RADIUS OF CURVATURE AND LOCATION ESTIMATION OF CYLINDRICAL OBJECTS WITH SONAR USING A MULTI-SENSOR CONFIGURATION

A THESIS
SUBMITTED TO THE DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING
AND THE INSTITUTE OF ENGINEERING AND SCIENCE
OF BILKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE

By

Ali Şafak Sekmen

July 1997

RADIUS OF CURVATURE AND LOCATION ESTIMATION OF CYLINDRICAL OBJECTS WITH SONAR USING A MULTI-SENSOR CONFIGURATION

A THESIS

SUBMITTED TO THE DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

AND THE INSTITUTE OF ENGINEERING AND SCIENCES

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

Ali Safak Sekmen.

By

Ali Şafak Sekmen

July 1997

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

[illegible]

Assist. Prof. Dr. Billur Barshan(Supervisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

[illegible]

Prof. Dr. Hayrettin Köymen

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

[illegible]

Assist. Prof. Dr. Orhan Arıkan

Approved for the Institute of Engineering and Sciences:

[illegible]

Prof. Dr. Mehmet Baray Y.

Director of Institute of Engineering and Sciences

ABSTRACT

RADIUS OF CURVATURE AND LOCATION ESTIMATION OF CYLINDRICAL OBJECTS WITH SONAR USING A MULTI-SENSOR CONFIGURATION

Ali Şafak Sekmen
M.S. in Electrical and Electronics Engineering
Supervisor: Assist. Prof. Dr. Billur Barshan
July 1997

Despite their limitations, sonar sensors are very popular in time-of-flight measuring systems since they are inexpensive and convenient. One of the most important limitations of sonar is its low angular resolution. An adjustable multi-sonar configuration consisting of three transmitter/receiver ultrasonic transducers is used to improve the resolution. The radius of curvature estimation of cylindrical objects is accomplished with this configuration. Two different ways of rotating the transducers are considered. First, the sensors are rotated around their *joints*. Second, the sensors are rotated around their *centers*. Also, two methods of time-of-flight estimation are implemented which are *thresholding* and *curve-fitting*. Sensitivity analysis of the radius of curvature with respect to some important parameters is made. The bias-variance combinations of both estimators are compared to the Cramér-Rao lower bound. Theory and simulations are verified by experimental data from real sonar systems. Data is smoothed by extended Kalman filtering. Rotating around the center works better than rotating around the joint. Curve-fitting method is shown to be better than thresholding method both in the absence and presence of noise. The best results are obtained when the sensors are rotated around their centers and the curve-fitting method is used to estimate the time-of-flight. There is about 30% improvement in the absence of noise and 50% improvement in the presence of noise.

ÖZET

ÇOKLU SONAR DÜZENLEŞİMİ KULLANARAK SİLİNDİRİK CİSİMLERİN YARIÇAP VE KONUM KESTİRİMİ

Ali Şafak Sekmen
Elektrik ve Elektronik Mühendisliği Bölümü Yüksek Lisans
Tez Yöneticisi: Yrd. Doç. Dr. Billur Barshan
Temmuz 1997

Ucuz ve kullanımlarının kolay olması nedeniyle, bazı sınırlamaları olmasına rağmen, sonar algılayıcılar uçuş zamanı ölçümlerinde çok sık kullanılmaktadırlar. Sahip oldukları sınırlamaların ilki düşük açısal çözünürlüktür. Bu çalışmada, çözünürlüğü arttırmak için, üç adet alıcı/verici ultrasonik sonar algılayıcıdan oluşan ayarlanabilir çoklu sonar düzenleşimi kullanıldı. Bu düzenleşim ile silindirik cisimlerin yarıçap kestirimleri yapıldı. Algılayıcı döndürümünde iki yöntem kullanıldı. İlk olarak, algılayıcılar *askıları* etrafında döndürüldüler. İkinci olarak algılayıcılar *merkezleri* etrafında döndürüldüler. Ayrıca, uçuş zamanı kestiriminde de *eşikleme* ve *eğri-uyarlama* yöntemleri uygulandı. Ayrıca kestiricilerin başarımını saptamak için Cramér-Rao alt sınır karşılaştırması yapılmıştır. Analiz ve benzetim sonuçları, gerçek sonar sistemlerinden alınan veriler kullanılarak desteklenmiştir. Genişletilmiş Kalman süzgeci ile veri düzleştirilmiştir. Algılayıcıların merkezleri etrafında döndürülmeleri daha iyi sonuç vermiştir. Eğri-uyarlama yönteminin eşikleme yönteminden daha iyi çalıştığı görülmüştür. Eğri-uyarlama yönteminin gürültüsüz bir ortamda yaklaşık %30 ve gürültülü bir ortamda yaklaşık %50 daha iyi sonuç verdiği gözlemlenmiştir.

ACKNOWLEDGMENTS

I am greatly thankful to my thesis advisor Dr. Billur Barshan for her supervision, guidance, suggestions and encouragement throughout the development of this thesis.

I would like to thank to my committee members Prof. Hayrettin Köymen and Dr. Orhan Arıkan.

Special thanks to İlknur for her help and patience.

Many thanks to Birsel Ayrulu and M. Ertuğrul Öner for their help and encouragement.

LIST OF SYMBOLS

p_{max}	propagation pressure
r_{min}	minimum range
r	range
$J_1(.)$	Bessel function of first order
a	radius of the transducer
θ	deviation angle
θ_o	half beam-width angle
λ	wavelength
ρ_c	reflection coefficient
A_{max}	maximum amplitude
t_o	time-of-flight
f_o	resonant frequency
d	transducer separation
h_{o1}	distance between the central transducer and the object
h_{l1}	distance between the left transducer and the object
h_{r1}	distance between the right transducer and the object
R	radius of curvature
r_{o1}	range of the central transducer
r_{l1}	range of the left transducer
r_{r1}	range of the right transducer
r_{o2}	range of the central transducer after rotation
r_{l2}	range of the left transducer after rotation
r_{r2}	range of the right transducer after rotation
α	inclination angle of the left transducer
η	inclination angle of the right transducer
c	the speed of sound
ϕ	rotation angle of the left transducer
β	rotation angle of the right transducer
R_{2l}	radius of curvature estimation after rotating the left sensor

R_{2r}	radius of curvature estimation after rotating the right sensor
R_2	radius of curvature estimation after rotating the sensors
$f(.)$	radius of curvature estimate
ΔR	perturbation on the radius
Δh_{o1}	perturbation on h_{o1}
Δh_{r1}	perturbation on h_{r1}
$\sigma^2_{\hat{R}}$	standard deviation of R estimate
$b^2(R)$	bias of R estimate
$\sigma^2_{\hat{r}}$	standard deviation of r estimate
$b^2(r)$	bias of r estimate
$\sigma^2_{\hat{\theta}}$	standard deviation of θ estimate
$b^2(\theta)$	bias of θ estimate
w_o	noise on h_{o1}
w_l	noise on h_{l1}
w_r	noise on h_{r1}
$\hat{\mathbf{z}}$	measurement vector
$p(\hat{\mathbf{z}}\|r, \theta, R)$	conditional probability density function of $\hat{\mathbf{z}}$
$\mathbf{x}(k)$	state vector
$\mathbf{z}(k)$	observation vector
$\mathbf{H}(k)$	Jacobian matrix
$\hat{r}$	maximum likelihood estimate of range
$\hat{\theta}$	maximum likelihood estimate of angle
$\hat{R}$	maximum likelihood estimate of radius
$\mathbf{J}$	Fisher information matrix
$\mathbf{C}$	error correlation matrix
$\mathbf{Q}$	process noise correlation matrix
$\mathbf{R}$	measurement noise correlation matrix
$\mathbf{P}$	state covariance matrix
$\mathbf{P}$	residual covariance matrix
$\mathbf{W}$	filter gain matrix

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

INTRODUCTION

Ultrasonic sensors are a convenient and inexpensive means for a mobile robot to build a model of its environment. However, these sensors have some limitations such as high beam-width which makes it difficult to localize objects correctly, and multiple reflections which may be difficult to interpret. In order to decrease the effect of these limitations, an adaptive multi-sensor configuration is used, composed of three transmitter/receiver ultrasonic transducers. This way, the radius of curvature and location of the reflecting object can be estimated. With the estimation of the radius of curvature, different types of reflectors such as walls, cylinders and edges can be discriminated. For large values of radius, the object can be assumed to be a planar wall, and for values close to zero, the object can be assumed to be an edge [1].

Barshan and Kuc differentiated sonar reflections from corners and planes by using a multi-transducer sensing system [2]. Kuc used a system which adaptively changes its position and configuration in response to the echoes it detects [3]. In [4], Kleeman and Kuc classified the target primitives as plane, corner, edge and unknown, and showed that in order to distinguish these, two receivers and two transmitters are necessary and sufficient in a non-adaptive configuration. In [5], Sasaki and Takano showed that the echo signals from the same object in different locations may have more variation than that from different objects. In order to decrease the limitations of sonar, Flynn used a multi-sensor configuration consisting of infrared and sonar sensors [6]. In [7, 8, 9], acoustic sensors were used for sonar mapping. Ohya and Yuta investigated how the information obtained by the ultrasonic sensor is affected by the characteristics of the sensing systems such as sensitivity and directivity [10]. In [11], Sabatini illustrated that advanced filtering methods are required for making data more accurate and reliable. He also proposed a digital-signal-processing technique for building a transducer array capable of automatically compensating for variations in the speed of sound due to temperature or any other atmospheric conditions [12]. Curran and Kyriakopoulos used an extended Kalman filter to combine dead-reckoning, ultrasonic, and infrared sensor data to estimate current position and orientation of a mobile robot [13]. Webb, Gibson and Wykes used novel sensors to measure the range and bearing of a target and guide the robot [14]. In [15, 16, 17], acoustic sensors were used for robot navigation. In [18], Ko, Kim, and Chung used sensors to extract multiple landmarks for the indoor navigation of a mobile robot. Chang and Song solved the beam-opening angle problem by fusing data from multiple ultrasonic sensors [19]. In [20, 21, 22], a sensor with the flexibility to track 3-D targets which are not necessarily in the same horizontal plane as the sensors is investigated. Moreover, radius of curvature estimation is important in image analysis to provide viewpoint-independent cues for shape classification [23].

In this thesis, the work done in [1] and [24] is extended to an adaptive tri-aural sensor for improved radius of curvature and location estimations of targets. When the object and the sensor are not perpendicular to each other, there is an exponential decline in the amplitude of the reflected sonar signal which decreases the signal-to-noise ratio. In order to avoid this problem, an adaptive tri-aural sensor is developed. Depending on the location of the object, the sensor can rotate its transducers to get a more accurate radius of curvature estimation. First, a tri-aural sensor composed of three sensors is employed. According to the measurements made with this set-up, the peripheral sensors are rotated around the joints. Also, the case in which the sensors are rotated around the centers is investigated.

In Chapter 2, background information on sonar sensors is given. The main reason why the sonar sensors are rotated is discussed. Also, rotated configurations of the sonar sensors are illustrated.

In Chapter 3, radius of curvature estimation is described. In order to estimate the time-of-flight values, two different methods are used. First, the traditional thresholding method is tested which is fast but biased and suboptimal. Second, the curve-fitting method which is slower but more accurate compared to the thresholding method is used. The sensitivity of estimation to the distance between sensors, the true radius, and the range is investigated. In Section 3.1, the case in which the sensors are rotated around their joints is investigated. In Section 3.1.1, the sensitivity analysis of the radius of curvature is performed. In Section 3.2, the case in which the sensors are rotated around their centers is investigated. Noiseless and noisy cases are studied separately. In order to obtain reliable results in a noisy environment, a 100-iteration Monte Carlo simulation study is performed. In Section 3.3, the curve-fitting method is described. In Section 3.4, in order to evaluate the estimation performance, a comparison with the Cramér-Rao lower bound is made.

In Chapter 4, the radius of curvature estimation is extended to location estimation.

In Chapter 5, an extended Kalman filter is used to estimate the radius of curvature and location of cylindrical objects.

In Chapter 6, the experimental results of both thresholding and curve-fitting methods are presented.

In Chapter 7, conclusions are drawn and directions for future work are motivated.

In Appendix A, the details of the Cramér-Rao lower bound calculation are given. In Appendix B, the model for the extended Kalman Filter which is used to estimate location and radius of curvature of cylindrical objects is detailed. In Appendix C, the properties of chi-square distributed random variables are presented. In Appendix D, the computer programs with which the simulations are performed and the experimental results are analyzed are presented.

Chapter 2

SONAR SENSING

In this chapter, some properties of sonar sensing are discussed. The multi-sonar configuration is introduced and the reasons why this configuration is used are given.

2.1 Acoustic Transducers

Ultrasonic transducers are acoustic devices having a resonant frequency higher than 20 kHz. They have been used widely in burglar alarm systems, proximity switches, anti-collision devices, counter for moving objects and TV remote control systems. Since the characteristics of the sound produced by an acoustic transducer change with distance, the near-field region (the neighborhood of the transducer) and the far-field region (beyond the near-field) are investigated separately. The near-field region is called as Fresnel diffraction zone and the far-field region is called as Fraunhofer zone [25]. The expression for the sound pressure within the near-field is relatively complex, and not in the scope of this thesis. The far-field characteristics at range r and angular deviation θ for a *single* frequency of excitation is described by [26, 27]

$$A(r,\theta) = \frac{p_{max} r_{min}}{r} \frac{J_1(ka\sin\theta)}{ka\sin\theta} \qquad \text{for} \qquad r \geq r_{min} \tag{2.1}$$

where $J_1(.)$ is the Bessel function of first order, and p_{max} is the propagation pressure amplitude on the beam axis at range r_{min} along the line-of-sight.

The half beam-width θ_o in the far-field region corresponds to the first zero of the Bessel function in Equation 2.1 which occurs at $ka\sin\theta = 1.22\pi$ and the

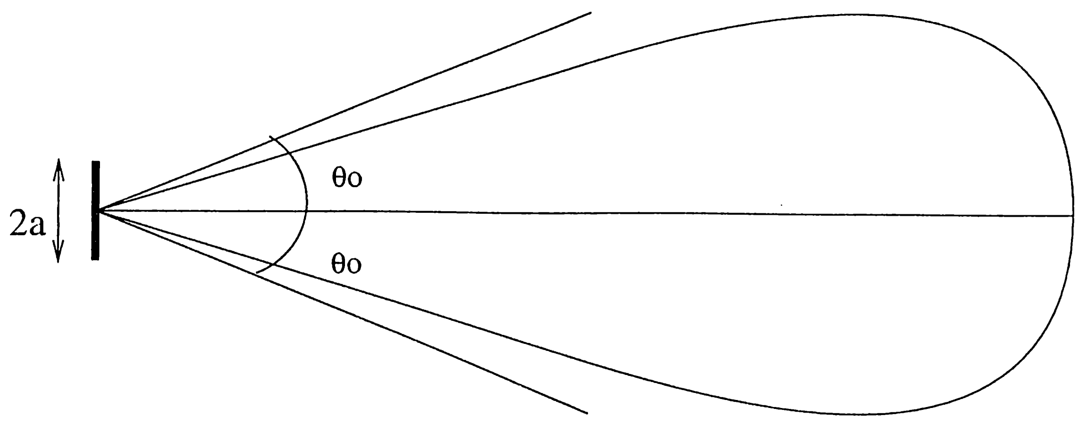


Figure 2.1: The main lobe of the radiation pattern of the Polaroid transducer.

following equation is obtained for the half beam-width angle [28]:

$$\theta_o = \sin^{-1}\left[\frac{0.61\lambda}{a}\right] \tag{2.2}$$

where $\lambda = c/f$ is the wavelength, and a is the radius of the transducer.[1] The radiation pattern is shown in Figure 2.1.

Since a range of frequencies around f_o are transmitted, the corresponding beam patterns are superposed and the resulting beam pattern can be approximated by a Gaussian beam profile centered around zero with standard deviation $\sigma_\theta = \frac{\theta_o}{2}$ [16, 27]:

$$\tilde{A}\,(r,\theta) = \frac{p_{max} r_{min}}{r} e^{-\frac{\theta^2}{2\sigma_\theta^2}} \qquad \text{for} \qquad r \geq r_{min} \tag{2.3}$$

Since the position and radius of curvature of cylindrical objects are going to be estimated, a reflected sonar signal model for cylindrical objects is needed. The signal model is basically a sinusoidal enveloped by a Gaussian which is given by [27, 29]:

$$A(r,\theta,d,t) = \rho_c \frac{A_{max} r_{min}^{3/2}}{r^{3/2}} e^{-\frac{\theta^2}{\sigma_\theta^2}} e^{-\frac{(t-t_o-\frac{3}{f_o})^2}{\sigma_t^2}} \sin\left[2\pi f_o(t-t_o)\right] \tag{2.4}$$

where ρ_c is the reflection coefficient depending on the radius of curvature (for $r > 5$ cm, $\rho_c = 0.0044946R - 0.0022471$ [29]), A_{max} is the maximum amplitude, r is the distance between the sensor and the center of the object, $r_{min} \cong a^2/\lambda$ (a is the radius of the transducer aperture, 0.65 cm for the Panasonic and 2

[1]$c = 331.4\sqrt{\frac{T}{273}}$ m/s, where T is absolute temperature in Kelvin. At room temperature, $c = 343.3$ m/s.

cm for the Polaroid), θ is the angle normal to the receiver with respect to the object, $\sigma_\theta = \theta_o/2$ (θ_o is the half beam-width angle), t_o is the time-of-flight, f_o is the resonant frequency (49.4 kHz for Polaroid, 40 kHz for Panasonic), and $\sigma_t = 1/f_o$.

The cross-section of the Panasonic transducer is pictured in Figure 2.2.

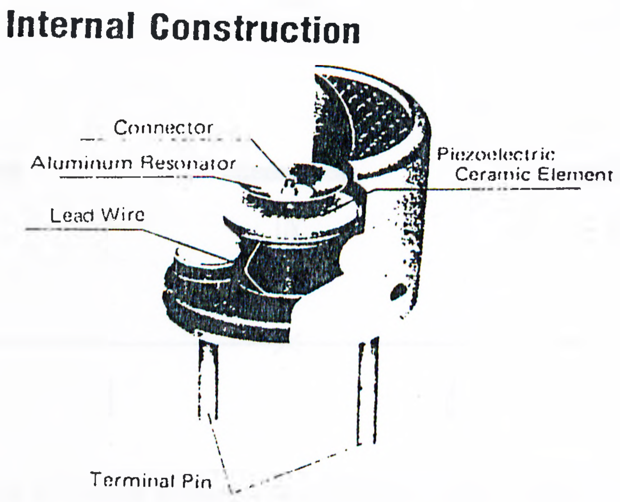


Figure 2.2: Cross-section of the Panasonic transducer.

2.2 Tri-aural Sensor Configuration

In this study, a tri-aural sensor composed of three sensors was employed as shown in Figure 2.3. Each one of the transducers is sensitive to echo signals reflected within its beam pattern. The actual beam pattern is similar to the region illuminated by a flashlight. All members of the tri-aural configuration can detect targets located within the overlap of the three beam patterns, which is called the *sensitivity region*, as illustrated in Figure 2.4. In fact, the boundaries of the sensitivity region change with the type of object. For example, for edge-like or pole-like targets, this region is much smaller but of similar shape, and for planes, it is more extended.

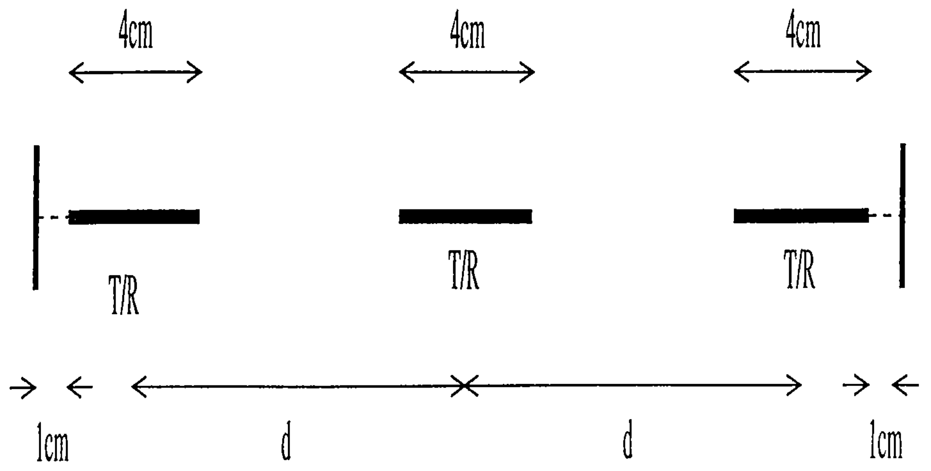


Figure 2.3: The tri-aural configuration where the sensors are aligned.

As seen from Equation 2.4, when the object and the sensor are not perpendicular to each other ($\theta \neq 0°$), there is an exponential decline in the amplitude which decreases the signal-to-noise ratio. Hence, information provided by sonar sensors is most reliable when the object is perpendicular to the sensor, and at nearby ranges due to the $1/r^{3/2}$ term in Equation 2.4. Because of this, the transducers are rotated to make them orthogonal to the object. In this study, two methods to rotate the transducers are used. First, the sensors are rotated around their joints as shown in Figure 2.5. Second, the sensors are rotated around their centers as will be shown later in Figure 3.14.

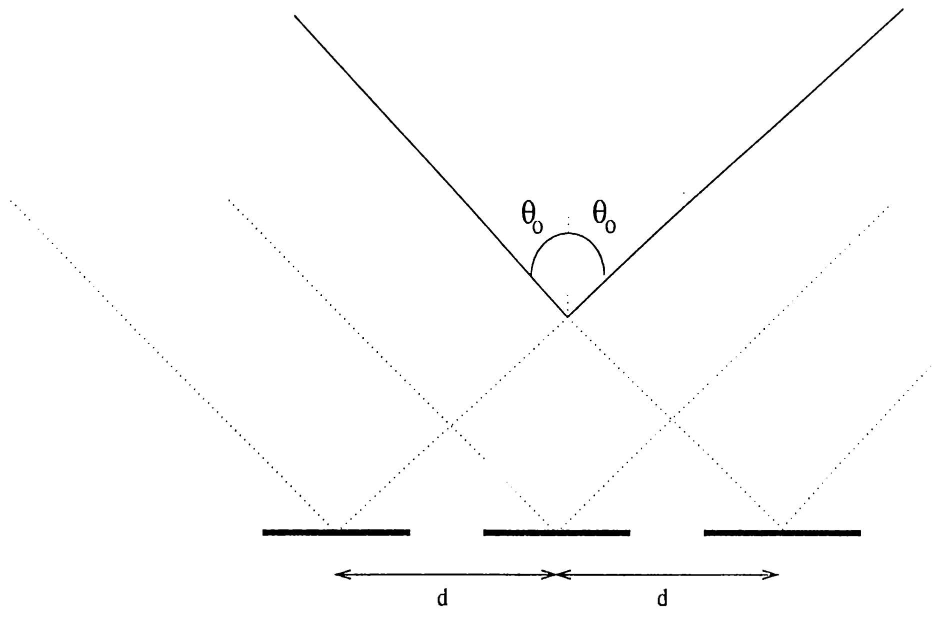


Figure 2.4: The beam patterns of the sensors (within dotted lines) and their sensitivity region (within solid lines).

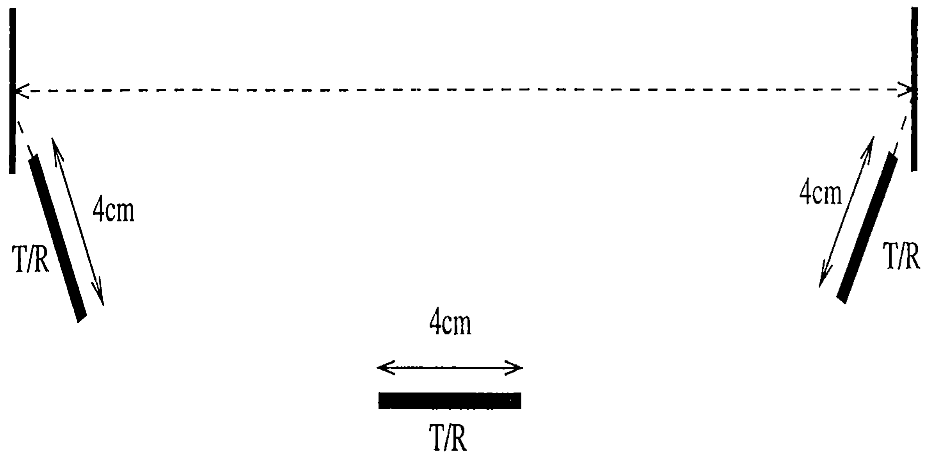


Figure 2.5: The tri-aural configuration where the peripheral sensors are rotated around their joints.

Chapter 3

RADIUS OF CURVATURE ESTIMATION

In this chapter, radius of curvature estimation of cylindrical objects is described. Two different ways of rotating the sensors are investigated. Two methods are used in order to estimate the time-of-flight. Sensitivity analysis for radius of curvature estimation is performed. Finally, the performances of the estimators are compared to the Cramér-Rao lower bound.

3.1 Sensors Rotated around their Joints

Figure 3.1 illustrates the object and the tri-aural configuration.

As seen from Equation 2.4, in order to model the sonar signal, the time-of-flight information is needed.

The following algorithm describes how the sensors can be rotated with respect to the target location in the simulations:

- Take the true values of h_{o1}, R, d and θ.

- The true distances between left transducer and the surface of the object (h_{l1}), and right transducer and the surface of the object (h_{r1}) are computed from the geometry:

$$\begin{aligned} h_{r1} &= \sqrt{r_{o1}^2 + d^2 - 2dr_{o1}\sin\theta} - R \\ h_{l1} &= \sqrt{r_{o1}^2 + d^2 + 2dr_{o1}\sin\theta} - R \end{aligned} \qquad (3.1)$$

- The distances of the three sensors to the center of the object and their inclination angles are calculated:

$$\begin{aligned} r_{l1} &= h_{l1} + R \\ r_{o1} &= h_{o1} + R \\ r_{r1} &= h_{r1} + R \end{aligned} \qquad (3.2)$$

$$\begin{aligned} \theta &= \sin^{-1}\left(\frac{r_{l1}^2 - r_{o1}^2 - d^2}{2dr_{o1}}\right) \\ \alpha &= \sin^{-1}\left(\frac{r_{l1}^2 - r_{o1}^2 + d^2}{2dr_{l1}}\right) \\ \eta &= \sin^{-1}\left(\frac{r_{r1}^2 - r_{o1}^2 + d^2}{2dr_{r1}}\right) \end{aligned} \qquad (3.3)$$

- The signal from each sensor is modeled by finding the time-of-flight values corresponding to h_{o1}, h_{l1} and h_{r1}.

- For the central sensor: $t_{oo} = \frac{2h_{o1}}{c}$, normal angle $= \theta$.
  For the left sensor: $t_{ol} = \frac{2h_{l1}}{c}$, normal angle $= \alpha$.
  For the right sensor: $t_{or} = \frac{2h_{r1}}{c}$, normal angle $= \eta$.

- The signals are modeled according to Equation 2.4 by using the time-of-flights obtained. The following model parameter values are used:

  A_{max} =1 cm
  r_{min} =10 cm
  $\rho_c = 0.44946R - 0.022471$
  $f_o = 50$ kHz
  $c = 34350$ cm/s

- White Gaussian noise $\mathcal{N}(0, \sigma)$ is added to the signal at every 1 μs and the noisy signal model is obtained.

The noiseless and noisy signal examples are shown in Figure 3.2(a) and (b) respectively.

- The time-of-flight can be estimated by choosing an appropriate threshold. In this case, the threshold is chosen as 5σ and the first time instant where the noisy signal exceeds the thresholding is detected. This is the noisy time-of-flight estimate. $h = \frac{ct_o}{2}$ is used to find the noisy h_{o1}, h_{l1} and h_{r1}.

- By using the noisy h_{o1}, h_{l1} and h_{r1}, the initial estimates of θ, α and η values are calculated, and the radius of curvature is estimated from the measurements:
$$R_1 = \frac{(h_{r1}^2 + h_{l1}^2) - 2(h_{o1}^2 + d^2)}{4h_{o1} - 2(h_{r1} + h_{l1})} \tag{3.4}$$

- The noisy distance of the left sensor is estimated and the rotation angle is calculated accordingly:
$$\begin{aligned} r_{l2} &= \sqrt{r_{l1}^2 + 6r_{l1}\sin\alpha} \\ a &= \frac{r_{l2}^2 - r_{l1}^2}{2r_{l2}} \qquad b = \frac{6 + 2r_{l1}\sin\alpha}{2r_{l2}} \\ z_l &= \frac{3 - ab}{b^2 - 1} - \frac{\sqrt{a^2 - 6ab + 9}}{b^2 - 1} \\ \phi &= \cos^{-1}\left(\frac{3}{z_l + 3}\right) \end{aligned} \tag{3.5}$$
The left sensor is rotated by ϕ.

- After the left sensor is rotated, the radius of curvature is estimated again:
$$\begin{aligned} c_1 &= \sqrt{(z_l + 3)^2 - 3^2} \\ h_{l2} &= r_{l2} - R_1 \\ c_2 &= h_{l2} - c_1 \\ c_3 &= d - z_l \\ R_{2l} &= \frac{c_2^2 - h_{o1}^2 + c_3^2 - 2c_2c_3\sin\phi}{2h_{o1} - 2c_2 + 2c_3\sin\phi} \end{aligned} \tag{3.6}$$

- The noisy distance of the right sensor is estimated and the rotation angle is calculated:
$$r_{r2} = \sqrt{r_{r1}^2 + 6r_{r1}\sin\eta}$$

$$
\begin{aligned}
a &= \frac{r_{r2}^2 - r_{r1}^2}{2r_{r2}} \qquad b = \frac{6 + 2r_{r1}\sin\alpha}{2r_{r2}} \\
z_r &= \frac{3 - ab}{b^2 - 1} - \frac{\sqrt{a^2 - 6ab + 9}}{b^2 - 1} \\
\beta &= \cos^{-1}\left(\frac{3}{z_r + 3}\right) \qquad (3.7)
\end{aligned}
$$

- After rotating the right sensor by β, the radius of curvature is estimated again:

$$
\begin{aligned}
c_4 &= \sqrt{(z_r + 3)^2 - 3^2} \\
h_{r2} &= r_{r2} - R_1 \\
c_5 &= h_{r2} - c_4 \\
c_6 &= d - z_r \\
R_{2r} &= \frac{c_5^2 - h_{o1}^2 + c_6^2 - 2c_5c_6\sin\beta}{2h_{o1} - 2c_5 + 2c_6\sin\beta} \qquad (3.8)
\end{aligned}
$$

- The average of the radius values calculated by rotating the left and right sensors are calculated:

$$R_2 = \frac{R_{2l} + R_{2r}}{2} \qquad (3.9)$$

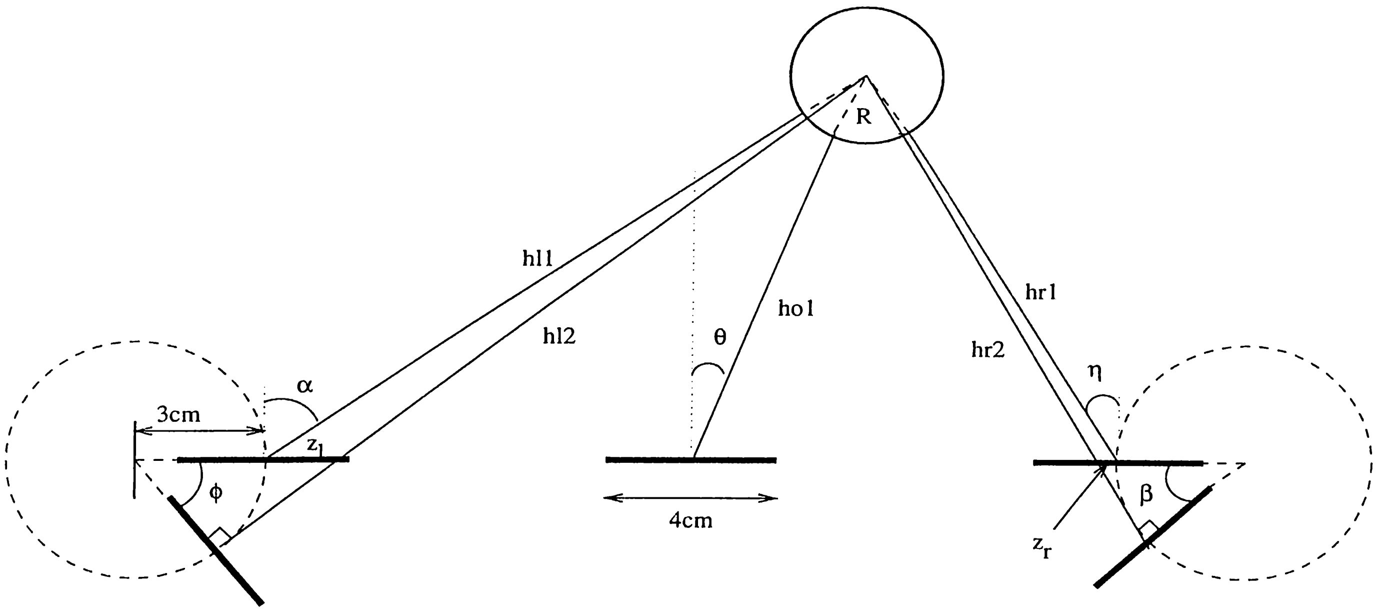


Figure 3.1: The object and the tri-aural configuration.

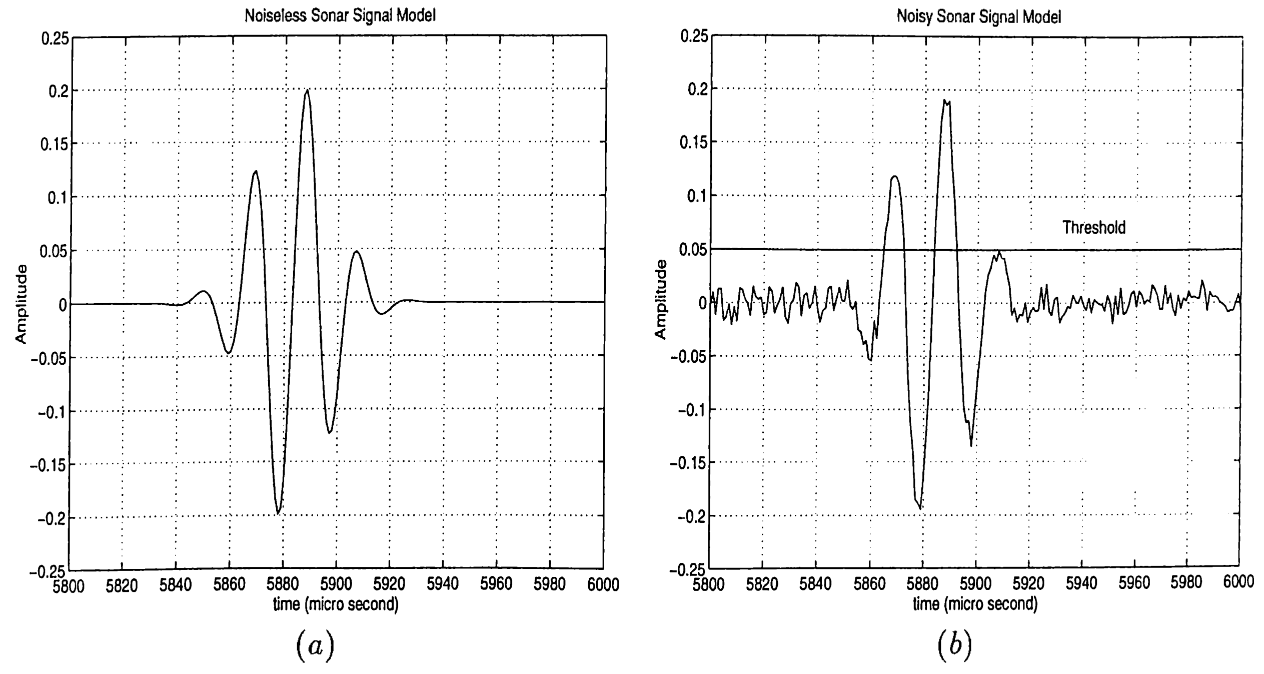


Figure 3.2: Noiseless and noisy signal models.

3.1.1 Sensitivity Analysis of the Radius of Curvature Estimate

In this section, the sensitivity analysis of the radius of curvature estimation to parameters such as h_{o1}, h_{l1}, h_{r1}, d, R and θ is presented. The sensitivity analysis can be summarized by the following steps:

- Function $f(.)$ for the radius of curvature estimate is defined as follows:

$$f(h_{o1}, h_{l1}, h_{r1}, d) = R_1 = \frac{(h_{r1}^2 + h_{l1}^2) - 2(h_{o1}^2 + d^2)}{4h_{o1} - 2(h_{r1} + h_{l1})} \tag{3.10}$$

- Perturbation is added to the variable for which sensitivity analysis is made. For example, the perturbation Δh_{o1} is added to h_{o1}.

