,an affine transformation in which $\{a, b, c, d\}$ are deformation parameters and $\{t_x, t_y\}$ are translation parameters. Image intensities are used in the computations. In order to compensate for contrast variations zero mean unit variance normalization is applied to the intensity values.

The particle filter and appearance model is initialized by selecting boundary points of the target and fitting a fourth degree implicit polynomial to the points. The target region is well represented with the fitted polynomial and the region inside it. The image transformation in equation 5.28 $T\{Y; x\}$ is realized as applying the affine transformation defined in x to the implicit polynomial and then cropping out the region represented with the new polynomial.

A tank recorded from an air-vehicle is tracked with the proposed approach. The video is gray-scale and contains 250 frames. The trajectory of the centroid of the tracked tank is not smooth and arbitrary due to the motion of the camera with respect to the tank. Some of the frames from the tracking results are shown below.

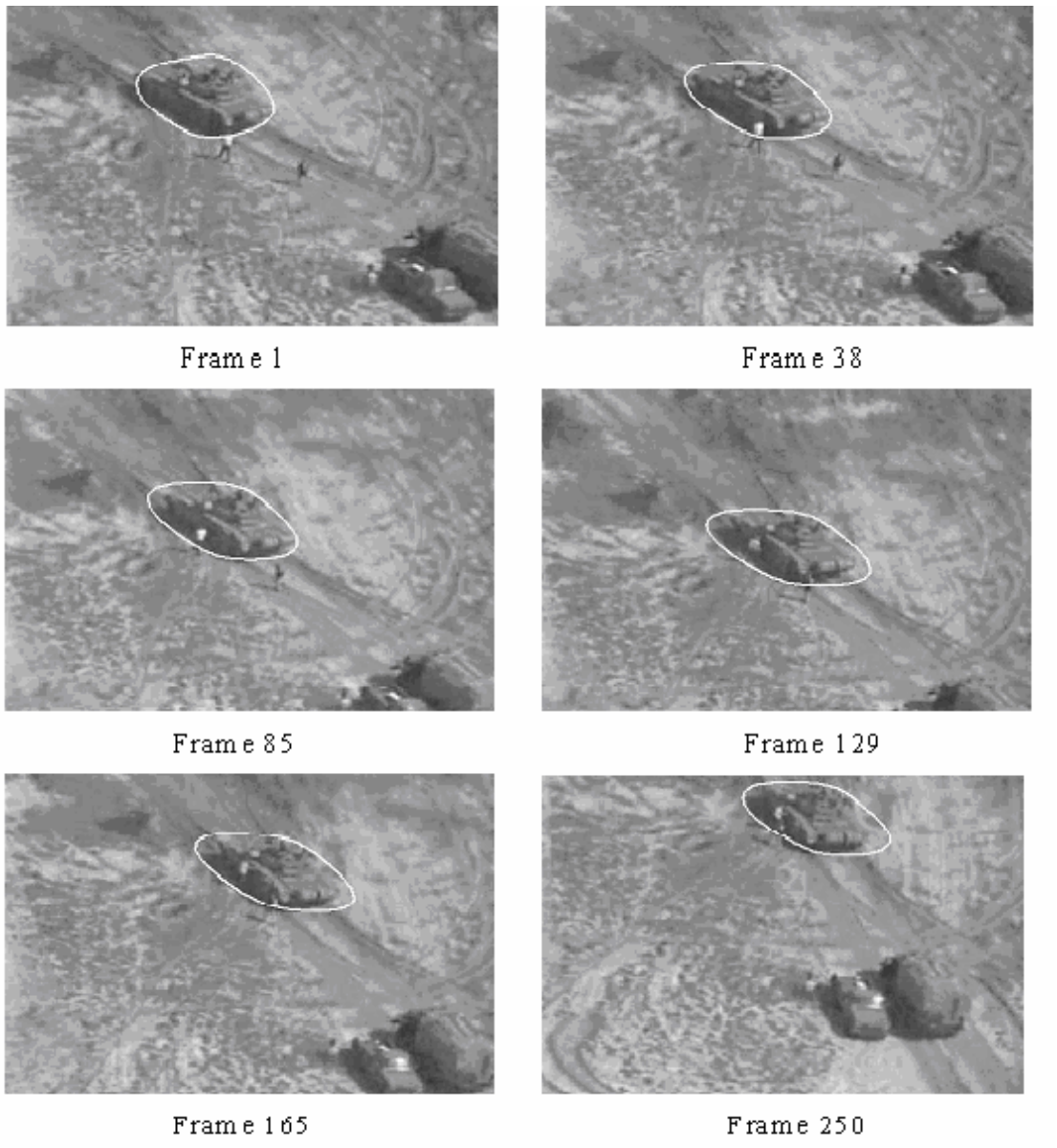


Figure 5-177 Selected Frames of the Tank Tracking by Implicit Polynomial Representation

Video includes not only tank but also other vehicles and people walking around the tank. The arbitrary 2D trajectory of the tank is drawn on to the first frame of the sequence and shown in the following figure.

