INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI

A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor MI 48106-1346 USA
313/761-4700 800/521-0600

Dynamic Object Localization via a Proximity Sensor Network

Gregory Allen Petryk

Department of Mechanical Engineering

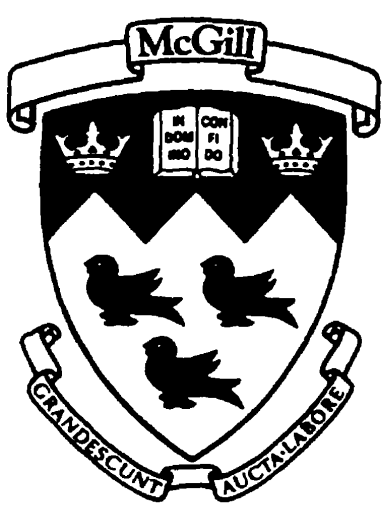
McGill University, Montreal

August, 1996

A Thesis submitted to the Faculty of Graduate Studies and Research

in partial fulfilment of the requirements of the degree of

Master of Engineering



Abstract

Autonomous robotic operation in an unstructured or partially known environment requires sensing and sensor-based control. To overcome the problems with current "eye-in-hand" systems, miniature amplitude-based, infra-red proximity sensors are being studied. Obtaining position and velocity estimates of a rigid body with these sensors is a non-linear parameter and state estimation problem. Among the methods examined in simulation, Extended Kalman Filtering (EKF) was selected for implementation. A novel approach for object localization was developed in which the object geometry is known, sensing is performed by a proximity sensing network (PSN) and the object's unknown reflective properties are estimated on-line. The method has been tested extensively in simulation and experiments in which a target object's planar position and velocity were successfully estimated. To the author's knowledge this is the first time amplitude based infra-red sensors have been used to estimate a rigid body's unknown trajectory.

Résumé

Un robot doit utiliser une commande à base de capteurs pour exécuter des tâches autonommes dans un environnement non-structuré. Pour éviter les problèmes associés avec les systèmes "caméra-en-main" d'aujourd'hui, des mini-capteurs d'intensité de lumière infra-rouge sont étudiés. Obtenir la position et la vitesse d'un objet avec ces capteurs est un problème d'estimation non-linéaire de paramètres et d'états. Parmi les méthodes d'estimation examinées en simulation, celle choisie pour application fût le filtre Kalman étendu (EKF). Une nouvelle méthode a été développé où nous utilisons un résau de capteurs de proximité (PSN) pour estimer la position, la vitesse et les proprietés de surface d'un objet dont la géométrie est connue. Des expériments où l'estimation fût accompli pour un objet manoeuvrant en 2D démontrent la practicabilité du système. Aux conaissances de l'auteur, ce fût la première foix que des capteurs pareils sont utilisés pour identifier la trajectoire inconnue d'un objet.

Acknowledgements

I would first like to acknowledge the financial support offered to me by the Quebec Government through its *Fonds pour la Formation de Chercheurs et l'Aide à la Recherche* (FCAR) scholarship program.

Much of the motivation I had for pursuing graduate work was provided by the other graduate students working in the ARL/AML laboratory. There are four people in particular I would like to thank. First, I would like to thank John Damianakis for building all the hardware on which my sensor fusion algorithms were validated. I hope he recovers from the hours spent soldering flea-sized components to other flea-sized components. Second, I thank Joey Mennitto for showing me what dedication can produce: beauty can truly be carved from metal. Third, I thank Mojtaba Ahmadi for teaching me that there is a science to the art of debugging. And fourth I thank Pedro Gregorio for his often surprising demonstrations that to be a roboticist one must be able to a true "jack-of-all-trades": designer, programmer, machinist, anatomist, electrician and musician.

Finally I wish to extend my gratitude to my advisor, Dr. Martin Buehler, for his unwavering patience and his sound guidance.

Contents

Chapter 1

Introduction

Object Localization

The determination of an object's location is basic to human interaction with the ever-changing outside world. The acquisition of this knowledge is necessarily sensor-based. Research in systems which mimic human object localization capabilities has produced such technologies as sonar, radar and forward-looking infra-red (FL-IR). These sensors have achieved extremely high levels of sophistication. Systems incorporating satellite mounted sensors are now capable of tracking maneuvering air and ground targets at the horizon and beyond. These systems are often composed of several independent sub-systems whose data streams must be fused to provide a single accurate estimate [13]. A typical example is depicted in Fig. 1.1.

For robots to operate autonomously in unstructured environments, they must be provided with similar capabilities. Object recognition is a well established field of study [21]. Typical systems have one or several overhead cameras and/or laser-range finders which act in unison to provide the robot with a model of its environment [1, 2].

Large vehicle tracking systems and smaller robotic systems are similar in many respects and thus suffer from the same problems. Of these, one of the most important is that, to acquire useful information, the object must be in view of the sensors: the

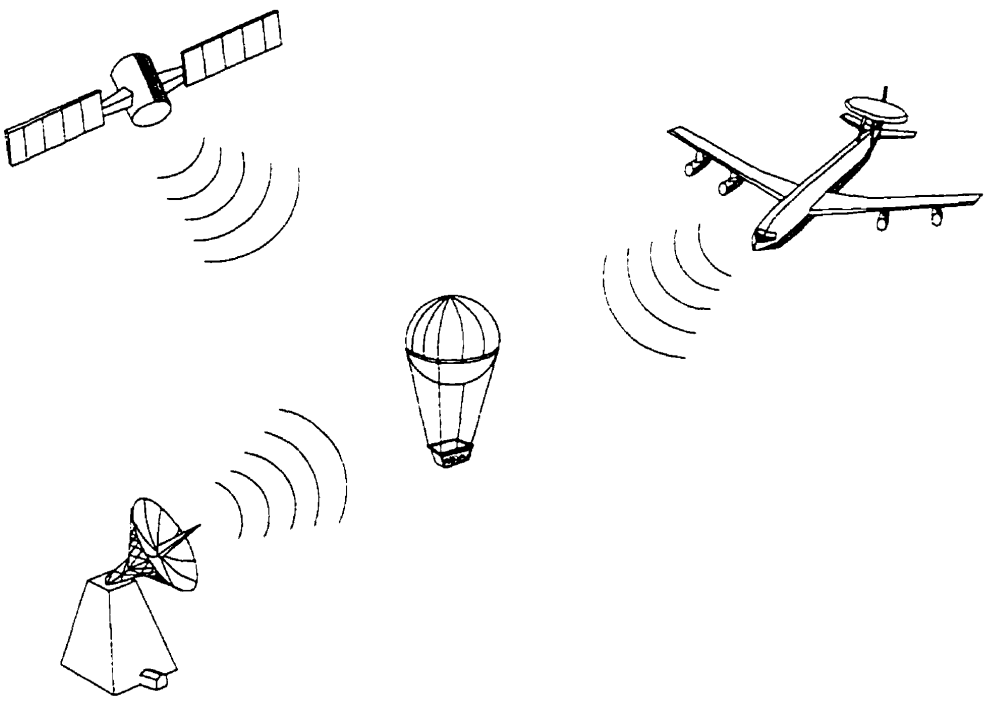


Figure 1.1: *Tracking a free flying object with a multi-platform sensor network.*

sensor's view must not be *occluded* by another object. For vehicle tracking systems, the approach has been to provide the sensor with an increasingly global point of view. The Airborne Warnings And Control Systems (AWACS) systems and satellites shown in Fig. 1.1 are examples of this. For robotic manipulators, "camera-in-hand" systems have been introduced to minimize the impact of occlusion. However, attaching directly to the end-effector is problematic due to space limitations and increased manipulator torque demands. Furthermore, since the camera is often attached to the link preceding the end-effector, it may still be occluded by the end-effector itself.

Infra-red Proximity Sensing

To avoid occlusion, proximity sensors could be embedded directly within the surface of the end-effector. This requires sensors which are small and light. Active infra-red sensors satisfy these conditions since miniature infra-red emitters and receivers are

available in diameters as small as 1.57mm [17].

Proximity sensing with such electro-optical sensors can be performed with one of three methods: triangulation [20, 31], difference of phase [28, 32] or received signal amplitude [5, 23, 27]. Of the three, only the amplitude-based scheme can be packaged small enough and retain a reasonable sensing range. Amplitude-based infra-red sensors with a head diameter of $5mm$ and a range of $\approx 15cm$ are available commercially but are not suitable for multiplexed proximity sensing networks [16]. Work is underway to produce an equivalent sensor with greater range and the multiplexing circuitry necessary for networked operation [12].

Wide-spread use of amplitude-based infra-red sensors in robotic applications has been hindered by the difficulties associated with their use. They are vulnerable to ambient light conditions; this can be eliminated with proper modulation [5]. They are also subject to cross-talk if a system has more than one sensor; this can be mitigated with proper multiplexing [11]. By far the greatest shortcoming of these sensors is their non-linear dependence on not only sensor-object distance but on the angle of incidence between sensor beam and the object's surface as well as the object's reflective properties (color, smoothness, collectively known as the reflectance gain) [15, 21]. Since the output is a function of three independent variables (object-sensor distance, object-sensor line-of-sight angle and object surface properties), all three must be estimated for any one to be measured.

Because of these difficulties, some research has been done in using the sensor output in a qualitative fashion, ie. using the magnitude of the sensor output as a measure of object "closeness". This approach has been implemented to perform obstacle avoidance [11], and automated robotic grasping [15].

Research has thus far concentrated on solving sub-problems of amplitude-based sensing by imposing restrictions on the object's motion, geometry or both.

A single proximity sensor was used to estimate surface properties and the one dimensional sensor-target distance in [15]. The estimation was performed as the

sensor was moved towards planar surface at a known velocity.

In [22], an end-effector equipped with three sensors was used to estimate the angle between it and a planar surface. For this experiment the object surface properties were presumed known and the sensor output was assumed to be independent of the angle between the sensor's line-of-sight and the target surface. Angle independence was made valid by restricting the angle between the end-effector and the target surface to less than 25^o.

In further experiments [23], a single sensor was used to estimate the sensor-target distance of a moving surface with unknown surface properties. The sensor output's dependence on the angle between the sensor's line-of-sight and the target surface was again assumed negligible.

Sensor Fusion

Tracking an object with any sensor, be it radar or an amplitude-based proximity sensor, requires a model of the relationship between the target's kinematic states and the sensor's output voltage. This model is often non-linear and thus estimation schemes which can handle such models must be employed. The estimation scheme should also accommodate multiple sensors. A network composed of multiple sensors provides redundancy in case of failure, minimizes the occurance of occlusion as well as increases the robustness of the state estimation by providing additional constraint equations [14, 35].

The study of how to assimilate and reason with data from disparate sources to produce the best estimate of the state of a system is collectively known as *sensor fusion* [26]. Although sensor fusion can be the study of how to fuse data from different types of sensors (eg. sonar and radar), it can also be the study of how to fuse the data streams from many similar sensors. This work is primarily concerned with the latter. From this body of knowledge, two state estimation schemes are especially suited for object localization. The first, batch estimation, fits sensed data to the sensor

model in a least squares sense. Although primarily intended for parameter estimation, the method has been adapted to perform state estimation [7]. The second, Kalman filtering and its derivatives (extended Kalman filtering and iterated extended Kalman filtering), is one of the most popular state estimation schemes [3, 8, 6, 9, 18, 30, 34]. This is due largely to its computationally inexpensive formulation but also to its many design variables and thus configurability.

Author's Contribution

Although research on using amplitude-based infra-red proximity sensors has a long history, estimating a non-planar object's position and velocity moving along an unknown trajectory with these sensors is new. Of all the assumptions made in previous experiments, one that seems to have been overlooked is assuming a known geometry for the object. I have shown that by knowing the object geometry and using a network composed of amplitude-based infra-red proximity sensors, an extended Kalman filter can be derived to perform object localization. Furthermore, I show simultaneous estimation of the object's reflectance properties. Simulations and experiments in which a maneuvering target object is acquired and tracked by the system demonstrate the effectiveness of this approach.

Organization of the Thesis

The thesis has the following structure. The state estimation problem is presented formally in Chapter 2 and the four approaches are examined in simulation and discussed. In Chapter 3, the proximity sensor network for which the estimation scheme is being constructed is described. Chapter 4 describes in detail the extended Kalman filter implemented to perform the sensor fusion for the proximity sensor network (PSN). The experiments and their results are presented in Chapter 5. A summary as well as an outline of possible avenues for future work are given in Chapter 6.

Chapter 2

Dynamic Object Localization

We wish to estimate the position and velocity of a free-flying, maneuvering object using infra-red sensing. The situation is depicted below in Fig. 2.1.

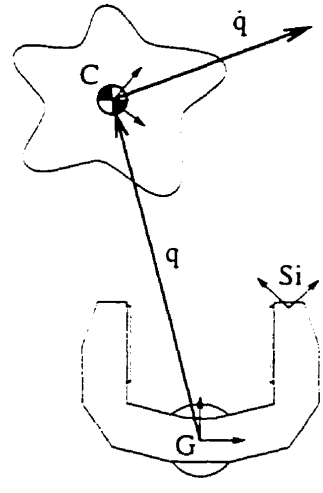


Figure 2.1: *Localization of an arbitrary object. Position, q, and velocity, $\dot{q}$, are estimated with respect to end-effector-fixed coordinates, G. The i sensors, each having local coordinate axes, Si, may be positioned and oriented arbitrarily over the surface of the end-effector. Object orientation is quantified using target fixed coordinates C.*

In general, object localization in this manner is a difficult problem. In order to build an estimation scheme, a relationship between the object's kinematic states and the measured output must be obtained. This relationship, being a function of the

object's geometry, may be complex or even impossible to derive. Furthermore, as the complexity of the object increases, additional parameters must be estimated.

To render the problem tractable, the crucial assumption was made that the geometry of the object to be tracked is known. Although this may appear to be very restrictive, in reality it is quite valid. In many industrial tasks, the geometries of the objects to be manipulated are well defined and known. Additionally, although the geometry must be specified prior to the state estimation sequence, a rudimentary object recognition phase can be performed by the sensing network prior to tracking. A limited form of object recognition can be accomplished concurrently with our object localization as described in Sec. 5.2.3.

2.1 System Formulation

The dynamic system model is:

$$\dot{\mathbf{x}} = A\,\mathbf{x} + B\,\mathbf{u} + \mathbf{v} \tag{2.1}$$

The state vector, $\mathbf{x}$, contains the object states as well as any parameters we may wish to estimate. A and B are the system and input matrices, respectively. The input vector, $\mathbf{u}$, is used to introduce any known inputs to the object dynamics. Note that $\mathbf{u}$ is zero for a free floating object. Model noise, $\mathbf{v}$, allows for unmodelled dynamics. Its statistics are system specific.

The system is being sensed by n sensors. The ith sensor is modelled as:

$$y_i = h_i(\mathbf{x}) + w_i \tag{2.2}$$

The relationships, h_i, between the state vector $\mathbf{x}$ and the sensed output, y_i, for each of the n sensors is non-linear and corrupted by noise, w_i, whose statistics are also sensor specific.

The tracking problem is therefore how to estimate the state vector, $\mathbf{x}$, whose evolution in time is modelled by (2.1) and being observed by n sensors modelled by (2.2).