- The perturbation on radius of curvature is calculated as follows:

$$f(h_{o1}+\Delta h_{o1}, h_{l1}, h_{r1}, d) = R_1+\Delta R = \frac{(h_{r1}^2 + h_{l1}^2) - 2[(h_{o1} + \Delta h_{o1})^2 + d^2]}{4(h_{o1} + \Delta h_{o1}) - 2(h_{r1} + h_{l1})} \tag{3.11}$$

- The effect of the perturbation on the radius of curvature estimate is found:

$$\Delta R = f(h_{o1} + \Delta h_{o1}, h_{l1}, h_{r1}, d) - f(h_{o1}, h_{l1}, h_{r1}, d) \tag{3.12}$$

Figure 3.3(a) is plotted for h_{o1} between 0–150 cm. The error in h_{o1} is positive and between 0–0.4 mm. A stationary cylindrical target at $\theta = 0°$ with radius 5 cm is used and transducer separation is assumed to be 10 cm. The error ΔR in R increases linearly with the error Δh_{o1} and nonlinearly with h_{o1}. Also a positive error in h_{o1} leads to a positive error in R since for constant h_{r1} and h_{l1}, increasing h_{o1} means increasing R, as the geometry of Figure 3.1 indicates. As range increases, the error also increases since the sensor has lower resolution for fixed d.

In Figure 3.3(b), the same parameters are used as in Figure 3.3(a) with a positive error in h_{r1} instead of h_{o1}. Since the terms h_{r1} and h_{l1} are used symmetrically in the radius of curvature equation, the sensitivity analysis of h_{l1} gives the same results. A positive error in h_{r1} or h_{l1} causes a negative error in R since a positive error on left and right measurements for constant central sensor measurement leads to a reduction in R as in Figure 3.1.

Figure 3.4(a) and 3.5(a) show the effect of d on the radius of curvature estimation. $\Delta h_{o1} = \Delta h_{r1} = 0.18$ mm errors are taken for Figure 3.4(a) and 3.5(a) respectively. Range is again varied between 0–150 cm. The error is plotted for d between 4–40 cm. As seen from the figures, for small d/h_{o1}, the error is high since the resolution of the left and right sensors decrease as shown in Figure 2.4. Hence, as range increases the transducer separation should also be increased to achieve higher resolution. Also for high values of d/h_{o1}, the error is large since the sensitivity patterns of the transducers will not overlap at the target [30].

Figures 3.4(b),(c) and 3.5(b),(c) illustrate the sensitivity of radius of curvature to measurement errors $\Delta h_{o1} = 0.18$ mm and $\Delta h_{r1} = 0.18$ mm, respectively. In Figures 3.4(c) and 3.5(c), θ is varied from $0°$ to $20°$ with $1°$ increments.

3.1.2 Simulation Results

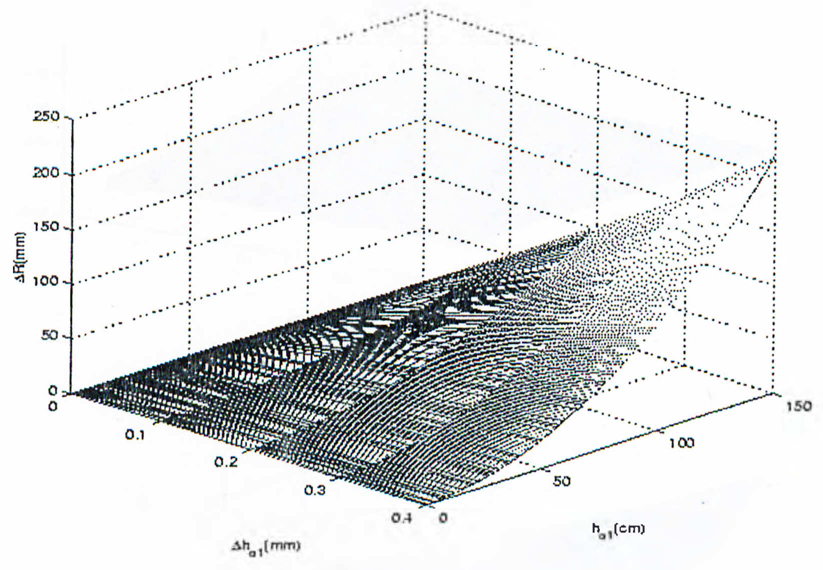
Now, the simulation results can be investigated by using the results of the sensitivity analysis.

Figure 3.6–3.9 show the simulation results for $\theta = 0°$.

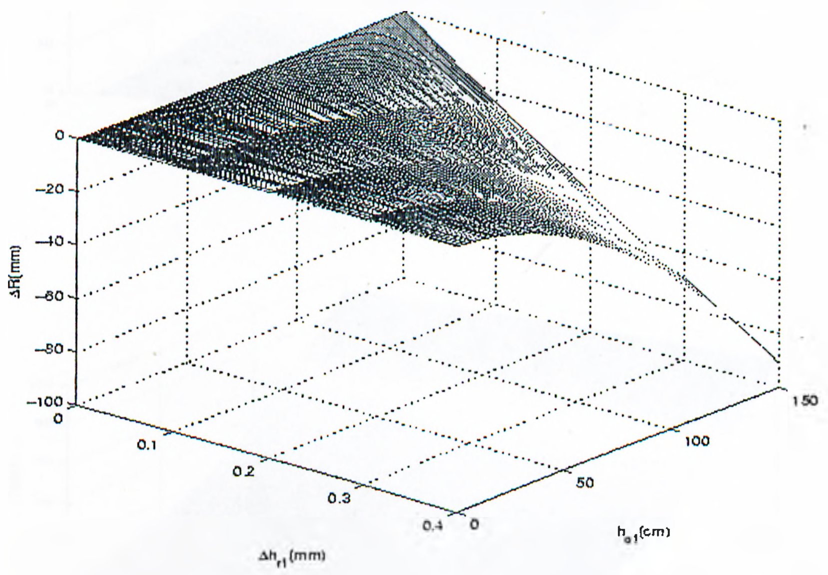
Figure 3.6 and 3.7 show how the estimated radius values R_1 and R_2 are affected from different variables in the absence of noise. Figure 3.8 and 3.9 show the same calculations obtained with the Monte Carlo simulation study.

The sensitivity analysis of the calculated radius of Equation 3.4 to $h_{o1}, h_{l1}, h_{r1}, d$ and R shows that the radius is very sensitive to these variables. A positive error in h_{o1} ($\Delta h_{o1} > 0$) causes the estimated radius to be greater than the true radius (according to Equation 3.4). A positive error in h_{l1} or h_{r1} ($\Delta h_{l1} > 0$) causes the estimated radius to be less than the true radius. Moreover, the error corresponding to the separation between the sensors is positive and it decreases as the separation increases. As the distance between the sensors and the object increases, the error increases in the negative direction.

The results are completely consistent with the sensitivity analysis predictions. Although the noise is zero in some cases, the estimated radius deviates from the true radius. The error sources are the bias error due to thresholding and the error due to sampling the signal.

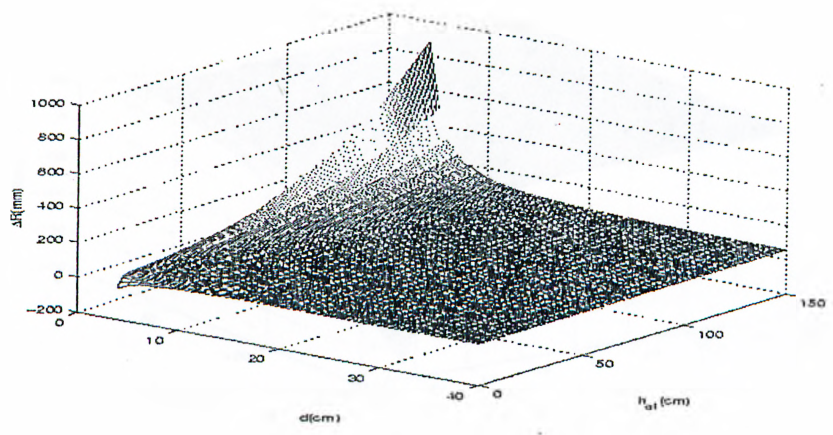


(a)

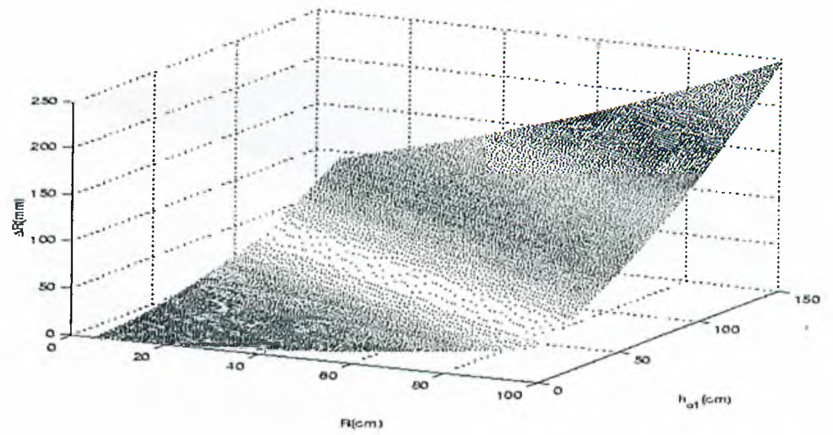


(b)

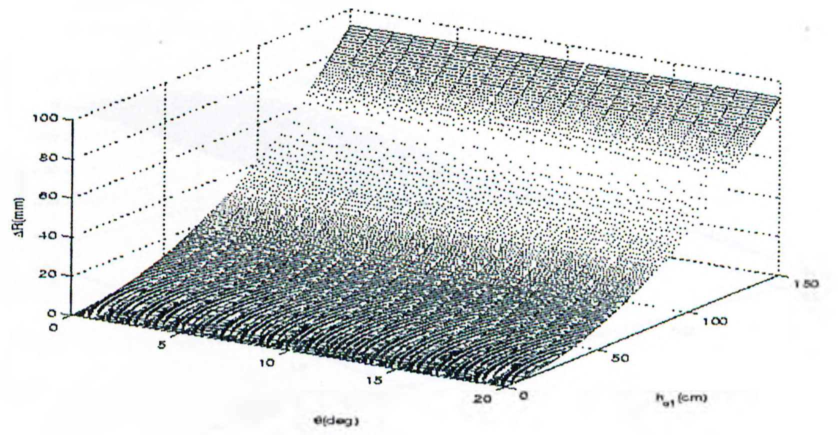
Figure 3.3: Sensitivity of R to distance measurements (a) h_{o1} (b) h_{r1} or h_{l1} for $d = 10$ cm, $R = 5$ cm and $\theta = 0°$.



(a)

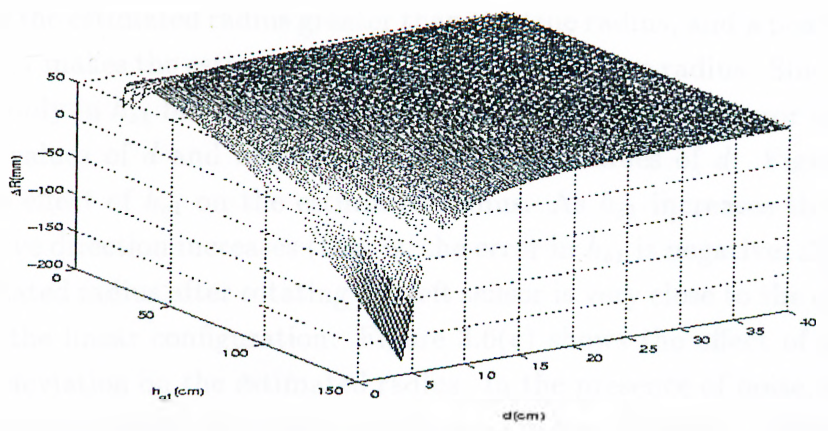


(b)

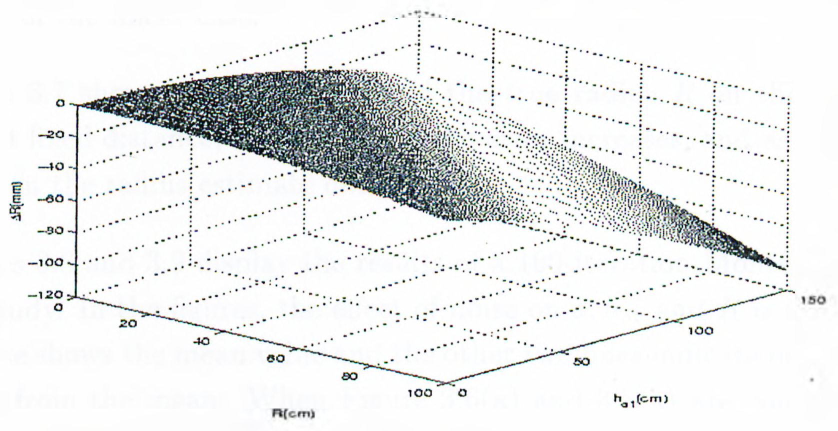


(c)

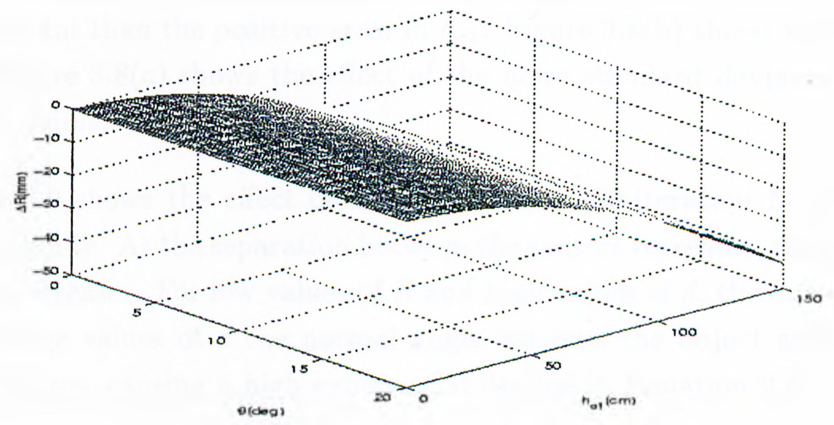
Figure 3.4: Sensitivity of R to (a) d, (b) R and (c) θ when $\Delta h_{o1} = 0.18$ mm.



(a)



(b)



(c)

Figure 3.5: Sensitivity of R to (a) d, (b) R and (c) θ when $\Delta h_{r1} = 0.18$ mm.

Figure 3.6(a) shows the relation of the estimated radius and the separation distance between the transducers. The error in the estimated radius values decreases as the separation increases. As stated previously, a positive error in h_{o1} makes the estimated radius greater than the true radius, and a positive error in h_{l1} or h_{r1} makes the estimated radius less than the true radius. Since there is error not only in h_{o1} but also in h_{l1} and h_{r1} in this estimation, error is positive for some values of d and negative for some other values of d. Figure 3.6(b) shows the effect of h_{o1} on the estimated radius. As h_{o1} increases the error in the negative direction increases (that is, the error in h_{o1} is negative, $\Delta h_{o1} < 0$). The estimated radius after rotating the left sensor is very close to the estimated radius of the linear configuration. Figure 3.6(c) shows the effect of the noise standard deviation on the estimated radius. In the presence of noise, the error after rotation is nearly the same as the error before rotation. However, for some values of the noise standard deviation ($2 \mathrm{x} 10^{-5}$ to $3 \mathrm{x} 10^{-5}$ V), it is greater than that of the linear case.

Figure 3.7 shows the effect of d and the true radius R on the estimated radius. At fixed distance, as R increases the error increases, and as d increases the error on the radius estimate decreases.

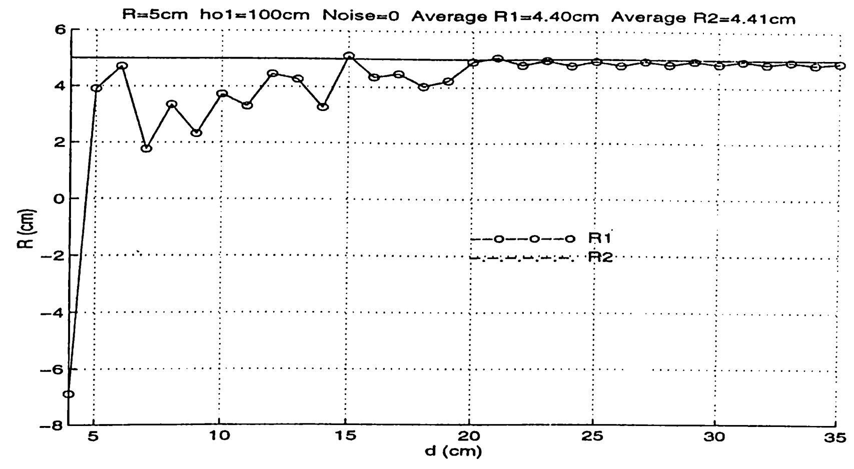
Figures 3.8 and 3.9 display the results of a 100-iteration Monte Carlo simulation study. In the figures, the effect of noise on d, h_{o1} and R is shown. The dashed line shows the mean value and the other two lines indicate one standard deviation from the mean. When Figure 3.6(a) and 3.8(a) are compared, the effect of noise on d can be observed. The estimated radius is less than the true radius for all values of d. This is because the negative error in h_{l1} and h_{r1} is more dominant than the positive error in h_{o1}. Figure 3.8(b) shows the effect of h_{o1} and Figure 3.8(c) shows the effect of the noise standard deviation on the estimated radius.

Figure 3.9 shows the effect of d and R over a 100-iteration Monte Carlo simulation study. As the separation between the sensors increases, the standard deviation decreases. For low values of R and high values of d, the error is large since for large values of d the normal angle between the object and the left sensor increases, causing a high exponential decline in Equation 2.4.

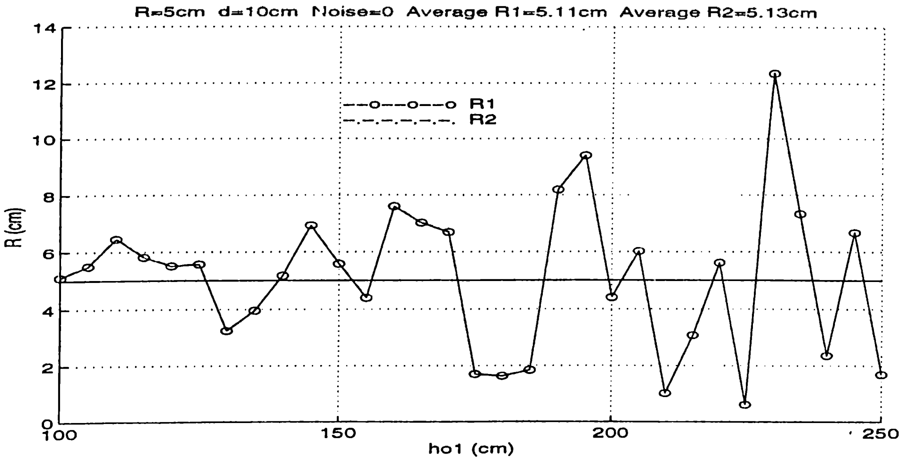
Figures 3.10 and 3.11 illustrate the results of $\theta \neq 0°$ case in the absence of noise. In Figure 3.10(a), the dependence of the estimated radius to the separation distance between the transducers is illustrated. As d increases, the error in the estimation of the radius of curvature corresponding to the linear configuration of the sensors decreases. Although the error in estimation after rotating the sensors is larger than that of the linear configuration case, it decreases with d. After a certain value of d (around $d = 27$ cm), the error for both cases begins increasing since at that point, the target is not within the beam pattern of the left or right sensor. Figure 3.10(b) shows the effect of h_{o1} on the estimation of the radius of curvature. As h_{o1} increases the error in both linear and rotated configurations increases. Figure 3.10(c) shows the effect of θ on the estimated radius. As θ increases, the error increases as expected, since a sonar sensor can make more accurate measurements along its line-of-sight. This is because the echo amplitude decreases exponentially with the square of θ according to Equation 2.4.

Figure 3.11 shows the effect of d and true radius R on the curvature estimation. As d increases, the estimation error corresponding to the linear configuration decreases according to the sensitivity analysis but the error for the rotated configuration increases. This is because as d increases, the object goes out of the sensitivity region and the rotation angle calculated by using the linear configuration is not correct.

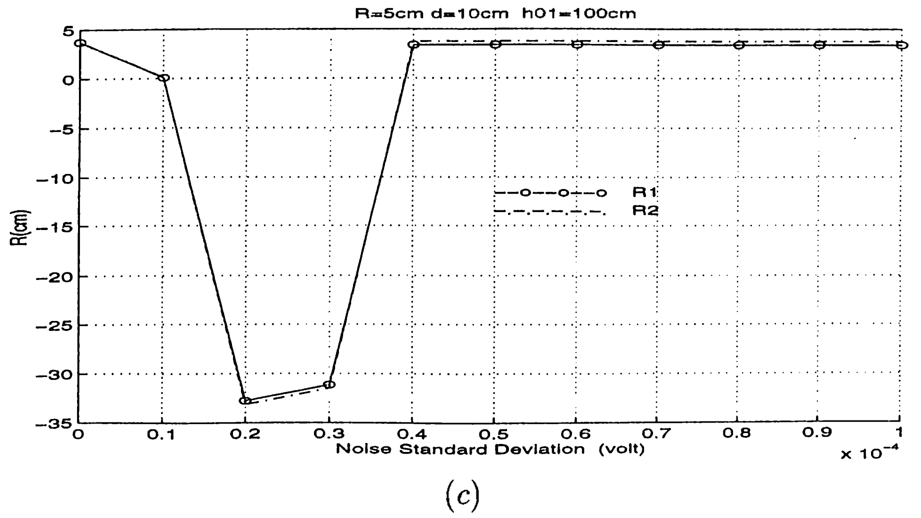
Figures 3.12 and 3.13 show the effect of noise on the above observations. In these figures, a 100-iteration Monte Carlo study was employed. Figure 3.12(a) shows the effect of d. In both cases, as d increases, initially the error decreases, and after a certain value of d (around $d = 16$ cm) the error increases substantially. Figure 3.12(b) shows the effect of h_{o1} and Figure 3.12(c) shows the effect of θ. When Figures 3.10 and 3.12 are compared, the effect of noise can be observed. Figure 3.13 shows the effect of d and R in the presence of noise. As d increases the estimates are degraded.



(a)

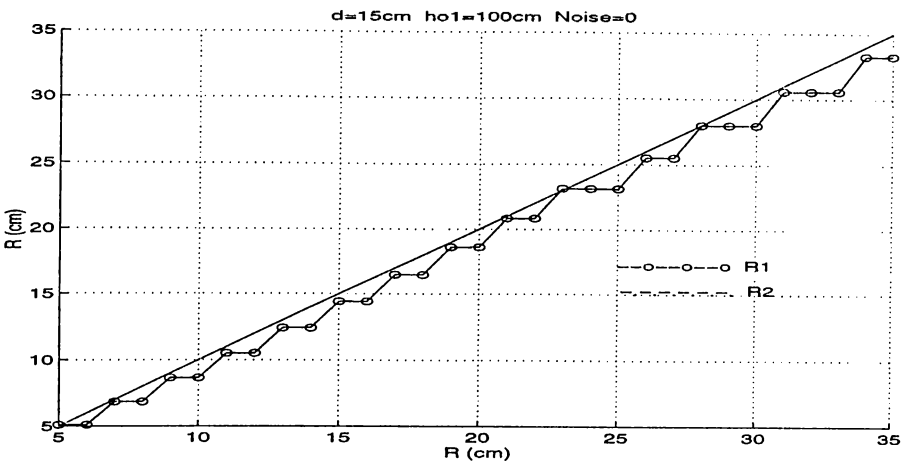


(b)

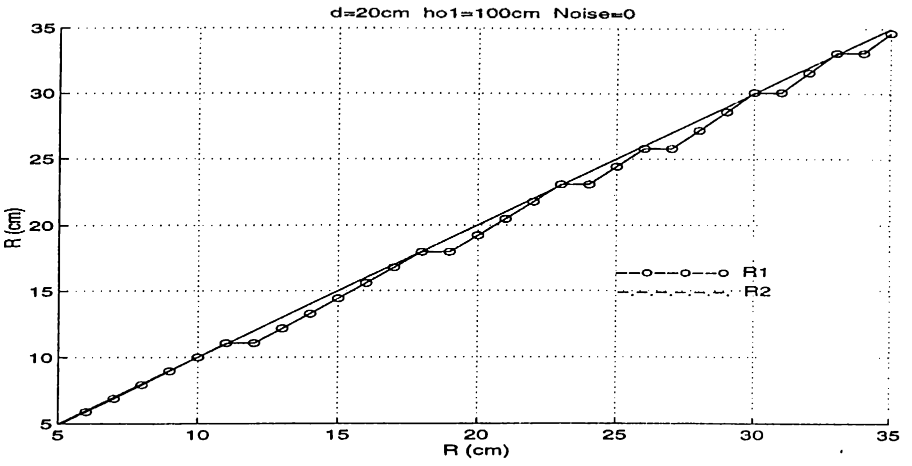


(c)

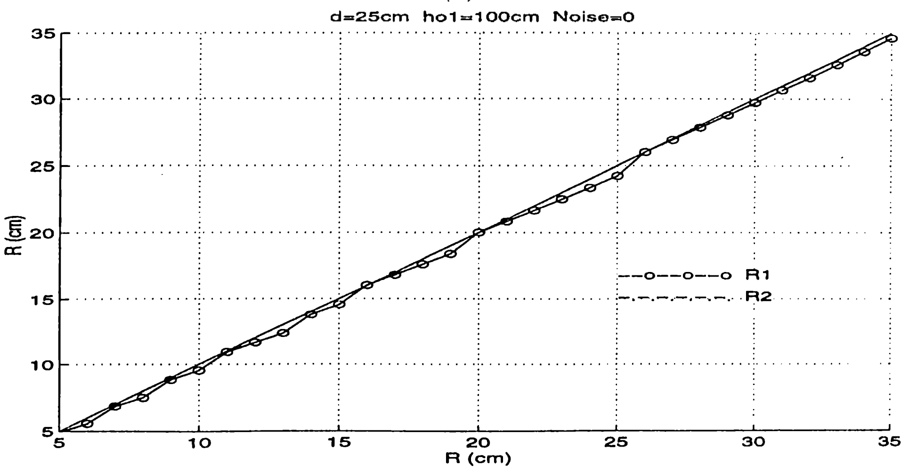
Figure 3.6: Estimated radius versus d, h_{o1} and noise standard deviation.



(*a*)

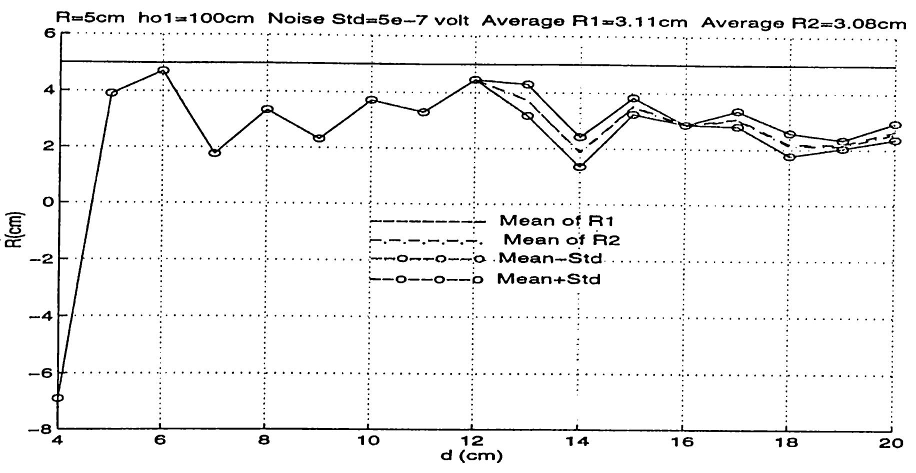


(*b*)

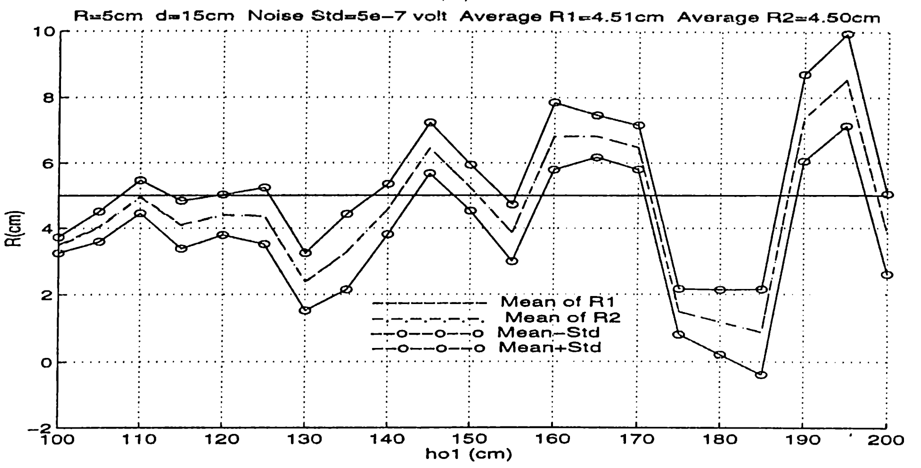


(*c*)

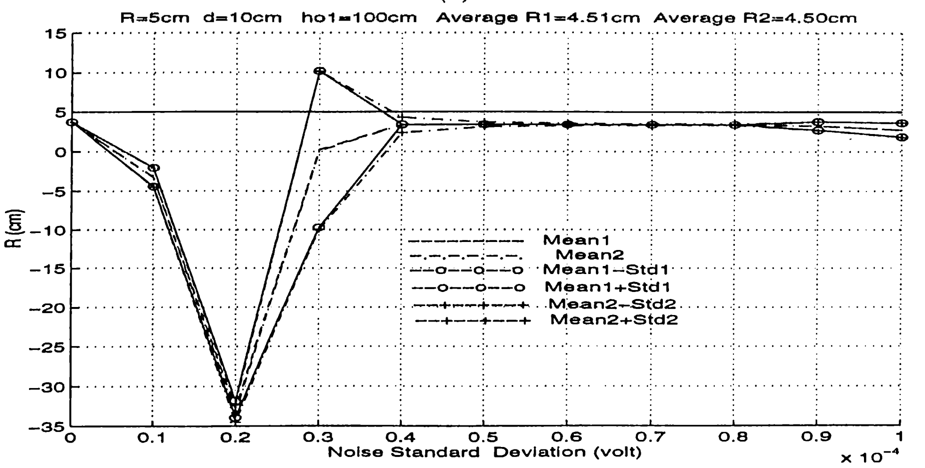
Figure 3.7: Estimated versus true radius for $d = 15$ cm, $d = 20$ cm, and $d = 25$ cm.



(a)

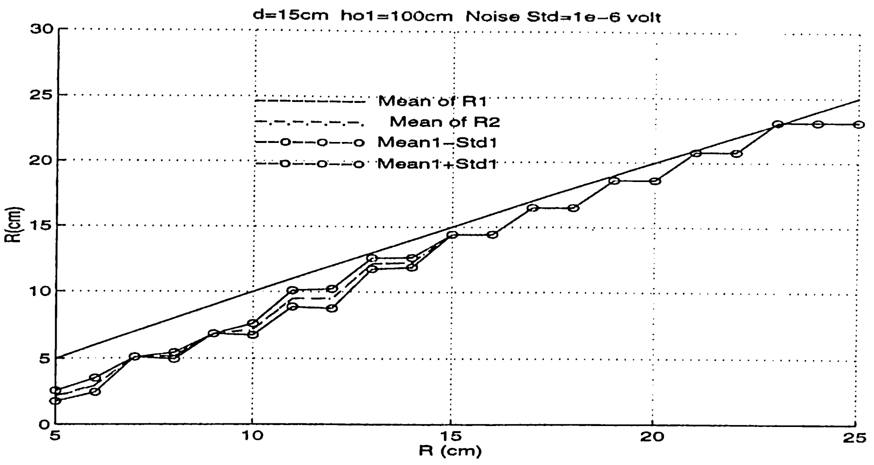


(b)

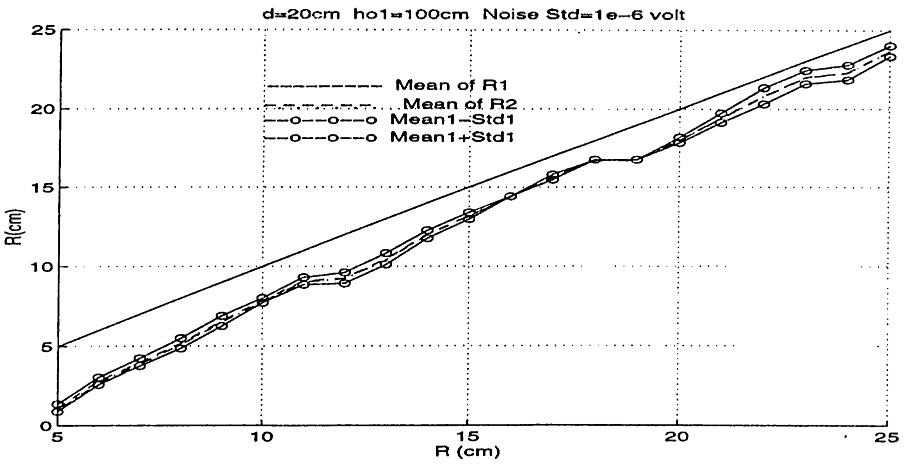


(c)

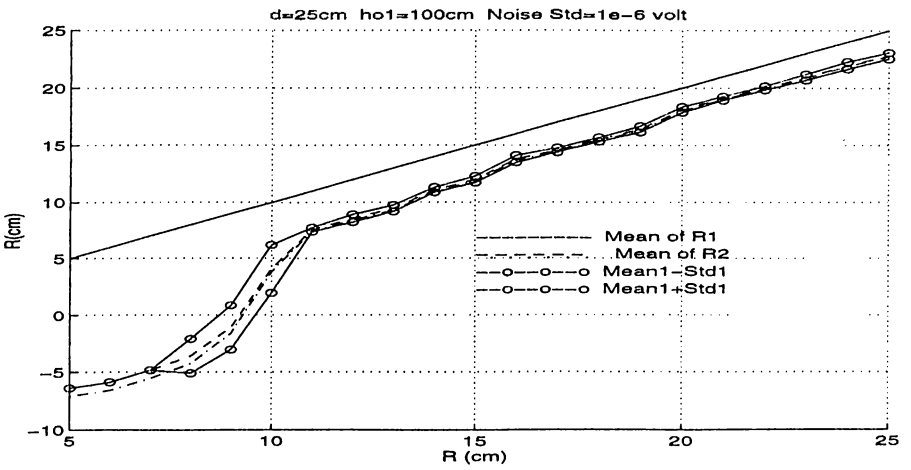
Figure 3.8: Estimated radius versus d, h_{o1} and noise standard deviation using a 100-iteration Monte Carlo simulation study.



(a)

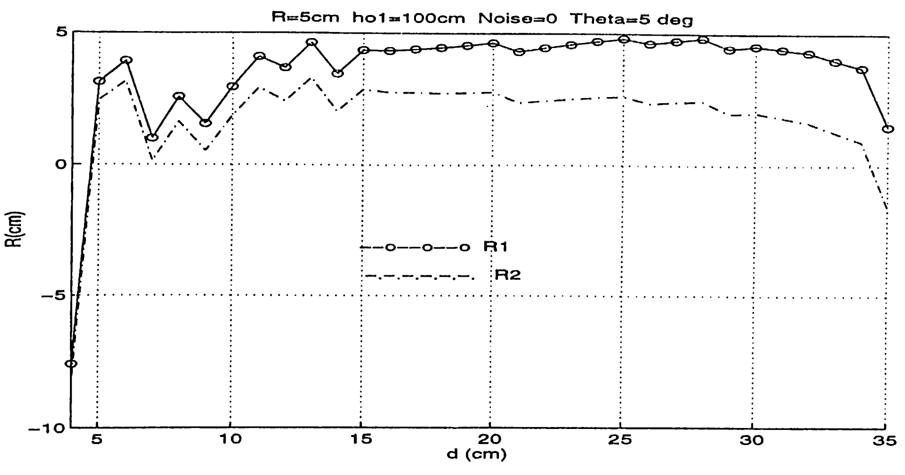


(b)

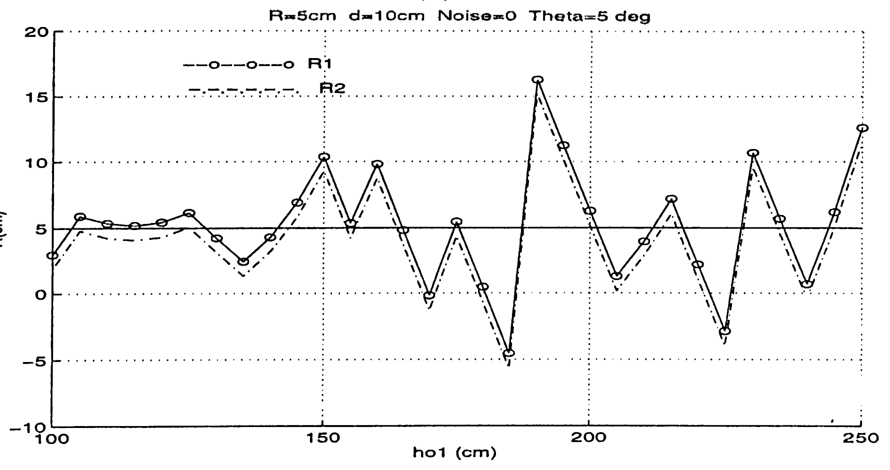


(c)

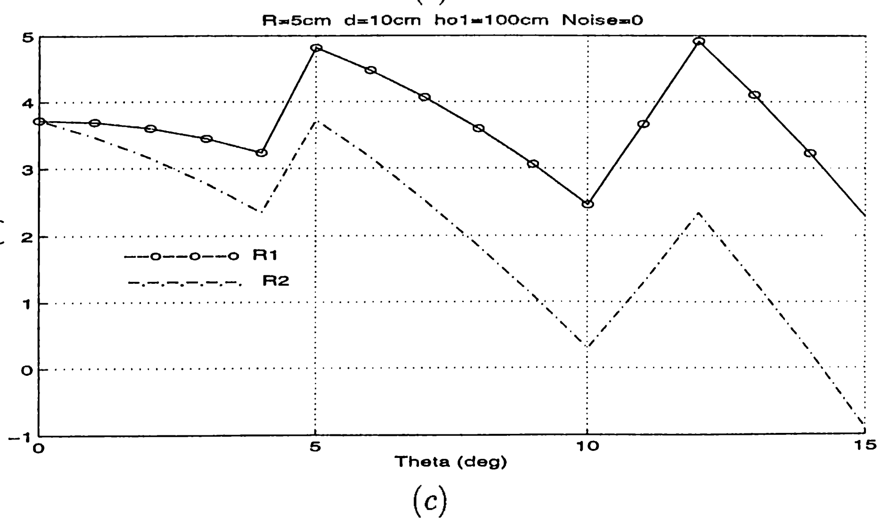
Figure 3.9: Estimated versus true radius for $d = 15$ cm $d = 20$ cm and $d = 25$ cm using a 100-iteration Monte Carlo simulation study.