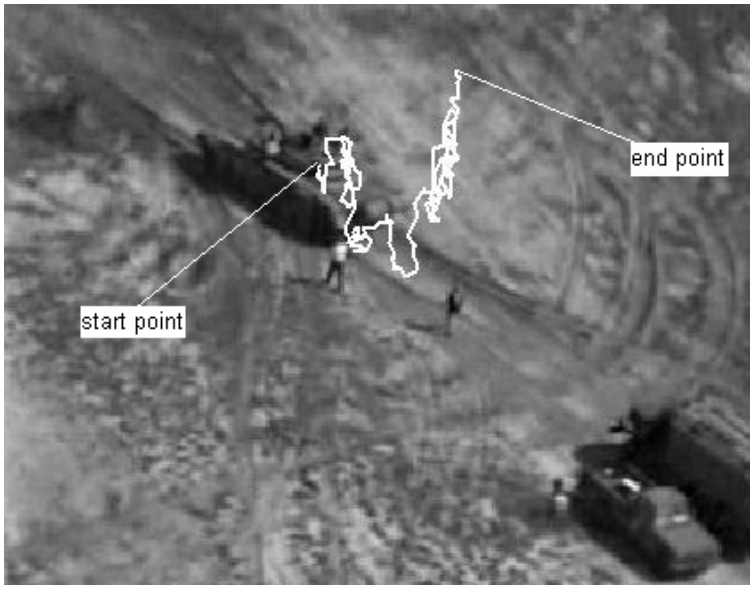


Figure 5-18 2D Trajectory of the Tank's Centroid

Number of particles used in each frame as a function of frame number is also shown in figure 5.19. 120 particles were used to initialize the filter.

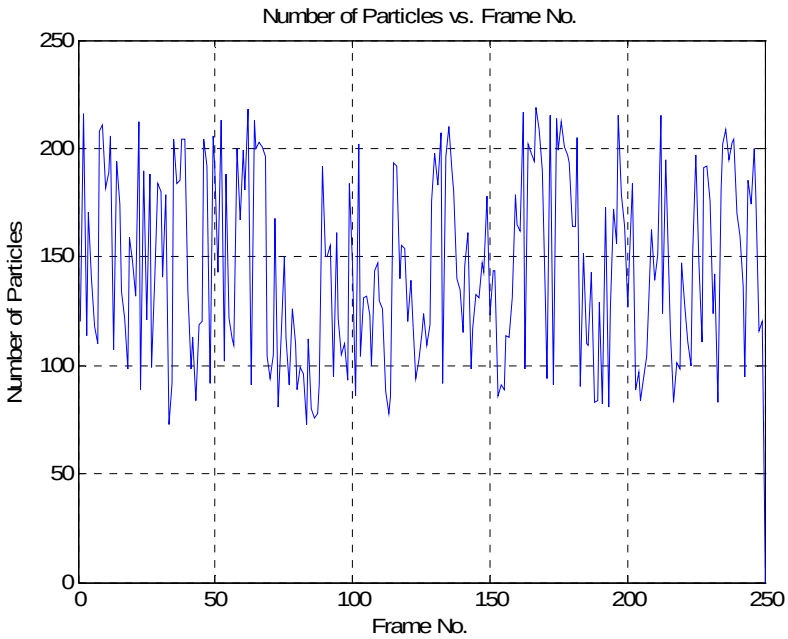
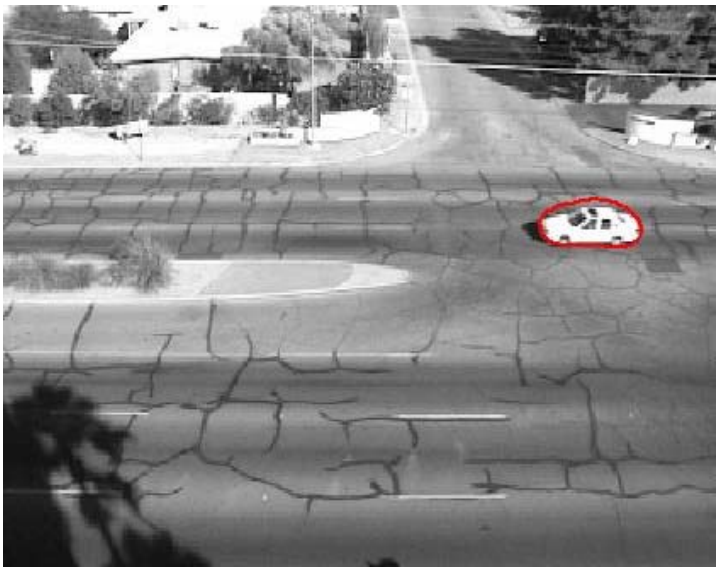