2.2 Estimation Schemes

Four methods were examined in simulation to perform the estimation: Batch. Kalman Filtering (KF). Extended Kalman Filtering (EKF) and Iterated Extended Kalman Filtering (IEKF).

2.2.1 Batch Estimation

Standard non-linear parameter estimation schemes can be used to estimate time-varying states. The one examined was recursive least squares (RLS). The algorithm used to perform the RLS was a boxed routine available as part of the mathematical modelling package used for the simulations. The advantages of using the packaged routine were that it was error-free and thus the time to develop the simulation was reduced. This allowed more time to examine its performance vis-a-vis the case in point.

The RLS method minimizes an error function g with respect to the states $\mathbf{x}$. For object localization the error function is:

$$g(\mathbf{x}) = \sum_{i=1}^{k} (h(\mathbf{x}) - y)^2 \tag{2.3}$$

where $h(\cdot)$ is the non-linear sensor model. y is the measured value and k is the number of measurements. To accommodate time-varying object positions we estimate locally the initial conditions and a constant velocity as parameters:

$$\mathbf{x} = \mathbf{x}_o + t\dot{\mathbf{x}} \qquad \ddot{x} = 0 \tag{2.4}$$

The primary advantage with this method is its simplicity. Little time was needed to produce simulations for batch estimation since only the error function had to be written. The actual RLS function was available as part of the mathematics software package used for the simulations [37].

There were, however, two important disadvantages with batch estimation. First, it was too computationally intensive and thus too slow to implement in real-time. This

computational overhead is due to the structure of the estimation scheme. For all estimation schemes, the computation time increases with the number of measurements, k used in the routine. This number can be increased by either increasing the number of sensors or, in the case of the RLS, retain past sets of measurements. The minimum k would be equal to the number of states, ie. a system of k equations for k unknowns. However, due to sensor noise, unmodelled dynamics and other sources of error, the number of measurements must be significantly larger. Second, the simplicity of (2.3) is misleading. To obtain even marginally acceptable results, the measurements, y, should be weighted: measurements from accurate sensors more than inaccurate ones, newest data more than old, etc. Proper assignment of these weights is non-trivial and has been adressed [36].

2.2.2 Kalman Filter

The Kalman filter is based on the assumption that an estimate of the system's next state can be obtained by combining a prediction of the system's state and the newest received measurement data such that the increase in the estimated covariance of the states is minimized. A full annotated derivation is given in Appendix B.

The Kalman Filter (KF) requires both state and measurement models to be linear. Since the measurement model (2.2) is non-linear, this method must be rejected. It is described, however, as the next two methods are derived from it.

There are several advantages to using a Kalman filter. First, the filter equations are simple matrix equations in which the most computationally expensive operation is inverting an $n \times n$ matrix. Since n is the number of sensors, which is often less than the number of states, this is an improvement over the batch method as processing time is reduced. Moreover, the model and sensor noise statistics appear in the filter equations explicitly as the covariance matrices, Q and R, respectively. Investigation into the effects of varying the elements of these matrices independently could proceed in a more structured manner than for the RLS method. For example, the weight

assigned to the velocity estimates could be altered independently from the position estimates providing insight into their relationship within the estimation scheme. This contrasts with the batch method where all the noise characteristics must be included in a single, composite weight for each state.

2.2.3 Extended Kalman Filter

Even though the measurement model (2.2) is non-linear, it can be used in the Kalman filter after linearization. A Kalman filter which employs linearized measurement models is called an Extended Kalman Filter (EKF). The linearization is a first order Taylor expansion of the measurement equation:

$$y_i = H_i \delta \mathbf{x} + v_i \tag{2.5}$$

where

$$H_i = \left[\frac{\partial h_i(\mathbf{x})}{\partial x_j} \right] \tag{2.6}$$

is the Jacobian of the non-linear measurement function, $h_i(\mathbf{x})$, of the ith sensor. When the system has more than one sensor they are stacked into a single matrix equation.

$$\mathbf{y} = H \delta \mathbf{x} + \mathbf{v}. \tag{2.7}$$

This linearized measurement equation may now be used in the derivation of the extended Kalman filter. See the complete derivation of the EKF in Appendix B Sec. B.2.

The extended Kalman filter has all of the advantages of the Kalman filter while accommodating non-linear measurement models. This flexibility comes at a cost however, the linearization may introduce numerical errors into the state estimates.

2.2.4 Iterated Extended Kalman Filter

As just mentioned, use of the Jacobian of the measurement model may cause numerical errors in the estimates. These errors may be significant depending on the curvature of the measurement function [6]. Iterated Extended Kalman Filtering (IEKF) minimizes this error by performing local iterations between measurements to improve the value of the Jacobian. The reasoning is as follows. In the EKF, the Jacobian, H, is evaluated with the last estimate of the states, ie. linearization is performed about the last estimate. It is then used to calculate the newest estimates. Since the new estimates are an improvement over the old ones (the filter would be diverging otherwise) these new estimates can be used to produce an improved Jacobian. This new Jacobian can then be used to reprocess the old data to increase the accuracy of the states. This is repeated until subsequent values of the states differ by less than a preset tolerance. The IEKF is derived in Appendix B Sec. B.3.

The advantage of this method is obvious; by reducing the effects of numerical errors, a more efficient filter is obtained. This improvement comes at the cost of increased processing time required by the local iterations.

2.3 EKF vs. IEKF

Prior to implementation, a choice had be be made between the EKF and IEKF. In simulation both methods gave equivalent results. Convergence was slightly faster with the IEKF but only when the simulated noise levels were unrealistically low. Because the gains accrued using the IEKF were not large enough to justify it's longer processing time, the EKF was chosen for final implementation.

Chapter 3

The Proximity Sensor Network

To perform accurate and robust object localization, a network of proximity sensors is necessary. A network composed exclusively of amplitude-based infra-red sensors was developed specifically for this task in a companion Masters of Engineering thesis [12] is described in Sec. 3.1. The first step after the construction of the sensors was their characterization, described in Sec. 3.2.

3.1 Sensor Hardware

The sensors and their associated circuitry were designed to conform to the following specifications:

- range of 0 - 100mm
- single-valued function in operational range
- ambient light rejection
- head diameter less than 7mm
- multiplexed sensor operation
- sensor scan less than 2ms

A commercial sensor was found whose range, ambient light rejection and diameter were suitable [16], but it did not support fast switched operation necessary for multiplexing.

One of the sensors developed is shown in Fig. 3.1 without its protective casing.

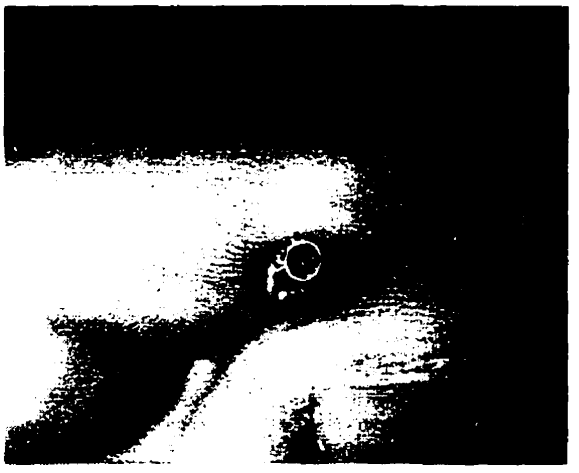


Figure 3.1: *Miniature infra-red proximity sensor.*

3.2 Sensor Characterization

The output voltage of an active infra-red sensor is a function of not only object distance, d, but also of the local surface angle, θ, the reflectance gain, λ, and local surface curvature. Object features were assumed to be large compared to the sensors and thus the effect of local surface curvature was neglected.

The sensors were characterized by experiment: sensor output voltage was recorded while a white paper covered cylindrical object, $60mm$ in diameter, whose longitudinal axis was perpendicular to the sensor's line-of-sight, was moved in front of the sensor. For every reading, the Cartesian coordinates of the center of the object were used to calculate the sensor-target distance, d, and the object surface angle, θ.

A typical raw data set is presented as a surface plot in Fig. 3.2.

The data obtained was used to establish the sensor model to be used in the extended

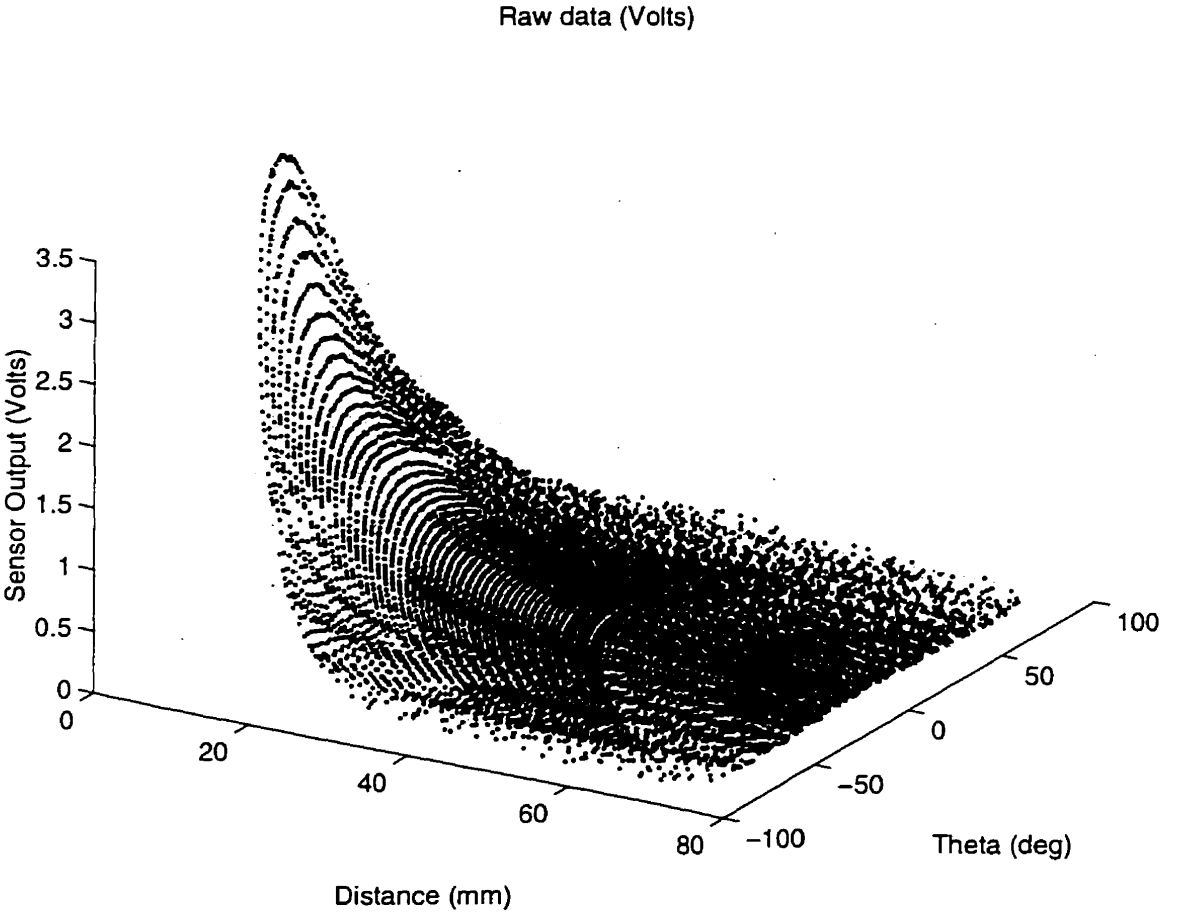


Figure 3.2: *Proximity sensor output voltage vs. object angle, θ and sensor-target distance, d.*

Kalman filter. Two avenues for constructing this model were pursued. At first, an analytical model was used but when this proved unsatisfactory, a numerical model had to be used. Both avenues are now discussed.

3.2.1 Analytical Formulation

In simulation, an analytical formulation was used for the sensor model. At first, as there was no actual sensor data for comparison, the following equation was used:

$$h = \frac{\lambda}{d^2}\cos(\theta) \tag{3.1}$$

The relationship between sensor output, h, (which is proportional to the intensity of the received infra-red light reflected by the object) and sensor-target distance, d, is the well known inverse square law [15, 21, 23]. The angle dependence relationship was obtained by assuming the object's surface was Lambertian [19, 21]. Finally, the surface properties of the object (color, texture, finish) were lumped into a single "reflectance gain" as in [15, 23].

Once the sensors were built and the calibration experiment conducted, (3.1) was used as the functional form to which the data was fit using a least squares parameter identification scheme. The parametrized sensor model was:

$$h_i(d_i, \theta_i, \lambda) = \frac{\lambda\beta_{1,i}}{(d_i + \beta_{4,i})^{\beta_{2,i}}} cos(\beta_{3,i}\theta_i). \tag{3.2}$$

The i subscript refers to the ith sensor in the network. In general, each sensor would have its own model but since all the sensors in the PSN are infra-red sensors, their structures are identical and only their parameters ($\beta_{1,i}, \beta_{2,i}, \beta_{3,i}, \beta_{4,i}$) differ. The fact that the parameters differ between sensors is due to variations in the sensor heads, in turn caused by our current manual production method. Eventually, all sensors will be characterized by the same $h(\cdot)$.

Note that while [23] assumes that surface angle dependence is negligible, Fig. 3.2 shows that this was not the case with our sensors as there is a definite attenuation of

the output signal with increasing angle. These findings agree with other researchers' findings and and the theoretical development found in [15].

The parameters $\beta_{1,2,3,4}$ were determined by fitting (3.2) to experimental data using recursive least squares (RLS). Due to the large variability between the output ranges of the different sensors (minimum range 1.4 volts, maximum range 3.3 volts) which were to make up the PSN, each was calibrated independently. Since the overall shape of each sensor's output curve was the same, each sensor's output was scaled in software to a range of $0 - 3.5$ volts prior to use in the parameter identification routine. The scaling and offset values for each sensor were saved for use in the filter. The relation (3.2) fit each sensor's data with a maximum root mean square (RMS) error of less than $6mV$.

Unfortunately, RMS error is a good indicator of model fit only if errors are due to gaussian noise [33]. Even though our RMS error was small, the surface plot of the error between measured data and the fit surface vs. object angle and sensor-target distance in Fig. 3.3 show that significant systematic errors existed. The lobes to either side of $\theta = 0°$ suggests that the angle dependence isn't modelled correctly by the cosine function. The error is due to the fact that our model assumes an incident light beam painting a point on the sensed object whereas, in reality, the sensor emits a cone and thus paints a circle on the object. The geometry involved in determining how much of that circle is reflected back to the sensor's receiver is complex [15].

3.2.2 Numerical Formulation

Acknowledging the presence of systematic errors raised the question of whether further theoretical analysis should be performed in hopes of producing a more faithful model or to forego the analytical model and resort to a numerical characterization. It was decided to follow the latter to keep the focus of the research on producing a working system.