(a)

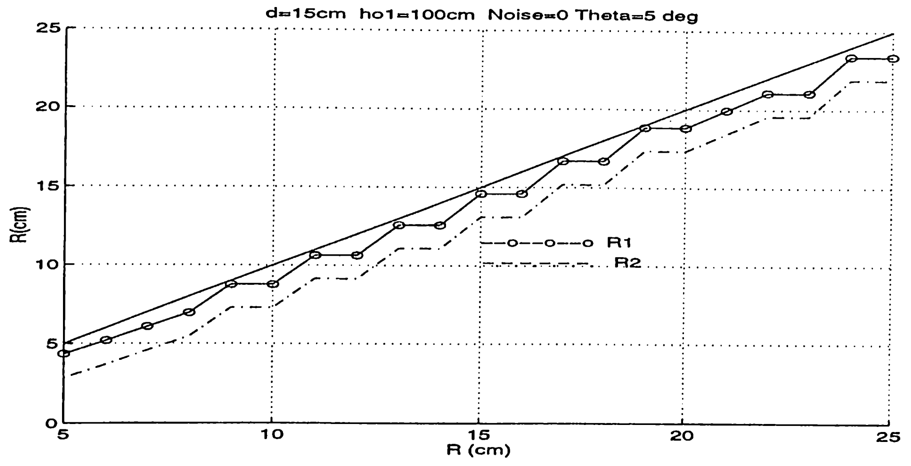


(b)

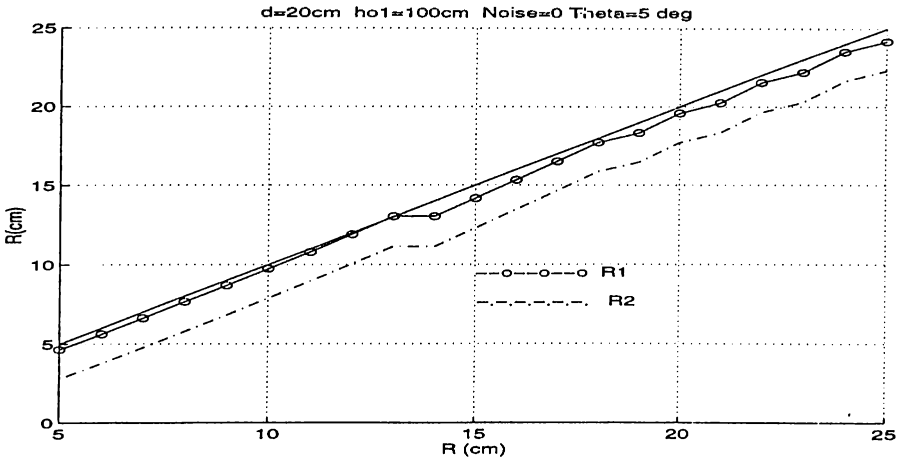


(c)

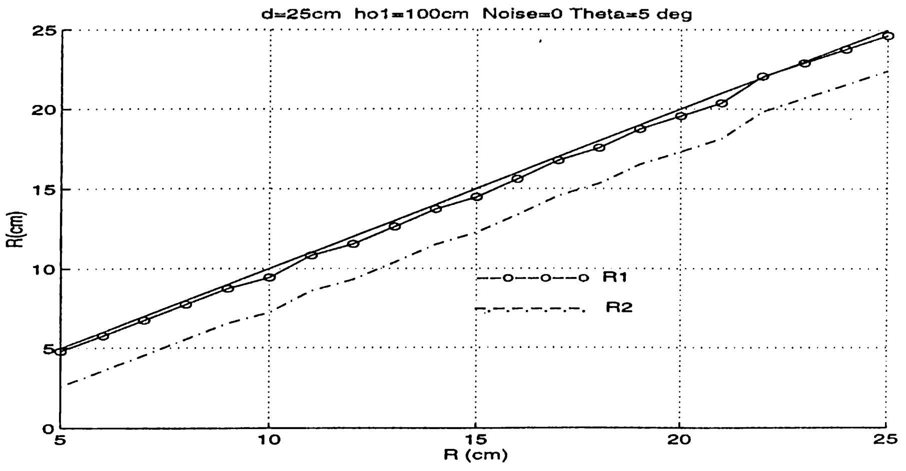
Figure 3.10: Estimated radius versus d, h_{o1} and θ in the absence of noise.



(a)

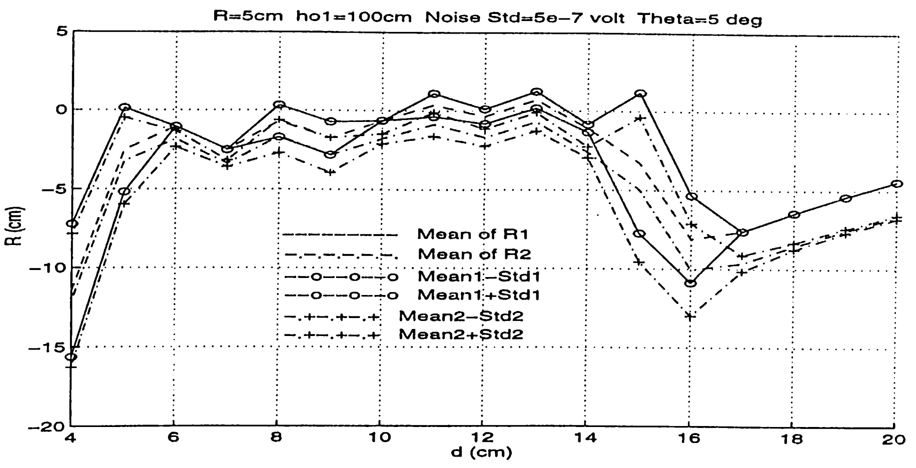


(b)

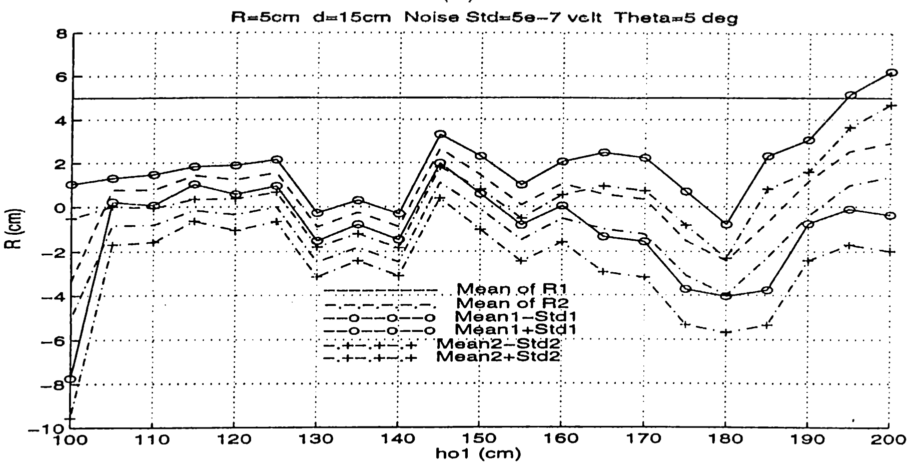


(c)

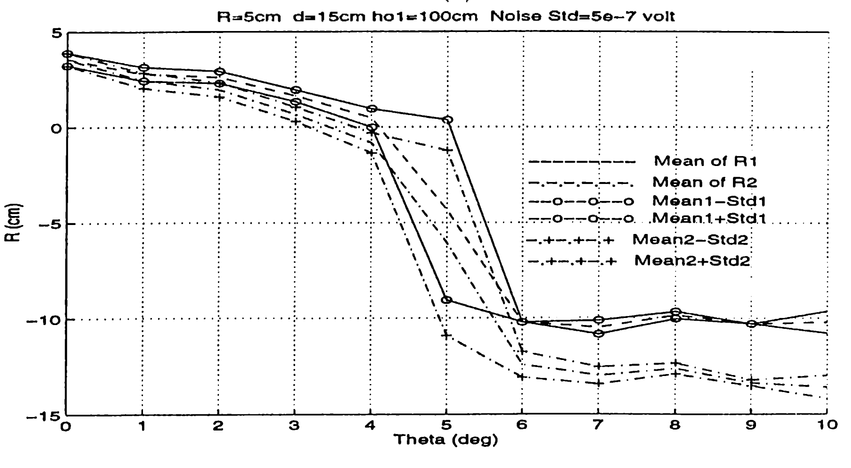
Figure 3.11: Estimated versus true radius for $d = 15$ cm $d = 20$ cm and $d = 25$ cm in the absence of noise.



(*a*)

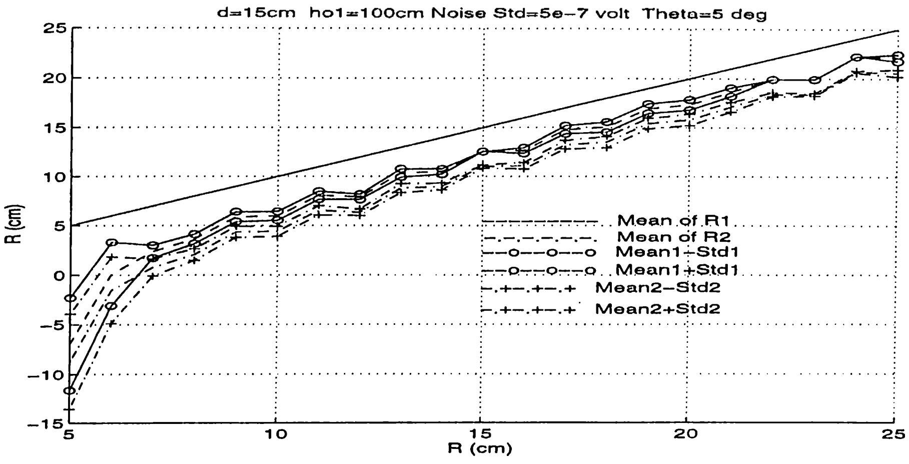


(*b*)

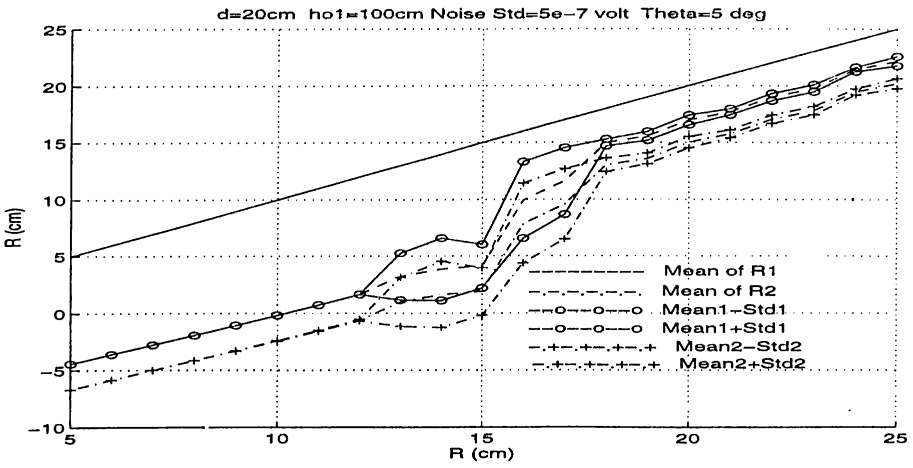


(*c*)

Figure 3.12: Estimated radius versus d, h_{o1} and θ in the presence of noise using a 100-iteration Monte Carlo simulation study.



(a)



(b)

Figure 3.13: Estimated versus true radius for d = 15 cm d = 20 cm in the presence of noise using a 100-iteration Monte Carlo simulation study.

3.2 Sensors Rotated around their Centers

In this part, we consider the case in which the left and right sensors are assumed to rotate around their centers as shown in Figure 3.14.

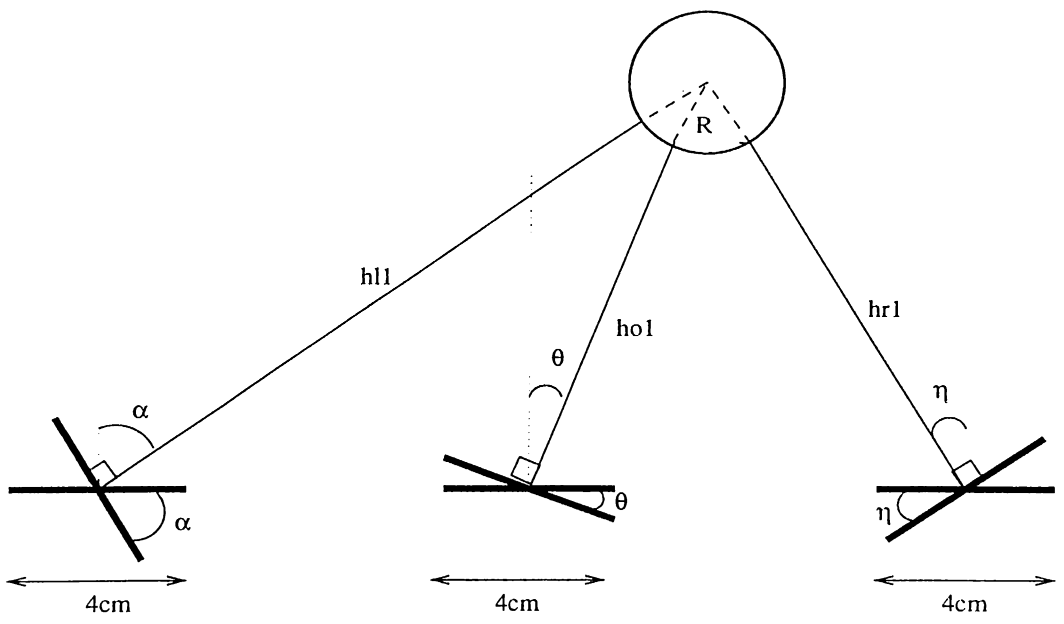


Figure 3.14: The object and the sensor configuration when the sensors are rotated around their centers.

The algorithm to find the rotation angle can be summarized as follows:

- The initial estimate of the radius of curvature (when the sensors are aligned) is the same as in the previous algorithm.
- From the first set of measurements, noisy θ, α and η are calculated:

$$\begin{aligned}
\theta &= \sin^{-1}\left(\frac{r_{l1}^2 - r_{o1}^2 - d^2}{2dr_{o1}}\right) \\
\alpha &= \sin^{-1}\left(\frac{r_{l1}^2 - r_{o1}^2 + d^2}{2dr_{l1}}\right) \\
\eta &= \sin^{-1}\left(\frac{r_{r1}^2 - r_{o1}^2 + d^2}{2dr_{r1}}\right) \qquad (3.13)
\end{aligned}$$

- After finding the rotation angles, θ, α and η, the central, right and left transducers are rotated by θ, α and η respectively. When the sensors are rotated, they are assumed to be perpendicular to the center of the object.

- The nominal values of the distances after rotation should be:

$$\begin{aligned} h_{o2} &= h_{o1} \\ h_{l2} &= h_{l1} \\ h_{r2} &= h_{r1} \end{aligned} \tag{3.14}$$

- New measurements are taken and the second estimation of the radius of curvature is made by using Equation 3.4.

3.2.1 Simulation Results

Figure 3.15 shows the effects of d, h_{o1} and noise standard deviation on the estimation of the radius of curvature when $\theta = 0°$. It can be seen that for all values of d, h_{o1} and noise standard deviation, the second estimation (after rotation) provides better results. Figure 3.15(a) shows that the deviation between the two estimates begin after a certain point in d (around $d = 37$ cm). Up to that point, the first and the second estimates are nearly the same since for low values of d, the normal angle of the left sensor is small and the object is nearly perpendicular to the left sensor. However, as d increases the normal angle of the left sensor also increases and the estimation after rotation provides improved results. Figure 3.15(b) shows that as h_{o1} increases, the first and second estimates become closer. This is because when h_{o1} is small, the deviation angle of the left sensor is large and the estimation after rotation gives improved results. Figure 3.15(c) shows the effect of the noise standard deviation. The second estimation is less affected from noise than the first estimation.

Figure 3.16 shows the effect of d and R simultaneously. As d increases both estimators improve and for the low values of R the second estimation is more accurate for the same reason as above. For lower values of R ($R \leq 10$ cm) and high values of d, the estimated radius after rotation is better since for high values of d, exponential decline in Equation 2.4 is very high.

Figures 3.17 and 3.18 show the same calculations done by using a 100-iteration Monte Carlo simulation study. Figure 3.17 shows the effect of d and h_{o1} on the estimated radius in the presence of noise. Figure 3.17(a) indicates that as d increases, the second estimation provides better results which is consistent with the above argument. When d is small, the object can be considered perpendicular to the sensors, hence we do not expect much improvement with the second estimation. Since there is error in the rotation angle, the estimation after rotation must be worse than that of the linear configuration.

Figure 3.17(b) shows that as h_{o1} increases the first estimate improves since the object becomes perpendicular to the sensors.

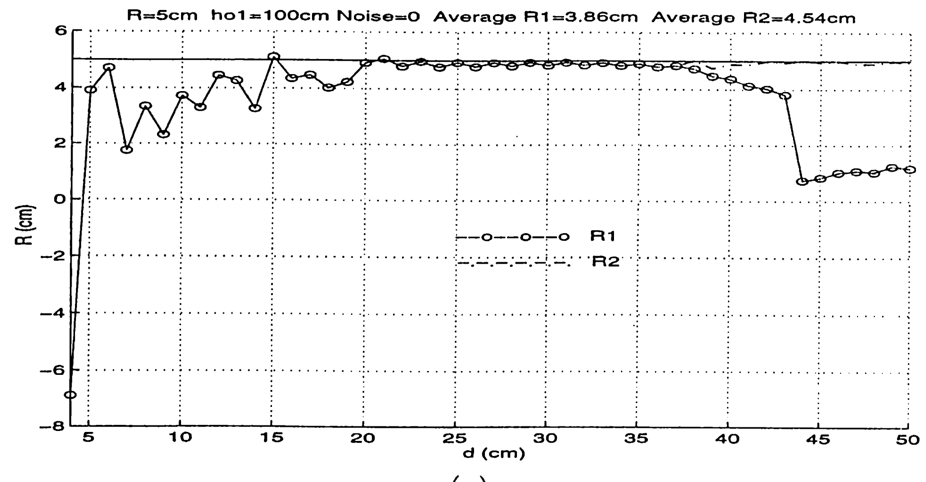
Figure 3.18 shows the effect of d and R simultaneously on the radius of curvature estimation. As d increases, the second estimation improves as expected. For low values of R, as d increases the error in the first estimation also increases since for high values of d and low values of R, the normal angle of the left and right sensors are high and the exponential decline in Equation 2.4 is high.

3.3 Curve-Fitting Method for Time-of-Flight Estimation

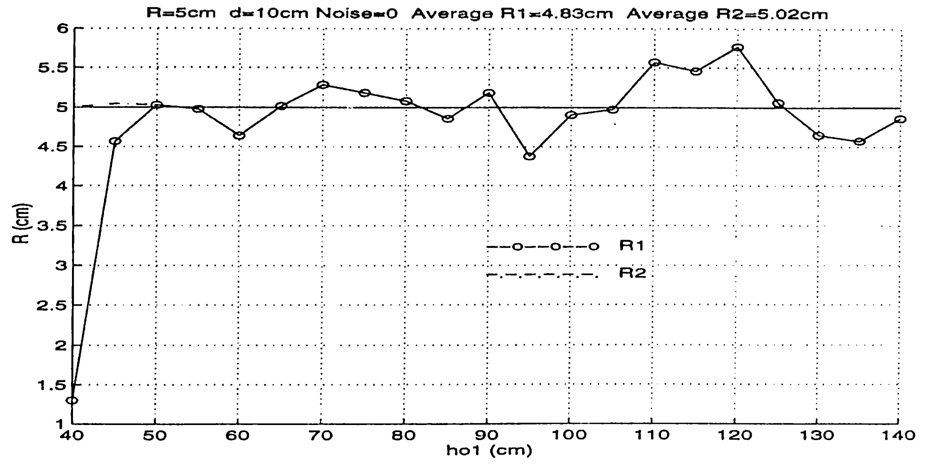
A curve-fitting approach is used in order to reduce the error in the time-of-flight estimations obtained from the thresholding method and improve the estimation procedure. Earlier, in [24], a similar method was used to improve the accuracy of point-target localization. It was shown that this method of time-of-flight estimation eliminated the bias resulted from thresholding and was comparable to thresholding in variance. Here, we generalize the method to include radius of curvature estimation. Figure 3.19 shows the parabola fitted noisy sonar signal model. It is expected to reduce the bias with this method.

The following algorithm is used when estimating the time-of-flight with curve fitting:

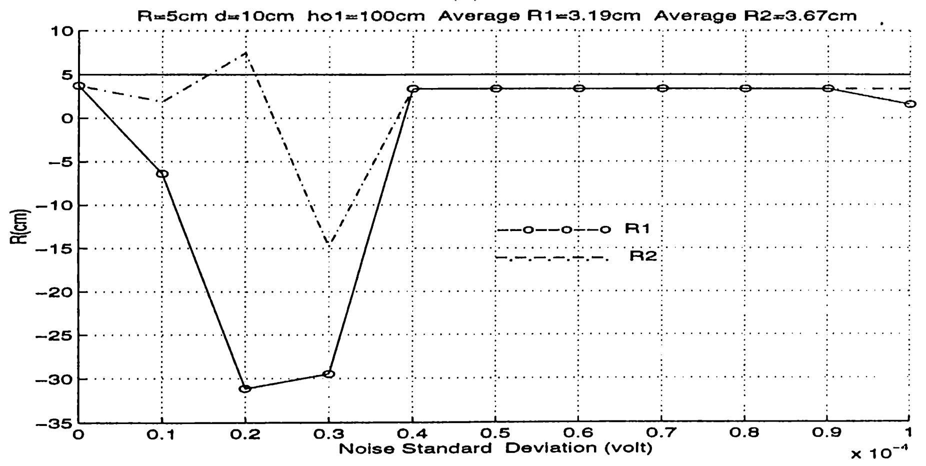
- An approximate time-of-flight estimate is obtained using the thresholding method.
- The amplitude of the signal is sampled at three time instants before the time-of-flight obtained by the thresholding method. The time interval between the time instants equals 3 μs.
- The zero-crossing parabola passing through these points is found. Parabola is assumed to be of the form $y = at^2 + bt + c$, and the coefficients a, b and c are calculated by using the three samples. If the parabola found by using the above samples does not take the value zero at any time, then new samples are taken and another parabola is obtained. This procedure continues until a zero-crossing parabola is found.



(a)

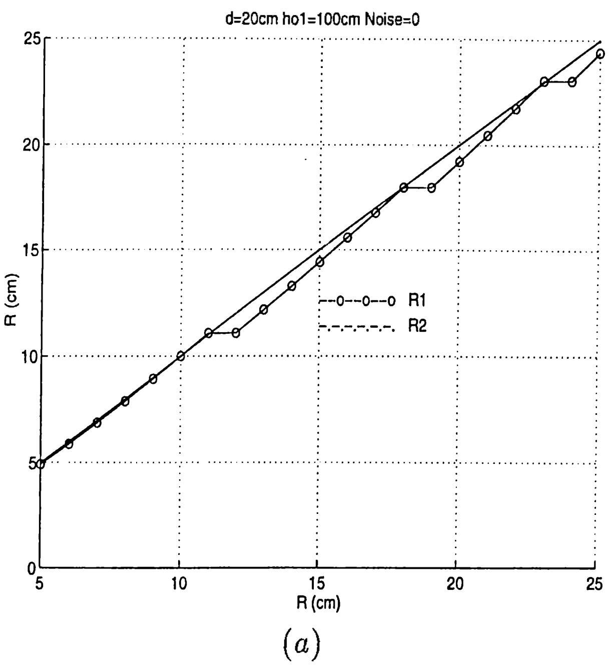


(b)



(c)

Figure 3.15: Estimated radius versus d, h_{o1} and noise standard deviation (when sensors are rotated around their centers).



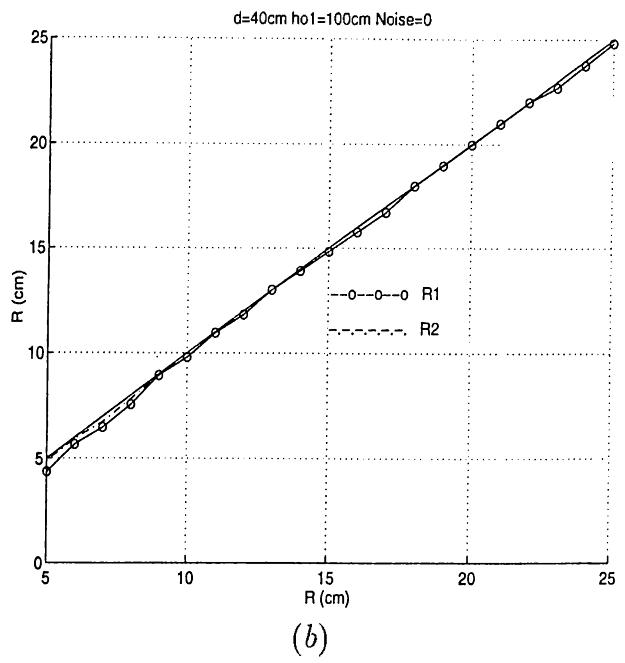
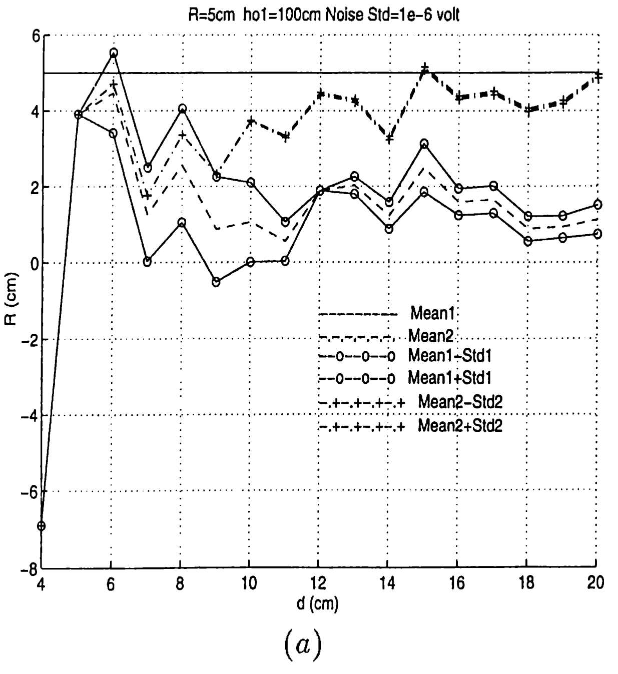


Figure 3.16: Estimated radius versus true radius when $d = 20$ cm and $d = 40$ cm (when sensors are rotated around their centers).



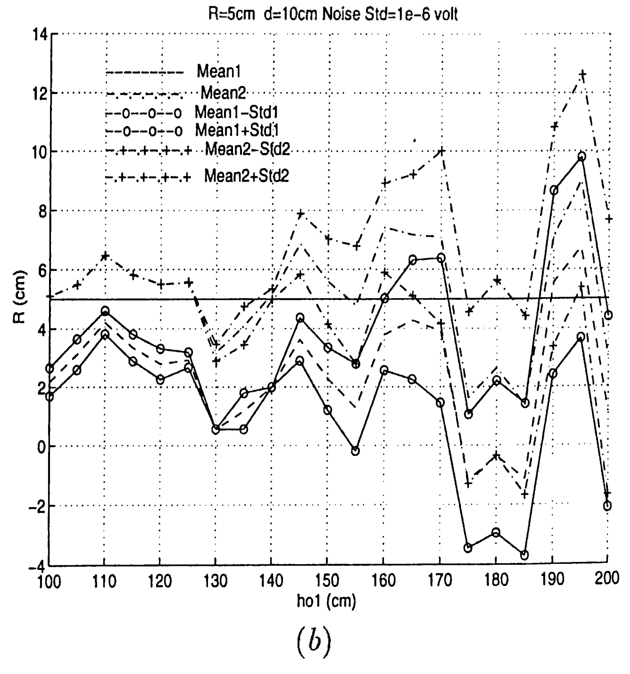
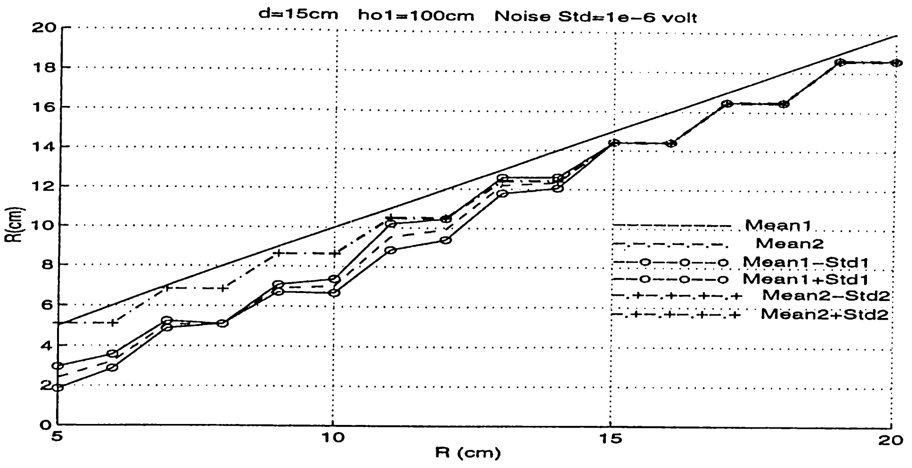
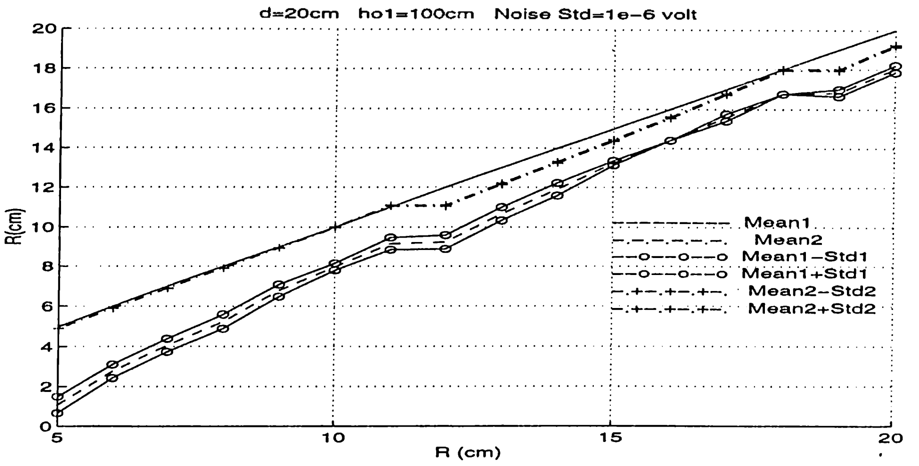


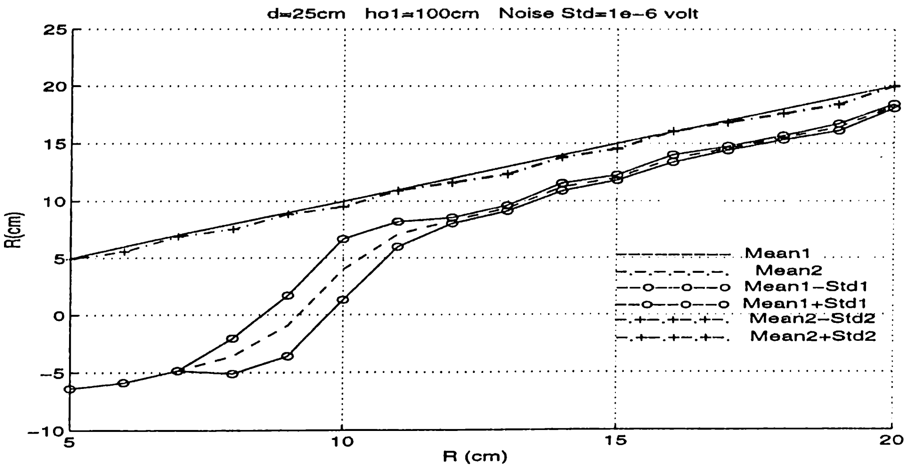
Figure 3.17: Estimated radius versus d, h_{o1} by using a 100-iteration Monte Carlo simulation study.



(a)



(b)



(c)

Figure 3.18: Estimated radius versus true radius when $d = 15$ cm $d = 20$ cm and $d = 25$ cm by using a 100-iteration Monte Carlo simulation study.

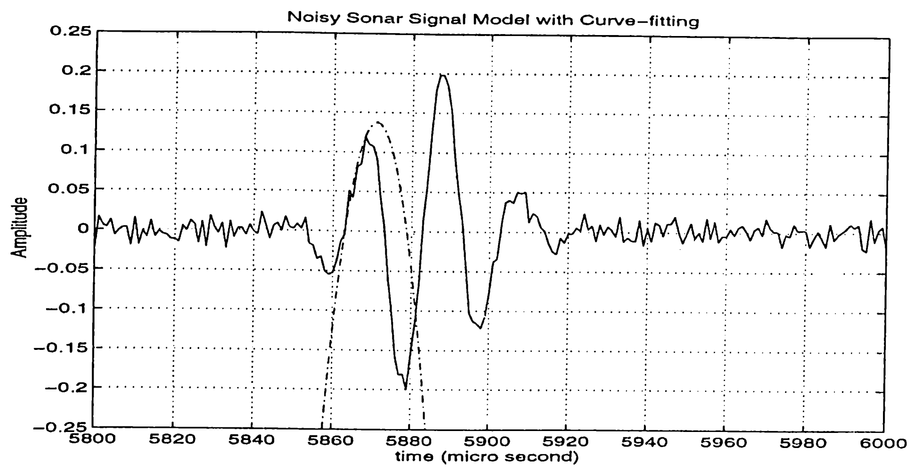


Figure 3.19: Curve-fitting method to estimate the time-of-flight.

- The time instant at which the value of this parabola is zero is the estimate of time-of-flight with the curve-fitting method.

3.3.1 Simulation Results

Figure 3.20 shows the results found by curve-fitting for $\theta = 0°$ case when the sensors are rotated around their joints. The improvement in the results is about 40% with respect to the thresholding method.

Figure 3.20 shows the effect of d for two different noise levels. When Figure 3.20(a) is compared with Figure 3.8(a), about 50% improvement is observed. The results improve especially after the rotation of the left sensor. When the curve-fitting approach is used, the error on the time-of-flight is sometimes positive and sometimes negative and likewise for h_{l1} and h_{r1}. Hence, the estimated radius is sometimes greater and sometimes less than the true radius. Figure 3.20(b) shows the effect of d and R simultaneously. When Figure 3.20(b) and 3.9(a) are compared, it can be seen that there is substantial improvement in the estimations both before rotation and after rotation of the sensors. The improvement is about 40%. Figure 3.20(c) shows the same results when the noise is small. When Figures 3.20(b) and 3.20(c) are compared, 20% improvement can be observed.

Figure 3.21 illustrates the effect of d, h_{o1} and R in a noiseless environment when the sensors are rotated around their centers. Figure 3.21(a) illustrates the effect of d. Up to $d = 43$ cm, both estimates have very small errors, but after this point, estimation before rotation gets worse and estimation after rotation continues improving. The corresponding d value for the thresholding method is 37 cm as can be seen from Figure 3.15. Figure 3.21(b) shows that there is only 2% error before and after rotation. Figure 3.21(c) illustrates that the error is 0% before and after rotation.