Figure 5-19 Number of Particle Variation as a Function of Frame Number

Second video used for testing contains a side view of a car moving along a road. The movement of the car is relatively linear. However, the tough part is that some objects by the road is very similar to the car in terms of color and pattern.

Hence, it was very probable that the tracker catches those objects instead of the car in some time and stuck there. However the tracking is still successful. Video is again gray-scale and contains 60 frames. Some of the frames from the tracking results are shown below.



Frame 1

Frame 10

Frame 30

Frame 45

Frame 60

Figure 5-20 Selected Frames of the Car Tracking by Implicit Polynomial Representation

Video includes not only tank but also other houses with similar pattern and color with car and also trees and other objects. Although the movement of the car starts inside the scene, it goes out of the scene at the end of the sequence.

The filter is initialized with 100 particles in this case and through the operation of the filter less number of particles is needed since the movement of the car is fairly linear and no outlier comes in front of the target object. When the target starts to leave the scene number of particles used increases adaptively.

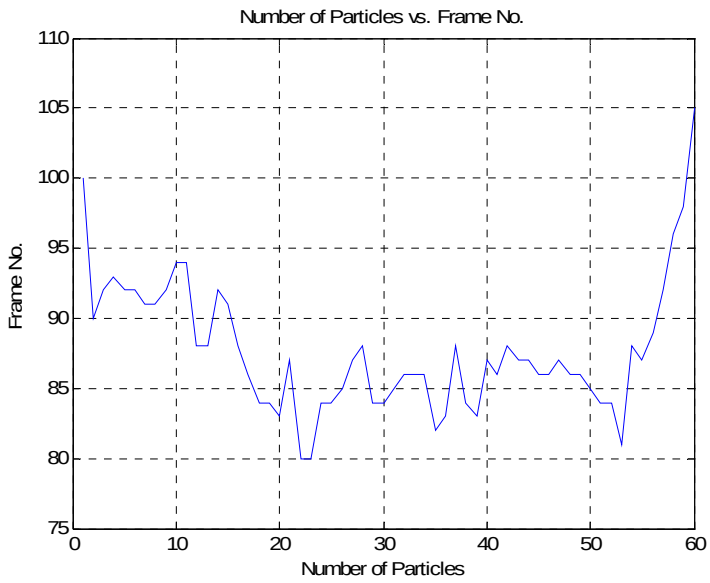


Figure 5-21 Number of Particle Variation as a Function of Frame Number

5.6.4 Assessment of Results

We have proposed a different usage of implicit polynomials in the context of visual tracking in real videos. The target region is represented by the implicit polynomial fitted at the first frame. The points inside the polynomial are used to define the target model in order to obtain an exact representation of the region to be tracked. Boundary curve of the target region is also tracked with the help of fitted implicit polynomial.

6 CONCLUSION

In this thesis, different approaches in order to efficiently apply the powerful implicit algebraic 2D curve representation to the phenomenon of visual tracking are experimented. Through the proposed concepts and algorithms, the computational burden of fitting algorithms is reduced.

The first approach was to do fitting only for certain frames in an image sequence and fill in the missing ones using Kalman filtering technique. A discrete steady-state Kalman filter is used to estimate the future positions and orientation of the target object. The orientation measurement technique which is shown by experiments to be robust against different causes of noise allows the proposed approach to be succesful.