The sensor model now takes the form of tabulated data and is evaluated by 2D linear

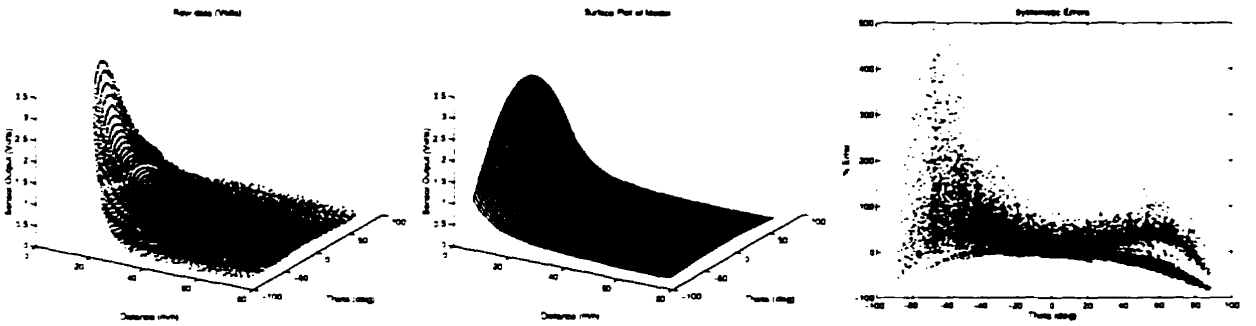


Figure 3.3: *Analytical formulation for the measurement model: raw data from characterization run (left), surface plot of model (middle). Surface plot of error (in percent of current value) between model and experimental data (right), note the lobes to both sides of* $\theta = 0^o$.

interpolation. The process to produce an array suitable for table lookup involved two steps. First, an interpolation scheme was used to produce a regularly spaced matrix from the irregular vectors of sampled data. The resulting $t \times d$ matrix mapped rows to increments of 1^o in θ and $1mm$ in distance, d. Drawing upon image processing techniques, the data was then smoothed by convolving it with the following mask, M:

$$M = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \tag{3.3}$$

and dividing the value of each element by 16, the sum of the elements in the mask. The resulting array was then stored for use in the filter. These steps were all performed using functions available in the mathematical software package used for the simulations [37].

It should be noted that, from a computation stand point, tabulating the function should have been the method to adopt from the start as it reduces computation time[1]. This is of prime importance since the goal was implementation in real-time.

[1]To evaluate the analytical model (3.2) 4 multiplications, 1 addition and 2 function calls (cosine and exponential) are needed. This is replaced with three linear interpolations which only require 4 additions, 1 division and 1 multiplication, each.

A second set of data, obtained in the same fashion as the data used to create the table, was used for validating the table. The error between the output voltage of the new data set and the output obtained from the table evaluated at the same coordinates was plotted to verify that the systematic errors had been eliminated. The covariance of this error was calculated and saved as it was required for implementing the extended Kalman filter.

Chapter 4

Data Fusion for the Proximity Sensor Network

4.1 Object Model

4.1.1 Geometry

As discussed in Chapter 2, the complexity of the filter depends to a large extent on the object's geometry. For simplicity, the object selected for localization was a circle which does not require the estimation of orientation. As can be seen in Fig. 4.1, the relationship between the sensed distance, d_i, the local surface angle, θ_i, and the center coordinates of the circle, $(x_{1,i}, x_{2,i})$, will be identical for each of the n sensors. The circle was also chosen because the task for which the PSN system was being developed was robotic grasping. Since many "grippable" objects, especially for anthropomorphic robot hands, are cylindrical in shape, using the simplest 2D projection of a cylinder seemed a natural starting point.

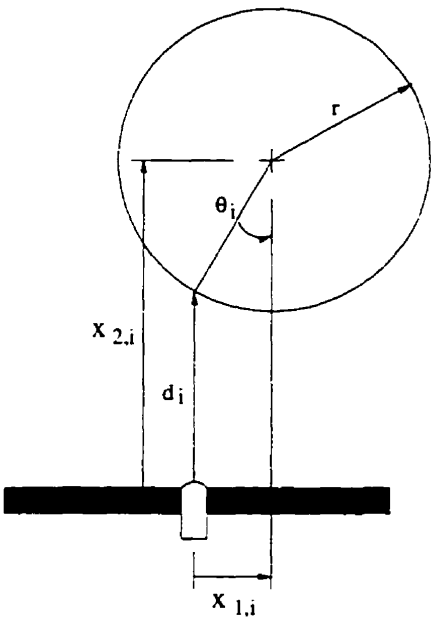


Figure 4.1: *Sensor i viewing object. Sensor output is a function of d_i and θ_i and the object surface properties, λ. The sensor model must relate these variables to the states of interest, $(x_{1,i}, x_{2,i})$.*

4.1.2 Dynamic Model

The object's dynamic equation was given in (2.1). Since the filter will be implemented in discrete time, that equation, discretized, is:

$$\mathbf{x}_{k+1} = \Phi \mathbf{x}_k + \Gamma \mathbf{u}_k + \mathbf{m} + \mathbf{v}, \tag{4.1}$$

where $\mathbf{x}_k$ is the state vector at step, k and we have added the vector $\mathbf{m}$ which captures known displacement of the manipulator. For the circle, the state vector is arranged as follows:

$$\mathbf{x} = [x_1, x_2, \dot{x}_1, \dot{x}_2, \lambda]^T \tag{4.2}$$

where x_j and $\dot{x}_j$ are the components of position of the center of the circle and their derivatives, respectively. We are estimating the albedo parameter λ on-line and append it to the state vector, assuming it has no dynamics, $\lambda_{k+1} = \lambda_k + v_k$.

The state transition matrix, Φ, is defined as:

$$\Phi = e^{A(t_{k+1}-t_k)} = e^{A\tau} \tag{4.3}$$

$$\Phi = \begin{pmatrix} 1 & 0 & \tau & 0 & 0 \\ 0 & 1 & 0 & \tau & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}. \tag{4.4}$$

(4.5)

We assume we have no knowledge about the system inputs, u, and therefore set $u_k = 0$. Thus accelerations are modelled as disturbances. Finally, the object model noise, $\mathbf{v}$, is assumed stationary and Gaussian. Furthermore, it is not correlated in time and therefore does not have a k subscript:

$$E[v_k\, v_l] \approx Q\delta_{kl}, \tag{4.6}$$

where Q is the system covariance matrix and δ_{kl} is the Kronecker delta.

4.2 Sensor Model

The measurement model (2.2) is composed of n non-linear equations, or n look-up tables, where n is the number of sensors acquiring valid data. The non-linear relationship, $h_i(\cdot)$, is a function of d_i, θ_i and λ which must be evaluated from the state variables $\mathbf{x}$. From the geometry in Fig. 4.1, the following kinematic relations are drawn:

$$\theta_i = \text{asin}(\frac{x_{1,i}}{r}) \tag{4.7}$$

$$d_i = x_{2,i} - r\cos(\theta_i). \tag{4.8}$$

The sensor noise, $\mathbf{w}$, was characterized as described in Sec. 3.2 and therefore,

$$E[w_k\, w_l] \approx R\delta_{kl}, \tag{4.9}$$

where the elements on the diagonal of the measurement covariance matrix, R, are equal to the experimentally obtained sensor covariances.

4.3 Calculating the Jacobian

The final step before using the extended Kalman filter is to obtain the Jacobian, H of the measurement function, h, with respect to the state vector in end-effector-local coordinates, ${}^{G}\mathbf{x}$. As described at the end of the previous section, h_i, is a function of d_i and θ_i which are in turn functions of the object state in the ith sensor-local coordinate system, ${}^{Si}\mathbf{x}$. Furthermore, ${}^{Si}\mathbf{x}$ is related to ${}^{G}\mathbf{x}$ through a coordinate transformation. Since we are considering only a planar scenario, this transformation is

$${}^{Si}T_G = \begin{bmatrix} \cos\alpha_i & -\sin\alpha_i & Si_x \\ \sin\alpha_i & \cos\alpha_i & Si_y \\ 0 & 0 & 1 \end{bmatrix}. \tag{4.10}$$

where α_i is the angle between the x-axis of the end-effector and sensor coordinate systems, and Si_x, Si_y are the components of the distance between the origins of the two coordinate systems.

Thus, the Jacobian with respect to ${}^{G}\mathbf{x}$ can be derived by applying the chain rule.

$$H_i = \frac{\partial[h_i]}{\partial[{}^{G}\mathbf{x}]} = \frac{\partial[h_i]}{\partial[\mathbf{p}_i]} \cdot \frac{\partial[\mathbf{p}_i]}{\partial[{}^{Si}\mathbf{x}]} \cdot \frac{\partial[{}^{Si}\mathbf{x}]}{\partial[{}^{G}\mathbf{x}]}. \tag{4.11}$$

The functional forms of the three terms can be found in Appendix A where the Jacobian is derived. Note that for the analytical formulation of the sensor model the first term in the Jacobian has an analytical form. For the numerical formulation the first term must be obtained from the look-up table. The partial derivatives of the function with respect to d and θ were obtained numerically and stored in two

additional tables. The partial with respect to λ is simply the original function divided by the latest estimate for λ.

4.4 An Extended Kalman Filter for the Proximity Sensor Network

The EKF equations are presented below:

Prediction

$$\mathbf{x}_{k+1|k} = \Phi \mathbf{x}_k \tag{4.12}$$

$$P_{k+1|k} = \Phi P_k \Phi^T + Q \tag{4.13}$$

Update

$$\mathbf{x}_{k+1|k+1} = \mathbf{x}_{k+1|k} + K[\mathbf{y}_{k+1} - \mathbf{h}(x_{k+1|k})] \tag{4.14}$$

$$P_{k+1|k+1} = [I - KH_{k+1}]P_{k+1|k}[I - KH_{k+1}]^T + KRK^T \tag{4.15}$$

the Kalman gain is defined as,

$$K = P_{k|k-1}H_k^T[H_k P_{k|k-1} H_k^T + R]^{-1} \tag{4.16}$$

The subscripts are in standard conditional probability notation. Conditional probability and other statistical concepts are covered in any statistics textbook, eg. [29].

4.5 Filter Implementation Issues

From the extensive simulations performed, several issues were raised and handled prior to experimentation.

4.5.1 State Vector Initialization

Although in simulation the initial values for the states, $\mathbf{x}_0$, can be set arbitrarily near or far from the actual values, in practice this cannot be done. Thus, a method to

initialize the states had to be devised.

The method adopted was a purely geometric one: at initialization, each sensor was checked to see if it was acquiring valid data. The lines of sight of those sensors which were valid were used to approximate the object's initial location. A graphical view of the method is given in Fig. 4.2.

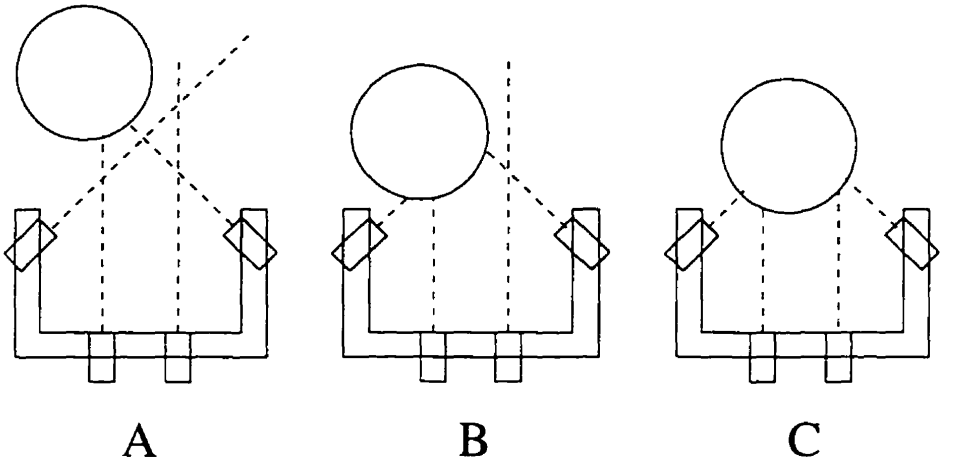


Figure 4.2: *Initialization procedure: from left to right examples of two, three and four point initialization.*

Single sensor initialization was not implemented since this would lead to an unobservable system. Observability is discussed in Sec. 4.5.5. The velocities were always initialized to zero and the reflectance gain to one.

4.5.2 Model Covariance Matrix Initialization

A more difficult problem was initializing the model covariance matrix, Q. This matrix greatly affects the speed of convergence and steady state noise of the state estimates. Theoretically, the Q matrix is a measure of the system model noise and is therefore used to account for unmodelled plant dynamics. Alternatively, the Q matrix can be thought of as a measure of the confidence the filter has in its estimation of each state. Thus large values in Q indicate that the confidence is low. Mathematically, large values weigh that state's estimation so that it is more sensitive to measurement data.

Since Q is an input to the filter, its elements can be designed to produce specific filter behaviour. The Q matrix for the PSN was designed as follows.

Since only the positions and the reflectance gain appear in the measurement equation, these are assigned higher confidence than that which is assigned to the velocities. Assigning low confidence to the velocity estimates has a positive side-effect. Since the accelerations are not being estimated, uncertainty in the velocities allows the object to maneuver without causing the filter to diverge. The drawback of assigning greater uncertainty to the velocity is that its estimate will be noisy even after the filter has converged. This will make it difficult to trust the velocity estimates for use in a control algorithm. Appropriate low-pass filtering can be used to smooth the velocity, however, since the estimate is reacting to sensor noise, not accelerations. A comparison between two filters with different Q's is made in Fig. 5.6 with experimental data.

4.5.3 State Covariance Matrix Behaviour

The state covariance matrix, P, performs a similar function to that of the model covariance matrix, Q: it weighs the state estimates by their uncertainty. It is therefore initialized in the same way as Q.

The P matrix can be used to measure the progress of the filter's performance. As the filter converges, its elements tend towards zero. Since the diagonal elements are the covariances of the estimated states, the trace of P is commonly used to determine filter convergence.

A final note on the P matrix. Since the calculation of the Kalman gain involves an inverse, numerical stability issues arise. As the P matrix forms the core of the matrix that is inverted, if it is badly conditioned, numerical errors may cause the filter to diverge. For this reason, the condition number of the P matrix provides an indication of the numerical stability of the filter.

4.5.4 State Covariance Resetting

When a Kalman filter (EKF or IEKF as well) converges, there is a danger that the state covariance matrix, P, will become so small that the filter ignores new measurements. Recalling the discussion in Sec. 4.5.2, this is equivalent to assigning a weight of zero to a state's estimate. This effect can be seen by examining the filter equations. If the state covariance matrix tends towards zero so does the Kalman gain (4.16). Thus, the measurements' contribution to the state update equation (4.14) is zero.

Vanishing covariance occurs when the object to be located by the PSN is stationary or moving with constant velocity for extended periods of time. This could be catastrophic if the object suddenly accelerated since the filter would not be able to respond and would diverge. To prevent this, the trace of the covariance matrix is checked against an arbitrary minimum and, if it is below, reset to its initial value [30].

4.5.5 Observability Analysis

Since the extended Kalman filter acts as an observer, observability theory was used to analyze the system. The observability matrix for the PSN system is:

$$\mathcal{O} = \begin{bmatrix} H \\ H\Phi \\ H\Phi^2 \\ H\Phi^3 \\ H\Phi^4 \end{bmatrix}, \tag{4.17}$$

where H and Φ are the Jacobian and state transition matrices, respectively.

Observability theory states that if the rank of $\mathcal{O}$ is less than the number states being observed, the system is unobservable [3]. Since the Jacobian, H, is a function of the relative positions of the sensors and the object, the system could become unobservable in certain configurations. An investigation into which configurations would be unobservable was therefore conducted.