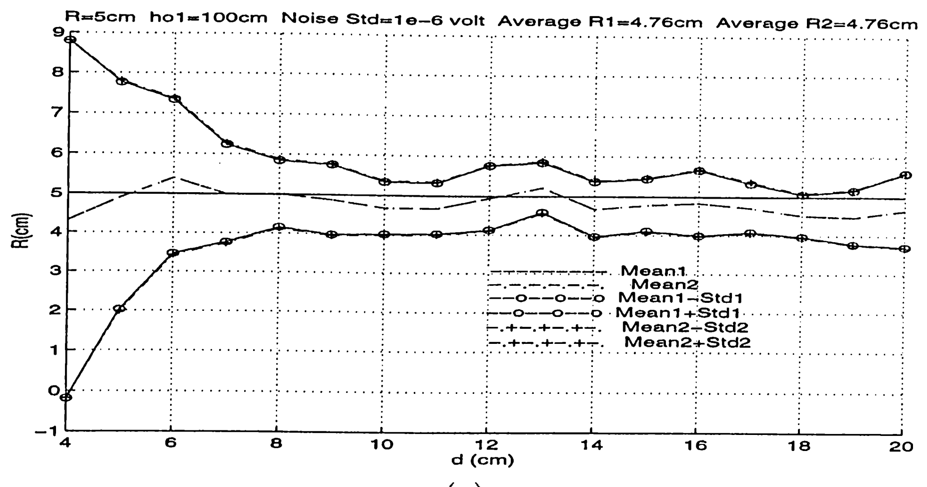
Figure 3.22 shows the effect of d, h_{o1} and R in a noisy environment when the sensors are rotated around their centers. In this case, a 100-iteration Monte Carlo simulation study is employed. Figure 3.22(a) illustrates the effect of d. When Figure 3.22(a) and 3.17(a) are compared, about 30% improvement is observed. Figure 3.22(b) illustrates that as h_{o1} increases standard deviations of both estimates increase. Also, when Figures 3.22(b) and 3.17(b) are compared, a 30% improvement is observed. Figure 3.22(c) illustrates the effect of the true radius. Estimate after rotation is better than estimate before rotation. When Figures 3.22(c) and 3.18(b) are compared, again a 30% improvement can be observed.

3.4 The Comparison of the Estimation Errors with the Cramér-Rao Lower Bound

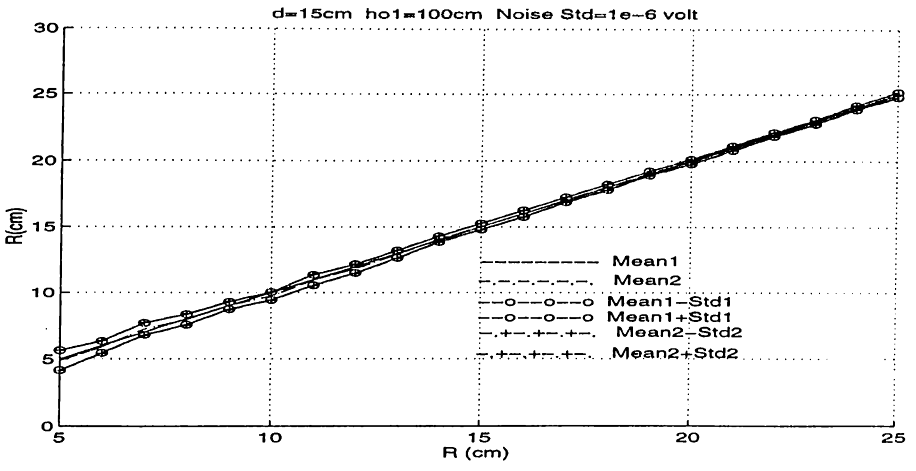
In order to evaluate the performances of the estimators, the results are compared to the Cramér-Rao lower bound (CRLB) which sets a lower bound on the variance of unbiased estimators [31]. The matched filter, which is the optimal method to estimate the time-of-flight, satisfies this lower bound asymptotically [31]. Since the CRLB is primarily derived for unbiased estimators, the variance and bias values are combined and compared with this lower bound. CRLBs for unbiased estimators of r, θ and R are derived in Appendix A.

Figures 3.23, 3.24 and 3.25 show how the CRLB ($\sqrt{J^{11}}$) and $\sqrt{\sigma_{\hat{R}}^2 + b^2(R)}$, the bias-variance combination, is affected by d, R and h_{o1}. $\theta = 0°$ is assumed for all cases. For each figure, parts (a) and (b) are the results obtained with the thresholding method and (c) and (d) are obtained with the curve-fitting method. Also, the sensors are assumed to rotate around their centers. In the curve-fitting method, the variance, and in the thresholding method, the bias term is dominant in the bias-variance combination.

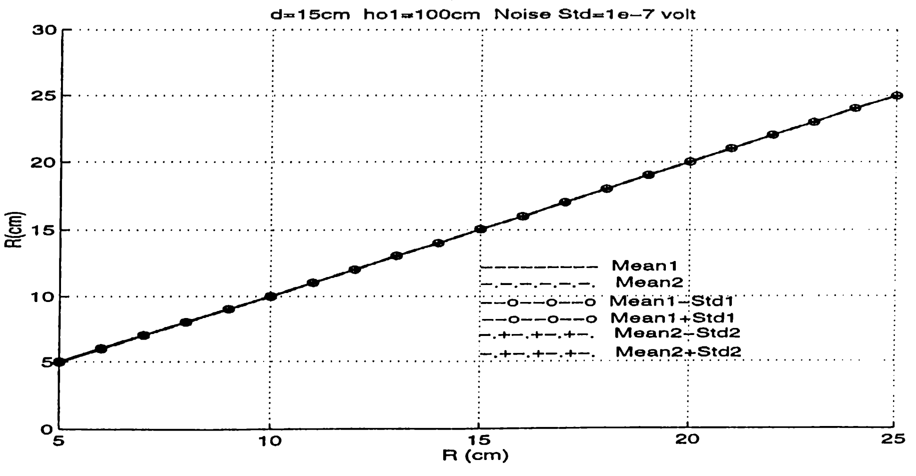
CRLB is small when compared with the bias-variance combination since Equation 3.4 is highly sensitive to d and h_{o1} and we calculated the time-of-flights with thresholding and curve-fitting methods which are very fast but



(*a*)

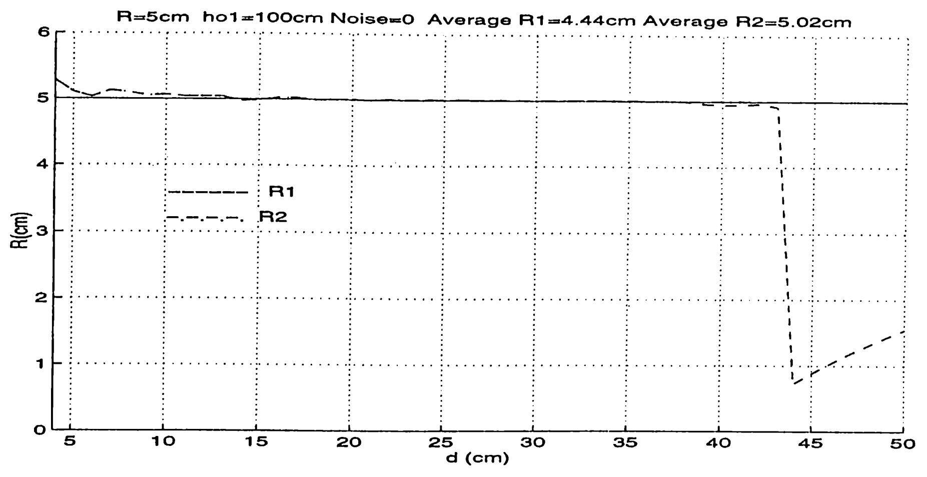


(*b*)

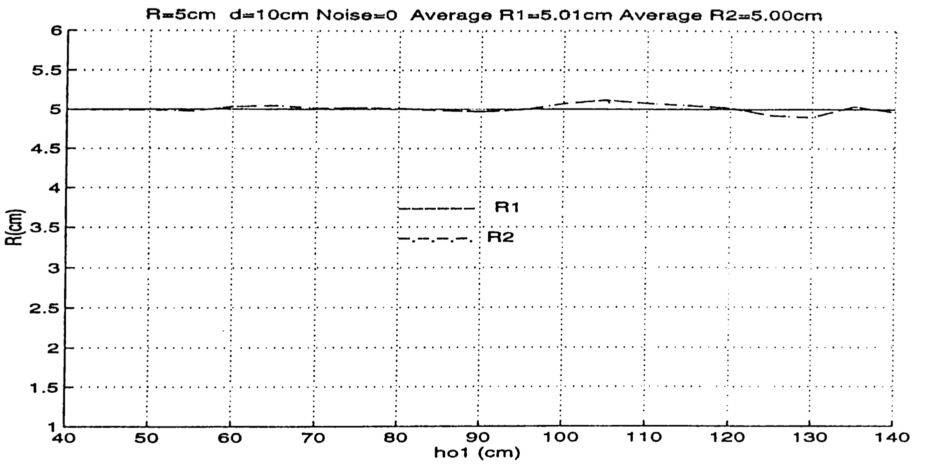


(*c*)

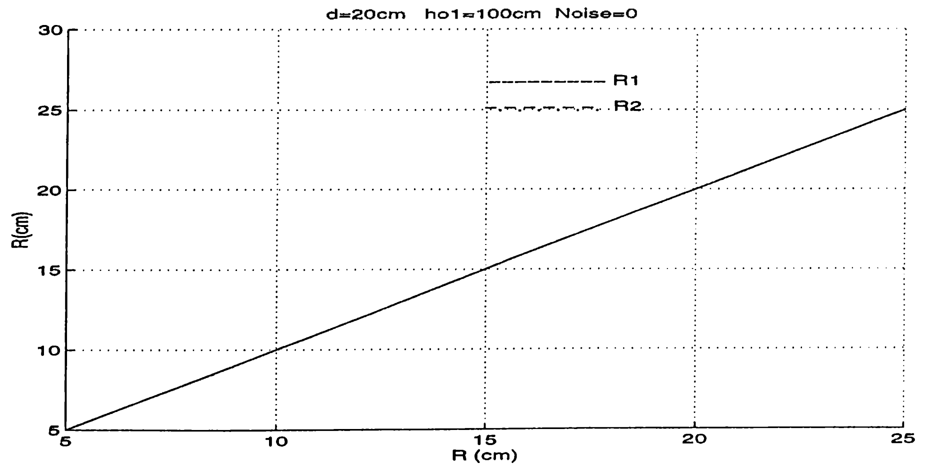
Figure 3.20: Estimated radius versus d and true radius when time-of-flight is estimated by the curve-fitting method when sensors are rotated around their joints.



(a)

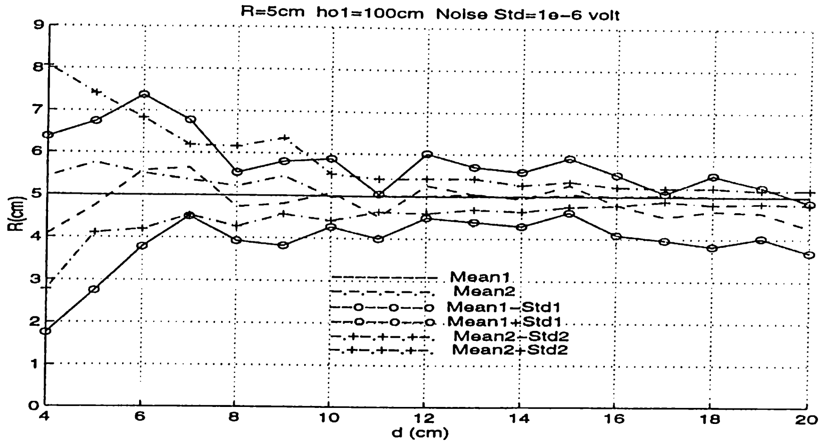


(b)

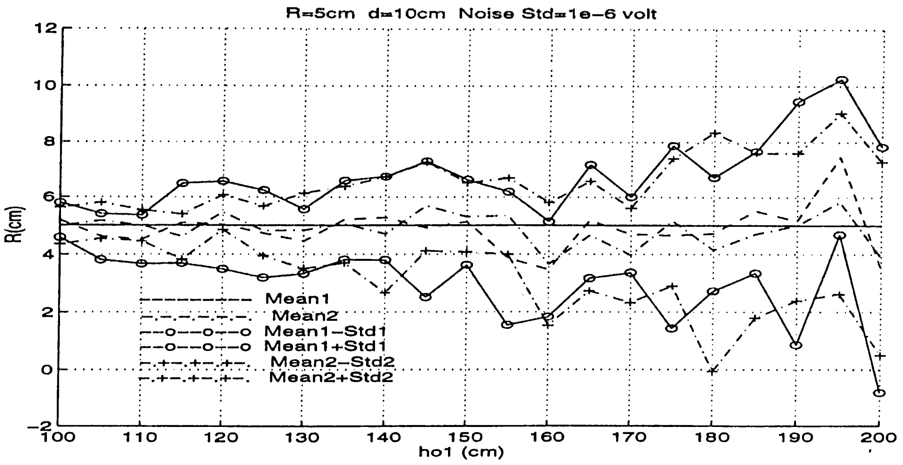


(c)

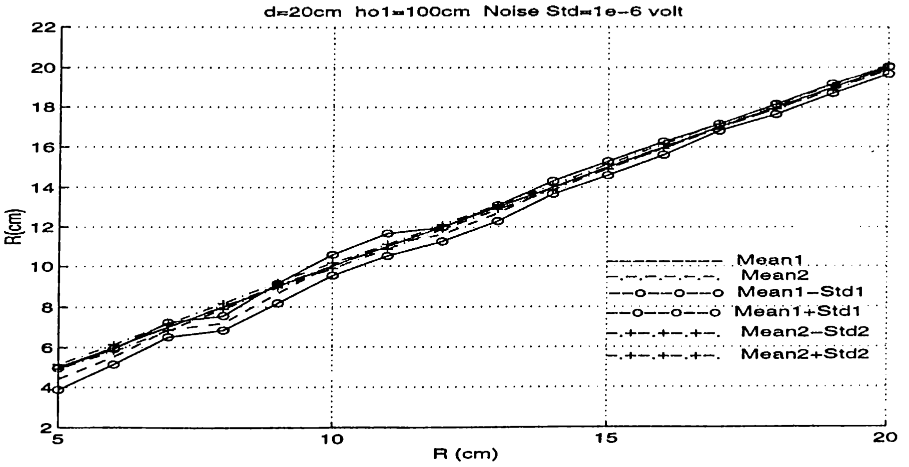
Figure 3.21: Estimated radius versus d, h_{o1} and R when sensors are rotated around their centers and the curve-fitting method is used in a noiseless environment.



(a)



(b)



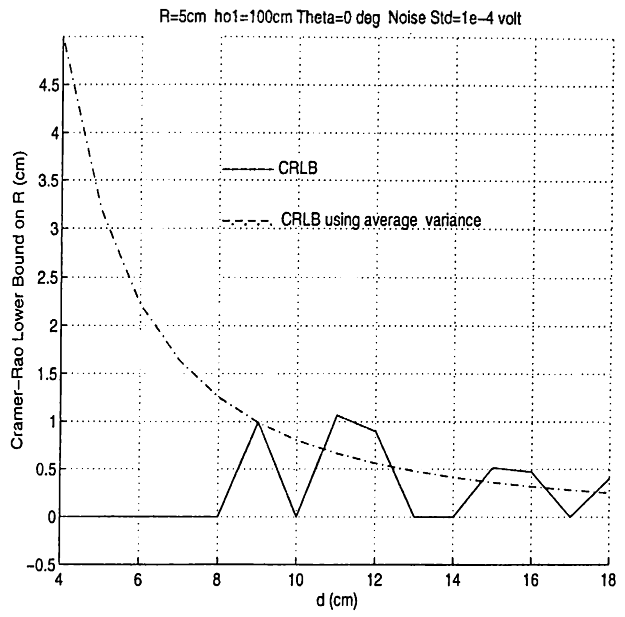
(c)

Figure 3.22: Estimated radius versus d, h_{o1} and R when sensor are rotated around their centers, the curve-fitting method and a 100-iteration Monte-Carlo simulation study are used.

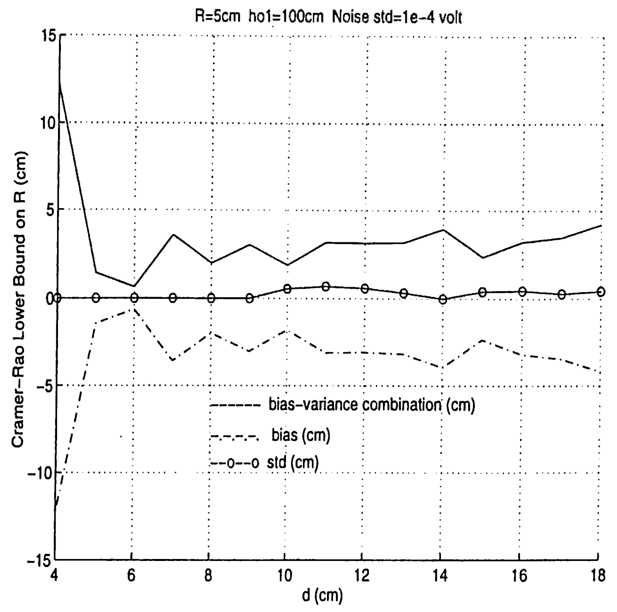
suboptimal. For the thresholding method, the difference between CRLB and bias-variance combination is more than that for the curve-fitting method. The difference for thresholding is about 5-fold and for curve-fitting about 1.5-fold. Also, with the curve-fitting method, the CRLB has a tendency to either steadily increase or decrease but with the thresholding method there is no such tendency. With the comparison of the estimators to CRLB, it is concluded that the estimator with curve-fitting method is better than the estimator with thresholding method. If the time-of-flights were to be estimated optimally, it would be necessary to store the templates of the signals corresponding to different objects at different distances and angles and to compare the returned signal with these and find the best matched one. This way, the processing time and the memory requirements would increase considerably. By using the thresholding method, a trade-off between the accuracy of estimation and system complexity is achieved.

3.5 Discussion

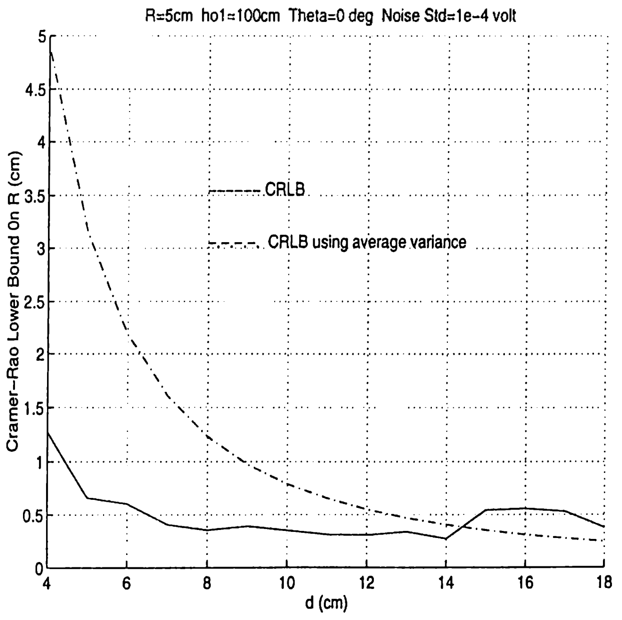
A method for radius of curvature estimation has been developed by using various configurations of sonar sensors. Two different methods of time-of-flight estimation are used. In order to assess the performances of the estimators, the comparison of the Cramér-Rao lower bounds with the bias-variance combination is included. When the time-of-flight is estimated by the thresholding method, the estimation of radius after the rotation of the left and right sensors is better than that of the linear case in the absence of noise. After adding the noise, the estimations after rotation degrade because of the error in the rotation due to noise. When the time-of-flight is estimated by the curve-fitting method, the results both before rotation and after rotation improve considerably. We obtain better results especially after rotation. When we use the curve-fitting method, the sensitivity of the estimation to noise decreases. In the cases for which the three sensors are rotated around their centers, considerable improvement is achieved due to the increased SNR. The effect of the error sources decrease very much. When sensors are rotated around their centers and curve-fitting method is used, further improvement is achieved. The comparison of the Cramér-Rao lower bounds with the bias-variance combination indicate that the difference is about 5-fold for thresholding and 1.5-fold for curve-fitting. This is caused by the suboptimality of these two methods used for the time-of-flight estimation. Also, it is seen that the difference is lower for the curve-fitting method than for the thresholding method.



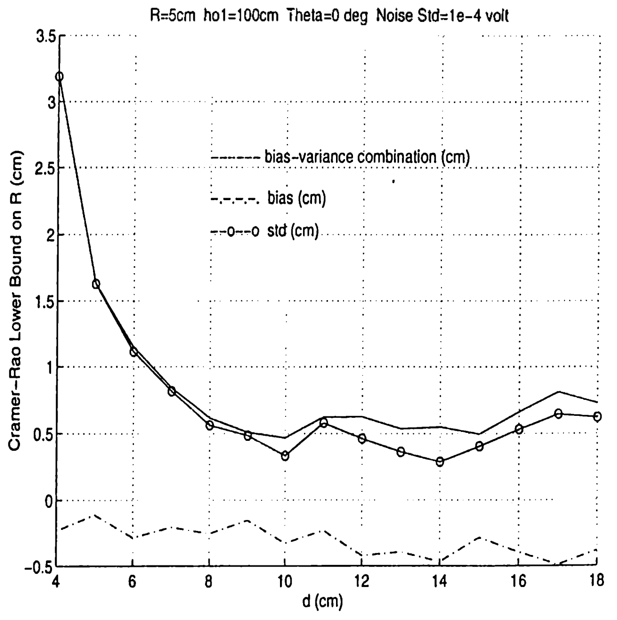
(a)



(b)

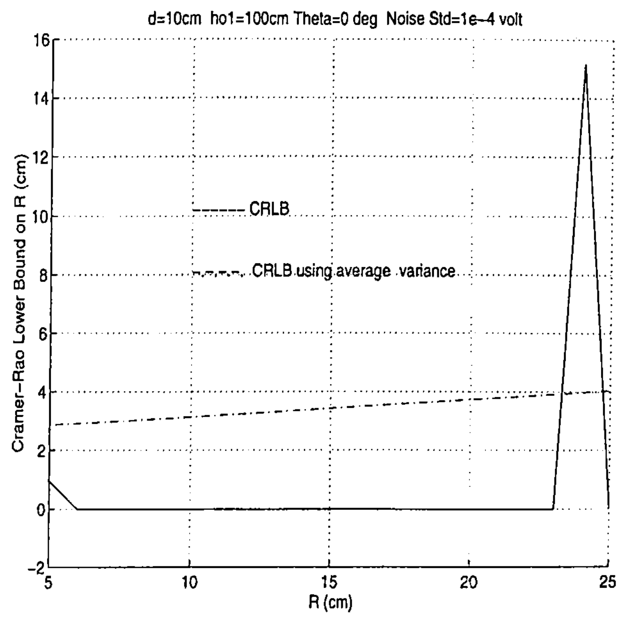


(c)

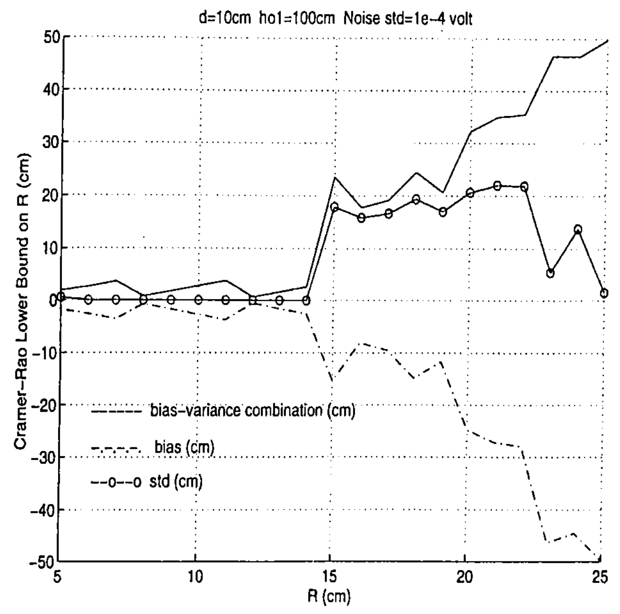


(d)

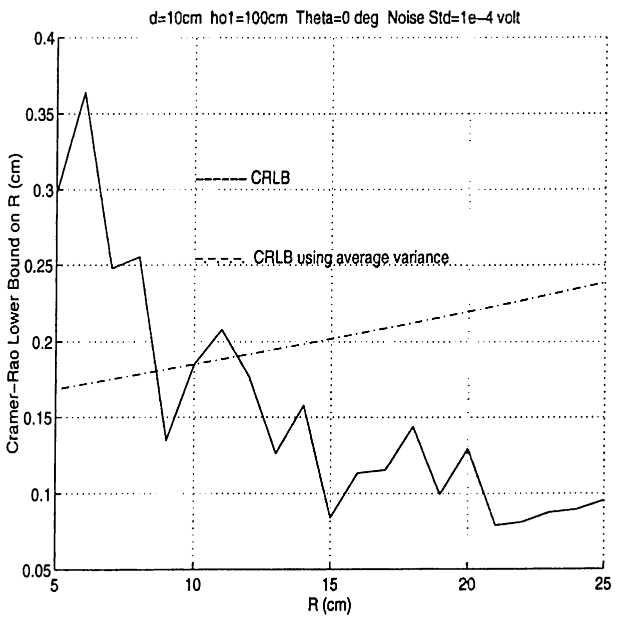
Figure 3.23: CRLB, bias, bias-variance combination versus d with (a),(b) thresholding and (c),(d) curve-fitting methods.



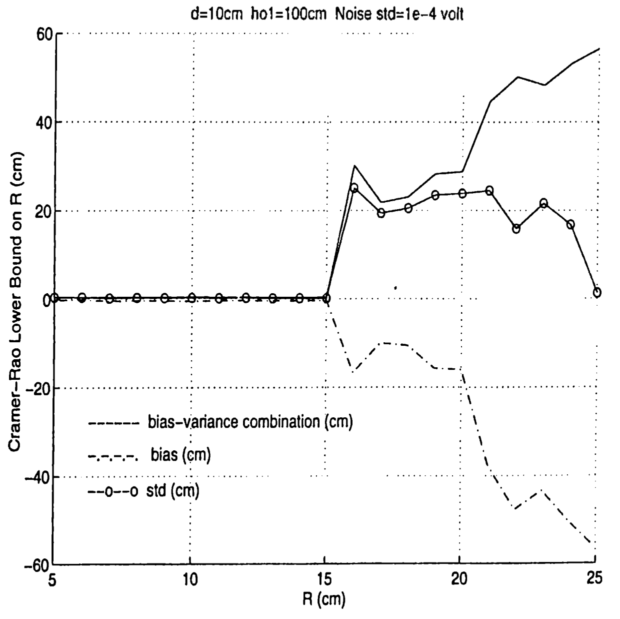
(a)



(b)

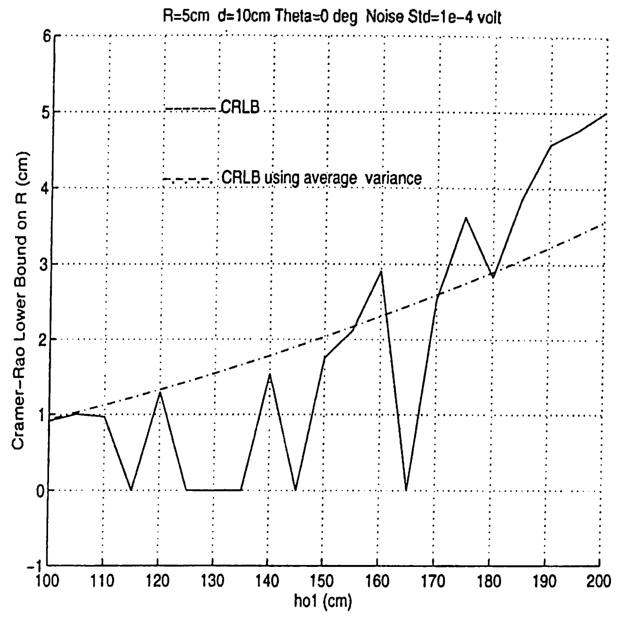


(c)

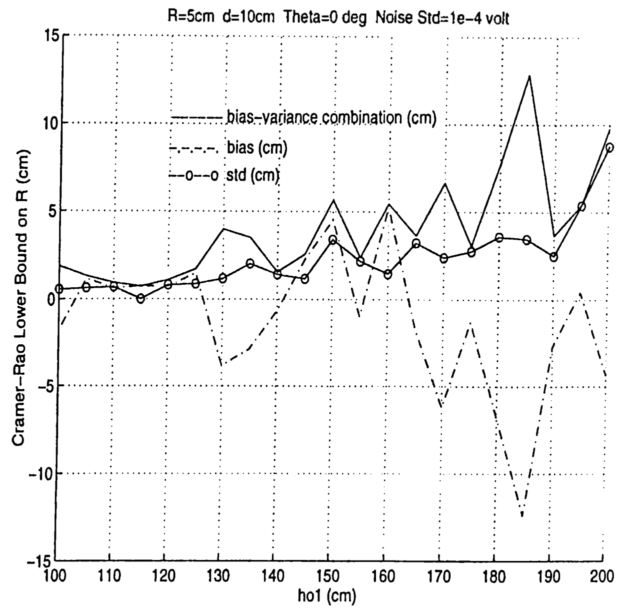


(d)

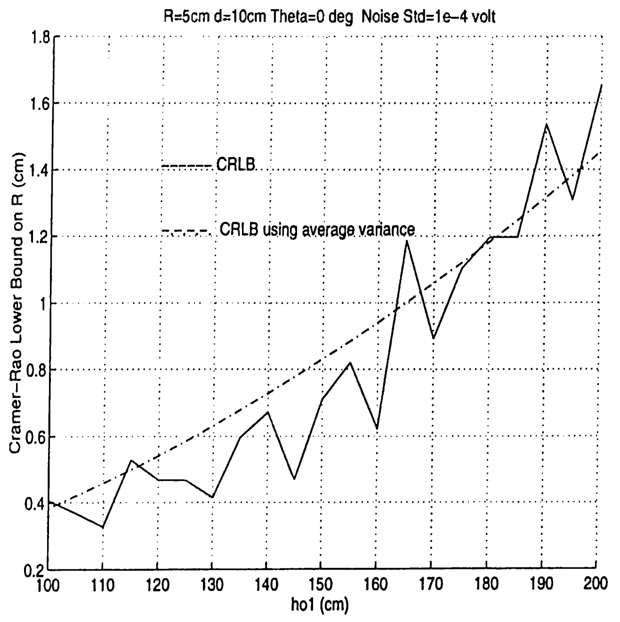
Figure 3.24: CRLB, bias, bias-variance combination versus R with (a),(b) thresholding and (c),(d) curve-fitting methods.



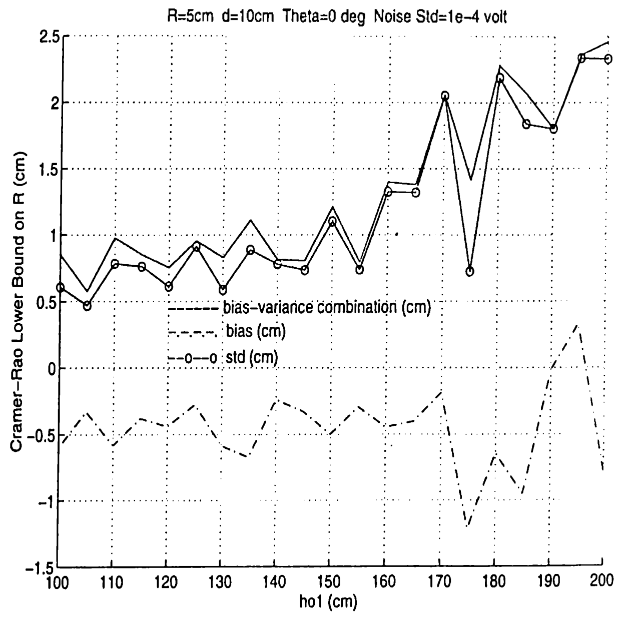
(a)



(b)



(c)



(d)

Figure 3.25: CRLB, bias, bias-variance combination versus h_{o1} with (a),(b) thresholding and (c),(d) curve-fitting methods.

Chapter 4

LOCATION AND RADIUS OF CURVATURE ESTIMATION

In this chapter, the radius of curvature estimation is extended to include estimation of target location. We investigate the case in which the sensors are rotated around their centers as shown in Figure 3.14 and estimate:

- The distance between the center of the object and the central sensor, $h_{o1} + R$.
- The normal angle of the central sensor, θ.
- The radius of curvature, R.

4.1 Algorithm for Location and Radius of Curvature Estimation

We have the following measurements:

$$\begin{aligned}
\hat{h}_o &= r - R + w_o(r, \theta, R) \\
\hat{h}_r &= \sqrt{r^2 + d^2 - 2dr\sin\theta} - R + w_r(r, \theta, R) \\
\hat{h}_l &= \sqrt{r^2 + d^2 + 2dr\sin\theta} - R + w_l(r, \theta, R) \qquad (4.1)
\end{aligned}$$

where $w_o(r,\theta,R)$, $w_r(r,\theta,R)$ and $w_l(r,\theta,R)$ are spatially uncorrelated zero-mean white Gaussian noise. Suppose the following vectors are defined:

$$\hat{\mathbf{z}} \triangleq \begin{bmatrix} \hat{h}_o \\ \hat{h}_r \\ \hat{h}_l \end{bmatrix} \qquad \mathbf{h}(r,\theta,R) = \begin{bmatrix} r - R \\ \sqrt{r^2 + d^2 - 2dr\sin\theta} - R \\ \sqrt{r^2 + d^2 + 2dr\sin\theta} - R \end{bmatrix} \tag{4.2}$$

In [1], it is shown that for Polaroid transducers the noise correlation coefficient is small since most of the noise on the sensors is dominated by the thermal noise in the electronics. Because of this, $w_o(r,\theta,R)$, $w_r(r,\theta,R)$ and $w_l(r,\theta,R)$ can be modeled as uncorrelated Gaussian noise. Hence, the error correlation matrix, its inverse, and the probability density function of $\hat{\mathbf{z}}$ are taken as follows:

$$\mathbf{C} = \begin{bmatrix} \sigma_{w_o}^2 & 0 & 0 \\ 0 & \sigma_{w_r}^2 & 0 \\ 0 & 0 & \sigma_{w_l}^2 \end{bmatrix} \tag{4.3}$$

$$\mathbf{C}^{-1} = \begin{bmatrix} \frac{1}{\sigma_{w_o}^2} & 0 & 0 \\ 0 & \frac{1}{\sigma_{w_r}^2} & 0 \\ 0 & 0 & \frac{1}{\sigma_{w_l}^2} \end{bmatrix} \tag{4.4}$$

$$p(\hat{\mathbf{z}}|r,\theta,R) = \frac{1}{2\pi|\mathbf{C}|}\exp\left\{-\frac{1}{2}[\hat{\mathbf{z}} - \mathbf{h}(r,\theta,R)]^T\mathbf{C}^{-1}[\hat{\mathbf{z}} - \mathbf{h}(r,\theta,R)]\right\} \tag{4.5}$$

The r, θ and R values maximizing Equation 4.5 are the maximum likelihood estimators which can be found by taking the inverse of $\hat{\mathbf{z}} = \mathbf{h}(\hat{r},\hat{\theta},\hat{R})$ as follows:

$$\hat{r} = \frac{2d^2 + 2(\hat{h}_l + \hat{h}_r)\hat{h}_o - 2\hat{h}_o^2 - \hat{h}_l^2 - \hat{h}_r^2}{2\hat{h}_r + 2\hat{h}_l - 4\hat{h}_o} \tag{4.6}$$

$$\hat{\theta} = \sin^{-1}\left[\frac{\hat{h}_l^2 - \hat{h}_r^2 + 2(\hat{h}_l - \hat{h}_r)\hat{R}}{4d(\hat{h}_o + \hat{R})}\right] \tag{4.7}$$

$$\hat{R} = \frac{(\hat{h}_o^2 + \hat{h}_l^2) - 2(\hat{h}_o^2 + d^2)}{4\hat{h}_o - 2(\hat{h}_r + \hat{h}_l\hat{h}_l)} \tag{4.8}$$

The algorithm to localize the object is outlined below:

- Noisy values of the distances are generated using Equation 4.1.
- By using these noisy values:
  - Radius of curvature,
  - Deviation angle (azimuth) of the central sensor,

- The distance between the central sensor and the object

  are calculated by using Equations 4.6–4.8.

- Noisy values of the normal angles of the left and right sensors are found as explained in the radius of curvature estimation algorithm.
- The sensors are rotated by their normal angles.
- New measurements are taken and the new estimates are made by using the above equations.

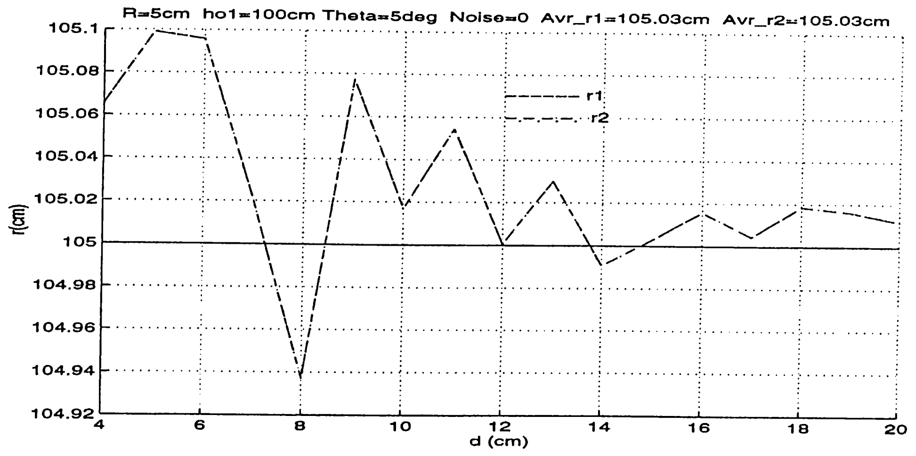
4.2 Simulation Results

Figures 4.1–4.3 show the simulation results obtained in a noiseless environment. Figures 4.4–4.7 show the same simulation results obtained by using a 100-iteration Monte Carlo simulation study in a noisy environment. For all simulation results, the curve-fitting method is used in order to estimate the TOF.