In the second proposed method, the computational burden is tried to be reduced further. In order to achieve this, the fitting is done once offline and an algebraic curve space is calculated. Then, in every frame, one curve from the search region of curve space that has the smallest error according to some error metric is chosen to be the best fit for that frame. If the first fitted polynomial is proper, fitting in all other frames is guaranteed to be proper.

The third aproach was to apply a region-based method using appearance-adaptive models and particle filters which make complete use of all available image intensity information. By employing implicit algebraic curves, the boundary of the target could be tracked and also the model could be adapted to fit inside an implicit curve rather than a rectangle or a priori known geometric shape.

Further work on trying to evolve the implicit algebraic curve during the tracking process as in active contour based applications will probably yield a more effective usage of this representation in this context.

REFERENCES

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

2. 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.

3. 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.

4. G. Taubin, "Estimation of planar curves, surfaces and non-planar space curves defined by implicit equations, with applications to edge and range image segmentation," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 13, No. 11, 1991.

5. D. Keren et al., "Fitting curves and surfaces to data using constrained implicit polynomials," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 23, No. 1, January 1999.

6. H. Civi, C. Christopher, A. Ercil, "The Classical Theory of Invariants and Object Recognition Using Algebraic Curve and Surfaces," Journal of Mathematical Imaging and Vision 19: 237–253, 2003.

7. Z. Lei, M. M. Blane and D. B. Cooper, "3L Fitting of Higher Degree Implicit Polynomials ," In proceedings of 3rd IEEE Workshop on Applications of Computer Vision, pp. 148-153, Florida 1996

8. H. Civi, *Implicit Algebraic Curves and Surfaces for Shape Modeling and Recognition*, Ph.D. thesis, Bogaziçi University, October 1997.

9. T. Sahin, M. Unel, "Globally stabilized 3L curve fitting" Proceedings of International Conference on Image Analysis and Recognition (ICIAR 2004), Porto, Portugal, September 2004.

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. R.E. Kalman, "A new approach to linear filtering and prediction theory" Transactions of the ASME – Journal of Basic Engineering Vol. 82: pp.35-45, 1960

12. D. Simon, *Kalman Filtering,* Embedded.com, viewed on Jun. 28, 2005, http://www.embedded.com/showArticle.jhtml?articleID=9900168 (2005)

13. L.D., Brock, G.T. Schmidt, General questions on Kalman filtering in navigation systems, Chapter 10 of "Theory and Applications of Kalman Filtering" C.T. Leondes, Editor, NATO AGARD (1970)

14. P. Gutman, M. Velger, "Tracking targets using adaptive Kalman filtering" IEEE Transactions on Aerospace and Electronic Systems Vol. 26, No. 5: pp. 691-699 , 1990

15. H.W. Sorenson, Comparison of Kalman, bayesian and maximum likelihood estimation techniques, Chapter 6 of "Theory and Applications of Kalman Filtering" C.T. Leondes, Editor, NATO AGARD (1970)

16. G. Welch, G. Bishop, " An introduction to the Kalman filter" ACM SIGGRAPH 2001, Course 8

17. M. Unel. "*Polynomial Decompositions for Shape Modeling, Object Recognition and Alignment*" PhD Thesis, Brown University, Providence, RI 02912, May 1999.

18. A. Jazwinski, "Stochastic Processes and Filtering Theory" Academic, New York, 1970

19. S. Zhou, R. Chellappa, and B. Moghaddam. "Visual tracking and recognition using appearance-adaptive models in particle filters" IEEE Transactions on Image Processing (IP), Vol. 11, pp. 1434-1456, November 2004.

20. A. D. Jepson, D. J. Fleet, and T. El-Maraghi,, "Robust Online Appearance Models for Visual Tracking," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 25, No. 10, pp. 1296-1311, October 2003.

21. C. Robert, G. Casella, "Monte Carlo statistical methods" Springer, New York, 1999

22. M. Arulampalam, S. Maskell, N. Gordon, T. Clapp, "A tutorial on particle filters for online non-linear/non-Gaussian Bayesian tracking," IEEE Transactions on Signal Processing, Vol. 50, No. 2, pp. 174-188, February 2002.

23. N. J. Gordon, D. J. Salmond, A. F. M. Smith, "Novel approach to nonlinear/non-Gaussian Bayesian state estimation," IEE Proceedings for Radar and Signal Processing, Vol. 140, No. 2, pp. 107-113, April 1993.

24. M. Pitt, N. Shephard, "Filtering via simulation: auxiliary particle filters," Journal of American Statistical Association, Vol. 94 , 1999