The observability matrix for the system is:

$$\mathcal{O} = \tau^2 \begin{bmatrix} (M_1 & N_1 & 0 & 0 & P_1) \\ (M_2 & N_2 & M_2 & N_2 & P_2) \\ (M_3 & N_3 & 2M_3 & 2N_3 & P_3) \\ (M_4 & N_4 & 3M_4 & 3N_4 & P_4) \\ (M_5 & N_5 & 4M_5 & 4N_5 & P_5) \end{bmatrix} \tag{4.18}$$

where,

$$M_i = \Theta_i \cos\alpha_i - \Omega_i \sin\alpha_i \tag{4.19}$$

$$N_i = \Theta_i \sin\alpha_i + \Omega_i \cos\alpha_i \tag{4.20}$$

$$P_i = \Pi_i \tag{4.21}$$

and each bracketed row is composed of n, the number of sensors, rows. The variables, Θ, Ω and Π are defined in Appendix A.

Let us consider the situation in which the PSN is composed of a single sensor. In this case, the first, second and fifth columns are identical, making the rank of this matrix, at most, two. Thus, a PSN composed of a single sensor could estimate, at most, two states simultaneously and the values for the other states would have to be obtained independently. Recalling that the state vector is ordered $(x_1, x_2, \dot{x}_1, \dot{x}_2, \lambda)$ and each column of $\mathcal{O}$ is associated with its respective state, the only possibility for an observable system is to estimate one of the velocities with one of the positions or the reflectance gain.

When the PSN contains more than one sensor, the first, third and fifth columns no longer have identical elements and thus the system should be observable. However, as mentioned above, the system may become unobservable in certain sensor-object configurations. Since a measure of how near a matrix is to losing rank is its condition number, this was used as a basis for a "generate and test" optimization scheme to find the optimal arrangement for the sensors on the end-effector. For each configuration, two indices were calculated:

1. the sum of the condition numbers of the observability matrix (4.18) over the number of discretized locations in a fixed sensing region.

2. the number of locations in the region in which the object is out of view of one or more of the sensors.

The first index is a measure of how observable the system is throughout the sensible region. Since large condition numbers arise from matrices which are near to losing rank, the larger the sum, the poorer the configuration. The second index is an indication of the coverage provided by the configuration. The more the object is in view of all the sensors, the better the configuration.

Since the general optimization problem (placing n sensors in arbitrary positions and orientations over the full surface of the end-effector) could not be solved in a feasible amount of time, some restrictions were imposed on the final sensor configuration. First there was to be four sensors: two on the horizontal edge of the end-effector and two on opposite vertical edges at $40mm$ from the horizontal edge. Second, the sensor arrayment was to be symmetric about the centerline of the sensible region. Finally, the orientation of the two sensors on opposite edges could only be outward looking, thus $0 < \theta < 90^o$. With these restrictions, only two parameters enter the optimization: the distance from the centerline, X, between the sensors on the horizontal edge and the angle, θ, of the other two sensors.

The results of the optimization are plotted in Fig. 4.3. As can be seen from the plot of condition number vs. X and θ, the regions to be avoided were at distances of $\approx 30mm$ between the angles of $40 - 50^o$ and $75 - 78^o$. The plot of the number of locations in which the object is out of view of one or more sensors shows that the angle should be greater than 60^o and distances less than $20mm$. The values used for our implementation were $X = 15mm$ and $\theta = 60^o$. The sensor arrangement is also shown in Fig. 4.3.

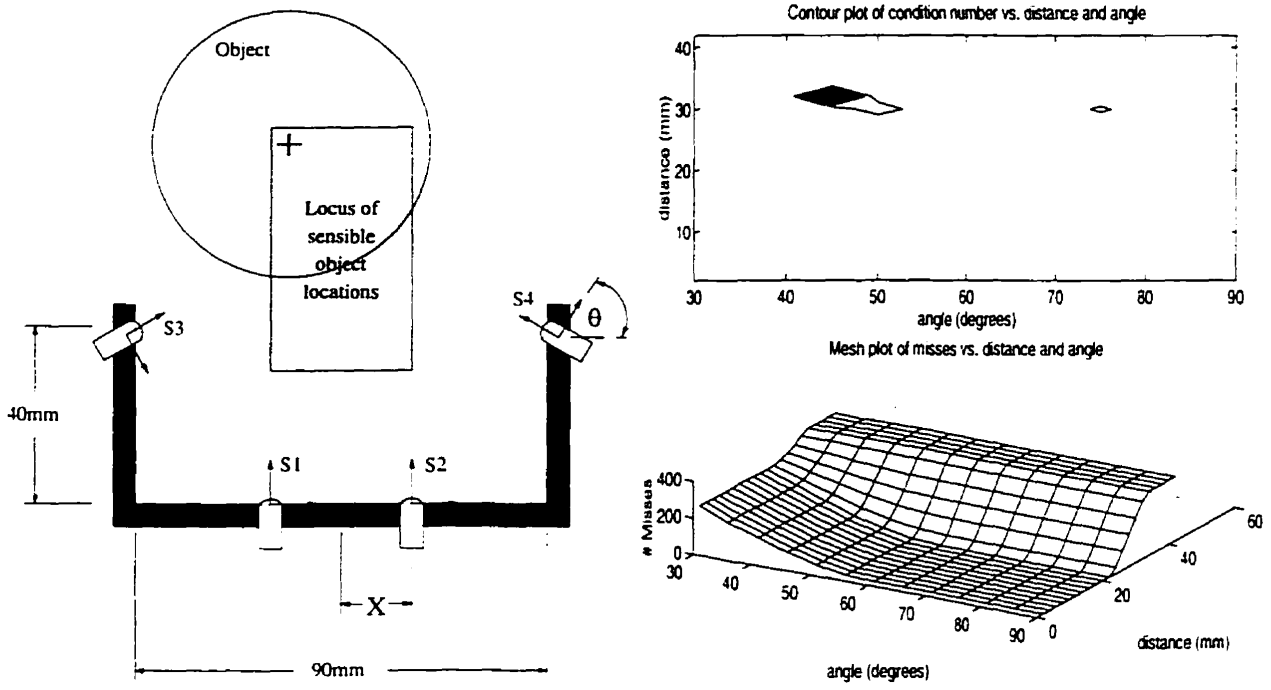


Figure 4.3: *Final sensor configuration, $\theta = 60^o$ and $X = 15mm$ (left). Regions of low observability characterized by high condition numbers in the observability matrix (top right). Number of locations in which the object is out of one or more of the sensors' views (bottom right).*

4.5.6 Sensor Validity Checking

A method for checking whether a sensor was providing valid data had to be implemented. Once a sensor was determined invalid, a method to remove it from the filter on-line was also developed.

Two checks were made to determine valid sensors. The first was based on the estimated position of the object: if the object's estimated position was out of a sensor's lateral field of view it was rejected. Second, if the measured value from the sensor was below the noise threshold, it was also removed from the filter.

To reform the filter equations on-line, a "Select" matrix was constructed based on which sensors were valid or not. This matrix operated on vectors, removing undesired elements but maintaining the order of the remaining elements. Once the valid sensors were selected, the new size of the measurement vector was propagated through the filter equations.

Chapter 5

Experimentation

5.1 Hardware Description

A custom autonomous robotic test-bed was constructed for sensor calibration and all experimental work. The test-bed is composed of two serial link manipulators: a PPR actuated linkage named "Calvin" and an unactuated RRR linkage named "Hobbes". All data was handled by custom software running on a two-node transputer network composed of a T800 and a T222 INMOS © processors [24, 25]. Control programs are downloaded from, and post-process data logging uploaded to, a UNIX workstation over an ethernet connection. The system is shown in Fig. 5.1

Calvin

The actuated manipulator configuration was chosen for its analytical simplicity: its forward kinematic equations are identical to its inverse kinematics. The base link, known as the x-stage, is a belt driven linear stage and has a travel of 600mm. Its actuator is a brushless DC motor. The second link, or y-stage, is a ballscrew driven linear stage with a travel of 300mm. It's motor was a brushed DC motor. The final link or θ-stage is a worm gear driven rotary stage. Its motor is a brushless DC motor. Each link's motor was instrumented with an optical encoder having a resolution of 0.09

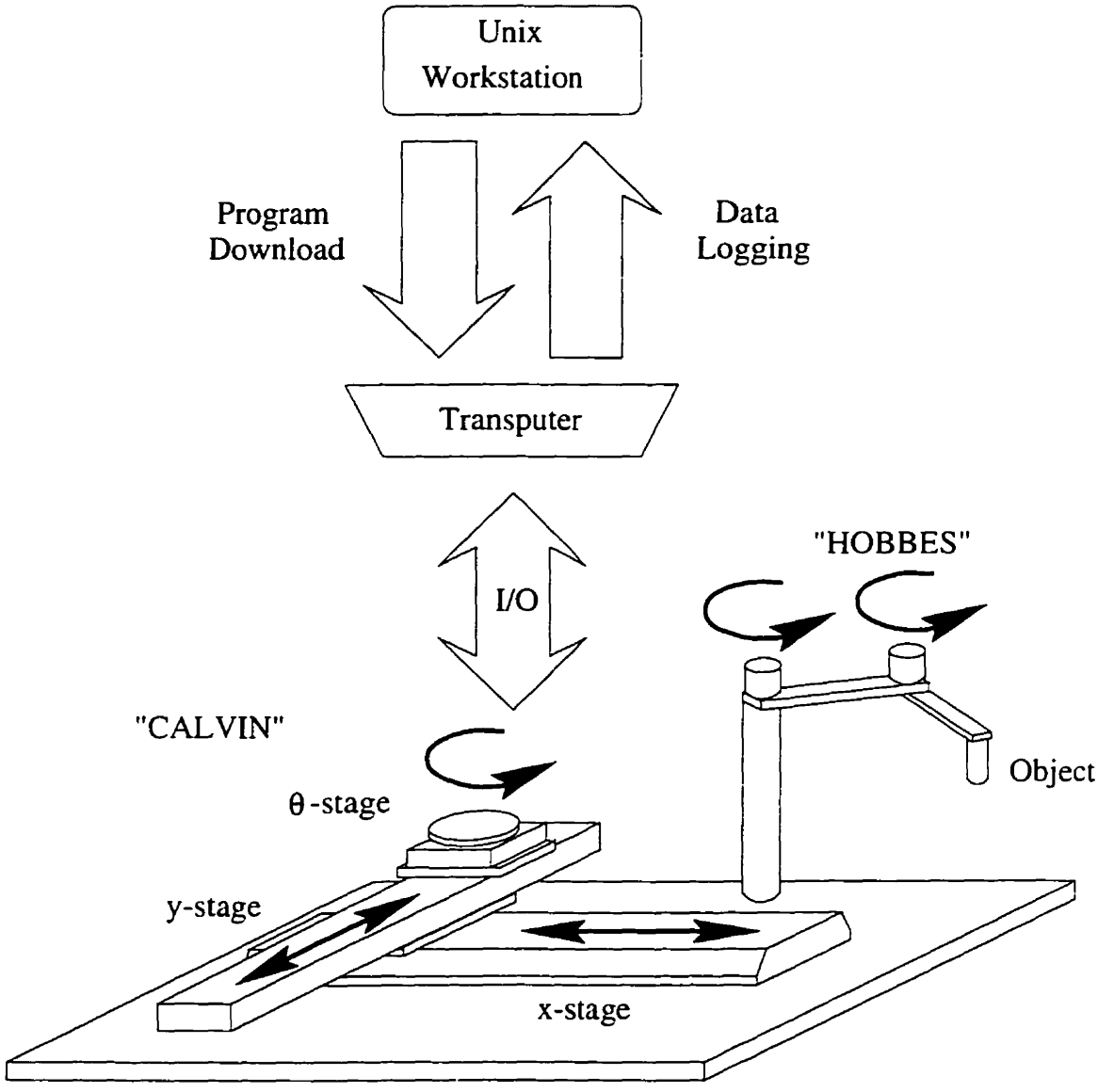


Figure 5.1: *Experimental setup*

degrees. Power to each motor was controlled by a servo-amplifier. The end-effector was a parallel jaw gripper.

Hobbes

To validate the estimates provided by the PSN, the object's actual position and velocities had to be determined. The second linkage was built for this purpose. Each of the three links was instrumented with an optical encoder having a resolution of 0.0071 degrees. Fig. 5.1 does not show Hobbes' distal link as it has zero length and thus not needed with a circular object.

The Transputer

The software controlling the I/O, running the servo-loops and handling postprocess storage, was written in C. However, because of the parallel architecture of the transputer network, the manufacturer's parallel process library routines also had to be used. The 32-bit T800 processor ran high level state machines, data storage and path planning calculations. The 16-bit T222 processor was responsible for I/O with the hardware as well as low-level data manipulation such as numerical differentiation of the position signal and low-level command calculations.

5.1.1 Calibration

A kinematic calibration was performed to identify Hobbes' position relative to Calvin's workspace (link0) as well as determining Hobbes' link lengths (link1, link2). Since Hobbes is initialized by pushing its links against their respective hard-stops, offset angles between their assigned coordinate axes and this position, $\theta 1_{off}$ and $\theta 2_{off}$, were also determined. Calibration followed the method described in [4]. First, the last link of both linkages were fixed together forming a redundant closed chain. Then the joint angles were recorded as the pose of the redundant chain was changed manually.

Finally, a recursive least squares algorithm was used to perform the parameter identification. The results are given in the table below.

Parameter	Nominal Value	Calibrated Value
link0_x (mm)	20	18.69
link0_y (mm)	-385	-369.33
link1 (mm)	400	400.34
link2 (mm)	200	200.02
$\theta 1_{off}$ (deg)	240	239.43
$\theta 2_{off}$ (deg)	-155	-156.95

5.2 Experimentation and Results

The behaviour of the proximity sensing system is presented in a combination of simulated and experimental results. Simulation results are presented first, showing the feasibility of the system. Experimental results of the working system are then presented as proof of implementability.

5.2.1 Dynamic Object Localization: Simulation

The following results were generated from the final version of the simulation. In this version, an object could be made to follow an arbitrary trajectory and this trajectory was estimated by a PSN fixed to an end effector controlled using the following control law:

$$\tau = K_p(\mathbf{x} - (\mathbf{x}_a + \delta)) + K_v(\dot{\mathbf{x}} - \dot{\mathbf{x}}_a) \tag{5.1}$$

where, τ is the two dimensional torque vector, K_p, K_v are the proportional and derivative gains respectively, $\mathbf{x}_a, \dot{\mathbf{x}}_a$ are the current end-effector position and velocity and $\mathbf{x}, \dot{\mathbf{x}}$ are the latest estimates of the target's position and velocity, respectively.

The components of all the vectors were in the x, and y directions. The offset, δ, is the desired distance between the end-effector-fixed coordinate system and the object-fixed coordinate system.

The end-effector was modelled as an undamped mass. The control law is a modified PD control in which actuator sensor feedback is replaced with the filter's estimates of the target's kinematic states. The errors go to zero when the end-effector's position and velocity match the target's estimated position and velocity [10].

The sensors were modelled using the analytical measurement model given in Sec. 3.2. The noise characteristics used matched those identified from the actual sensors.

An important assumption was made concerning the placement of the sensors. It was assumed that the location of each sensor was exact and known. When the orientation or position differ from that assumed by the filter systematic errors are introduced into the filter which may cause it to diverge.