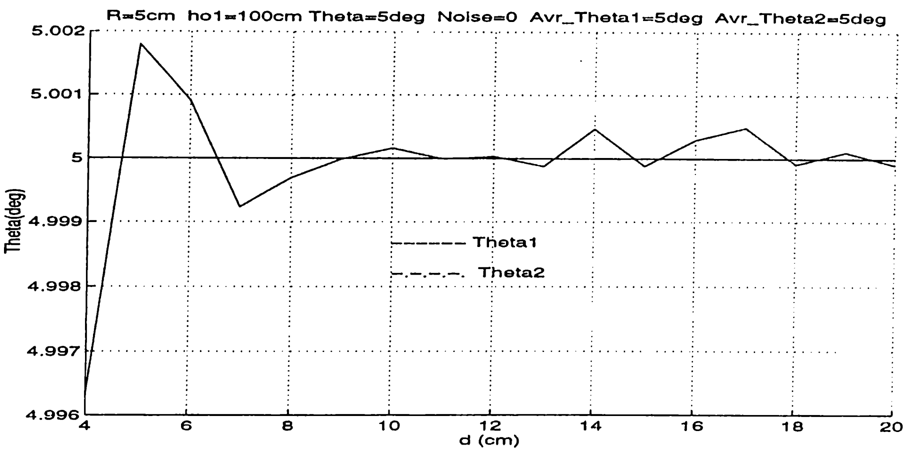
Figure 4.1 shows the dependence of the estimates to the transducer separation d. The estimates before and after rotation are the same. Figure 4.1(a) illustrates the dependence of the estimation of range to the transducer separation d. As d increases, the error decreases. The average error is 0.03%. Figure 4.1(a) illustrates the dependence of the angle estimate to d. As d increases, the estimate improves. The average error over d is 0.0%. Figure 4.1 illustrates the dependence of the radius of curvature estimation to d. Again as d increases, the error decreases. The average error over d is 0.6%.

Figure 4.2 displays how the estimates depend on the true distance between the central sensor and the object, h_{o1}. Figure 4.2(a), 4.2(b) and 4.2(c) show the dependence of the distance, the angle and the radius of curvature to h_{o1}, respectively. The average errors for the distance, the angle and the radius of curvature estimates are 3.6%, 0.0% and 0.2%, respectively.

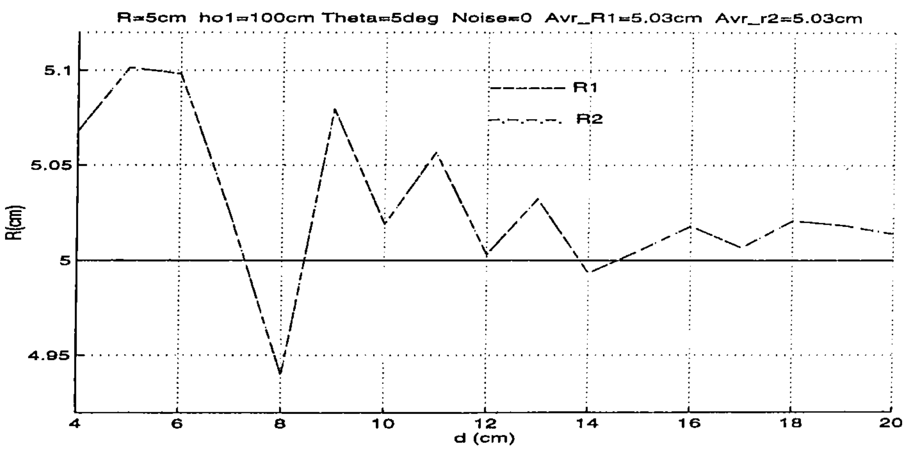
Figure 4.3 illustrates the dependence of the estimates on the true radius, R. Figure 4.3(a) shows the dependence of the distance to R. The average error is 0.0%. Figure 4.3(b) shows the dependence of the normal angle to R. Figure 4.3(c) shows the dependence of the radius of curvature estimation to the true radius. The average error is 0.0%.



(*a*)

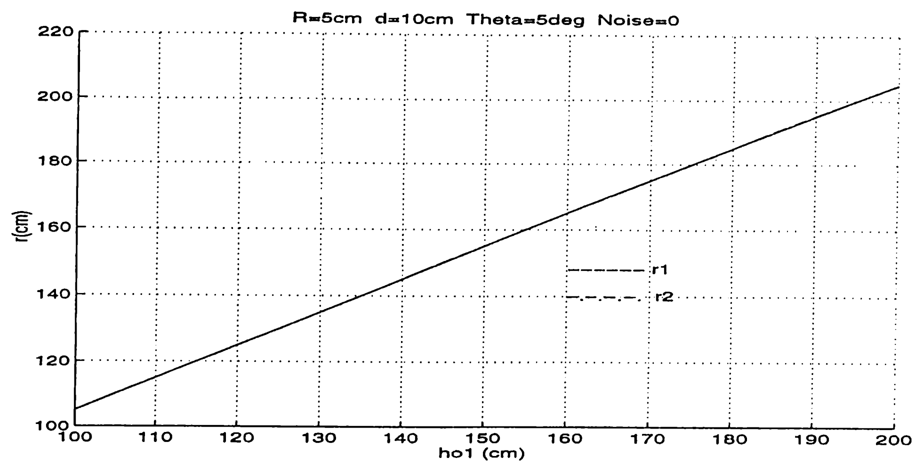


(*b*)

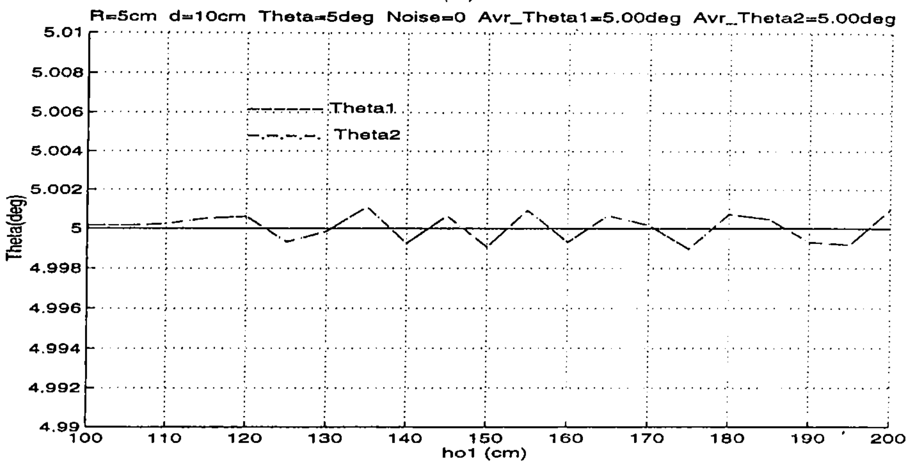


(*c*)

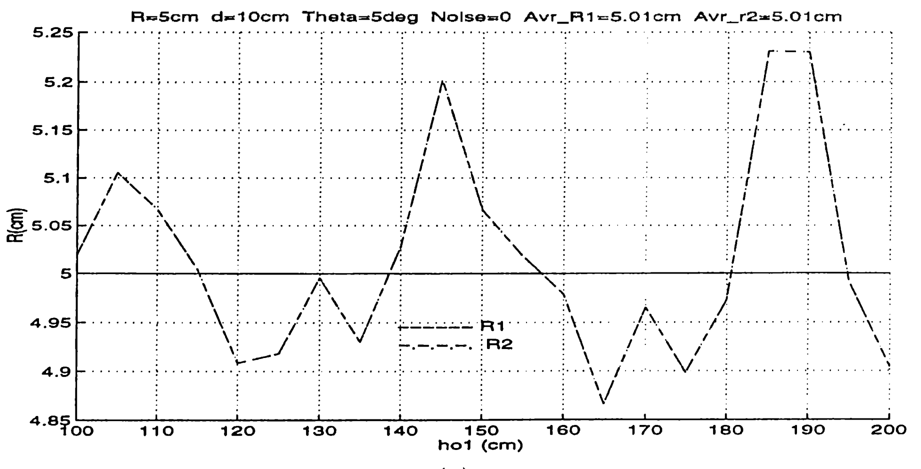
Figure 4.1: Location and radius of curvature estimation versus d in a noiseless environment with the curve-fitting method when sensors are rotated around their centers (note that both estimates are the same).



(a)

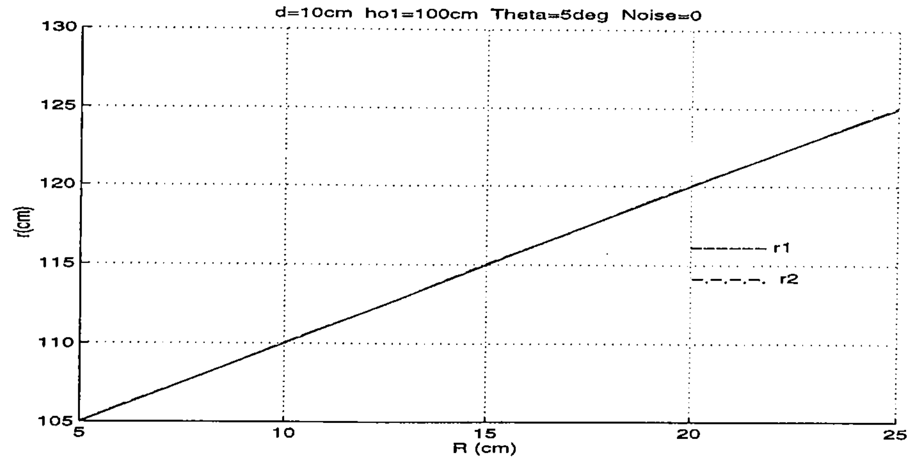


(b)

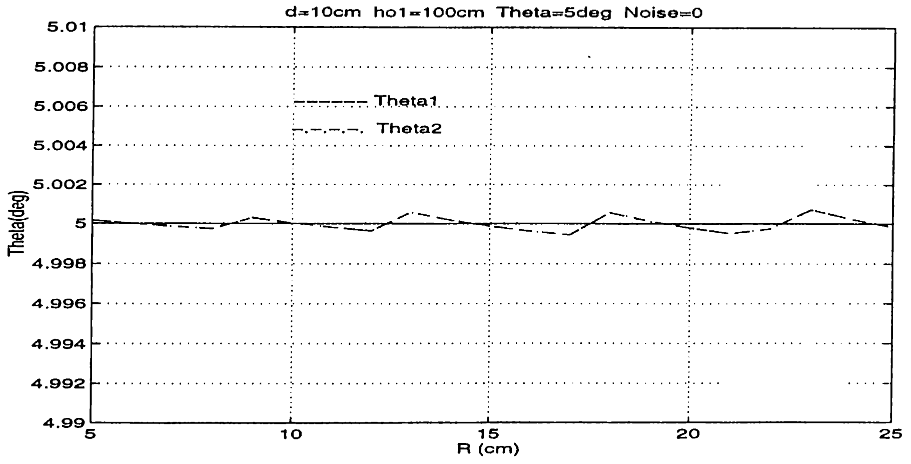


(c)

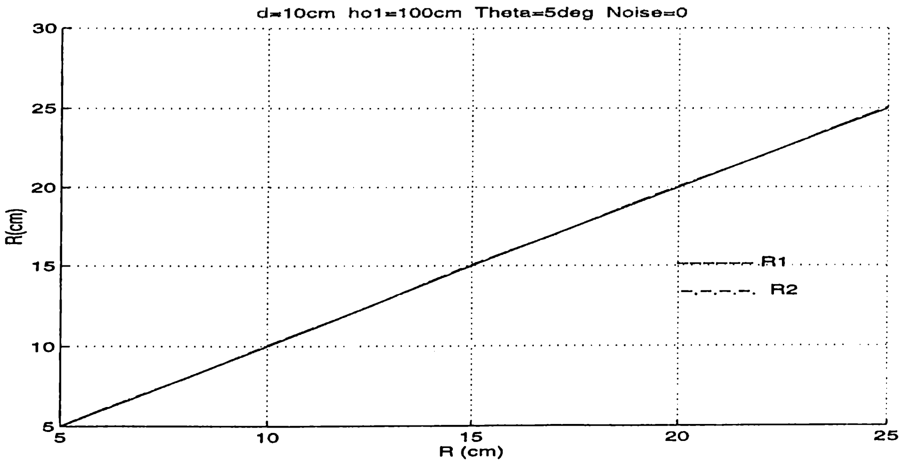
Figure 4.2: Location and radius of curvature estimation versus h_{o1} in a noiseless environment with the curve-fitting method when sensors are rotated around their centers (note that both estimates are the same).



(a)



(b)



(c)

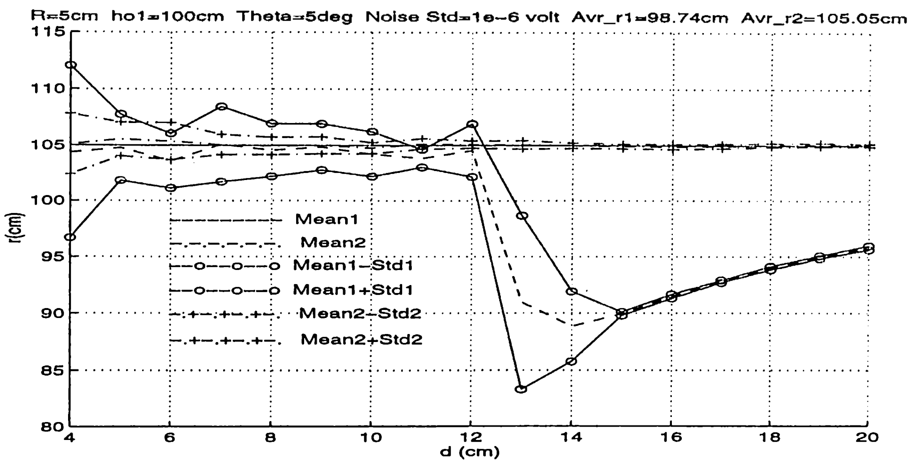
Figure 4.3: Location and radius of curvature estimation versus R in a noiseless environment with the curve-fitting method when sensors are rotated around their centers (note that both estimates are the same).

Figure 4.4 shows the dependence of the estimates to d in a noisy environment. Figure 4.4(a) illustrates that as d increases the estimate after rotation improves. The estimation before rotation improves up to $d = 12$ cm, and after that value it gets worse since the target is now located either at very low SNR regions of the sensitivity pattern or outside it (as d increases the normal angles of the left and right sensors increase). The average error before rotation is 5.96% and the error after rotation is 0.05%. Figure 4.4(b) illustrates the dependence of the normal angle to d. The error figures before and after rotation are 6.0% and 0.0% respectively. Figure 4.4(c) illustrates the dependence of the radius of curvature estimation to d. The average error before rotation is 11.4% and the average error after rotation is 1.2%. Note that the standard deviation of the estimates after rotation are lower than those before rotation by a factor of four approximately.

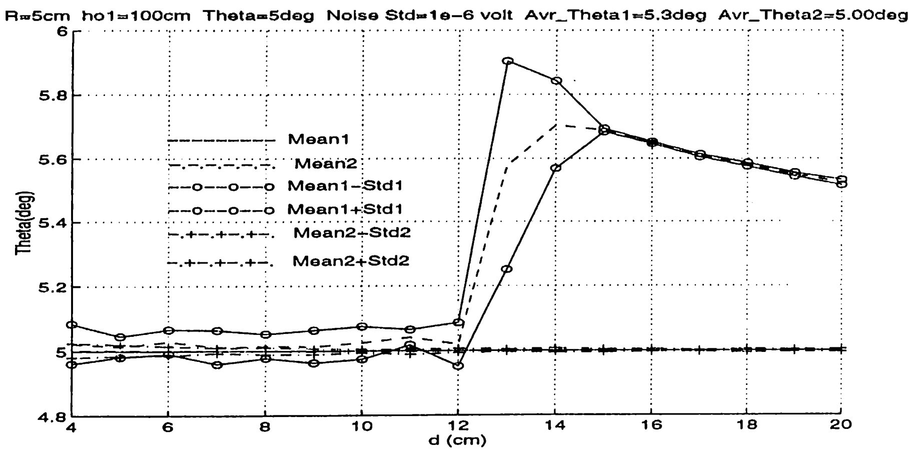
Figure 4.5 shows the dependence of the estimates to the distance h_{o1} in a noisy environment. For small values of h_{o1}, the normal angles of the left and right sensors are high and the estimates are not very good. For large values of h_{o1}, the normal angles decrease and estimations improve. Figure 4.5(a) illustrates the distance estimation. Figure 4.5(b) illustrates the angle estimation. The average error before rotation is 6.0% and after rotation it is 0.0%. Figure 4.5 illustrates the radius of curvature estimation. The average error before rotation is 19.0% and after rotation it is 0.0%. Note that the standard deviations after rotation are lower than those before rotation.

Figure 4.6 shows the effect of R in a noisy environment. As R increases, all three estimates improve. Figure 4.6(a) illustrates the range estimate. Figure 4.6(b) illustrates the azimuth angle estimate. The average error before rotation is 0.4% and after rotation it is 0.0%. Figure 4.6(c) illustrates the radius of curvature estimation.

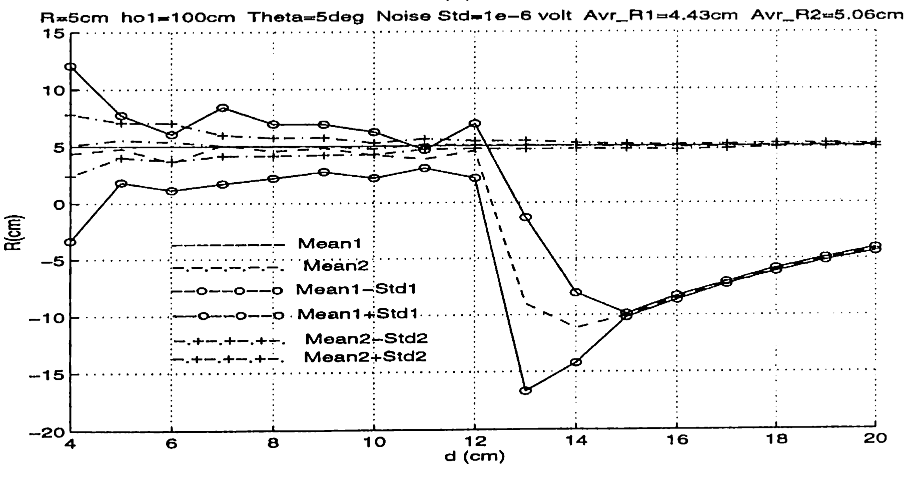
Figure 4.7 shows the effect of the azimuth angle θ in a noisy environment. Up to $\theta = 6°$ the estimates improve, but after that value the target is either outside the sensitivity region or within a very low SNR region. Figure 4.7(a) illustrates the distance estimate. The average error before rotation is 5.71% and after rotation it is 0.0%. Figure 4.7(b) illustrates the angle estimation. Figure 4.7(c) illustrates the radius of curvature estimation. The average error before rotation is 85.6% and after rotation it is 0.6%. Note that again the standard deviations after rotation are smaller than those before rotation.



(a)

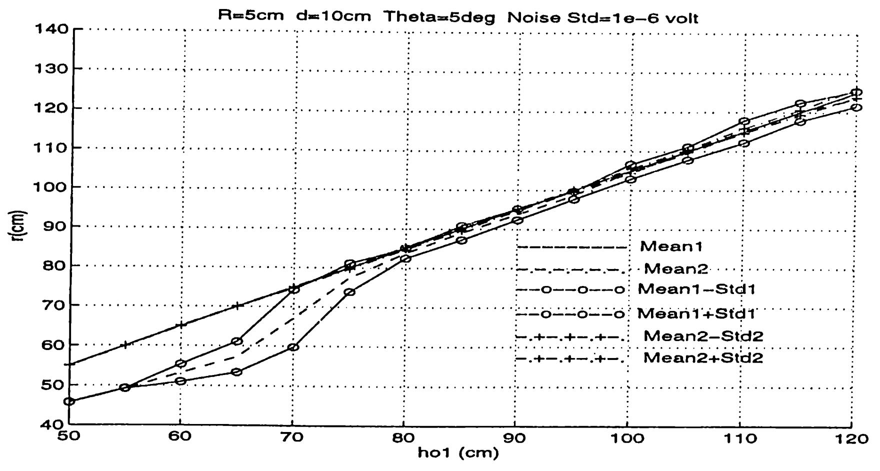


(b)

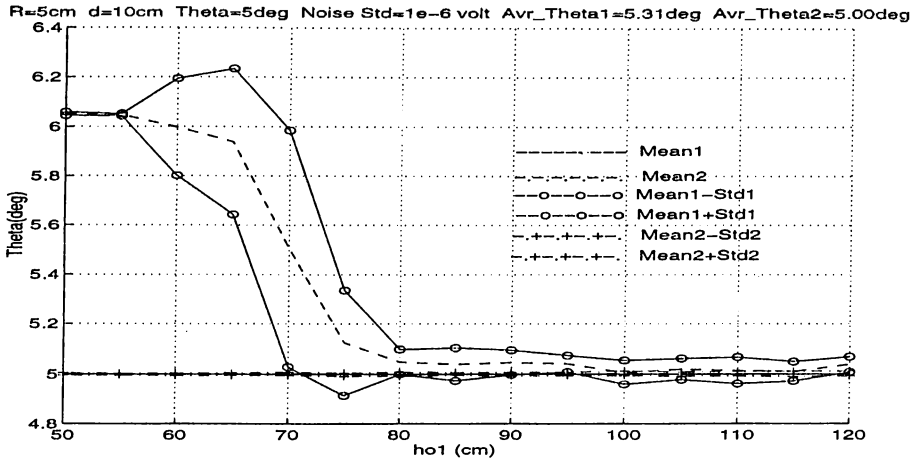


(c)

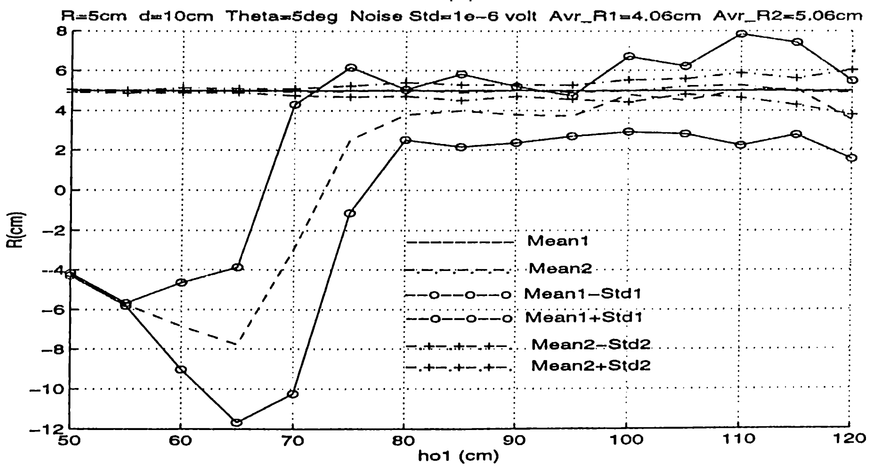
Figure 4.4: Location and radius of curvature estimation versus *d* under noise with the curve-fitting method when the sensors are rotated around their centers and a 100-iteration Monte Carlo simulation study is employed.



(a)

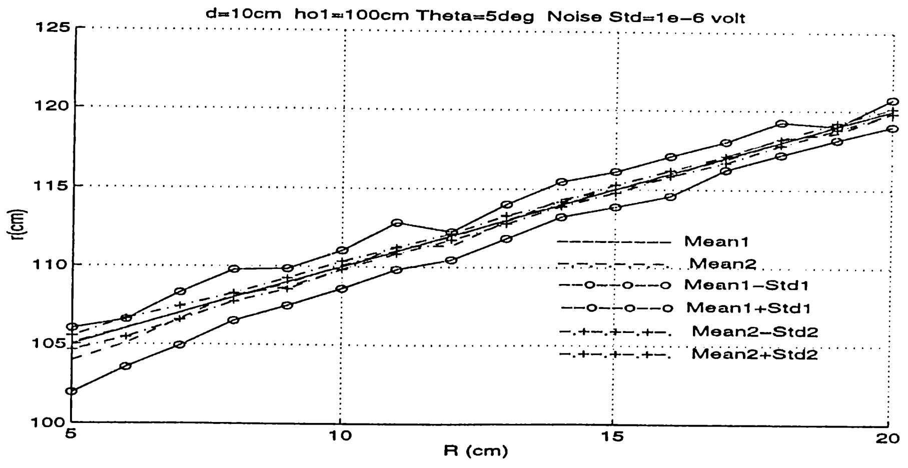


(b)

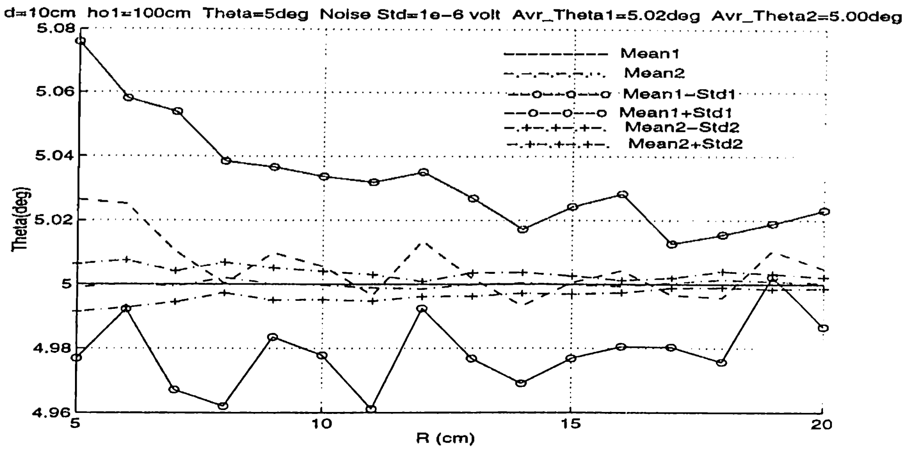


(c)

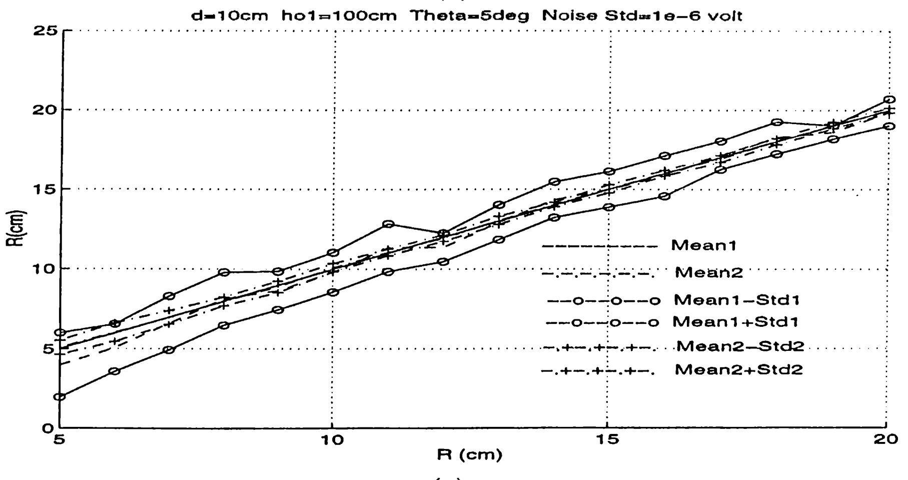
Figure 4.5: Location and radius of curvature estimation versus h_{o1} under noise with the curve-fitting method when the sensors are rotated around their centers and a 100-iteration Monte Carlo simulation study is employed.



(*a*)

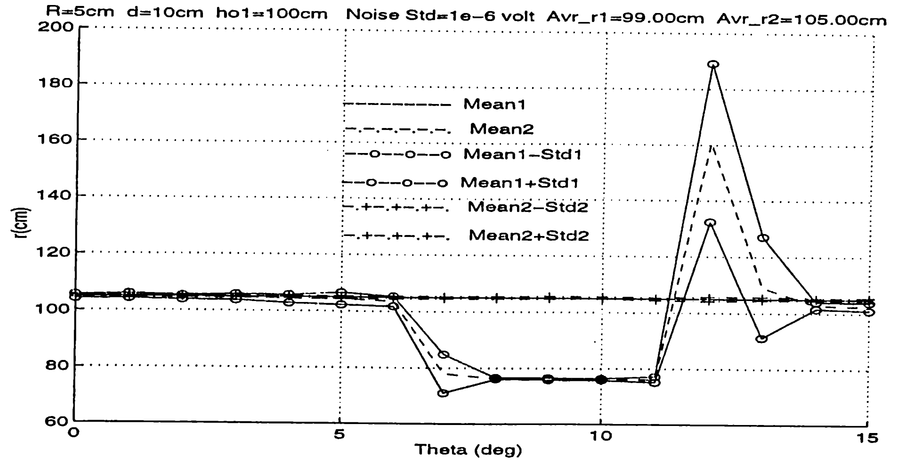


(*b*)

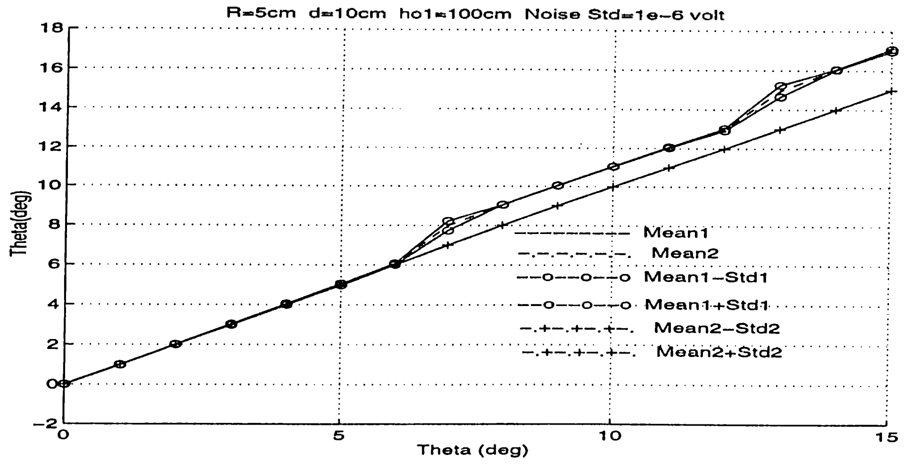


(*c*)

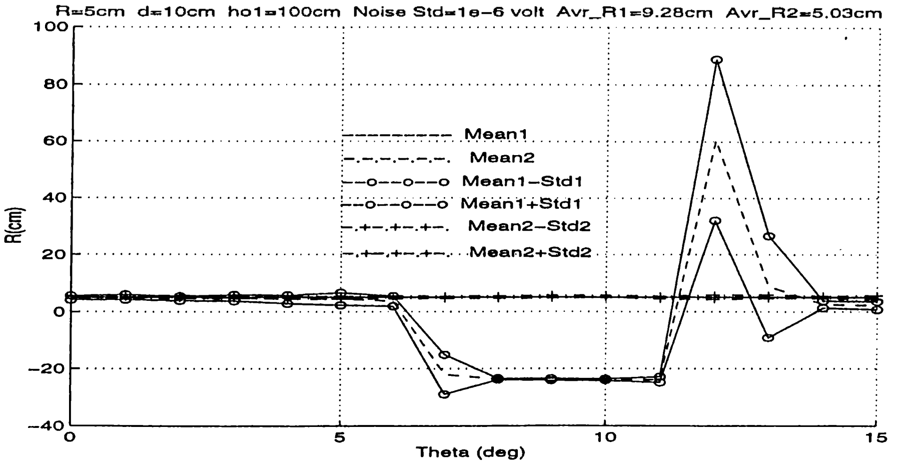
Figure 4.6: Location and radius of curvature estimation versus R under noise with the curve-fitting method when the sensors are rotated around their centers and a 100-iteration Monte Carlo simulation study is employed.



(a)



(b)



(c)

Figure 4.7: Location and radius of curvature estimation versus θ under noise with the curve-fitting method when the sensors are rotated around their centers and a 100-iteration Monte Carlo simulation study is employed.

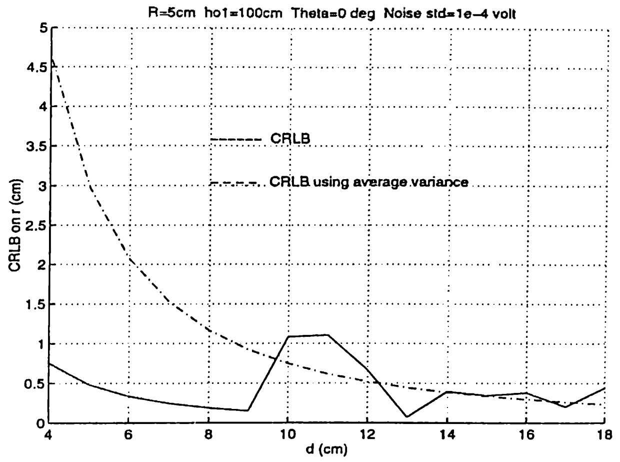
4.3 The Comparison of the Estimation Errors with the Cramér-Rao Lower Bound

As explained in the radius of curvature estimation section, in order to evaluate the performances of the estimators, the results are compared to the CRLB which sets a lower bound of the variance of unbiased estimators [31]. In this section, we combine the variance and bias values to compare to this lower bound. The derivation is provided in the Appendix.

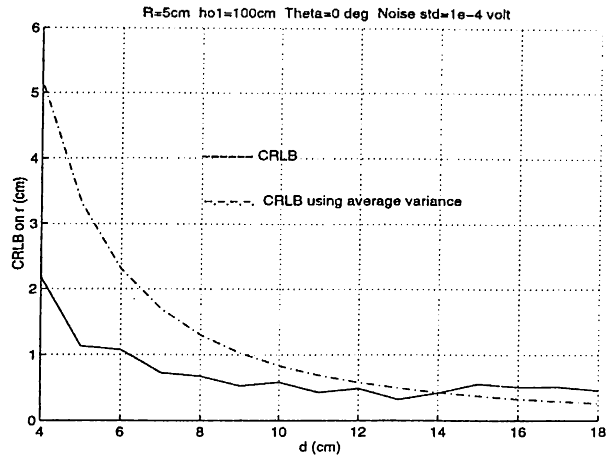
Figures 4.8–4.11 show how the Cramér-Rao lower bounds $\sqrt{J^{11}}$, $\sqrt{J^{22}}$, $\sqrt{J^{33}}$ and $\sqrt{\sigma_{\hat{r}}^2 + b^2(r)}$, $\sqrt{\sigma_{\hat{\theta}}^2 + b^2(\theta)}$, $\sqrt{\sigma_{\hat{R}}^2 + b^2(R)}$, which are the combinations of biases and variances, are affected by d, h_{o1}, R and θ. The simulation results are presented both for the thresholding method and the curve-fitting method. The first columns of the figures illustrate the results of the thresholding method, and the second columns show the results of the curve-fitting method.

Figure 4.8 and 4.9 illustrate the effect of d. Figures 4.8(a),(c) and (e) illustrate the results with thresholding method. When Figures 4.8(a) and (e) are compared to Figures 4.9(a) and (e), a 3-fold difference can be observed. When Figure 4.8(c) and 4.9(c) are compared, the difference is 10-fold. Figures 4.8(b),(d) and (f) illustrate the result with the curve-fitting method. When Figures 4.8(b) and (f) are compared to Figures 4.9(b) and (f), it is observed that they are very close to each other. When Figure 4.8(d) and 4.9(d) are compared, a 2-fold difference can be observed. From these results, it is concluded that the curve-fitting estimator is more accurate than the estimator with thresholding. Figure 4.9 illustrates that for the thresholding method the bias term, and for the curve-fitting method the variance term is dominant.

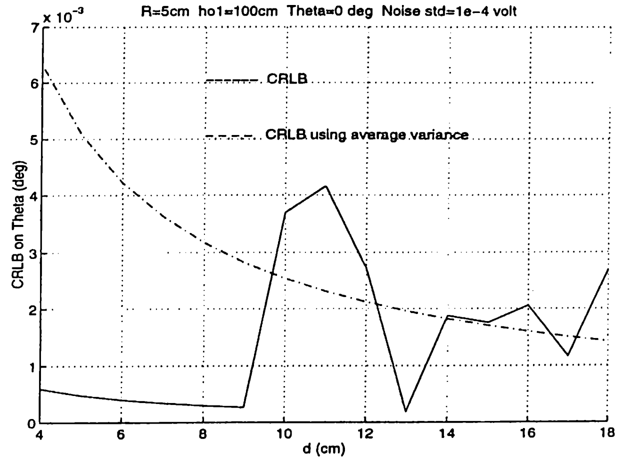
Figure 4.10 and 4.11 illustrate the effect of h_{o1}. Figure 4.10(a),(c) and (e) illustrate the results with thresholding. When Figures 4.10(a) and (e) are compared to Figures 4.11(a) and (e), it is seen that they are nearly the same. Figure 4.10(c) and Figure 4.11(c) show a 10-fold difference. Figures 4.10(b),(d) and (f) illustrate the results with curve-fitting. When Figures 4.10(b) and (f) are compared to Figures 4.11(b) and (f), they seem to be the same. When Figure 4.10(d) and 4.11(d) are compared, a 5-fold difference can be observed. It is concluded that the estimator with curve-fitting has higher performance. Again, the bias term is dominant for the thresholding method, and variance term is dominant for the curve-fitting method.