Results from a typical simulation are given in Fig. 5.2. In the simulation shown, the object followed a linear trajectory to the right at 3.5 cm/sec upon which sinusoidal disturbances in the X and Y directions were superimposed. The disturbance in the X direction had an amplitude of 0.2 cm and a period of 0.4 seconds. The disturbance in the Y direction had an amplitude of 0.5 cm and a period of 2.86 seconds. The resulting object trajectory is shown in the upper left panel of the figure. The filter's estimate of the object's trajectory is presented in the same graph. The errors in each of the five states are presented in the other five panels. The effect of assuming linear behaviour for the object in the filter is highlighted by the periodic nature of the errors. The filter's performance in the linear regions of the trajectory is very good but degrades significantly in the regions with more curvature.

To keep the object in view of the PSN, the end-effector was allowed to move according to the control law stated in (5.1). The two plots in Fig. 5.3 demonstrate how the filter's estimates of the object's position and velocity can be used as feedback signals for use in controlling the end-effector. The plots show the trajectory of the center of

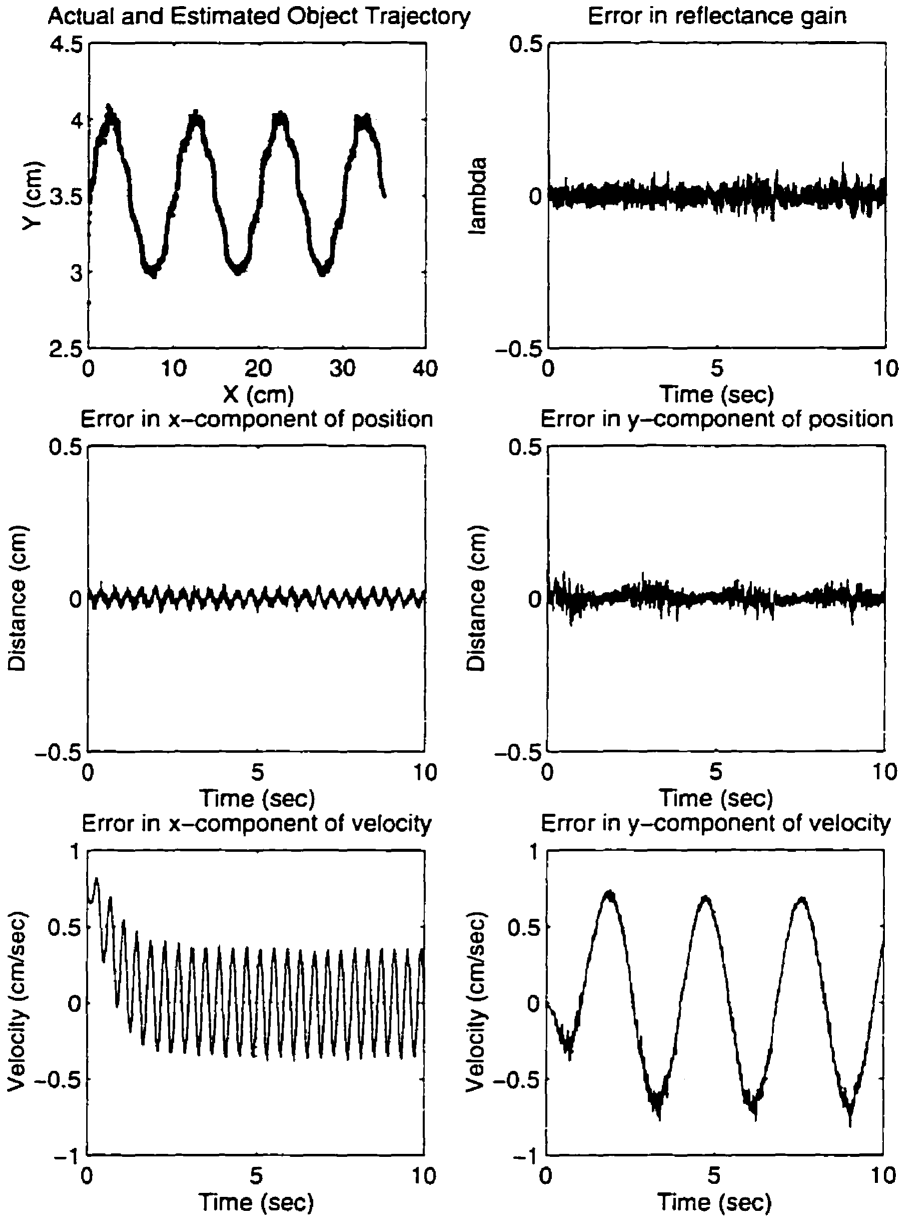


Figure 5.2: *Output from a typical simulation experiment: actual and estimated object trajectories (top left), reflectance gain error (top right), error in the X and Y components of position (middle left and right, respectively) and error in the X and Y components of velocity (bottom left and right, respectively).*

the end-effector superimposed on the trajectory of the object. The trajectories were superimposed by adding the offset, δ in (5.1), to the end-effector's trajectory. This was done so that the performance of the filter may be more easily evaluated.

From the first plot we see that the end-effector acquires and tracks the object closely. In the second plot, however, we see a phase lag between the object and end-effector's trajectory. This lag is due to the simplistic control law used to control the end-effector and could be mitigated by adding an integral term. The control law would then be a modified PID controller which would exhibit better tracking properties.

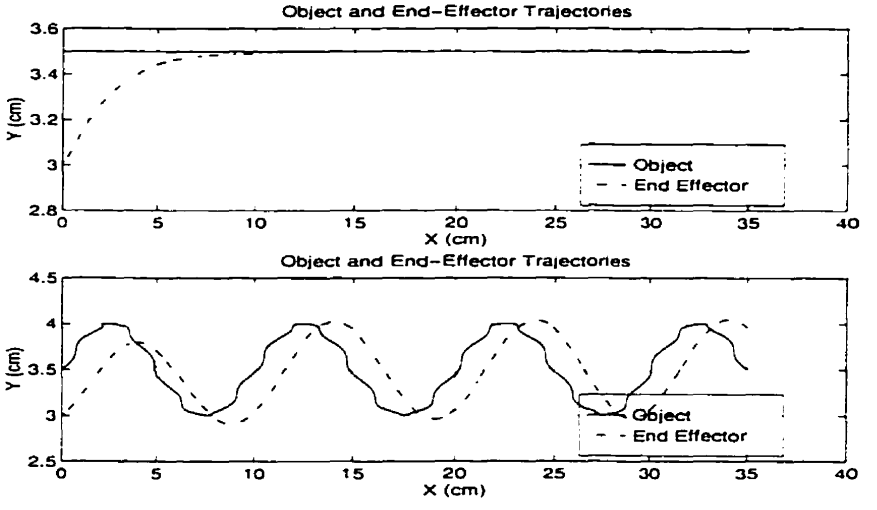


Figure 5.3: *Object and end-effector trajectories using estimated object position and velocity for feedback: object following linear trajectory (above) and object following trajectory with disturbances (below).*

From Fig. 5.2 we see that the filter provides sufficiently accurate for object tracking.

Simulations were also conducted to determine the filter's behaviour when faced with a changing reflectance gain. The results of a simulation in which the reflectance gain changed suddenly are also presented in Fig. 5.4. Note that the actual reflectance gain and not the error is plotted in the top right panel.

The simulation changes the reflectance gain from 0.8 to 3.0 at $t = 3$ seconds and changes it again to 0.45 at $t = 7$ seconds. As can be seen from the plots the error

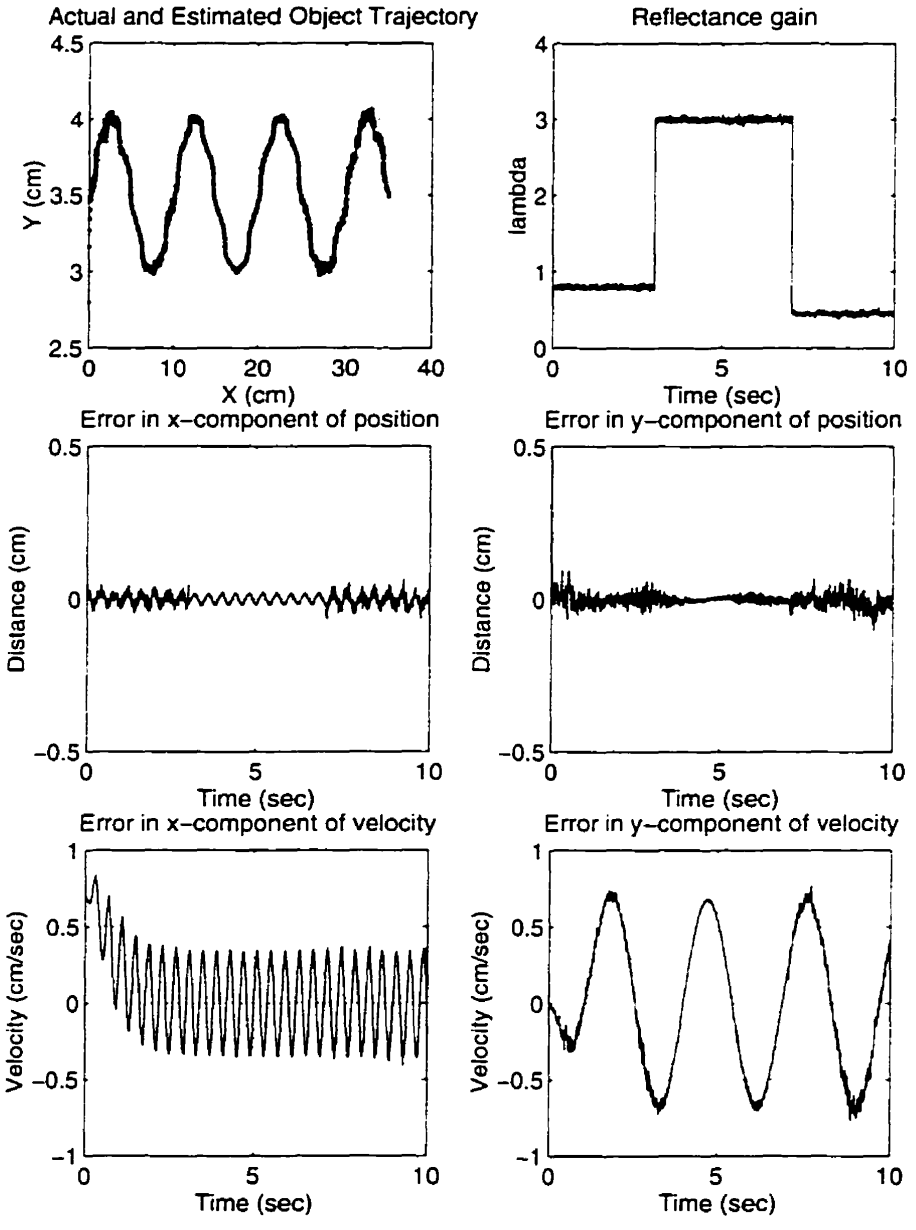


Figure 5.4: *Tracking the reflectance gain. The reduction of the signal to noise ratio at large reflectance gains causes the efficiency of the filter to increase.*

of the states is significantly reduced at larger albedo parameters. This is due to the fact that when the reflectance gain is large, the signal to noise ratio is reduced, thus improving the efficiency of the filter.

As seen in Fig. 5.4 the system is able to track objects of unknown surface properties. This is important since the measurement model for a given geometric class (in this case a circle) obtained in calibration is valid for other, naturally occurring, objects of the same class. Thus the PSN calibrated with a white paper covered surface could track green paper covered objects, wooden objects, etc. as well.

5.2.2 Dynamic Object Localization: Experimental

As a first iteration experiment the PSN was attached to a stationary fixture and the object was mounted on the Calvin robot so that its motion could be controlled.

As mentioned in the previous section an important assumption was overlooked during the simulations: the filter needs to know the position of each of it's sensors or systematic errors are introduced in calculating the measurement function and the Jacobian. This is a definite drawback since proper implementation hinges on correct knowledge of where each sensor is located. For the experiments conducted a precisely machined fixture was used. An alternative method is suggested for future work in Chapter 6.

In the experiment, the object moved from $(-5, 55)mm$ to $(0, 40)mm$ in $5sec$. The coordinates are in the PSN's local coordinate system. The object was in view of all the sensors throughout the experiment.

Thus, with proper tuning of the system covariance matrix, Q, a real object can be successfully tracked. The system acquired the target and tracked it over a period of 5 seconds. The state errors converged to zero in less than 0.5 seconds.

The data from this experiment was also used to demonstrate the effects of tuning the system noise covariance matrix, Q. The raw data from the run was filtered post-process twice, once with $Q1 = \text{Diag}(0.0001, 0.01, 0.0001, 0.05, 0.001)$ and then with

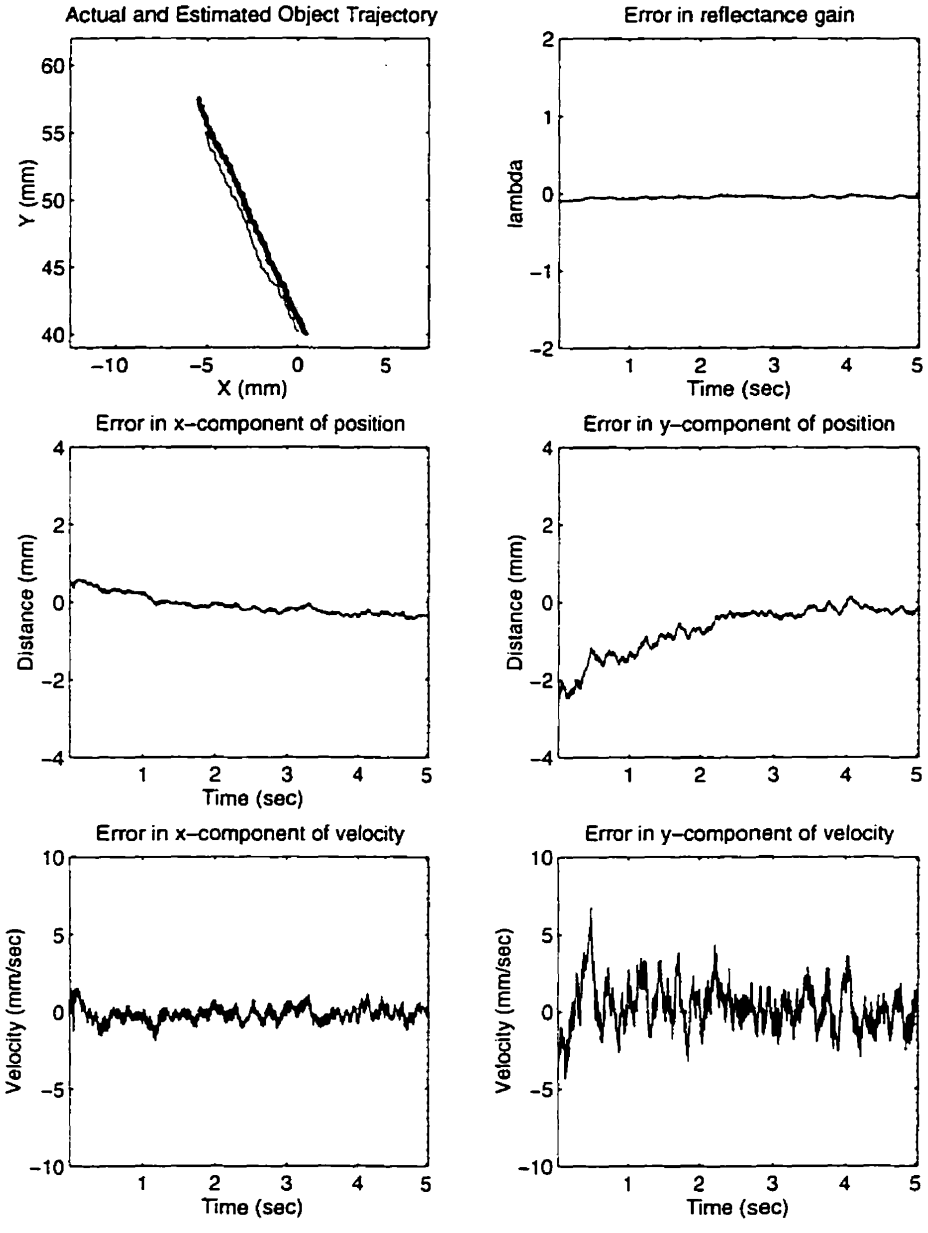


Figure 5.5: *Experimental Results: actual and estimated trajectories (top left), reflectance gain error (top right), errors in X and Y position (middle left and right, respectively), errors in X and Y velocity (bottom left and right, respectively).*

$Q2 = \text{Diag}(0.1, 10, 0.1, 20, 0.01)$. The results are given in Fig. 5.6.

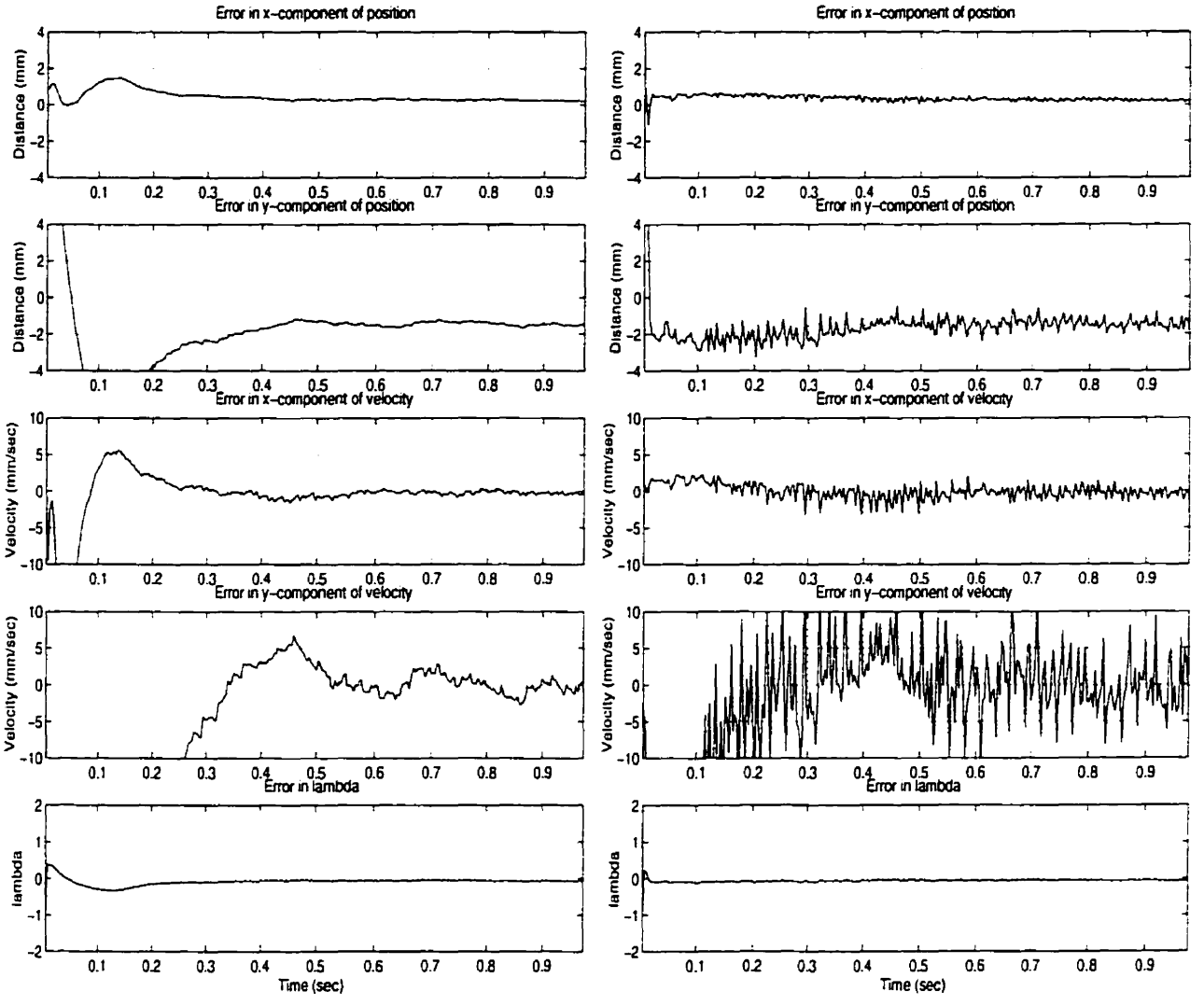


Figure 5.6: *State errors of two filters with different Q matrices. The filter which generated the output on the left had smaller elements than the filter on the right. Note the differences in speed of convergence and steady state noise.*

The errors in the states estimates are given for both filters, $Q1$ on the left and $Q2$ on the right. Comparing the convergence of the velocity estimates, it can be seen that by assigning larger weight, convergence time decreases and steady-state noise increases, as explained in Chapter 4.

5.2.3 Object Recognition: Experimental

The EKF assumes a known object shape. If the PSN acquires data from a different shape, the filter will diverge. This observation led to a series of experiments to demonstrate the system's ability to perform rudimentary object recognition.

The object recognition procedure was as follows. At initialization, several filters, each assuming a different object geometry, were launched in parallel. Each filter was given a finite period of time in which to converge. The object in the filter which converges is assumed to be the object being sensed.

The experiment performed launched several filters, each assuming a different radius for the object. The object is stationary. The results are given in Fig. 5.7.

As can be seen in this first iteration experiment, rudimentary object recognition can be performed by the system. From these experiments, however, the system could not discriminate between objects whose radii differed by less than 5mm. This is due to the measurement function where a change in radius can be compensated by a change in reflectance gain. Thus, although the system can be used to perform object recognition, it can only do so with objects which have sufficiently different geometries.

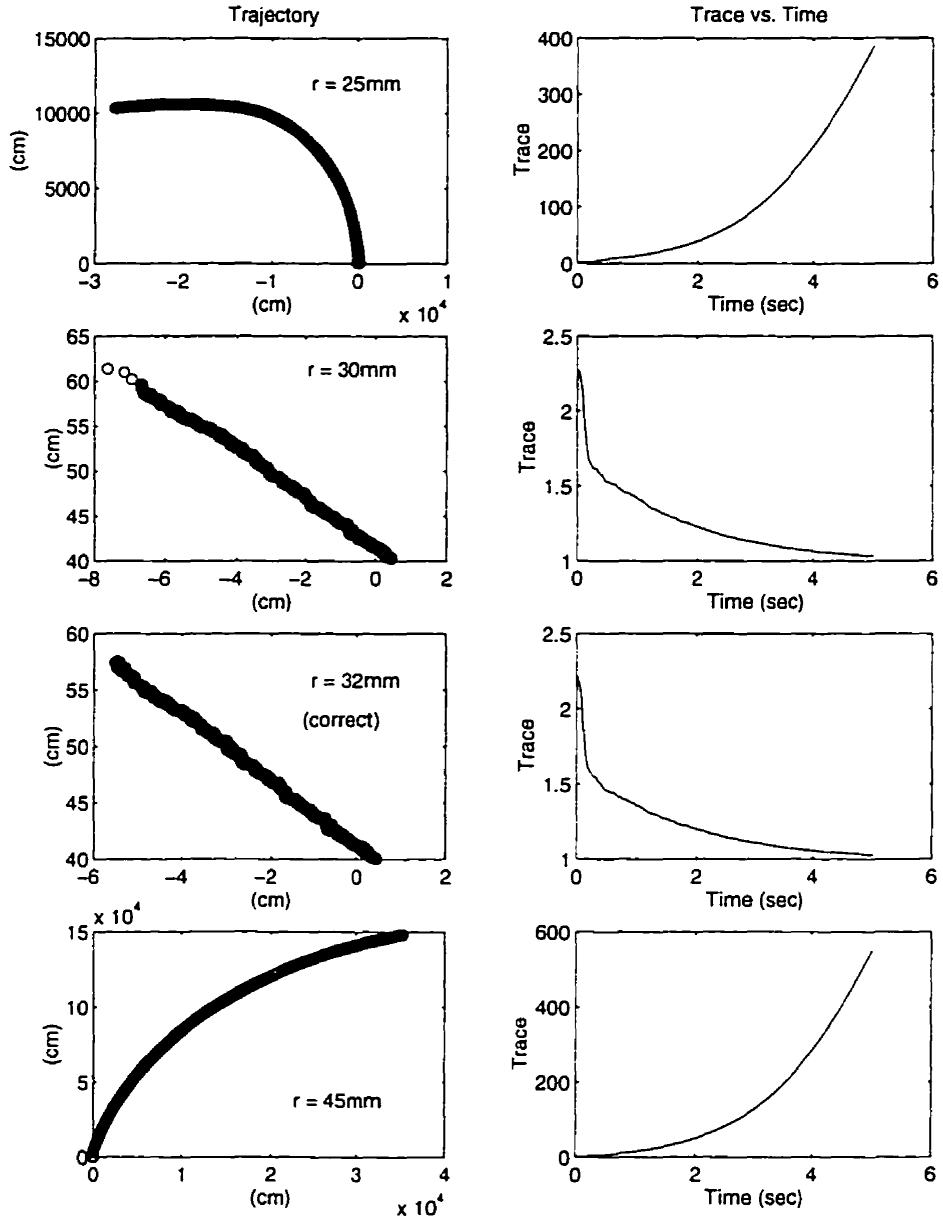


Figure 5.7: *Object recognition with a PSN. Four filters, each assuming a different radius for the object are given 5 seconds in which to converge. The estimated trajectory (left) as well as the trace of the state covariance (right) matrix are given for each filter. Two filters converge in the alloted time of 5 seconds.*

Chapter 6

Conclusion

Accurate sensing of one's environment is a precursor to autonomous operation within it. A network of miniature, amplitude-based, proximity sensors coupled with an extended Kalman filter has demonstrated it's ability to provide estimates of a sensed object's velocity and position. This information has been used in a sensor-based control algorithm to track a moving object. Although the vehicle used for demonstration was autonomous grasping, this technology can be applied to any other task requiring precise, non-contact object localization.

6.1 Future Work

Although experimentation on the C&H robotic testbed demonstrated the basic tracking of all parameters, the full extent of the system's capabilities, and limitations, have yet to be tested. The following avenues of experimentation should be explored.

The simulation results should be reproduced in reality. The system should be tested with data from the PSN used for control of the Calvin robot.

The object recognition phase should be studied further. How sensitive is the procedure to errors in the object's radius? What is the system's resolution? Later, when filters are derived for other objects such as ellipses or edged shapes, other object

recognition experiments could be conducted.

The effect of sensing ellipses with a filter which assumes circular objects was not investigated. Since the goal is to grasp 3D objects, the cylinders to be grasped will not always have their longitudinal axes perpendicular to the grasping plane. An analysis of how the filter degrades with increasingly elliptical cross-sections should be performed.

The system has only been tested on the 2D testbed. Generalization to a 3D environment is a natural extension of this work.

As mentioned in Chapter 5, the relative positions of all the sensors which make up the PSN must be known by the filter, otherwise systematic errors are introduced. Since making a precision fixture for the PSN is expensive and inconvenient, a method to perform the kinematic calibration of the PSN should be developed. One method which used the sensor outputs in an identification scheme to estimate the positions and orientations of the sensors was tried with mixed success. This research should be continued.

Finally, all experiments conducted were with a PSN with a static configuration. A PSN in which the sensors can move with respect to each other could modify it's coverage to adapt to certain conditions. Thus, a PSN could widen its coverage when no object is in sight to increase the likelihood of acquiring a target. The benefits and drawbacks of a "dynamic PSN" should be investigated.

Appendix A

Jacobian Derivation

The full analytical derivation of the Jacobian is presented here.

The measurement model (3.2) for the ith sensor is repeated below:

$$h_i = \frac{\lambda \beta_{1,i}}{(d_i + \beta_{4,i})^{\beta_{2,i}}} cos(\beta_{3,i}\theta_i). \tag{A.1}$$

where the $\beta_{k,i}$ are the curve fitting coefficients. Let us define the parameter vector

$$\mathbf{p} = [d_i, \theta_i, \lambda]^T. \tag{A.2}$$

As described in Sec. 3.2, d_i is the sensor-target distance, θ_i is the object angle with respect to the sensor's line of sight and λ is the lumped parameter of the object's surface properties. The parameters, $\mathbf{p}$, are related to the object states in sensor-fixed coordinates, ${}^{Si}\mathbf{x}$, by geometry as presented in Sec. 4.2, repeated below:

$$\theta_i = \text{asin}\left(\frac{{}^{Si}\mathbf{x}_1}{r}\right) \tag{A.3}$$

$$d_i = {}^{Si}\mathbf{x}_2 - r\cos(\theta_i). \tag{A.4}$$

The object states in the end-effector-fixed frame of reference, ${}^{G}\mathbf{x}$, are related to the

states in the sensor-fixed frame of reference, ${}^{Si}\mathbf{x}$, through a simple transformation:

$$\begin{bmatrix} {}^{Si}\mathbf{x} \\ {}^{Si}\dot{\mathbf{x}} \\ \lambda \end{bmatrix} = \begin{bmatrix} {}^{Si}R_G \ {}^{G}\mathbf{x} + {}^{Si}P_G \\ {}^{Si}R_G \ {}^{G}\dot{\mathbf{x}} \\ \lambda \end{bmatrix} \tag{A.5}$$

where,

$${}^{Si}R_G = \begin{bmatrix} c\alpha_i & s\alpha_i \\ -s\alpha_i & c\alpha_i \end{bmatrix} \tag{A.6}$$

is the rotation matrix relating the two frames of reference and,

$${}^{Si}P_G = \begin{bmatrix} -P_{x,i}c\alpha_i - P_{y,i}s\alpha \\ P_{x,i}s\alpha_i - P_{y,i}s\alpha \end{bmatrix}. \tag{A.7}$$

is the distance between the origins of the two coordinate frames. The angle of rotation between the sensor-fixed and end-effector-fixed frames of reference is α_i. The cosine and sine functions have been abbreviated to c and s, respectively.