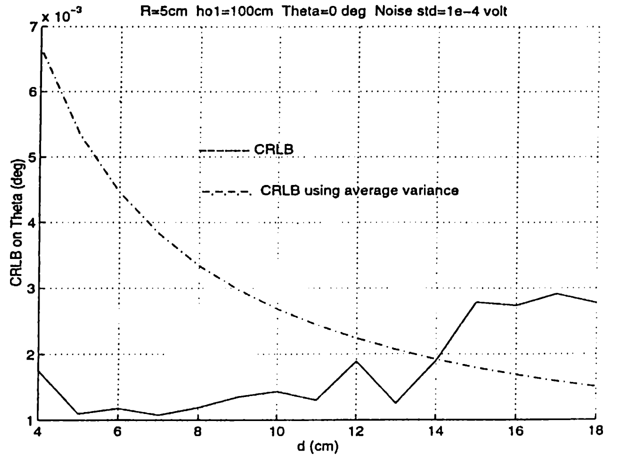
(a)



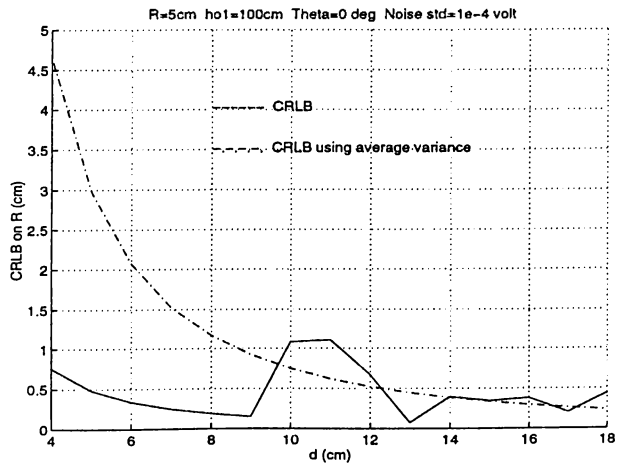
(b)



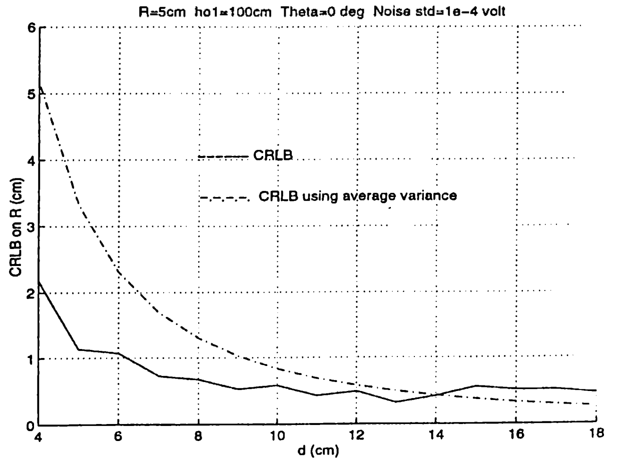
(c)



(d)

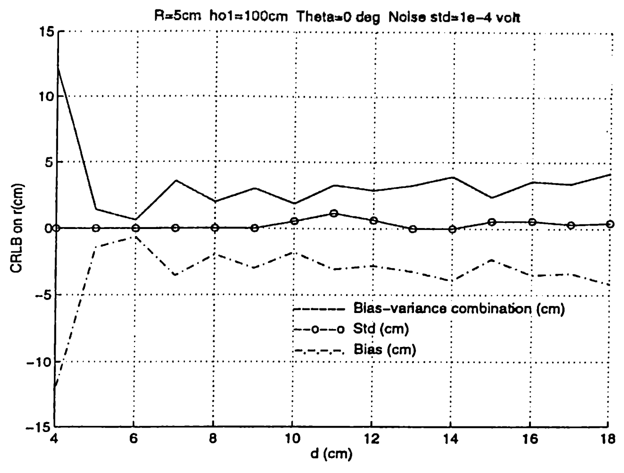


(e)

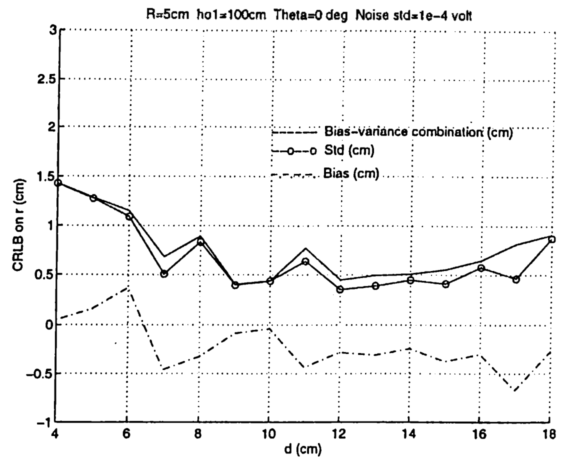


(f)

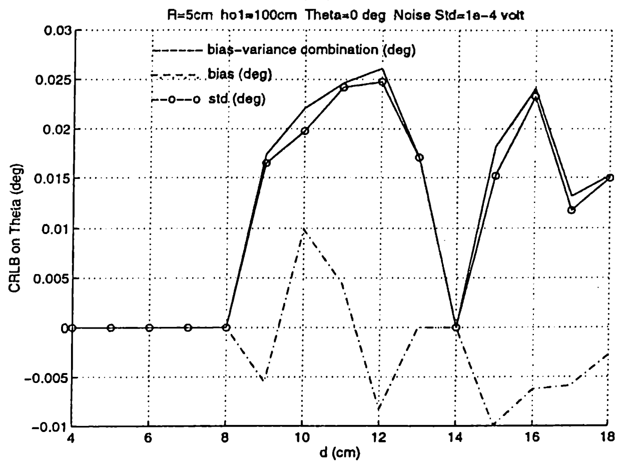
Figure 4.8: CRLB versus d. First column is with thresholding and second column is with curve-fitting.



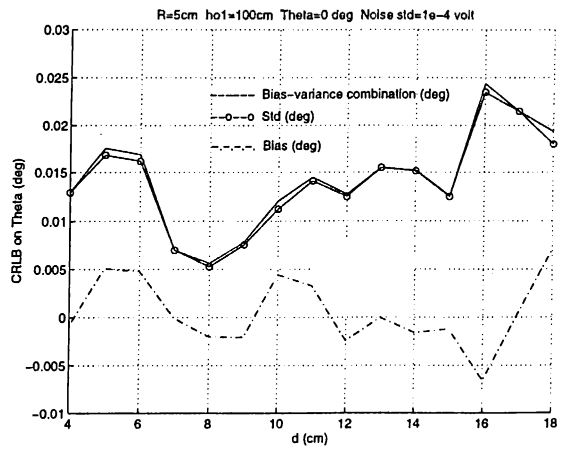
(a)



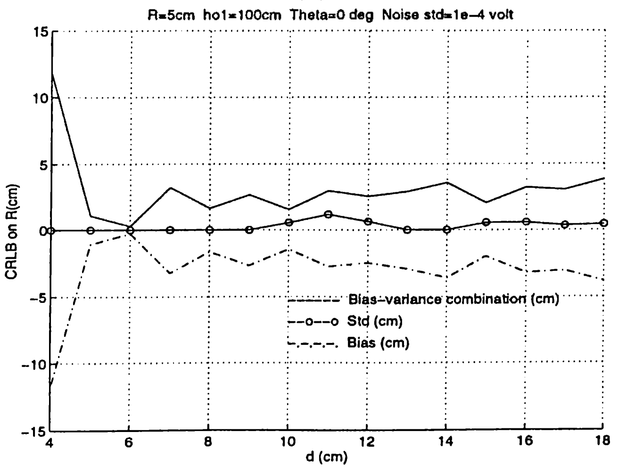
(b)



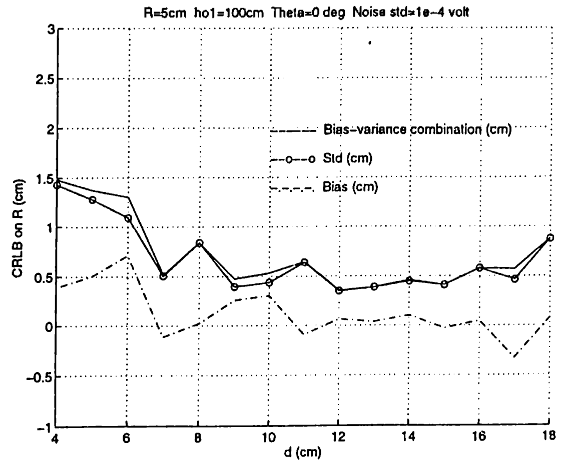
(c)



(d)

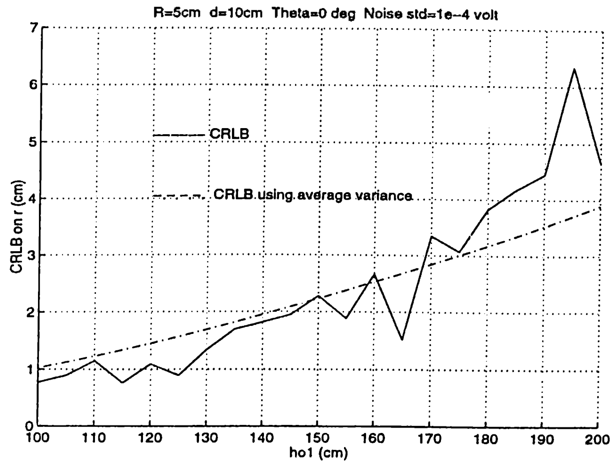


(e)

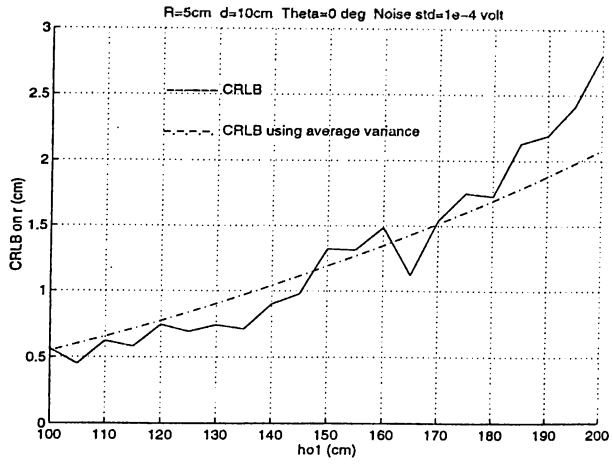


(f)

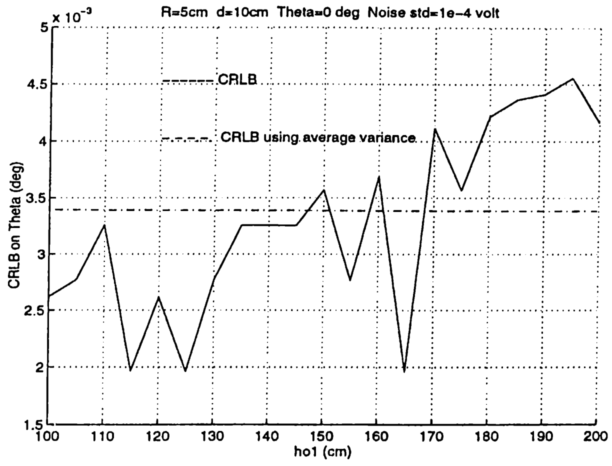
Figure 4.9: Bias, bias-variance combination, standard deviation versus d with thresholding (first column) and curve-fitting methods (second column).



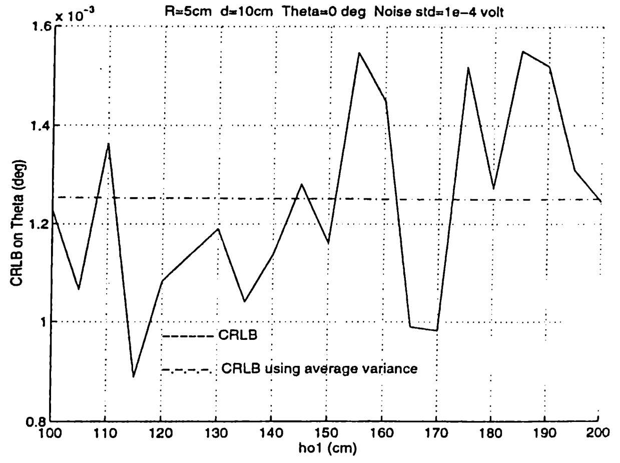
(a)



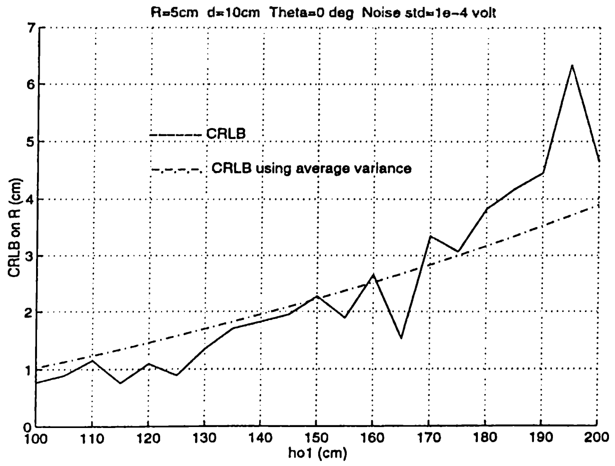
(b)



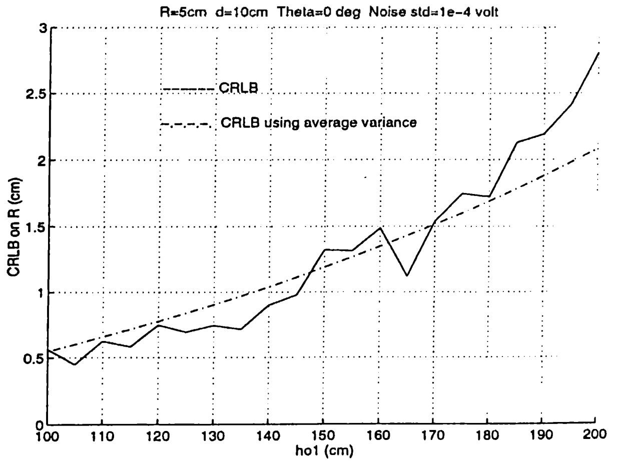
(c)



(d)

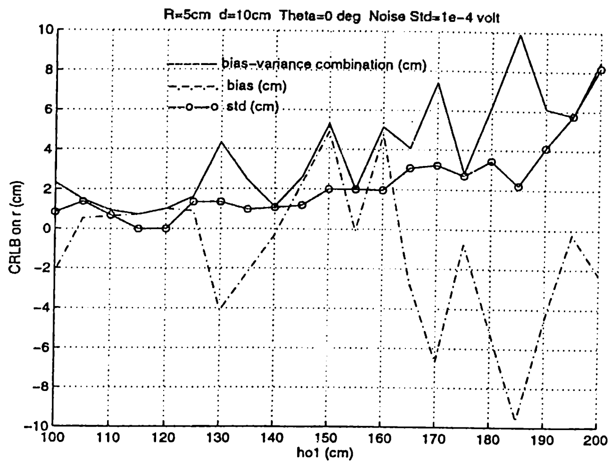


(e)

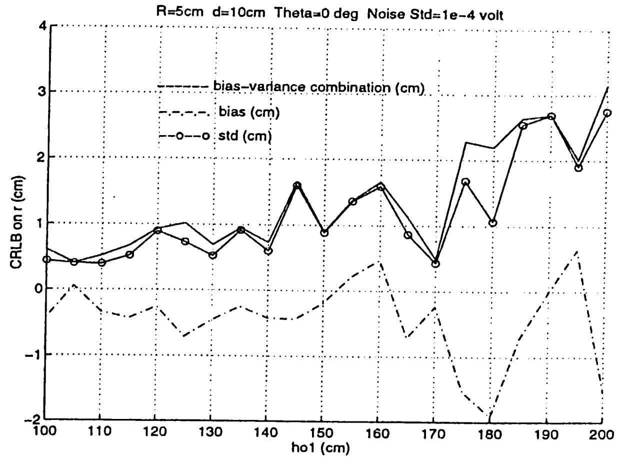


(f)

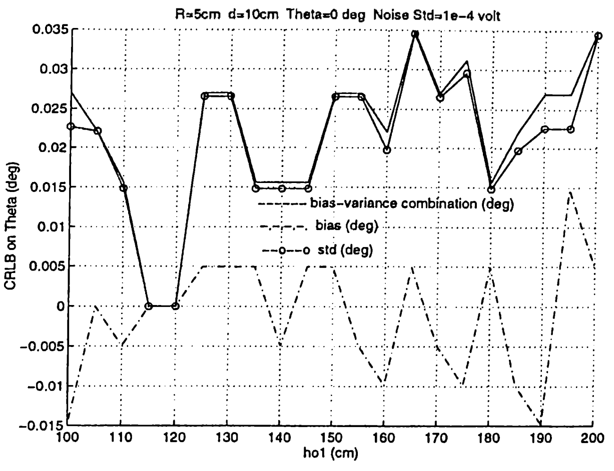
Figure 4.10: CRLB versus h_{o1}. First column is with thresholding and second column is with curve-fitting.



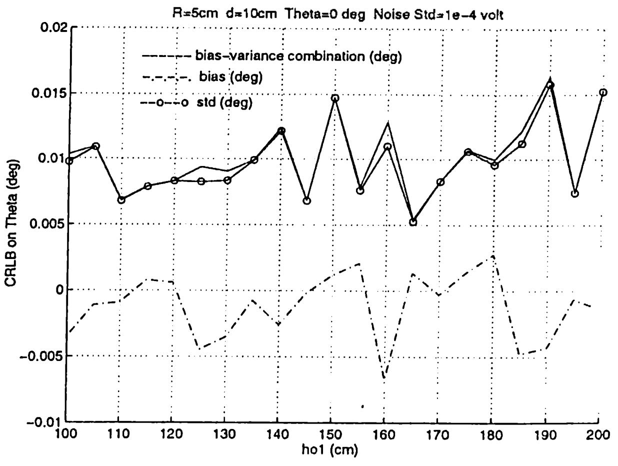
(a)



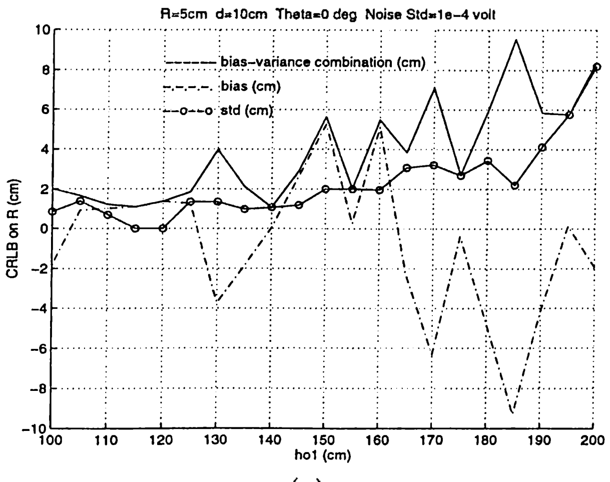
(b)



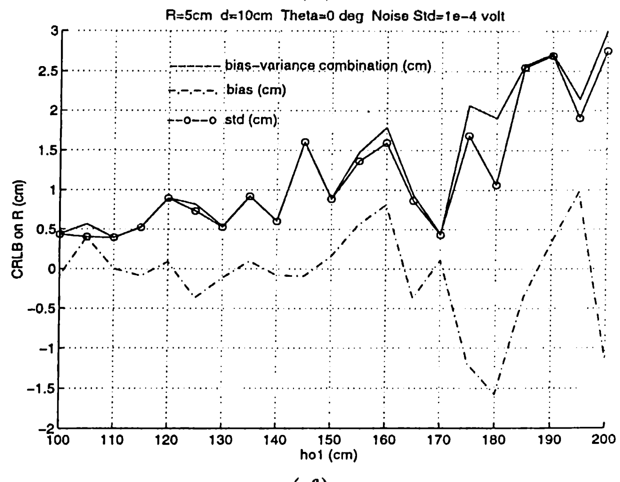
(c)



(d)



(e)



(f)

Figure 4.11: Bias, bias-variance combination, standard deviation versus h_{o1} with thresholding (first column) and curve-fitting methods (second column).

Chapter 5

SMOOTHING OF THE ESTIMATES USING EXTENDED KALMAN FILTERING

In this chapter, an *extended Kalman filter* (EKF) is used to estimate the location and radius of curvature of the target. The case in which the sensors are aligned is investigated as shown in Figure 3.1. Appendix B gives a brief summary of extended Kalman filtering. A more detailed treatment can be found in [32].

5.1 Algorithm

The following procedure is used to estimate the location and the radius of curvature of the cylindrical object.

- The state vector is defined as follows:

$$\mathrm{x}(k) \triangleq \begin{bmatrix} r(k) \\ \theta(k) \\ R(k) \end{bmatrix} \tag{5.1}$$

- The observation model is

$$\mathrm{z}(k) = \begin{bmatrix} h_{o1}(k) \\ h_{r1}(k) \\ h_{l1}(k) \end{bmatrix} = \mathrm{h}\left[\mathrm{x}(k)\right] + \mathrm{w}(k) \tag{5.2}$$

where

$$\mathrm{h}\left[\mathrm{x}(k)\right] = \begin{bmatrix} r(k) - R(k) \\ \sqrt{r^2(k) + d^2 - 2dr(k)\sin\theta(k)} - R(k) \\ \sqrt{r^2(k) + d^2 + 2dr(k)\sin\theta(k)} - R(k) \end{bmatrix} \tag{5.3}$$

- Since the target is assumed to be stationary, the state transition model is

$$\mathrm{x}(k+1) = \mathbf{F}\mathrm{x}(k) + \mathrm{v}(k) = \begin{bmatrix} r(k) \\ \theta(k) \\ R(k) \end{bmatrix} + \begin{bmatrix} v_r(k) \\ v_\theta(k) \\ v_R(k) \end{bmatrix} \tag{5.4}$$

where v_R, v_r and v_θ are the additive process noise for radius, range and angle, respectively. **F** matrix in this case is an identity matrix. Note that the state model is linear but the observation model is nonlinear.

- The Jacobian matrix **H** is found as follows:

$$\mathbf{H}(k) = \nabla\mathrm{h}(k) = \begin{bmatrix} -1 & 1 & 0 \\ -1 & \frac{r(k) - d\sin\theta(k)}{\sqrt{r(k)^2 + d^2 - 2dr(k)\sin\theta(k)}} & -\frac{dr(k)\cos\theta(k)}{\sqrt{r(k)^2 + d^2 - 2dr(k)\sin\theta(k)}} \\ -1 & \frac{r(k) + d\sin\theta(k)}{\sqrt{r(k)^2 + d^2 + 2dr(k)\sin\theta(k)}} & \frac{dr(k)\cos\theta(k)}{\sqrt{r(k)^2 + d^2 + 2dr(k)\sin\theta(k)}} \end{bmatrix} \tag{5.5}$$

where $r(k)$ and $\theta(k)$ are the predicted values of range and normal angle.

5.2 Simulation Results

Figure 5.1(a), (b) and (c) show the measurement residuals of h_{o1}, h_{r1} and h_{l1}, respectively. All three figures illustrate that innovations are white as expected. When a three degree-of-freedom chi-square test is applied, it is observed that one out of 30 points falls outside of the 97.5% confidence region. Although the maximum error allowed is 2.5%, the innovations with error 3.3% can be considered as white Gaussian distributed since only 30 iterations are processed. The details of chi-square test are given in Appendix C.

Figure 5.2 illustrates the estimated and predicted states. Figure 5.2(a), (b), (c) illustrate how the estimation of the range, normal angle and radius of curvature improve as the number of steps increase, respectively. For Figures 5.1 and 5.2, d = 10 cm, R = 5 cm, h_{o1} = 100 cm, $\theta = 0°$, measurement noise standard deviation = 10^{-6} V, the variance of the radius noise = 10^{-3} cm^2 and the variance of the angle noise =10^{-4} rad^2.

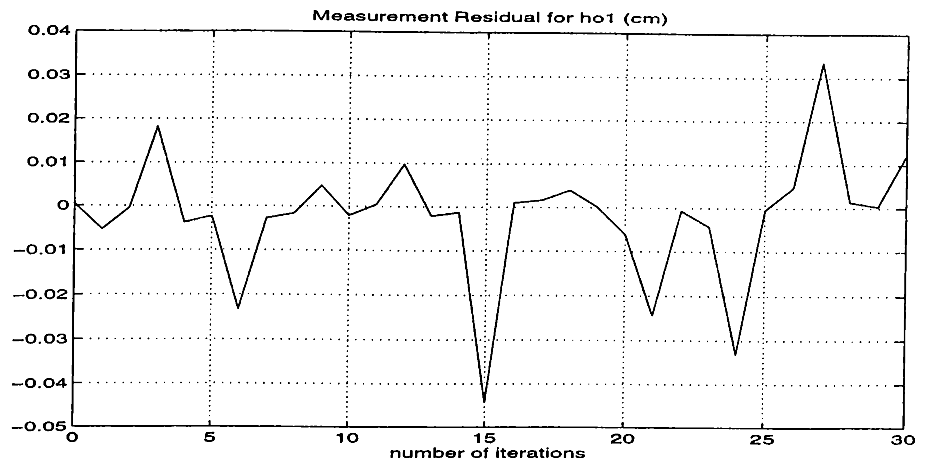
Figure 5.4(a), (b) and (c) illustrate how the range, angle and radius estimations are affected by d. For small values of d, the radius and range estimation are not accurate. The average error is about 0.8% for the range, and 10% for the radius. For Figure 5.4, R = 5 cm, h_{o1} = 100 cm, $\theta = 0°$, measurement noise standard deviation = 10^{-6} V, the variance of the radius noise = 10^{-3} cm^2 and the variance of the angle noise =10^{-4} rad^2.

Figure 5.5 presents the effect of R on the estimations. Figure 5.5(a), (b) and (c) illustrate the effect of R on the range, angle and radius, respectively. All of the three estimations are very accurate for different values of R. The errors on the range and radius of curvature are about 0.6% and 5%, respectively. For Figure 5.5, d = 10 cm, h_{o1} = 100 cm, $\theta = 0°$, measurement noise standard deviation = 10^{-6} V, the variance of the radius noise = 10^{-3} cm^2 and the variance of the radius noise =10^{-4} rad^2.

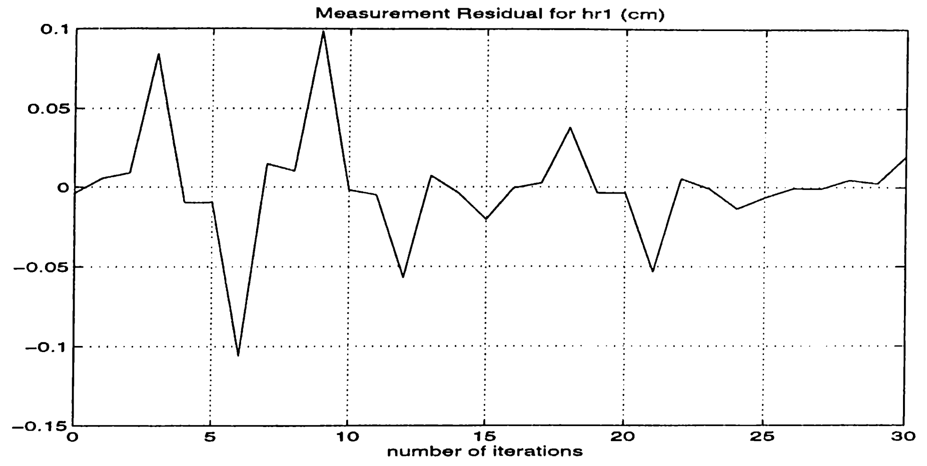
Figure 5.6 shows how h_{o1} affects the estimation. Figure 5.6(a) is drawn for the range estimation and the average error is about 1.2%. Figure 5.6(b) is for the azimuth angle estimation. Figure 5.6(c) is for the radius of curvature estimation and the average error is about 18%. For Figure 5.6, d = 10 cm, R = 5 cm, $\theta = 0°$, measurement noise standard deviation = 10^{-6} V, the variance of the radius noise = 10^{-3} cm^2 and the variance of the angle noise =10^{-4} rad^2.

Figure 5.7 displays the effect of θ on the estimation process. Figure 5.7(a), (b) and (c) illustrate how the range, angle and radius estimations are affected by θ. As θ increases, the estimations degrade a little due to increase in uncertainty on the measured values. For Figure 5.7, d = 10 cm, R = 5 cm, h_{o1} = 100 cm, measurement noise standard deviation = 10^{-6} V, the variance of the radius noise = 10^{-3} cm^2 and the variance of the angle noise =10^{-4} rad^2.

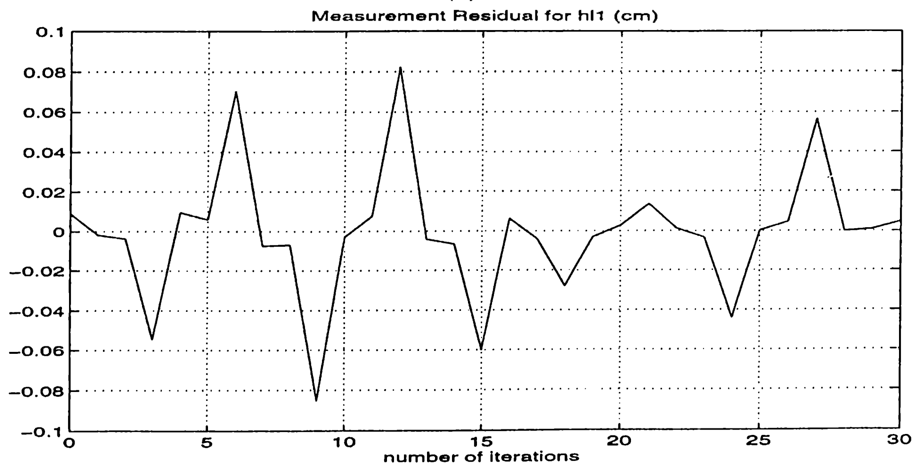
Figure 5.3 illustrates the range, angle and radius estimations respectively by using extended Kalman filtering and raw data over a single data sequence. That is, in the first case, estimates are smoothed by the EKF, in the second, estimates are directly derived from the raw data.



(a)

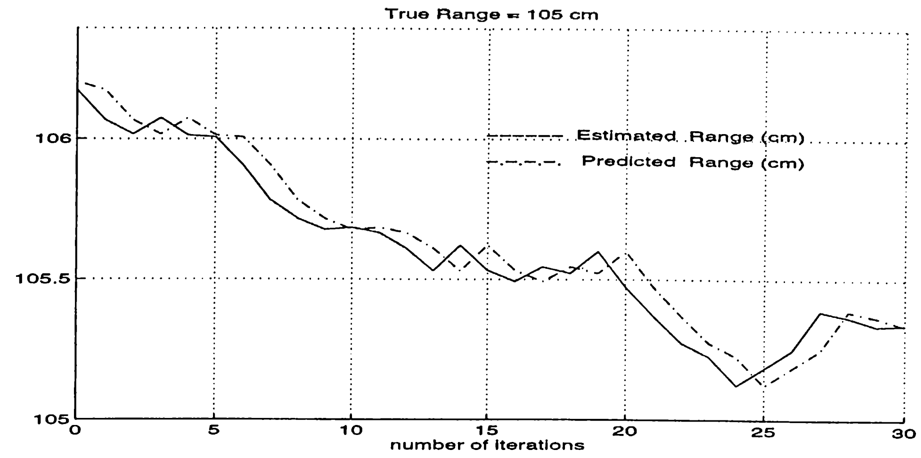


(b)

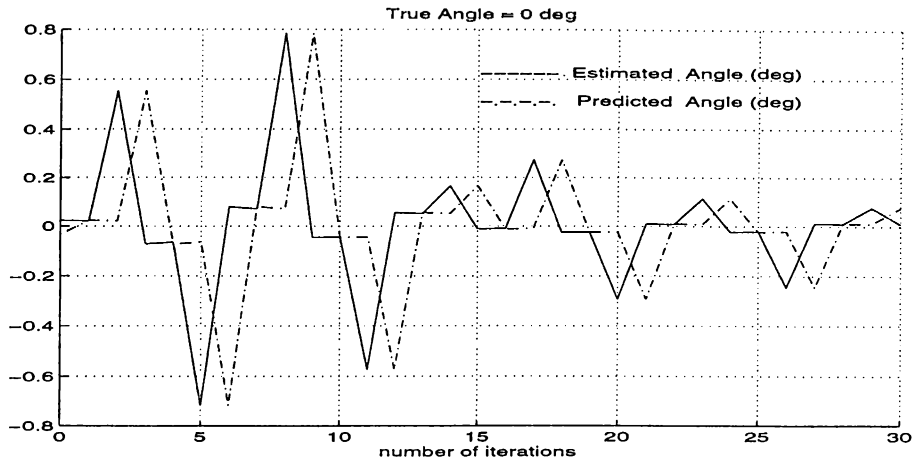


(c)

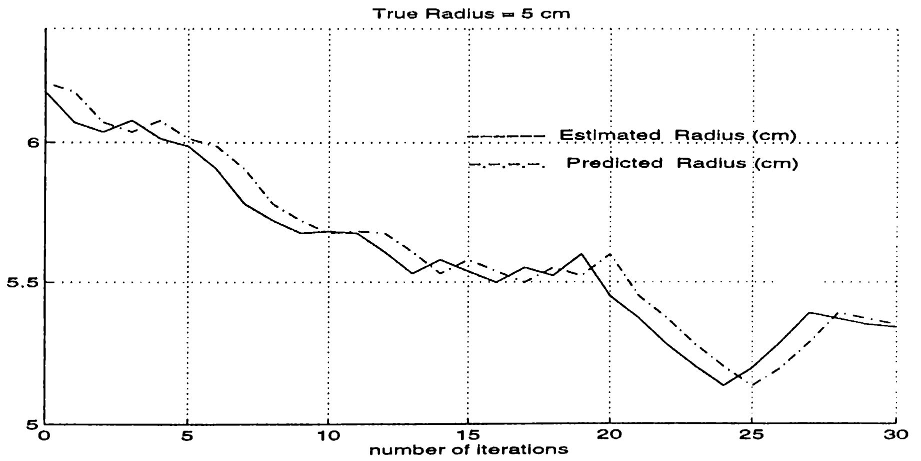
Figure 5.1: Measurement residuals of Kalman filtering for d = 10 cm, R = 5 cm, h_{o1} = 100 cm, $\theta = 0°$, measurement noise std = 10^{-6} V, variance of radius noise = 10^{-3} cm^2, variance of angle noise = 10^{-4} rad^2.



(*a*)

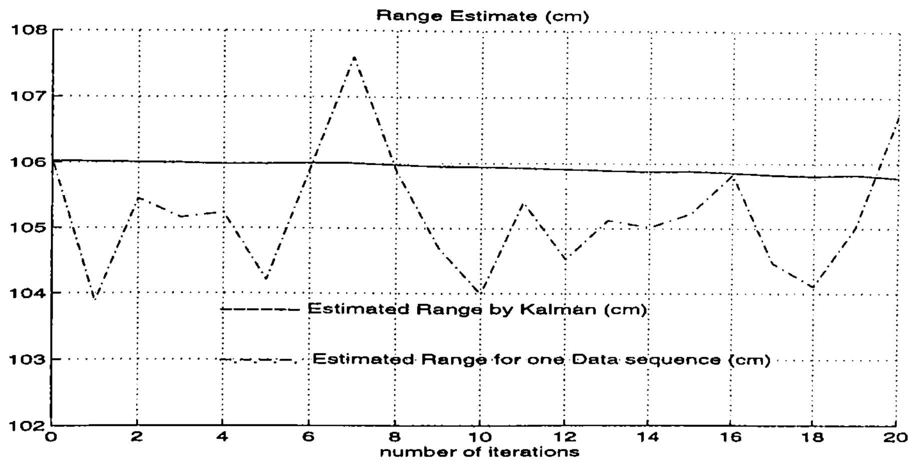


(*b*)

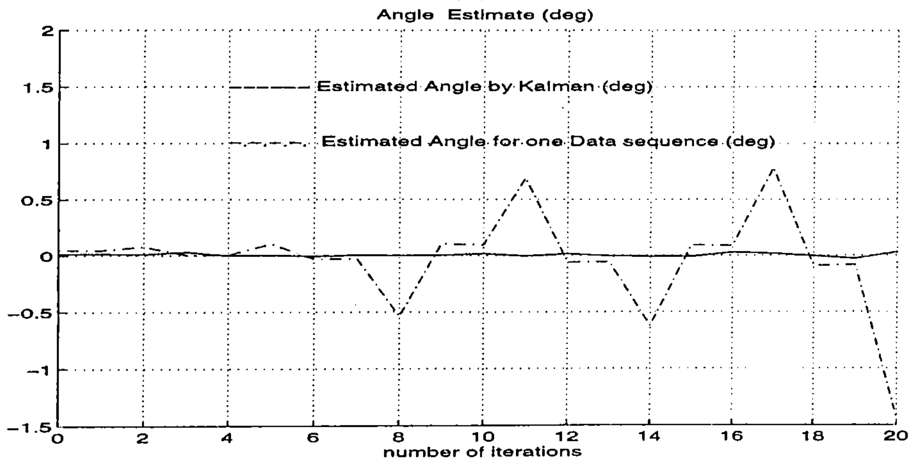


(*c*)

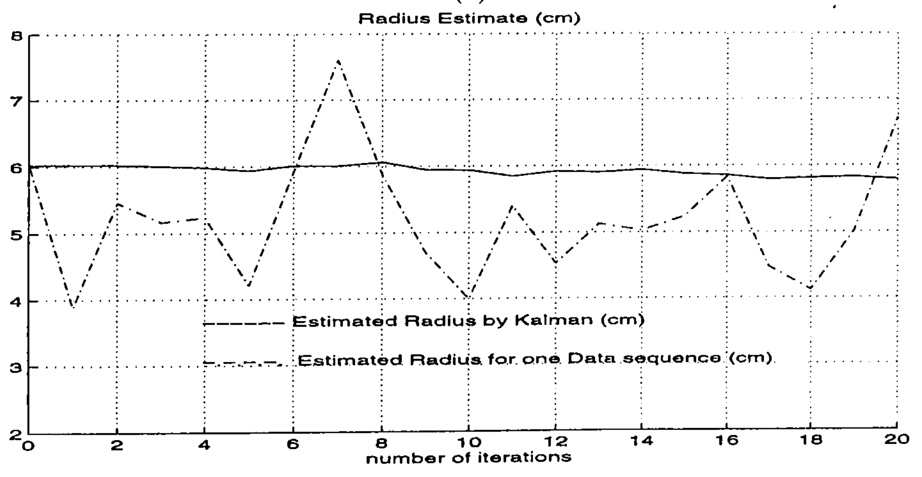
Figure 5.2: Estimated and predicted values of Kalman filtering for $d = 10$ cm, $R = 5$ cm, $h_{o1} = 100$ cm, $\theta = 0°$, measurement noise std $= 10^{-6}$ V, variance of radius noise $= 10^{-3}$ cm^2, variance of angle noise $= 10^{-4}$ rad^2.



(a)

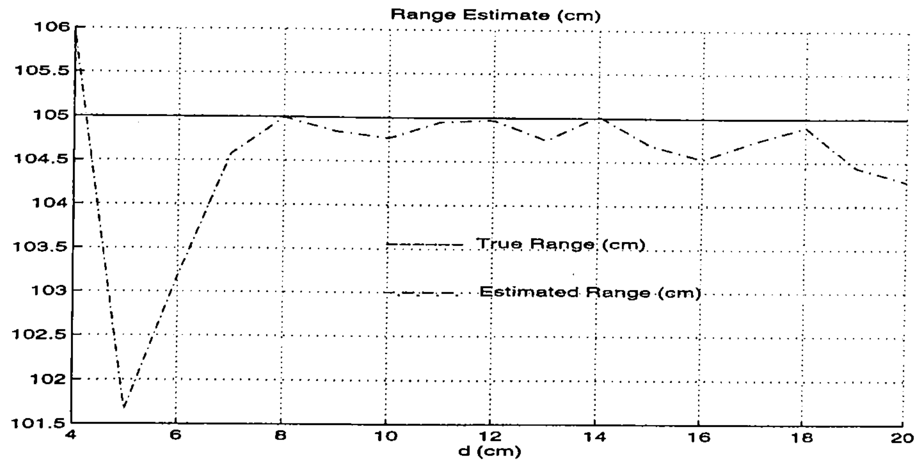


(b)

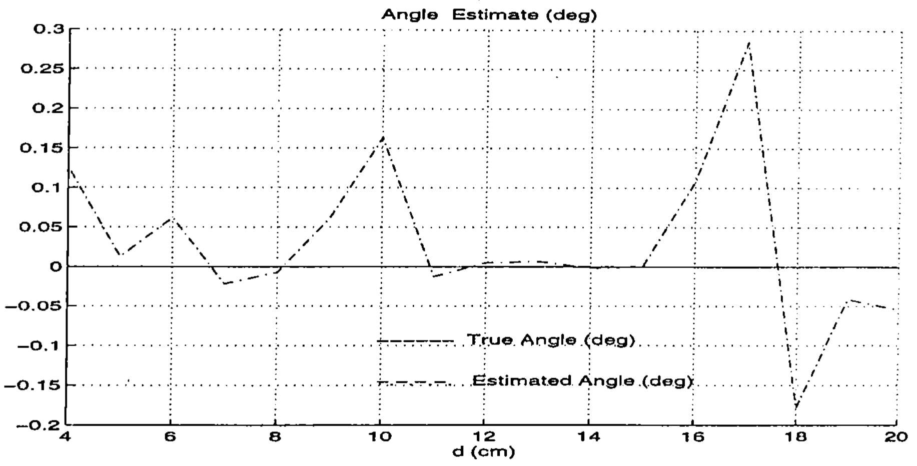


(c)

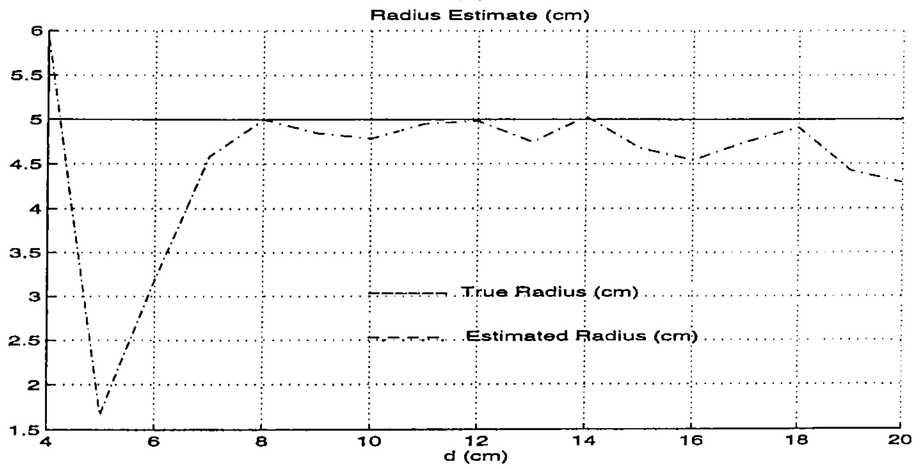
Figure 5.3: Kalman filtering and estimates by one raw data sequence versus iteration numbers for $d = 10$ cm, $R = 5$ cm, $h_{o1} = 100$ cm, $\theta = 0°$, measurement noise std $= 10^{-6}$ V, variance of radius noise $= 10^{-3}$ cm^2, variance of angle noise $= 10^{-4}$ rad^2.



(a)

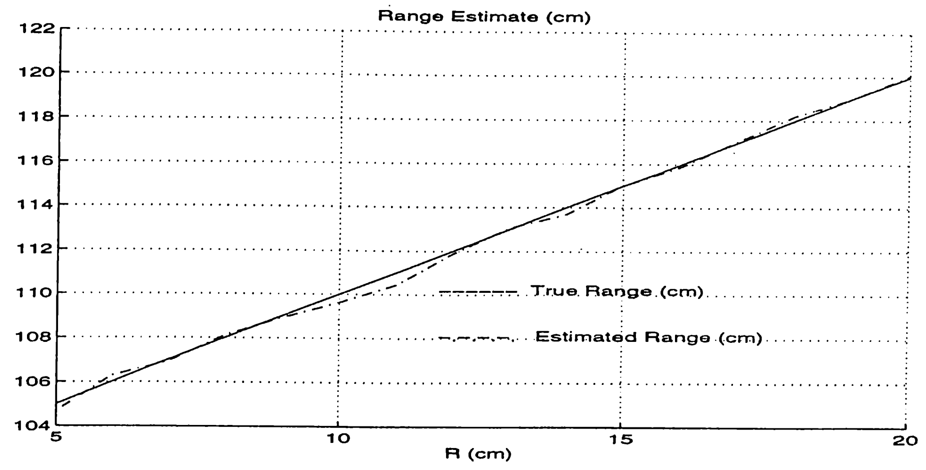


(b)

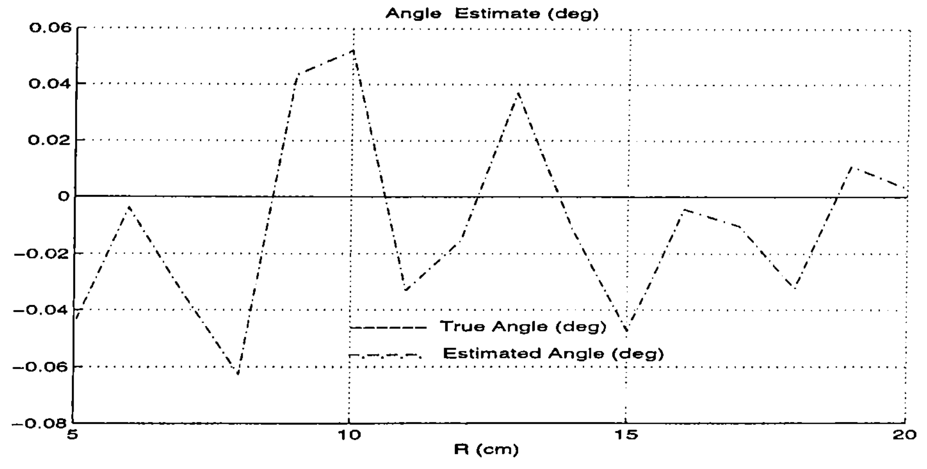


(c)

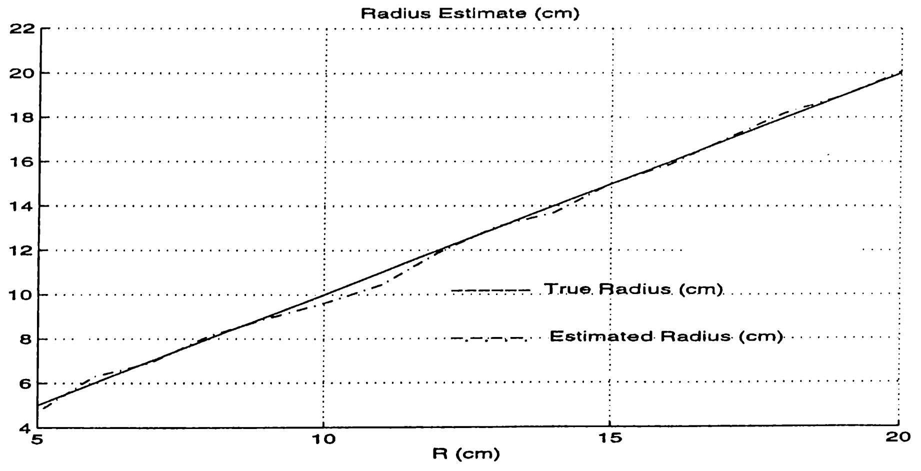
Figure 5.4: Kalman filtered R, r, θ versus d for true $R = 5$ cm, $h_{o1} = 100$ cm, $\theta = 0°$, measurement noise std $= 10^{-6}$ V, variance of radius noise $= 10^{-3}$ cm^2, variance of angle noise $= 10^{-4}$ rad^2, number of iterations $= 25$.



(a)

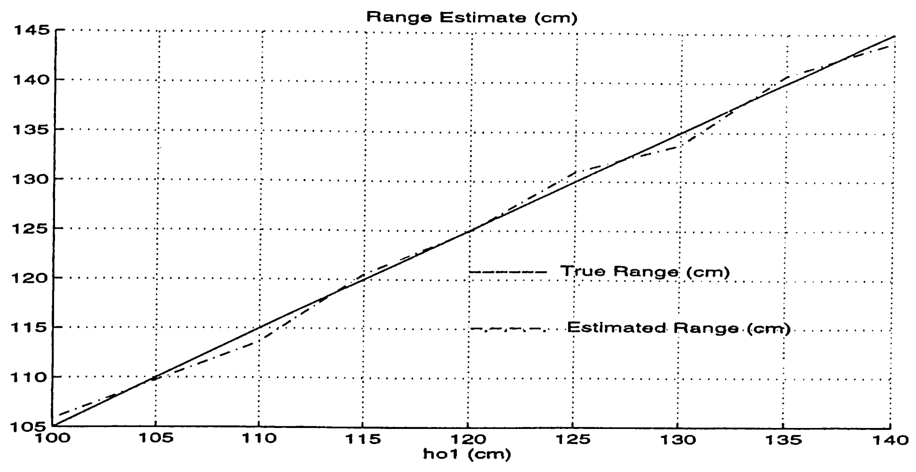


(b)

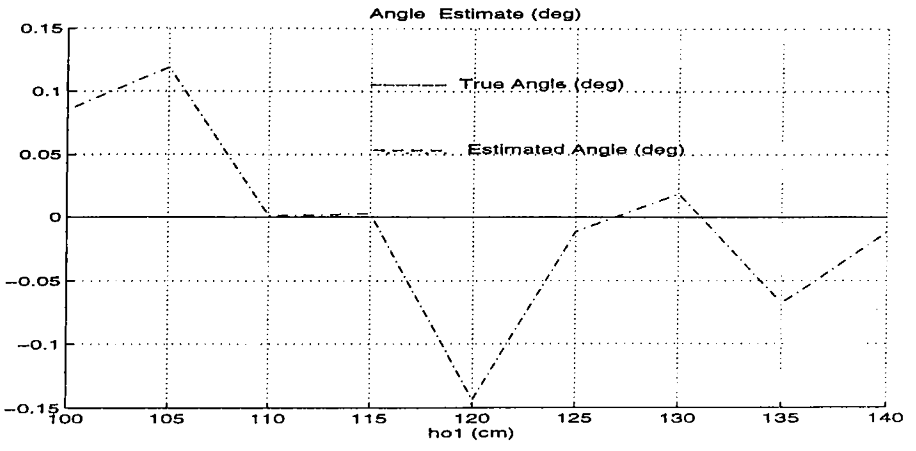


(c)

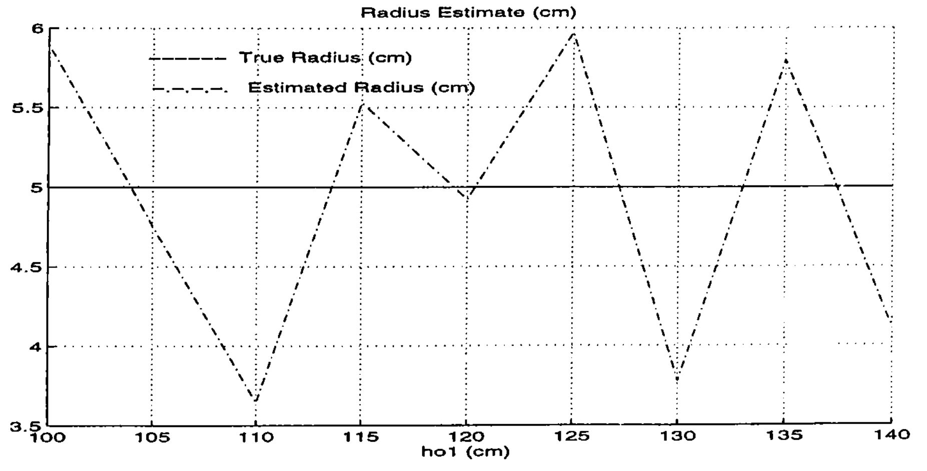
Figure 5.5: Kalman filtered R, r, θ versus true R for $d = 10$ cm, $h_{o1} = 100$ cm, $\theta = 0°$, measurement noise std $= 10^{-6}$ V, variance of radius noise $= 10^{-3}$ cm^2, variance of angle noise $= 10^{-4}$ rad^2, number of iterations $= 25$.



(a)

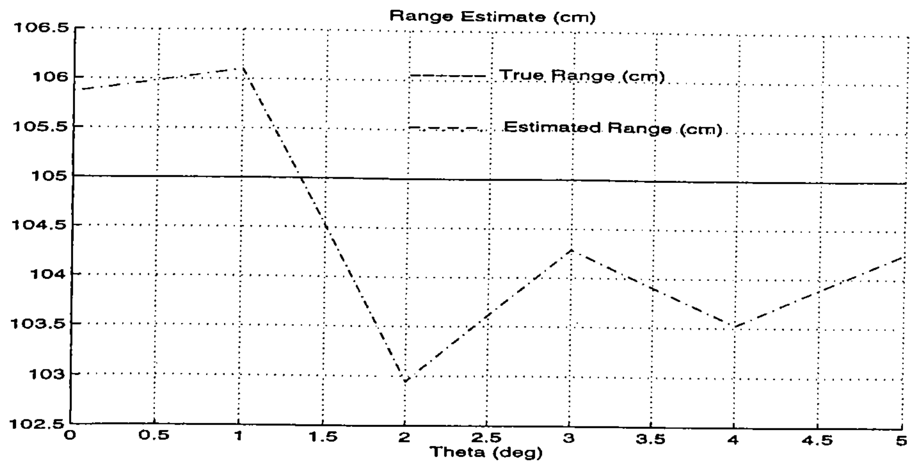


(b)

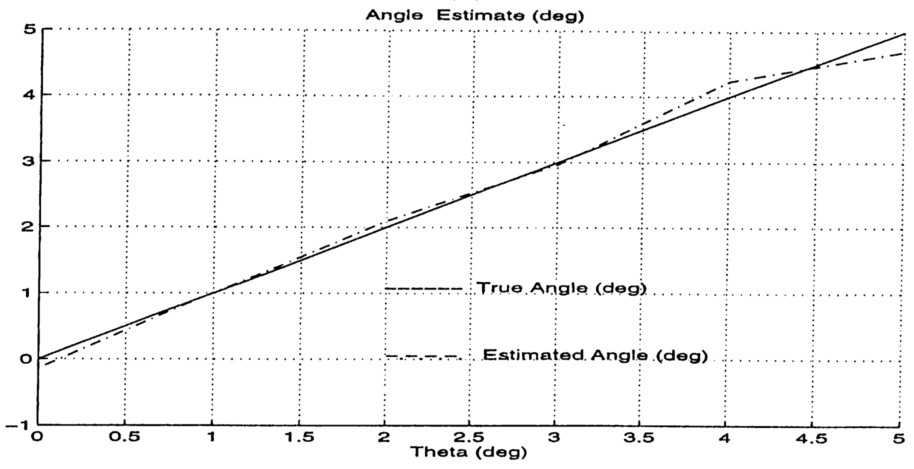


(c)

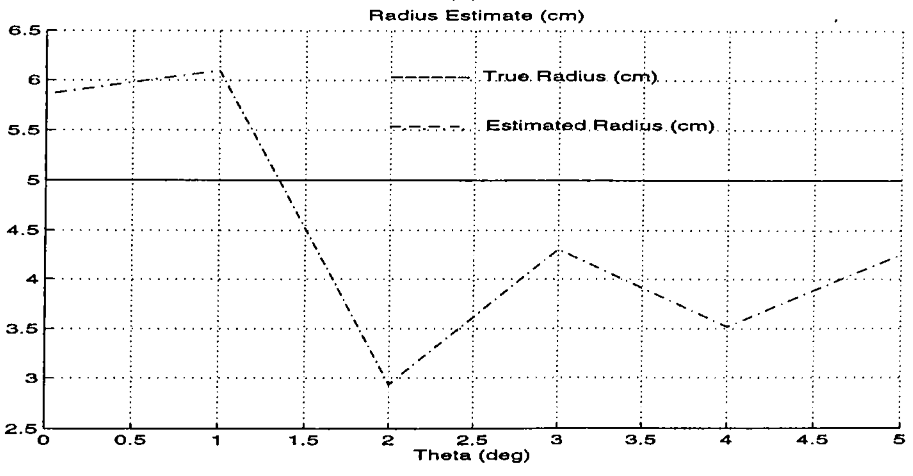
Figure 5.6: Kalman filtered R, r, θ versus h_{o1} for $R = 5$ cm $d = 10$ cm, $\theta = 0°$, measurement noise std $= 10^{-6}$ V, variance of radius noise $= 10^{-3}$ cm^2, variance of angle noise $= 10^{-4}$ rad^2, number of iterations $= 25$.



(a)



(b)



(c)

Figure 5.7: Kalman filtered R, r, θ versus true θ for $R = 5$ cm $d = 10$ cm, $h_{o1} = 100$ cm, measurement noise std $= 10^{-6}$ V, variance of radius noise $= 10^{-3}$ cm^2, variance of angle noise $= 10^{-4}$ rad^2, number of iterations $= 25$.

Chapter 6

EXPERIMENTAL RESULTS

In this chapter, two different set-ups using transducers at different frequencies are employed to verify the simulation results. In the first set-up, Panasonic transducers are used to estimate the position and radius of curvature using the algorithm mentioned in Chapter 4. In the second, Polaroid transducers are used to estimate the position and radius of curvature estimation using an EKF described in Chapter 5.

6.1 Experimental Set-up with Panasonic Transducers

The set-up is constructed for 2-D applications. Since the Panasonic transducers are manufactured separately as transmitter and receiver with different characteristics, the same transducer was not used as both transmitter and receiver. Because of this, two different transducers, with very close vertical separation, are used to transmit and receive the echoes. In this case, the pair can be considered as a transmitter/receiver transducer. The experimental set-up consists of three such pairs as shown in Figure 6.1. The aperture radius of each Panasonic transducer is $a = 0.65$ cm, the resonant frequency is $f_o = 40$ kHz and $\theta_o \cong 54°$. The block diagram for the hardware is shown in Figure 6.2.

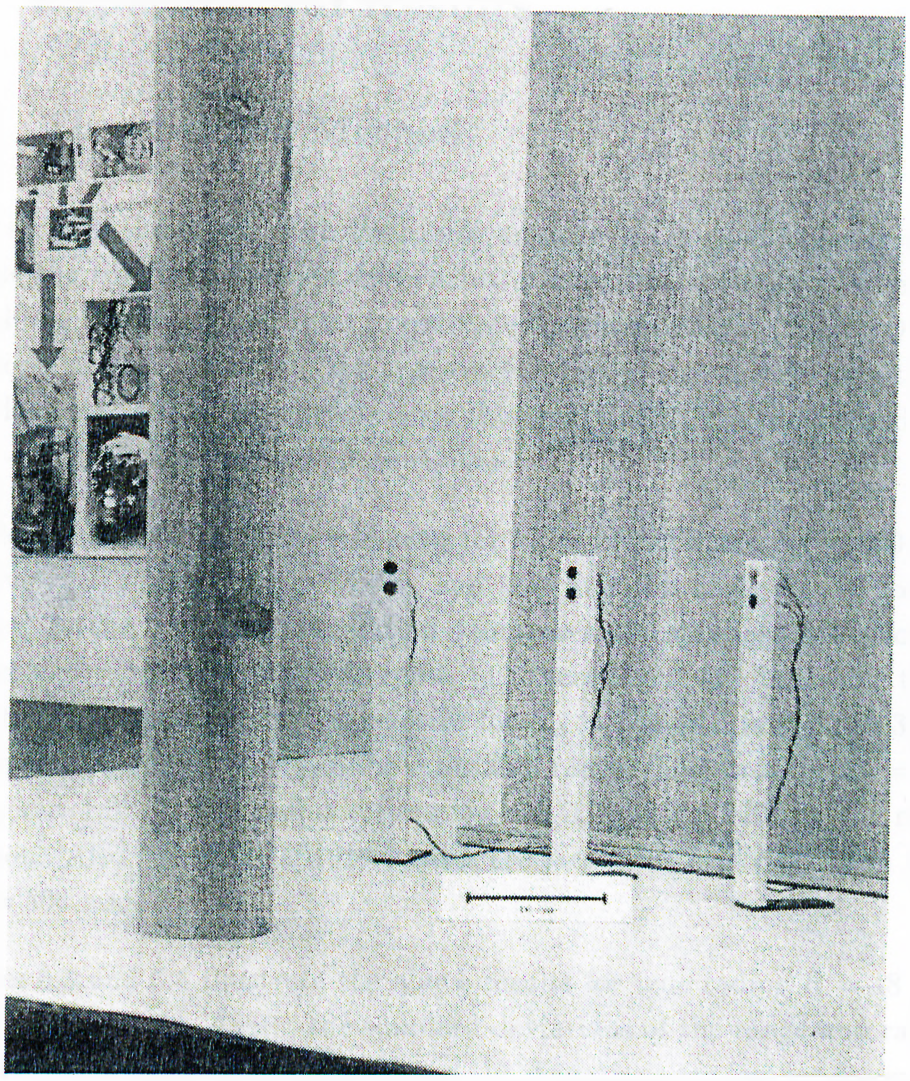


Figure 6.1: The experimental set-up consisting of Panasonic transducers.

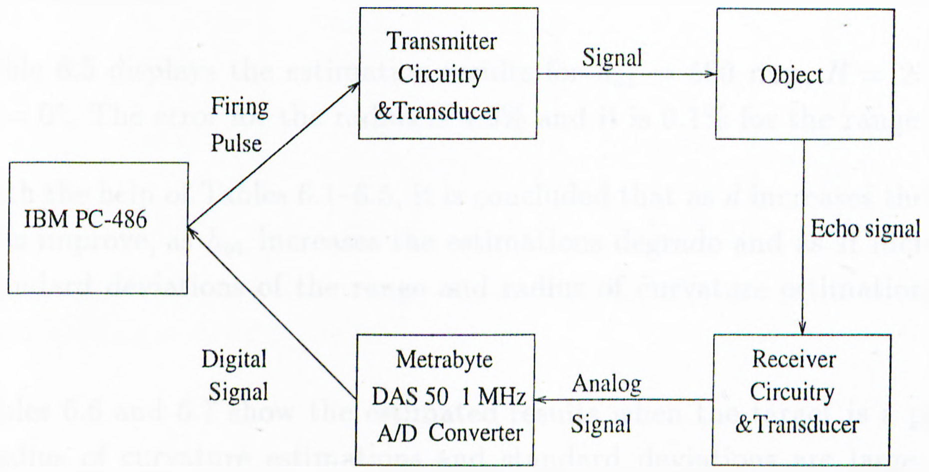

Transmitter Circuitry &Transducer
Signal
Object
Firing Pulse
IBM PC-486
Echo signal
Digital Signal
Metrabyte DAS 50 1 MHz A/D Converter
Analog Signal
Receiver Circuitry &Transducer

Figure 6.2: The block diagram of experimental set-up with Panasonic transducers.

6.2 Experimental Results

6.2.1 Thresholding Method

In this section, the set-up with Panasonic transducers is used in order to investigate the results of position and radius of curvature estimation when the thresholding method is employed to estimate the time-of-flight (TOF). During the process of data collection, the transducers' line-of-sights were maintained perpendicular to the target surface since this way, the SNR is maximized. Tables 6.1–6.8 illustrate the estimation results.

Tables 6.1 and 6.2 show the estimates for $h_{o1} = 500$ mm and $h_{o1} = 600$ mm, respectively. True radius $R = 75$ mm and true azimuth angle $\theta = 0°$ for the two cases. As the transducer separation d increases, the standard deviations of the estimated range and radius decrease, but there is not an observable trend in the standard deviation of θ. The error for $h_{o1} = 500$ mm is about 1.3% in the estimated radius and 0.9% in the estimated range. The error for $h_{o1} = 600$ mm is about 1.3% in the radius estimation but it is about 1.8% in the range estimation. Also, the standard deviations are larger for $h_{o1} = 600$ mm than for $h_{o1} = 500$ mm.

Tables 6.3 and 6.4 illustrate the above results for true radius $R = 48$ mm and true angle $\theta = 0°$. Again, the standard deviations of the range and radius decrease as the separation d increases and the deviations are less for $h_{o1} = 500$ mm. The error in the range is 0.5% for $h_{o1} = 500$ mm, and 0.8% for $h_{o1} = 600$ mm. The error in the estimated radius is 4.4% for $h_{o1} = 500$ mm, and 4.6% for $h_{o1} = 600$ mm.

Table 6.5 displays the estimation results for $h_{o1} = 500$ mm, $R = 25$ mm and $\theta = 0°$. The error for the radius is 4.0% and it is 0.1% for the range.

With the help of Tables 6.1–6.5, it is concluded that as d increases the estimations improve, as h_{o1} increases the estimations degrade and as R increases the standard deviations of the range and radius of curvature estimations decrease.

Tables 6.6 and 6.7 show the estimated results when the target is a plane. The radius of curvature estimations and standard deviations are large. By looking at the radius estimates, it is confidently concluded that the object is a plane.

Finally, Table 6.8 illustrates the effect of the azimuth angle. As the azimuth angle increases, the standard deviations tend to increase. Also, the estimates degrade as the true azimuth angle θ increases. The average error in the angle estimation is about 16%.

d (mm)	$E\{r\}$ (mm)	σ_r (mm)	$E\{\theta\}$ (deg)	σ_θ (deg)	$E\{R\}$ (mm)	σ_R (mm)
250	571.90	16.96	0.23	0.19	76.76	15.88
300	569.74	10.88	0.07	0.13	74.18	9.81
350	569.78	7.38	-0.06	0.15	73.66	6.82
400	569.73	5.27	0.11	0.19	73.89	8.09
450	570.07	4.93	-0.19	0.19	74.74	7.23

Table 6.1: Experimental results when h_{o1} = 500 mm, R = 75 mm, $\theta = 0°$ and the thresholding method is used.

d (mm)	$E\{r\}$ (mm)	σ_r (mm)	$E\{\theta\}$ (deg)	σ_θ (deg)	$E\{R\}$ (mm)	σ_R (mm)
250	661.73	15.93	0.36	0.16	73.26	15.17
300	663.84	15.15	0.21	0.10	74.87	13.90
350	661.98	13.02	-0.14	0.17	73.15	13.08
400	664.07	12.77	0.10	0.15	74.32	11.21
450	664.86	11.42	-0.43	0.15	74.45	9.91

Table 6.2: Experimental results when h_{o1} = 600 mm, R = 75 mm, $\theta = 0°$ and the thresholding method is used.

d (mm)	$E\{r\}$ (mm)	σ_r (mm)	$E\{\theta\}$ (deg)	σ_θ (deg)	$E\{R\}$ (mm)	σ_R (mm)
200	550.39	29.88	-2.10	0.41	50.18	28.77
250	548.97	8.39	2.48	0.12	48.18	7.73
300	552.19	8.50	0.01	0.20	51.50	8.18
350	549.55	5.23	-0.19	0.09	48.93	4.62
400	555.66	7.62	-0.06	0.18	55.18	6.75
450	549.97	7.06	-0.32	0.18	49.58	6.20

Table 6.3: Experimental results when h_{o1} = 500 mm, R = 48 mm, $\theta = 0°$ and the thresholding method is used.

d (mm)	$E\{r\}$ (mm)	σ_r (mm)	$E\{\theta\}$ (deg)	σ_θ (deg)	$E\{R\}$ (mm)	σ_R (mm)
200	636.97	27.52	-0.49	0.30	40.38	26.88
250	637.96	24.18	-0.56	0.33	41.79	23.01
300	645.36	16.90	-0.11	0.12	49.16	15.38
350	643.29	13.03	-0.25	0.16	47.22	11.72
400	648.86	14.04	-0.18	0.19	52.30	12.68
450	643.10	10.26	-0.21	0.22	46.99	9.34

Table 6.4: Experimental results when $h_{o1} = 600$ mm, $R = 48$ mm, $\theta = 0°$ and the thresholding method is used.

d (mm)	$E\{r\}$ (mm)	σ_r (mm)	$E\{\theta\}$ (deg)	σ_θ (deg)	$E\{R\}$ (mm)	σ_R (mm)
250	524.72	6.31	0.09	0.13	26.40	6.00
300	529.16	8.23	0.03	0.21	27.61	7.88
350	525.41	5.16	-0.18	0.13	24.03	4.64
400	522.92	7.03	0.35	0.23	24.85	6.89
450	522.91	3.85	-0.23	0.18	24.87	3.85

Table 6.5: Experimental results when $h_{o1} = 500$ mm, $R = 25$ mm, $\theta = 0°$ and the thresholding method is used.

d (mm)	$E\{r\}$ (mm)	σ_r (mm)	$E\{\theta\}$ (deg)	σ_θ (deg)	$E\{R\}$ (mm)	σ_R (mm)
200	2824.56	789.76	0.10	1.95	2344.87	789.74
250	2527.98	270.57	-0.98	0.17	2048.72	266.34
300	1531.40	476.66	-0.40	0.90	1051.84	475.77

Table 6.6: Experimental results when the thresholding method is used to estimate the radius of curvature of a plane at $h_{o1} = 500$ mm and $\theta = 0°$.

d (mm)	$E\{r\}$ (mm)	σ_r (mm)	$E\{\theta\}$ (deg)	σ_θ (deg)	$E\{R\}$ (mm)	σ_R (mm)
200	3122.65	860.69	-0.01	4.12	2545.12	848.52
250	2561.10	693.35	-0.98	1.78	1981.51	688.54
300	1354.48	480.81	-1.20	3.14	775.72	476.15

Table 6.7: Experimental results when the thresholding method is used to estimate the radius of curvature of a plane at $h_{o1} = 600$ mm and $\theta = 0°$.

θ (deg)	$E\{r\}$ (mm)	σ_r (mm)	$E\{\theta\}$ (deg)	σ_θ (deg)	$E\{R\}$ (mm)	σ_R (mm)
0	522.92	7.03	0.35	0.23	24.85	6.89
3	522.96	3.19	2.53	0.10	24.65	0.35
5	522.66	4.79	4.11	0.19	24.43	4.76
8	524.12	6.02	6.76	0.18	22.98	5.89

Table 6.8: Experimental results when the thresholding method is used to estimate the radius of curvature of a cylinder at $h_{o1} = 500$ mm, $d = 400$ mm and $R = 25$ mm.

6.2.2 Curve-fitting Method

In this section, the experimental results of estimation with the curve-fitting method are presented. The set-up with Panasonic transducers is used and the transducers are rotated around their centers. Tables 6.9–6.16 summarize the experimental results.

Table 6.9 illustrates the estimation results for $h_{o1} = 500$ mm, $R = 75$ mm and $\theta = 0°$. As d increases, the standard deviations decrease for the range and radius. The average error for the range is about 0.9% and it is 1.1% for the radius. Table 6.10 shows the results for $h_{o1} = 600$ mm, $R = 75$ mm and $\theta = 0°$. Again as d increases, the standard deviations decrease. The range and radius errors are about 1.4% and 2.7%, respectively.

When Tables 6.1, 6.2 and 6.9, 6.10 are compared, it is observed that the standard deviations of the range, radius and angle estimates are comparable for the thresholding and curve-fitting methods.

Table 6.11 and 6.12 illustrates the results of $r = 48$ mm and $\theta = 0°$ for $h_{o1} = 500$ mm and $h_{o1} = 600$ mm, respectively. As d increases, standard deviations decrease and they are comparable to the results of the thresholding method (Tables 6.3 and 6.4). The error in the range is 0.4% for $h_{o1} = 500$ mm and 0.7% for $h_{o1} = 600$ mm. The error in the radius is 3.5% for $h_{o1} = 500$ mm and 6.9% for $h_{o1} = 600$ mm.

Table 6.13 displays the effect of d for $h_{o1} = 500$ mm, $R = 25$ mm and $\theta = 0°$. The error for the radius is 3.3%, and it is 0.5% for the range.

Tables 6.14 and 6.15 illustrate the estimation results when the target is a plane. The radius of curvature estimations and standard deviations are huge. As in the thresholding case, it is concluded that the target is a plane.