We can now derive the Jacobian of the measurement model with respect to the states in end-effector-fixed frame representation. Using the chain rule of differentiation we have:

$$H_i = \frac{\partial[h_i]}{\partial[{}^{G}\mathbf{x}]} = \frac{\partial[h_i]}{\partial[\mathbf{p}_i]} \cdot \frac{\partial[\mathbf{p}_i]}{\partial[{}^{Si}\mathbf{x}]} \cdot \frac{\partial[{}^{Si}\mathbf{x}]}{\partial[{}^{G}\mathbf{x}]} \tag{A.8}$$

where,

$$\frac{\partial[h_i]}{\partial[\mathbf{p}_i]} = \begin{bmatrix} \frac{-\beta_{1,i}\beta_{2,i}\lambda\cos(\beta_{3,i}\theta_i)}{(d_i+\beta_{4,i})^{\beta_{2,i}+1}} & \frac{-\beta_{1,i}\beta_{3,i}\lambda\sin(\beta_{3,i}\theta_i)}{(d_i+\beta_{4,i})^{\beta_{2,i}}} & \frac{-\beta_{1,i}\cos(\beta_{3,i}\theta_i)}{(d_i+\beta_{4,i})^{\beta_{2,i}}} \end{bmatrix}^T \tag{A.9}$$

$$\frac{\partial[\mathbf{p}_i]}{\partial[{}^{Si}x]} = \begin{bmatrix} \tan\theta_i & 1 & 0 & 0 & 0 \\ \frac{1}{\cos\theta_i} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}^T \tag{A.10}$$

$$\frac{\partial[{}^{Si}\mathbf{x}]}{\partial[{}^{G}\mathbf{x}]} = \begin{bmatrix} {}^{Si}R_G & 0_{2\times2} & 0_{2\times1} \\ 0_{2\times2} & {}^{Si}R_G & 0_{2\times1} \\ 0_{1\times2} & 0_{1\times2} & 1 \end{bmatrix}^T \tag{A.11}$$

For the case in which the sensor model is numeric rather than analytic, the values for (A.9) are tabulated rather than calculated. The rest of the derivation is identical.

Multiplying the three matrices yields the Jacobian.

$$H_i = \begin{bmatrix} \Theta_i \cos\alpha_i - \Omega_i \sin\alpha_i \\ \Theta_i \sin\alpha_i - \Omega_i \cos\alpha_i \\ 0 \\ 0 \\ \Pi_i \end{bmatrix} \tag{A.12}$$

where,

$$\Omega_i = \frac{-\mathcal{J}_{1,i}\mathcal{J}_{2,i}\lambda\cos(\mathcal{J}_{3,i}\theta_i)}{(d_i + \mathcal{J}_{4,i})^{\mathcal{J}_{2,i}-1}} \tag{A.13}$$

$$\Theta_i = \frac{\mathcal{J}_{1,i}\mathcal{J}_{2,i}\lambda\cos(\mathcal{J}_{3,i}\theta_i)\tan\theta_i}{(d_i + \mathcal{J}_{4,i})^{\mathcal{J}_{2,i}-1}} - \frac{\mathcal{J}_{1,i}\mathcal{J}_{3,i}\lambda\sin(\mathcal{J}_{3,i}\theta_i)}{r\cos\theta_i(d_i + \mathcal{J}_{4,i})^{\mathcal{J}_{2,i}}} \tag{A.14}$$

$$\Pi_i = \frac{\mathcal{J}_1}{(d_i + \mathcal{J}_4)^{\mathcal{J}_2}}\cos(\mathcal{J}_3\theta_i). \tag{A.15}$$

Appendix B

Filter Derivations

The derivations of the Kalman filter, extended Kalman filter and iterated extended Kalman filter are presented below. These derivations may be found in the literature, but not all together, nor with a consistent notation from one derivation to the next. Furthermore, derivation was not just an academic excercise. In deriving the filters, important insights were gained in how to tune, initialize and implement them.

B.1 The Kalman Filter

We are observing a discretized linear dynamical system:

$$x_{k+1} = \Phi x_k + \Gamma_k w_k. \tag{B.1}$$

with the following linear measurement equation:

$$y_k = C_k x_k + v_k. \tag{B.2}$$

where w_k and v_k are white, zero mean, gaussian noise processes.

If $\hat{x}_k$ is the latest estimate of the state, then the estimated state at $k+1$ given the state at k will be:

$$\hat{x}_{k+1|k} = \Phi \hat{x}_k. \tag{B.3}$$

The Kalman Filter forms a linear combination of the latest state estimate and the latest measurement in which the increase in the state estimate covariance matrix, P, is minimized.

The linear combination takes the following form, constants J and K to be determined:

$$\hat{x}_{k+1} = J_{k+1}\hat{x}_{k+1|k} + K_{k+1}y_{k+1} \tag{B.4}$$

We now take the expected value of both sides of B.4:

$$\begin{aligned} E[\hat{x}_{k+1}] &= E[J_{k+1}\hat{x}_{k+1|k}] + E[K_{k+1}y_{k+1}] \\ E[\hat{x}_{k+1}] &= J_{k+1}E[\hat{x}_{k+1|k}] + K_{k+1}E[C_{k+1}x_{k+1} + v_{k+1}] \\ x_{k+1} &= J_{k+1}x_{k+1} + K_{k+1}C_{k+1}x_{k+1} \end{aligned}$$

Therefore,

$$I = J_{k+1} + K_{k+1}C_{k+1} \tag{B.5}$$

and,

$$J_{k+1} = I - K_{k+1}C_{k+1} \tag{B.6}$$

By substituting the newly found expression for J into B.4 we get the familiar Kalman Filter update equation:

$$\begin{aligned} \hat{x}_{k+1} &= (I - K_{k+1}C_{k+1})\hat{x}_{k+1|k} + K_{k+1}y_{k+1} \\ \hat{x}_{k+1} &= \hat{x}_{k+1|k} + K_{k+1}[y_{k+1} - C_{k+1}\hat{x}_{k+1|k}] \end{aligned} \tag{B.7}$$

We now derive the expression for the Kalman gain K. We do so recalling that the linear combination in equation B.7 must minimize the increase of $\hat{x}_{k+1|k}$'s covariance matrix, P.

We begin by deriving the covariance matrix update equation.

$$\begin{aligned} P_k &= E[(\hat{x}_k - E[\hat{x}_k])(\cdot)^T] \\ &= E[(\hat{x}_{k|k-1} + K_k[y_k - C_k\hat{x}_{k|k-1}] - x_k)(\cdot)^T] \\ &= E[((I - K_kC_k)(\hat{x}_{k|k-1} - x_k) + K_k(y_k - C_kx_k))(\cdot)^T] \end{aligned}$$

$$
\begin{aligned}
&= E[((I - K_kC_k)(\hat{x}_{k|k-1} - x_k))(\cdot)^T] \\
&\quad + E[(I - K_kC_k)(\hat{x}_{k|k-1} - x_k)(K_k(y_k - C_kx_k))^T] \\
&\quad + E[(K_k(y_k - C_kx_k))(\hat{x}_{k|k-1} - x_k)^T(I - K_kC_k)^T] \\
&\quad + E[(K_k(y_k - C_kx_k))(\cdot)^T] \qquad \text{(B.8)} \\
&= E[(I - K_kC_k)(\hat{x}_{k|k-1} - x_k)(\cdot)^T(I - K_kC_k)^T] + E[K_k(y_k - C_kx_k)(\cdot)^TK_k^T] \\
&= (I - K_kC_k)E[(\hat{x}_{k|k-1} - x_k)(\cdot)^T](I - K_kC_k)^T + K_kE[(v_k)(\cdot)^T]K_k^T \\
&= (I - K_kC_k)E[(\Phi\hat{x}_{k-1} - \Phi x_{k-1} + \Gamma_{k-1}w_{k-1})(\cdot)^T](I - K_kC_k)^T + K_kR_kK_k^T \\
&= (I - K_kC_k)\Phi E[(\hat{x}_{k-1} - E[\hat{x}_{k-1}])(\cdot)^T]\Phi^T + E[(\Gamma_{k-1}w_{k-1})(\cdot)^T](I - K_kC_k)^T \\
&\quad + K_kR_kK_k^T \\
&= (I - K_kC_k)[\Phi P_{k-1}\Phi^T + \Gamma Q_k\Gamma^T](I - K_kC_k)^T + K_kR_kK_k^T \\
&= (I - K_kC_k)P_{k|k-1}(I - K_kC_k)^T + K_kR_kK_k^T \\
&= (I - K_kC_k)P_{k|k-1}(I - C_k^TK_k^T) + K_kR_kK_k^T \qquad \text{(B.9)}
\end{aligned}
$$

The middle two terms of equation B.8 vanish under the assumption that the system error and the measurement error are uncorrelated random processes.

We now minimize ΔP with respect to a change ΔK.

$$
\begin{aligned}
\Delta P &= P(K + \Delta K) - P(K) \\
&= (I - (K_k + \Delta K)C_k)P_{k|k+1}(I - C_k^T(K_k + \Delta K)^T) + (K_k + \Delta K)R_k(K_k + \Delta K)^T \\
&\quad - (I - K_kC_k)P_{k|k-1}(I - C_k^TK_k^T) + K_kR_kK_k^T \\
&\approx \Delta K_k(-C_kP_{k|k-1}(I - C_k^TK_k^T) + R_kK_k^T) + (-(I - K_kC_k)P_{k|k-1}C_k^T + K_kR_k)\Delta K_k^T \\
&= \Delta K_k\Psi + \Upsilon\Delta K_k^T \qquad \text{(B.10)}
\end{aligned}
$$

In deriving this last equation we have neglected second order terms. We now force ΔP to zero by setting Ψ and Υ to zero. Setting $\Psi = 0$ we can solve for K_k.

$$
\begin{aligned}
0 &= -C_kP_{k|k-1}(I - C_k^TK_k^T) + R_kK_k^T \\
0 &= -C_kP_{k|k-1} + C_kP_{k|k-1}C_k^TK_k^T + R_kK_k^T \\
0 &= C_kP_{k|k-1} - (C_kP_{k|k-1}C_k^T + R_k)K_k^T
\end{aligned}
$$

$$\begin{aligned} K_k^T &= (C_k P_{k|k-1} C_k^T + R_k)^{-1} C_k P_{k|k-1} \\ K_k &= [C_k P_{k|k-1}]^T (C_k P_{k|k-1} C_k^T + R_k)^{-T} \\ K_k &= P_{k|k-1}^T C_k^T (C_k P_{k|k-1}^T C_k^T + R_k^T)^{-1} \\ K_k &= P_{k|k-1} C_k^T (C_k P_{k|k-1} C_k^T + R_k)^{-1} \end{aligned} \tag{B.11}$$

Equation B.11 is obtained by recognizing that P and R matrices are symmetric. Had we started with Υ equal to zero the result would have been identical to the one above.

We can now write the Kalman Filter equations.

Given

$$x_{k+1} = \Phi x_k + \Gamma w_k \tag{B.12}$$

$$y_k = C_k x_k + v_k \tag{B.13}$$

Prediction

$$\hat{x}_{k+1|k} = \Phi \hat{x}_k \tag{B.14}$$

$$P_{k+1|k} = \Phi P_k \Phi^T + \Gamma Q_k \Gamma^T \tag{B.15}$$

Update

$$\hat{x}_{k+1} = \hat{x}_{k+1|k} + K_{k+1}[y_{k+1} - C_{k+1}\hat{x}_{k+1|k}] \tag{B.16}$$

$$P_{k+1} = [I - K_{k+1}C_{k+1}]P_{k+1|k}[I - C_{k+1}^T K_{k+1}^T] + K_{k+1}R_{k+1}K_{k+1}^T \tag{B.17}$$

where,

$$K_{k+1} = P_{k|k-1}C_k^T[C_k P_{k|k-1} C_k^T + R_k]^{-1} \tag{B.18}$$

B.2 The Extended Kalman Filter

Extended Kalman Filtering is a derivative of the regular Kalman filter which allows for non-linear measurement and system model equations. In this case we are solely concerned with measurement non-linearity and thus the following derivation will be specific to this case.

We are observing a linear dynamical model:

$$x_{k+1} = \Phi x_k + \Gamma_k w_k, \tag{B.19}$$

with the following non-linear measurement equation:

$$y_k = h(x_k) + v_k \tag{B.20}$$

where w_k and v_k are again white, zero-mean, gaussian noise processes.

Since we wish to use linear filtering theory, we will linearize the measurement function about some nominal trajectory $\bar{x}_k$:

$$\delta x_k = x_k - \bar{x}_k \tag{B.21}$$

Let us therefore define the nominal measurement as:

$$\bar{y}_k \triangleq h(\bar{x}_k) \tag{B.22}$$

and the measurement variation as:

$$\delta y_k \triangleq y_k - \bar{y}_k \tag{B.23}$$

Therefore, using B.22 and B.23 in B.20 we have:

$$\begin{aligned} \delta y_k &= h(x_k) - h(\bar{x}_k) + v_k \\ &\approx H[\bar{x}]\delta x_k + v_k \end{aligned} \tag{B.24}$$

Equation B.24 is the first order Taylor expansion of $h(x)$. The linearized measurement matrix, H, is defined as:

$$H_k = \left[\frac{\partial h_i(\bar{x}_k)}{\partial x_j}\right] \tag{B.25}$$

which is the Jacobian of the measurement equation. The index, i, on the measurement equation, h, reminds us that it is possible to have more than one measurement equation. The index on the state x in the denominator specifies that the partial is to be taken with respect to each variable in the state vector. Thus, H, is an $i \times j$ matrix.

After linearizing the measurement equation, we have the following pair of linear equations:

$$\begin{aligned} x_{k+1} &= \Phi x_k + w_k \\ \delta y_k &= H_k \delta x_k + v_k \end{aligned} \tag{B.26}$$

but what is the nominal trajectory $\bar{x}_k$? The obvious choice is the latest estimate $\hat{x}_{k+1|k}$.

We now use these linear equations in the Kalman Filter. The only equation affected is the update equation. Substituting δy_{k+1} for y_{k+1} and $H_{k+1}\delta\hat{x}_{k+1|k}$ for $C_{k+1}\hat{x}_{k+1|k}$ in equation B.7 we get:

$$\hat{x}_{k+1} = \hat{x}_{k+1|k} + K_k[\delta y_{k+1} - H_{k+1}\delta\hat{x}_{k+1|k}] \tag{B.27}$$

but,

$$\delta\hat{x}_{k+1|k} = 0 \tag{B.28}$$

Equation B.28 requires some additional explanation. Let us consider the definition of δx in equation B.21. Following its form, we see that:

$$\delta\hat{x}_{k|k} = \hat{x}_{k|k} - \bar{x}_k \tag{B.29}$$

When $k = 0$, the initial condition is $\bar{x}_0 = \hat{x}_{0|0}$. Therefore,

$$\delta\hat{x}_{0|0} = \hat{x}_{0|0} - \hat{x}_{0|0} = 0 \tag{B.30}$$

And thus,

$$\delta\hat{x}_{1|0} = \Phi\delta\hat{x}_{0|0} = \Phi \cdot 0 = 0 \tag{B.31}$$

When we go to the next time step, the above repeats, since when $k = 1$, $\bar{x}_1 = \hat{x}_{1|1}$. Thus for all k:

$$\delta\hat{x}_{k|k} = \delta\hat{x}_{k+1|k} = 0 \tag{B.32}$$

Therefore, to complete the derivation we have:

$$\begin{aligned} \hat{x}_{k+1} &= \hat{x}_{k+1|k} + K_k[\delta y_{k+1} - H_{k+1} \cdot 0] \\ \hat{x}_{k+1} &= \hat{x}_{k+1|k} + K_k[\delta y_{k+1}] \\ \hat{x}_{k+1} &= \hat{x}_{k+1|k} + K_k[y_{k+1} - h(\hat{x}_{k+1|k})] \end{aligned} \tag{B.33}$$

which is the update equation for the EKF with measurement non-linearity.

The filter equations are therefore:

Prediction

$$\hat{x}_{k+1|k} = \Phi \hat{x}_k \tag{B.34}$$

$$P_{k+1|k} = \Phi P_k \Phi^T + Q \tag{B.35}$$

Update

$$\hat{x}_{k+1|k+1} = \hat{x}_{k+1|k} + K[y_{k+1} - h(x_{k+1|k})] \tag{B.36}$$

$$P_{k+1|k+1} = [I - KH_{k+1}]P_{k+1|k}[I - KH_{k+1}]^T + KRK^T \tag{B.37}$$

the Kalman gain is,

$$K = P_{k|k-1}H_k^T[H_k P_{k|k-1} H_k^T + R]^{-1} \tag{B.38}$$

B.3 The Iterated Extended Kalman Filter

The logic behind the IEKF was explained in Sec. 2.2.4. The derivation for the local iterations begins with equation B.27, repeated below:

$$\hat{x}_{k+1} = \hat{x}_{k+1|k} + K_k[\delta y_{k+1} - H_{k+1}\delta\hat{x}_{k+1|k}] \tag{B.39}$$

In the EKF it was shown that $\delta\hat{x}_{k|k}$ and $\delta\hat{x}_{k+1|k}$ were always zero. For the IEKF, however, additional iterations between successive measurement updates occur and thus the linearization is being performed around the lth local iteration's estimate $\hat{x}_{k+1|l}$, not $\hat{x}_{k+1|k}$. The update equation should therefore read:

$$\hat{x}_{l+1} = \hat{x}_{k+1|k} + K_l[\delta y_{k+1} - H_l\delta\hat{x}_{k+1|l}] \tag{B.40}$$

In the IEKF $\delta\hat{x}_{k+1|l}$ does not go to zero as did the analogous term in the EKF. This is seen by noting, for local iterations i:

$$\begin{aligned}\delta\hat{x}_{i|i} &= \hat{x}_{i|i} - \bar{x}_i \\ \delta\hat{x}_{0|0} &= \hat{x}_{0|0} - \bar{x}_0 \\ \delta\hat{x}_{0|0} &= 0\end{aligned}$$

but for local iterations,

$$\delta\hat{x}_{1|0} \neq \Phi\delta\hat{x}_{0|0} \tag{B.41}$$

Therefore,

$$\begin{aligned}\delta\hat{x}_{1|0} &= \hat{x}_{1|0} - \bar{x}_0 \\ \delta\hat{x}_{1|0} &= \hat{x}_{1|0} - \hat{x}_{0|0} \neq 0\end{aligned}$$

Note that in the before last equation above $\bar{x}_0 = \hat{x}_{1|0}$ cannot be used since $\hat{x}_{1|0}$ is a function of $\hat{x}_{0|0}$.

The derivation is now completed, continuing from equation B.40.

$$\begin{aligned}\hat{x}_{l+1} &= \hat{x}_{k+1|k} + K_l[\delta y_i - H_l(\hat{x}_{k+1|k} - \hat{x}_l)] \\ \hat{x}_{l+1} &= \hat{x}_{k+1|k} + K_l[y_{k+1} - h_{\hat{x}_l} - H_l(\hat{x}_{k+1|k} - \hat{x}_l)]\end{aligned} \tag{B.42}$$

which is the IEKF update equation. Thus the iterated extended Kalman filter equations are:

Prediction

Same as for extended Kalman filter.

Iterations

$$\hat{x}_{l=0} = \hat{x}_{k+1|k} \tag{B.43}$$

$$\hat{x}_{l+1} = \hat{x}_{k+1|k} + K_l[y_{k+1} - h(\hat{x}_l) - H_l(\hat{x}_{k+1|k} - \hat{x}_l)] \tag{B.44}$$

$$K_l = P_{k+1|k}H_l^T[H_lP_{k+1|k}H_l^T + R]^{-1} \tag{B.45}$$

Iterations stop when the difference between two successive estimates of $\hat{x}$ is less than an arbitrarily set tolerance. The resulting estimate at the end of the local iterations, ie. at $l = lf$, are:

Estimation

$$\hat{x}_{k+1|k+1} = \hat{x}_{lf} \tag{B.46}$$

$$P_{k+1|k+1} = [I - K_{lf}H_{lf}]P_{k+1|k}[I - K_{lf}H_{lf}]^T + K_{lf}RK_{lf}^T \tag{B.47}$$

Bibliography

[1] J.K. Aggarwal and M.J. Magee. Determining motion parameters using intensity guided range sensing. *Pattern Recognition*, 19(2):169–180, 1986.

[2] P.K. Allen, A. Timcenko, B. Yoshimi, and P. Michelman. Automated tracking and grasping of a moving object with a robotic hand-eye system. *IEEE Trans. Robotics and Automation*, 9(2):152–165, 1993.

[3] K.J. Astrom and B. Wittenmark. *Computer-Controlled Systems: Theory and Design*. Prentice-Hall Inc., Toronto, 1990.

[4] D.J. Bennett and J.M. Hollerbach. Autonomous calibration of single-loop closed kinematic chains formed by manipulators with passive endpoint constraints. *IEEE Trans. Robotics and Automation*, 7(5), Oct 1991.

[5] A. Bonen, M. Parent, K.C. Smith, and B. Benhabib. Development of a robust electro-optical proximity sensor. In *IEEE/RSJ Intelligent Robots and Systems*, pages 986–990, Yokohama, Japan, Jul 1993.

[6] T.J. Broida. Kinematic and statistical models for data fusion using kalman filtering. In M.A. Abidi and R.C. Gonzalez, editors, *Data Fusion in Robotics and Machine Intelligence*, pages 311–365. Academic Press, London, 1992.

[7] T.J. Broida and R. Chellappa. Estimation of object motion parameters from noisy images. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 8:90–99, Jan 1986.

[8] T.J. Broida and R. Chellappa. Estimating the kinematics and structure of a rigid object from a sequence of monocular images. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 13(6):497–512, Jun 1991.

[9] C. Brown, H. Durrant-Whyte, J. Leonard, B. Rao, and B. Steer. Distributed data fusion using kalman filtering: A robotics application. In M.A. Abidi and R.C. Gonzalez, editors. *Data Fusion in Robotics and Machine Intelligence*, pages 267–309. Academic Press, London, 1992.

[10] M. Buehler, D. E. Koditschek, and P. J. Kindlmann. Planning and Control of Robotic Juggling and Catching Tasks. *Int. J. Robotics Research*, 13(2):101–118, 1994.

[11] E. Cheung and V. Lumelsky. Development of sensitive skin for a 3d robot arm operating in an uncertain environment. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 1056–1061, Piscataway, NJ, 1989.

[12] J. Damianakis. Sensor based manipulation using a proximity sensor network. M.Eng. Thesis, McGill University, Dec 1996.

[13] F.E. Daum and R.J. Fitzgerald. Decoupled kalman filters for phased array radar tracking. *IEEE Trans. Automatic Control*, 28(3):269–283, Mar 1985.

[14] H.F. Durrant-Whyte. Sensor models and multisensor integration. *Int. J. Robotics Research*, 7(6):97–113, Dec 1988.

[15] B. Espiau and J.Y. Catros. Use of optical reflectance sensors in robotics applications. *IEEE Trans. Systems, Man, and Cybernetics*, 10(12):903–912, Dec 1980.

[16] Sensor Technologie München GmbH. *STM*, Germany, 1995.

[17] Optek Technology Inc. *Data Book*, Carrollton, TX, 1990.

[18] A.H. Jazwinski. *Stochastic Processes and Filtering Theory.* Academic Press Inc. Ltd., London, 1970.

[19] S.M. Juds. *Photoelectric Sensors and Controls.* Marcel Dekker, Inc., NY, 1988.

[20] T. Kanade and T. Sommer. An optical proximity sensor for measuring surface position and orientation for robot manipulation. In M. Brady and R. Paul, editors, *Robotics Research: The First International Symposium*, pages 547–563. The MIT Press, 1984.

[21] M. D. Levine. *Vision in Man and Machine.* McGraw Hill, Montreal, 1985.

[22] Y.F. Li. Robot end-effector orientation control using proximity sensors. *Robotics and Computer-Integrated Manufacturing*, 10(5):323–331, 1993.

[23] Y.F. Li. Characteristics and signal processing of a proximity sensor. *Robotica*, 12:335–41, 1994.

[24] INMOS Limited. *IMS T222 transputer.* Bristol, U.K., 1988.

[25] INMOS Limited. *IMS T805 transputer.* Bristol, U.K., 1988.

[26] R.C. Luo and M.G. Kay. Data fusion and sensor integration: State-of-the-art 1990s. In M.A. Abidi and R.C. Gonzalez, editors, *Data Fusion in Robotics and Machine Intelligence*, pages 7–108. Academic Press, San Diego, 1992.

[27] J. Marszalec. A proximity sensing system for an intelligent optically-powered robot gripper. *I. J. Optoelectronics*, 4(3/4):343–355, May-Aug 1989.

[28] R. Masuda. Multifunctional optical proximity sensor using phase modulation. *J. Robotic Systems*, 3(2):137–147, 1986.

[29] W. Mendenhall and R.J. Beaver. *Introduction to Probability and Statistics.* PWS-Kent Publ. Co., Boston, 1991.

[30] J.P. Norton. *An Introduction to Identification.* Academic Press Inc. Ltd., London, 1986.

[31] T. Okada and U. Rembold. Proximity sensor using a spiral-shaped light-emitting mechanism. *IEEE Trans. Robotics and Automation*, 7(6):798–805, Dec 1991.

[32] O. Partaatmadja, B. Benhabib, and A. Goldenberg. Analysis and design of a robotic distance sensor. *J. Robotic Systems*, 10(4):427–445, 1993.

[33] W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling. *Numerical Recipies in C: The Art of Scientific Computing.* Cambridge University Press, NY, 1988.

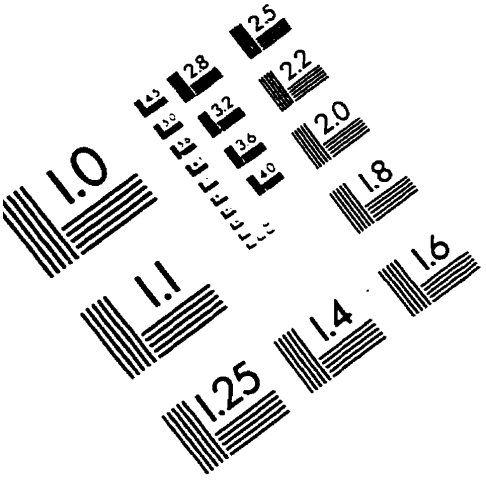
[34] I.B. Rhodes. A tutorial introduction to estimation and filtering. *IEEE Trans. Automatic Control*, 16(6):688–706, 1971.

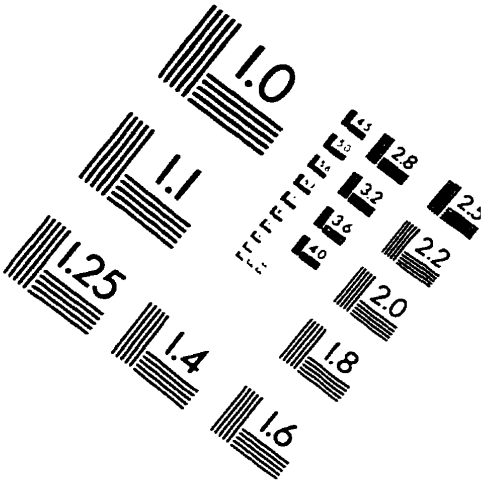
[35] G. Skofteland and G. Hirzinger. Computing position and orientation of a freeflying polyhedron from 3d data. *Proc. IEEE Int. Conf. Robotics and Automation*, pages 150–155, Apr 1991.

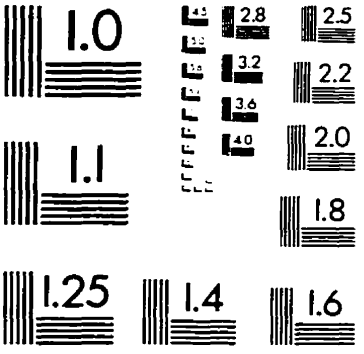
[36] H. W. Sorenson. Least-squares estimation from gauss to kalman. *IEEE Spectrum*, 7:63–68, Jul 1970.

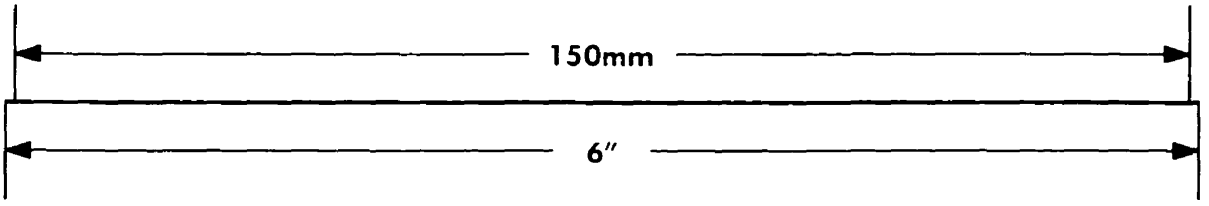
[37] The MathWorks Inc. *Matlab v4.2c.* Natick, MA, 1994.

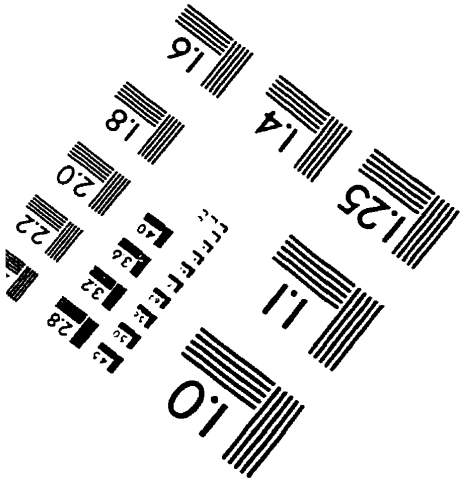
IMAGE EVALUATION
TEST TARGET (QA-3)

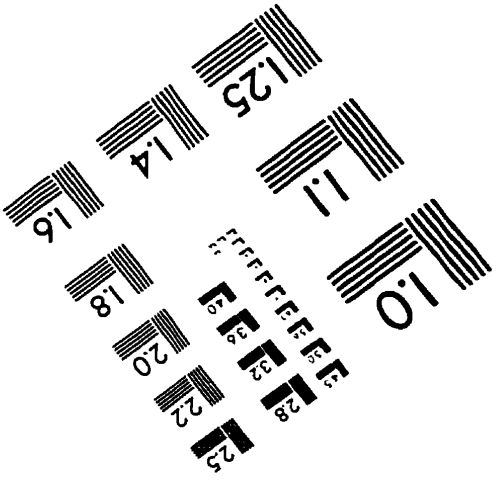












APPLIED IMAGE, Inc
1653 East Main Street
Rochester, NY 14609 USA
Phone: 716/482-0300
Fax: 716/288-5989