Table 6.16 shows the effect of the inclination angle. As the angle increases, the standard deviations increase. Also, the estimates degrade as the true inclination angle θ increases. The average error in the angle estimation is about 11.4%.

d (mm)	$E\{r\}$ (mm)	σ_r (mm)	$E\{\theta\}$ (deg)	σ_θ (deg)	$E\{R\}$ (mm)	σ_R (mm)
250	569.47	15.64	0.24	0.18	74.71	14.60
300	571.40	12.94	0.05	0.15	76.71	11.64
350	570.58	8.04	-0.09	0.15	74.59	7.25
400	569.45	8.75	0.12	0.18	73.57	8.02
450	569.51	8.36	-0.21	0.18	74.30	7.17

Table 6.9: Experimental results when $h_{o1} = 500$ mm, $R = 75$ mm, $\theta = 0°$ and the curve-fitting method is used.

d (mm)	$E\{r\}$ (mm)	σ_r (mm)	$E\{\theta\}$ (deg)	σ_θ (deg)	$E\{R\}$ (mm)	σ_R (mm)
250	664.49	21.71	0.13	0.24	75.80	20.85
300	664.04	16.43	0.23	0.13	75.14	15.23
350	667.68	17.49	-0.16	0.15	77.44	15.78
400	665.78	14.47	0.08	0.19	75.70	12.87
450	666.00	12.06	-0.40	0.20	75.40	10.53

Table 6.10: Experimental results when $h_{o1} = 600$ mm, $R = 75$ mm, $\theta = 0°$ and the curve-fitting method is used.

d (mm)	$E\{r\}$ (mm)	σ_r (mm)	$E\{\theta\}$ (deg)	σ_θ (deg)	$E\{R\}$ (mm)	σ_R (mm)
200	550.93	30.18	-2.25	0.44	50.64	29.27
250	548.92	8.56	2.45	0.13	48.15	8.01
300	550.10	10.62	-0.05	0.21	49.69	10.01
350	549.22	6.56	-0.20	0.11	48.81	5.89
400	556.30	8.05	-0.11	0.21	55.92	7.35
450	549.95	7.39	-0.31	0.19	49.80	6.48

Table 6.11: Experimental results when $h_{o1} = 500$ mm, $R = 48$ mm, $\theta = 0°$ and the curve-fitting method is used.

d (mm)	$E\{r\}$ (mm)	σ_r (mm)	$E\{\theta\}$ (deg)	σ_θ (deg)	$E\{R\}$ (mm)	σ_R (mm)
200	635.59	25.49	-0.60	0.25	38.99	24.81
250	645.27	29.32	-0.52	0.31	49.20	28.24
300	644.99	17.84	-0.12	0.13	48.92	16.29
350	648.39	14.86	-0.27	0.16	51.74	13.44
400	649.20	12.85	-0.16	0.24	52.76	11.83
450	643.41	12.84	-0.24	0.22	47.16	11.63

Table 6.12: Experimental results when $h_{o1} = 600$ mm, $R = 48$ mm, $\theta = 0°$ and the curve-fitting method is used.

d (mm)	$E\{r\}$ (mm)	σ_r (mm)	$E\{\theta\}$ (deg)	σ_θ (deg)	$E\{R\}$ (mm)	σ_R (mm)
250	520.75	8.96	0.06	0.17	22.56	8.49
300	524.91	11.04	0.01	0.24	23.64	10.34
350	526.53	8.63	-0.15	0.17	25.30	7.53
400	522.81	6.76	0.39	0.26	24.76	6.66
450	520.93	5.73	-0.29	0.12	23.46	4.91

Table 6.13: Experimental results when $h_{o1} = 500$ mm, $R = 25$ mm, $\theta = 0°$ and the curve-fitting method is used.

d (mm)	$E\{r\}$ (mm)	σ_r (mm)	$E\{\theta\}$ (deg)	σ_θ (deg)	$E\{R\}$ (mm)	σ_R (mm)
200	2860.72	559.80	0.10	0.47	2381.13	559.73
250	2471.04	465.58	-0.85	1.09	1991.88	462.32
300	1610.82	99.42	-0.27	0.37	1130.96	99.40

Table 6.14: Experimental results when the curve-fitting method is used to estimate the radius of curvature of a plane at $h_{o1} = 500$ mm and $\theta = 0°$.

d (mm)	$E\{r\}$ (mm)	σ_r (mm)	$E\{\theta\}$ (deg)	σ_θ (deg)	$E\{R\}$ (mm)	σ_R (mm)
200	3503.52	4330.69	-0.31	0.79	2924.89	4929.26
250	2644.16	685.01	-0.54	1.76	2065.07	678.08
300	1467.02	103.38	-1.22	0.21	886.53	101.96

Table 6.15: Experimental results when the curve-fitting method is used to estimate the radius of curvature of a plane at $h_{o1} = 600$ mm and $\theta = 0°$.

θ (deg)	$E\{r\}$ (mm)	σ_r (mm)	$E\{\theta\}$ (deg)	σ_θ (deg)	$E\{R\}$ (mm)	σ_R (mm)
0	522.81	6.76	0.39	0.26	24.76	6.66
3	522.73	4.35	2.52	0.14	24.50	4.10
5	524.19	5.54	4.20	0.21	26.01	5.30
8	525.91	6.41	6.81	0.20	24.71	6.19

Table 6.16: Experimental results when the curve-fitting method is used to estimate the radius of curvature of a cylinder at $h_{o1} = 500$ mm, $d = 400$ mm and $r = 25$ mm.

REMARKS:

- When the transducer separation d exceeds 20–22 cm, the transducer cannot observe the object with the linear configuration. Because of this, all experimental data is collected when the object and the transducers are perpendicular to each other.
- The transducers are maintained perpendicular to the object surface while experimental data is taken.

6.3 Experimental Set-up with Polaroid Transducers

The set-up is constructed for 3-D sonar applications. The system consists of five Polaroid transducers, each of which can be used both as transmitter or receiver. One of them is at the center, and the others symmetrically surround the central sensor as shown in Figure 6.3(a). The separation between the fixed central sensor and each surrounding sensor can be adjusted manually between 7.5 – 12 cm. The central sensor can be moved backward and forward, so that the others are perpendicular to the target, by a single stepper motor, as shown in Figure 6.3(b). A 12-bit 4 channel A/D converter, Metrabyte DAS-50 1MHz, samples the analog signals reflected by the target. The aperture radius of the transducers is $a = 2$ cm, the resonant frequency $f_o = 49.4$ kHz, and $\theta_o \cong 12°$.

6.3.1 Experimental Results with the EKF

In this section, the simulation results of the EKF for the position and radius of curvature estimation are verified using the set-up with Polaroid transducers.

Figures 6.4 and 6.5 illustrate the Kalman filtered results of the experimental data for $R = 5$ cm, $d = 7.5$ cm, $h_{o1} = 100$ cm and $\theta = 0°$. $R = 5.1$ cm, $h_{o1} = 99$ cm and $\theta = 0.1°$ are taken as the initial conditions. Figures 6.4(a),(b) and (c) show the innovations for h_{o1}, h_{r1} and h_{l1} respectively. Figure 6.5(a),(b) and (c) illustrate that the estimates for range, angle and radius are close to the real values. When a three degree-of-freedom chi-square test is applied, it is observed that four out of 30 points (the error is 13.3%) falls outside the 97.5% confidence region which means the innovations are not white Gaussian distributed.

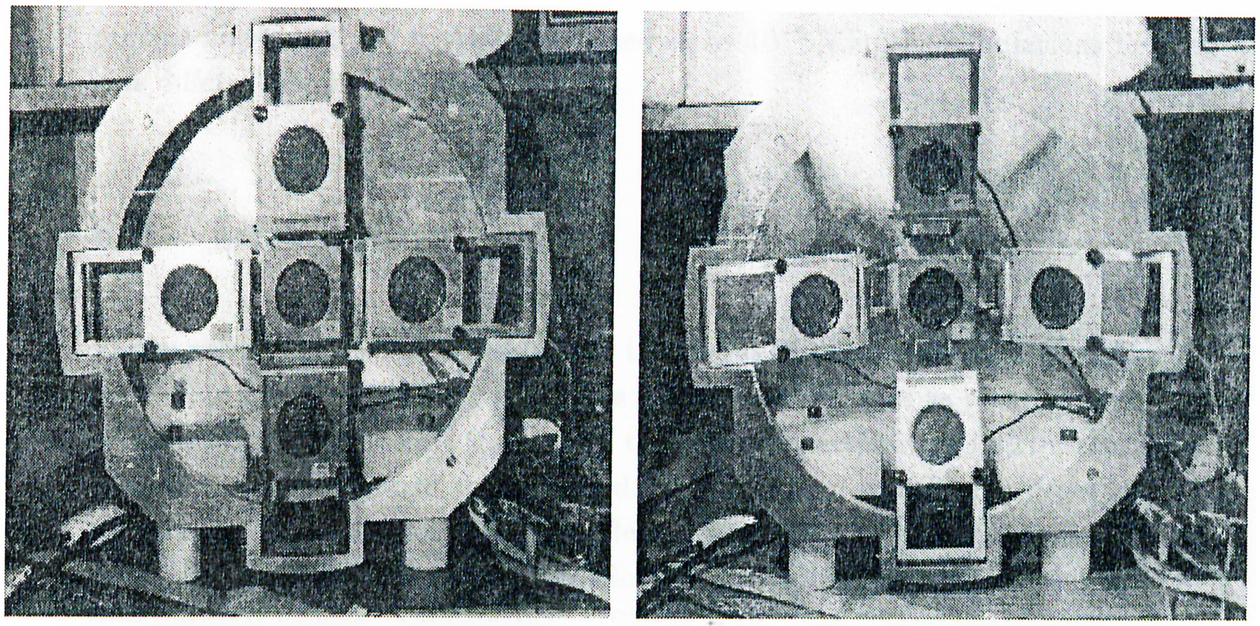

(a) (b)

Figure 6.3: Two extreme positions of the sensing device.

Figures 6.6 and 6.7 display the Kalman filtered results of the experimental data for $R = 10$ cm, $d = 7.5$ cm, $h_{o1} = 100$ cm and $\theta = 0°$. $R = 10.2$ cm, $h_{o1} = 99$ cm and $\theta = 0.1°$ are taken as the initial conditions. Figures 6.6(a) and (c) illustrate that the innovations for h_{o1} and h_{l1} are white sequences, but Figure 6.6(b) shows that the innovation for h_{r1} is not a white sequence. Figure 6.7 illustrates that the estimated values do not converge to the real values. Since all of the innovations are not white sequences, this result is expected. When a three degree-of freedom chi-square test is applied, it is observed that six out of 30 points (the error is 20%) fall outside the 97.5% confidence region, that is, the innovations are not white Gaussian distributed.

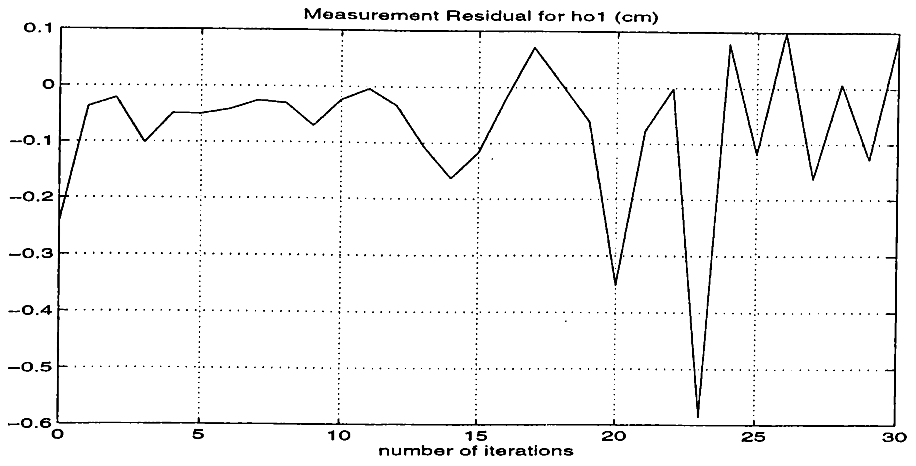
Figures 6.8 and 6.9 illustrate the Kalman filtered results of the experimental data for $R = 5$ cm, $d = 7.5$ cm, $h_{o1} = 140$ cm and $\theta = 0°$. $R = 5.1$ cm, $h_{o1} = 139$ cm and $\theta = 0.1°$ are taken as the initial conditions. In Figure 6.8, all of the innovations seem to be white sequences. Also, the estimated values are close to the true values. When a three degree-of freedom chi-square test is applied, it is observed that one out of 30 points fall outside the 97.5% confidence region and the innovations are white Gaussian distributed.

When the simulation results (Figures 5.2) and the experimental results (Figure 6.5) are compared, a huge difference is observed. There are two main reasons for this: First, the curve-fitting method is used for the simulations and the thresholding method is used for the experimental results. Second, the process noise standard deviation is tuned to be 10^{-6} V in the simulations but it is different for the real data.

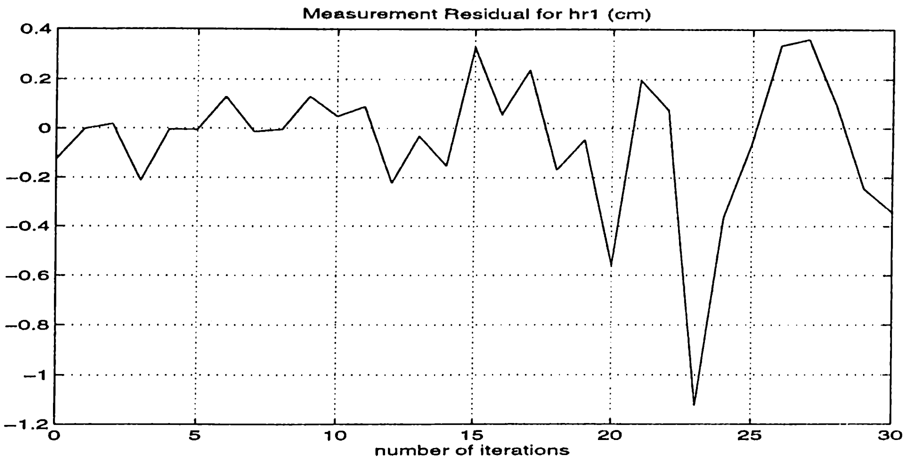
6.4 Discussion

The experimental results indicate that the radius of curvature estimation can be used to classify the target primitives into three categories as plane, edge or cylinder. In order to assess the type of the object by estimating its curvature, all reflectors are assumed to be curved. A continuum of reflector types are considered, $R = 0$ for an edge-like reflector going all the way up to $R = \infty$ for a plane-like reflector. The classification procedure, consistent with the experimental results, is illustrated in Figure 6.10. The uncertainty region of each radius estimate is considered to be between $[R - 3\sigma_R, R + 3\sigma_R]$ assuming zero mean Gaussian distributed estimation error. In the figure, σ_R changes with the type of object (reflected signal depends on the type of the object) and seems to increase with radius of curvature.

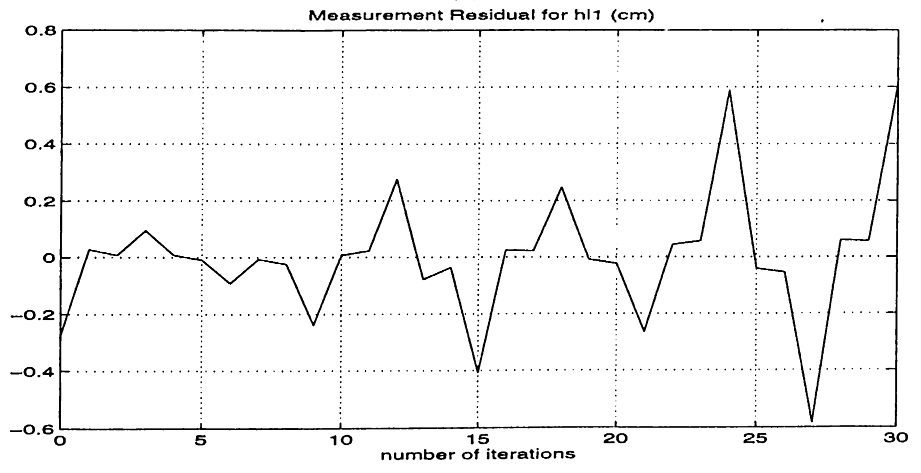
Given two targets with constant curvature, if there is overlap between their uncertainty regions, then these targets may not be distinguished for estimates which fall within the overlap region.



(a)

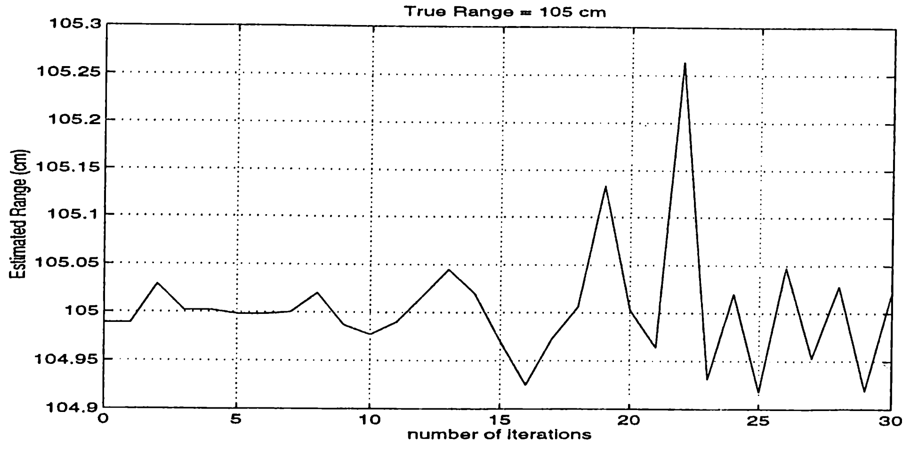


(b)

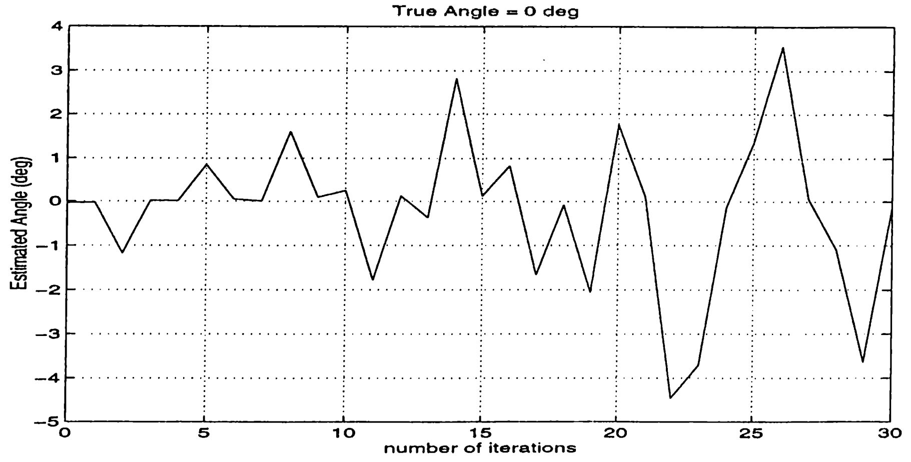


(c)

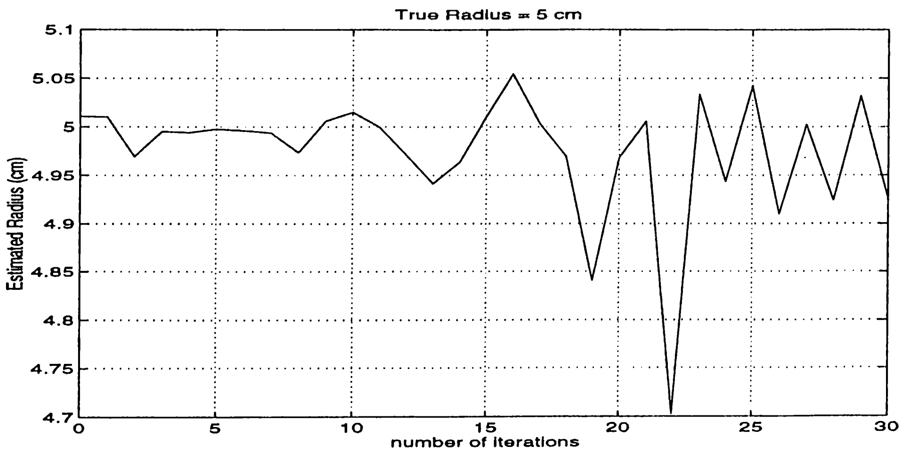
Figure 6.4: Measurement residuals of the Kalman filter for $d = 7.5$ cm, $R = 5$ cm, $h_{o1} = 100$ cm, $\theta = 0°$, variance of radius noise $= 10^{-3}$ cm^2, variance of angle noise $= 10^{-4}$ rad^2, initial estimates $R = 5.1$ cm, $h_{o1} = 99$ cm and $\theta = 0.1°$.



(a)

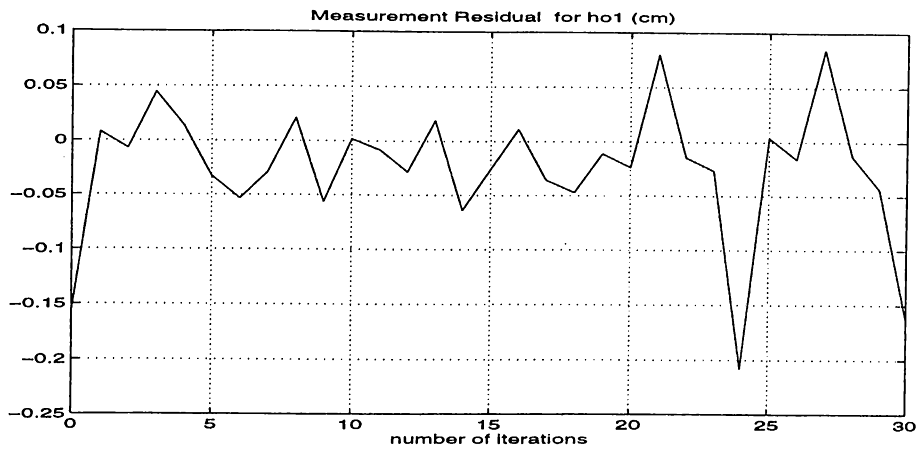


(b)

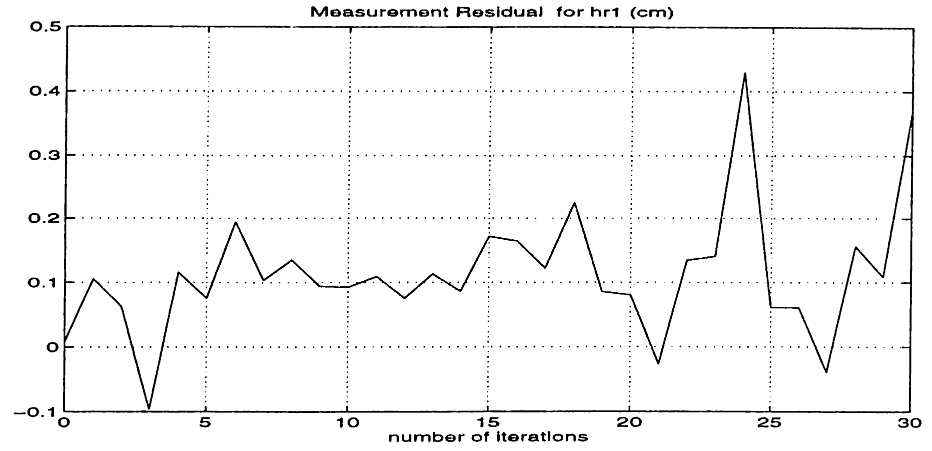


(c)

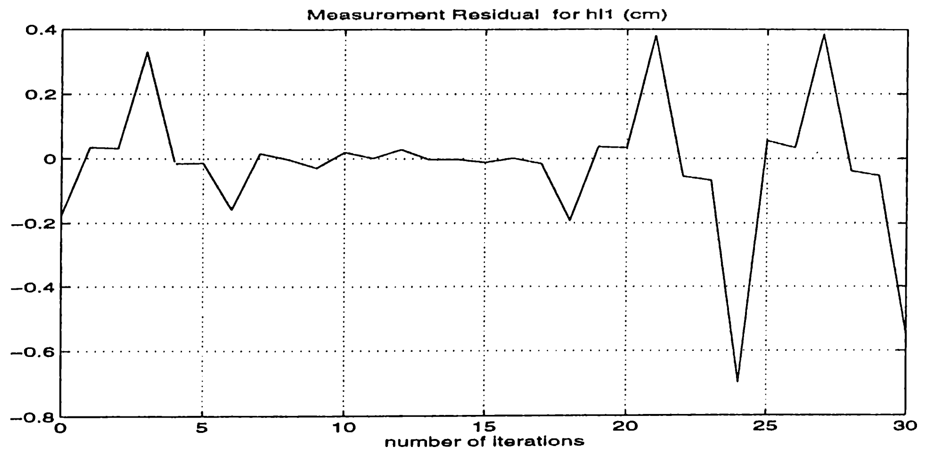
Figure 6.5: Estimates of the Kalman filter for $d = 7.5$ cm, $R = 5$ cm, $h_{o1} = 100$ cm, $\theta = 0°$, variance of radius noise $= 10^{-3}$ cm^2, variance of angle noise $= 10^{-4}$ rad^2, initial estimates $R = 5.1$ cm, $h_{o1} = 99$ cm and $\theta = 0.1°$.



(a)

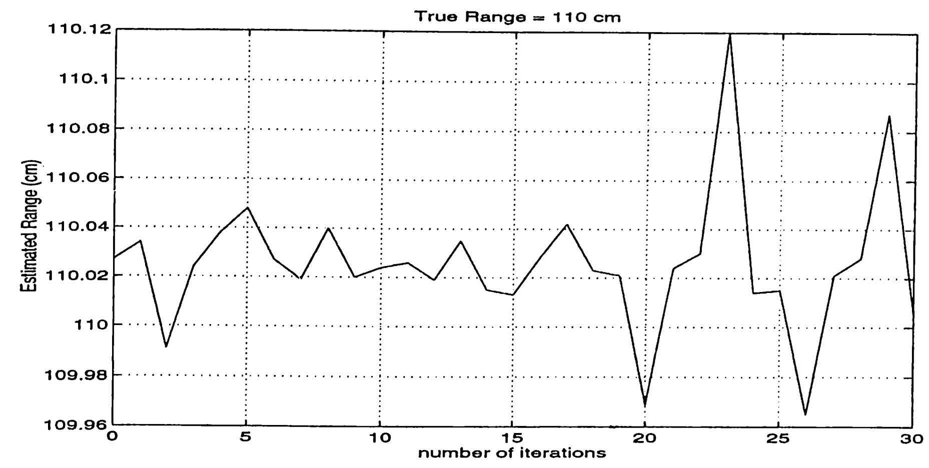


(b)

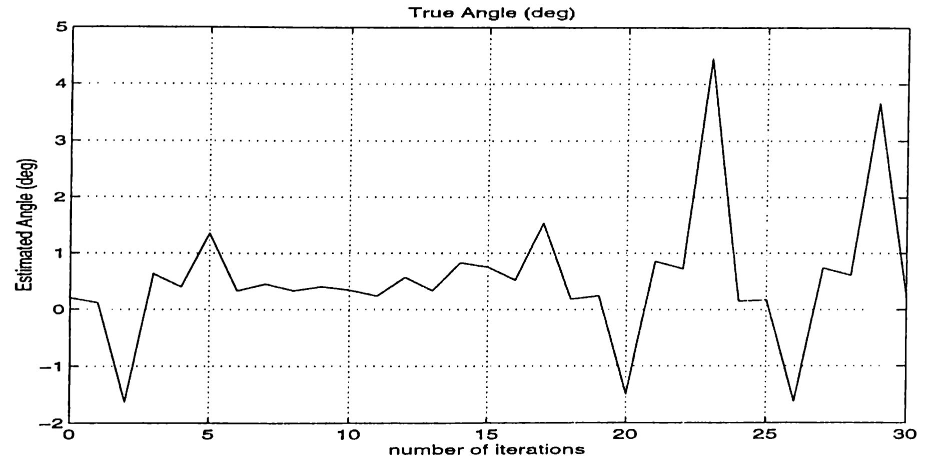


(c)

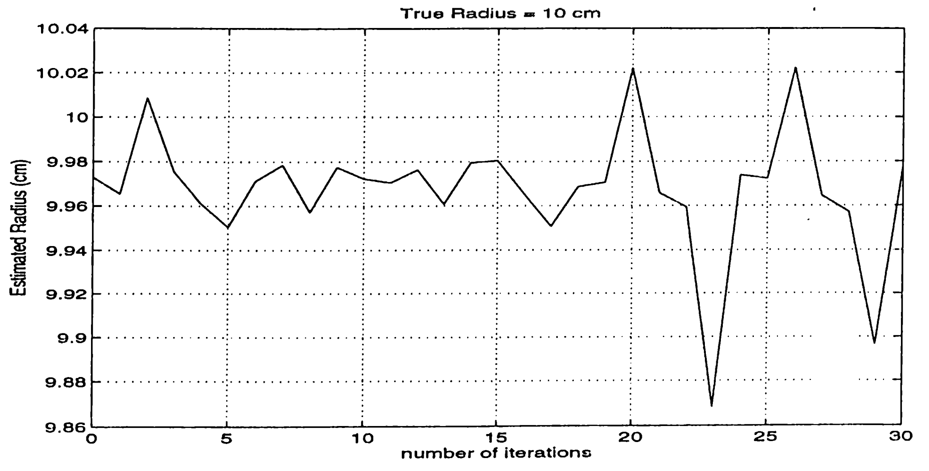
Figure 6.6: Measurement residuals of the Kalman filter for $d = 7.5$ cm, $R = 10$ cm, $h_{o1} = 100$ cm, $\theta = 0°$, variance of radius noise $= 10^{-3}$ cm^2, variance of angle noise $= 10^{-4}$ rad^2, initial estimates $R = 10.1$ cm, $h_{o1} = 99$ cm and $\theta = 0.1°$.



(a)

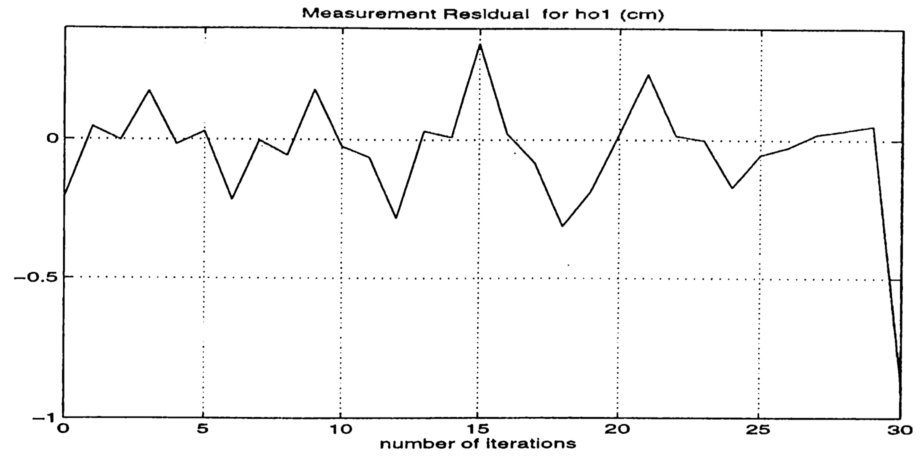


(b)

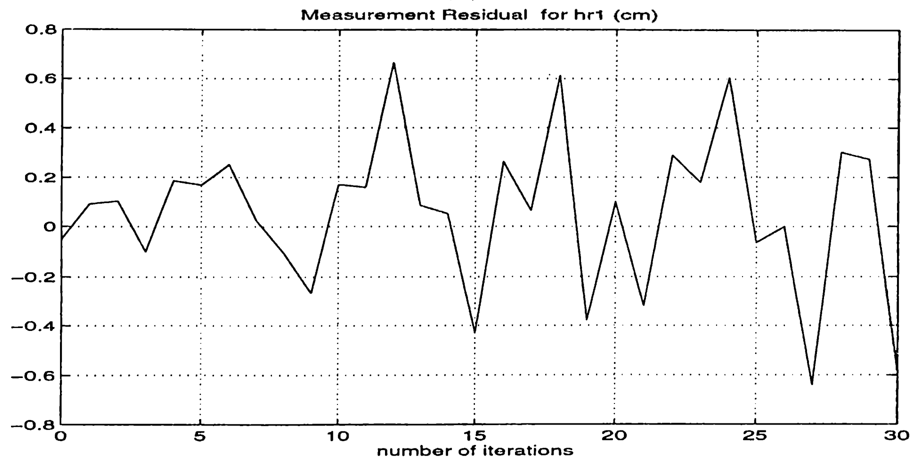


(c)

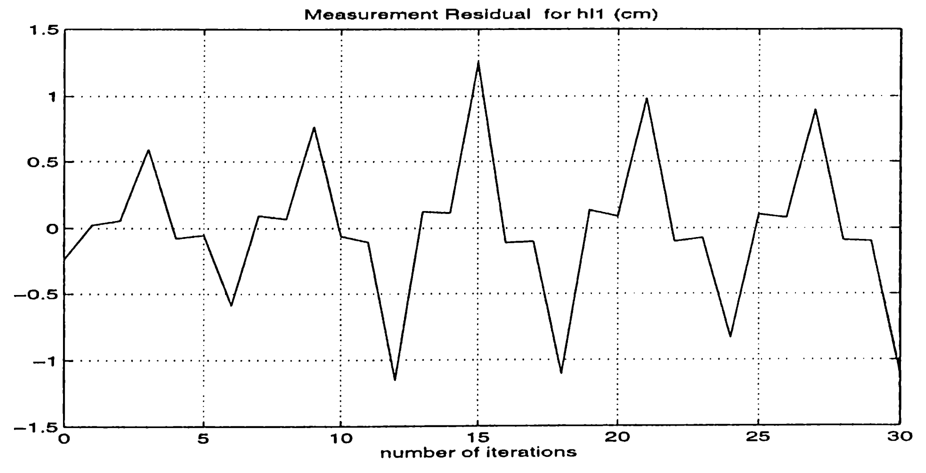
Figure 6.7: Estimated values of the Kalman filter for $d = 7.5$ cm, $R = 10$ cm, $h_{o1} = 100$ cm, $\theta = 0°$, variance of radius noise $= 10^{-3}$ cm^2, variance of angle noise $= 10^{-4}$ rad^2, initial estimates $R = 10.1$ cm, $h_{o1} = 99$ cm and $\theta = 0.1°$.



(*a*)

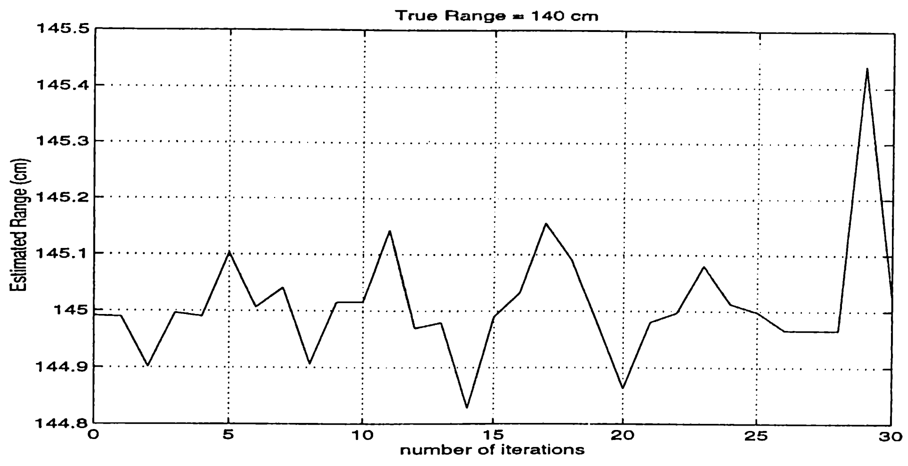


(*b*)

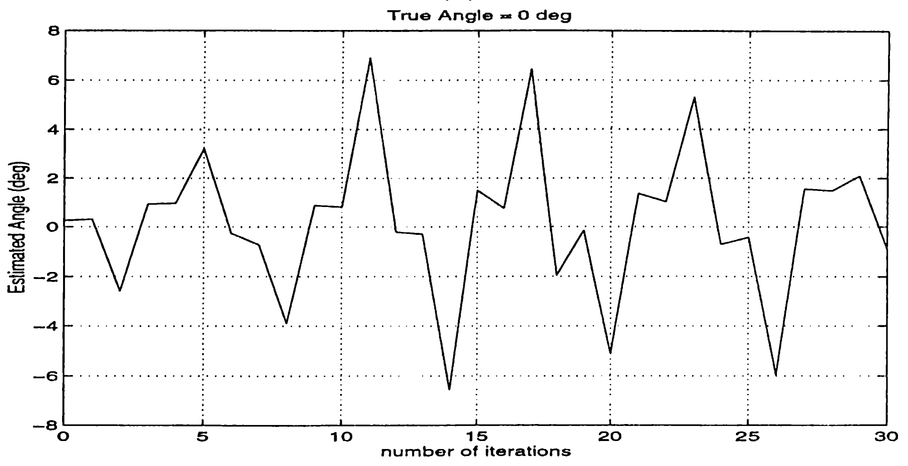


(*c*)

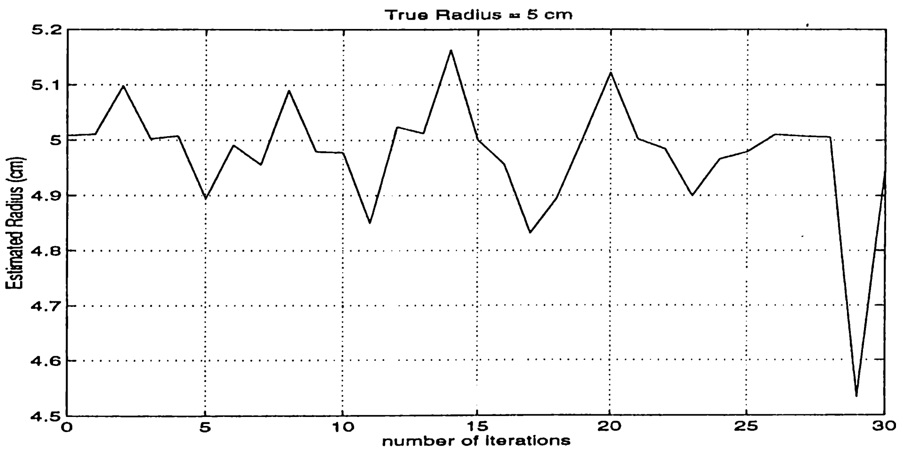
Figure 6.8: Measurement residuals of the Kalman filter for $d = 7.5$ cm, $R = 5$ cm, $h_{o1} = 140$ cm, $\theta = 0°$, variance of radius noise $= 10^{-3}$ cm^2, variance of angle noise $= 10^{-4}$ rad^2, initial estimates $R = 5.1$ cm, $h_{o1} = 49$ cm and $\theta = 0.1°$.



(a)



(b)



(c)

Figure 6.9: Estimated values of the Kalman filter for $d = 7.5$ cm, $R = 5$ cm, $h_{o1} = 140$ cm, $\theta = 0°$, variance of radius noise $= 10^{-3}$ cm^2, variance of angle noise $= 10^{-4}$ rad^2, initial estimates $R = 5.1$ cm, $h_{o1} = 139$ cm and $\theta = 0.1°$.

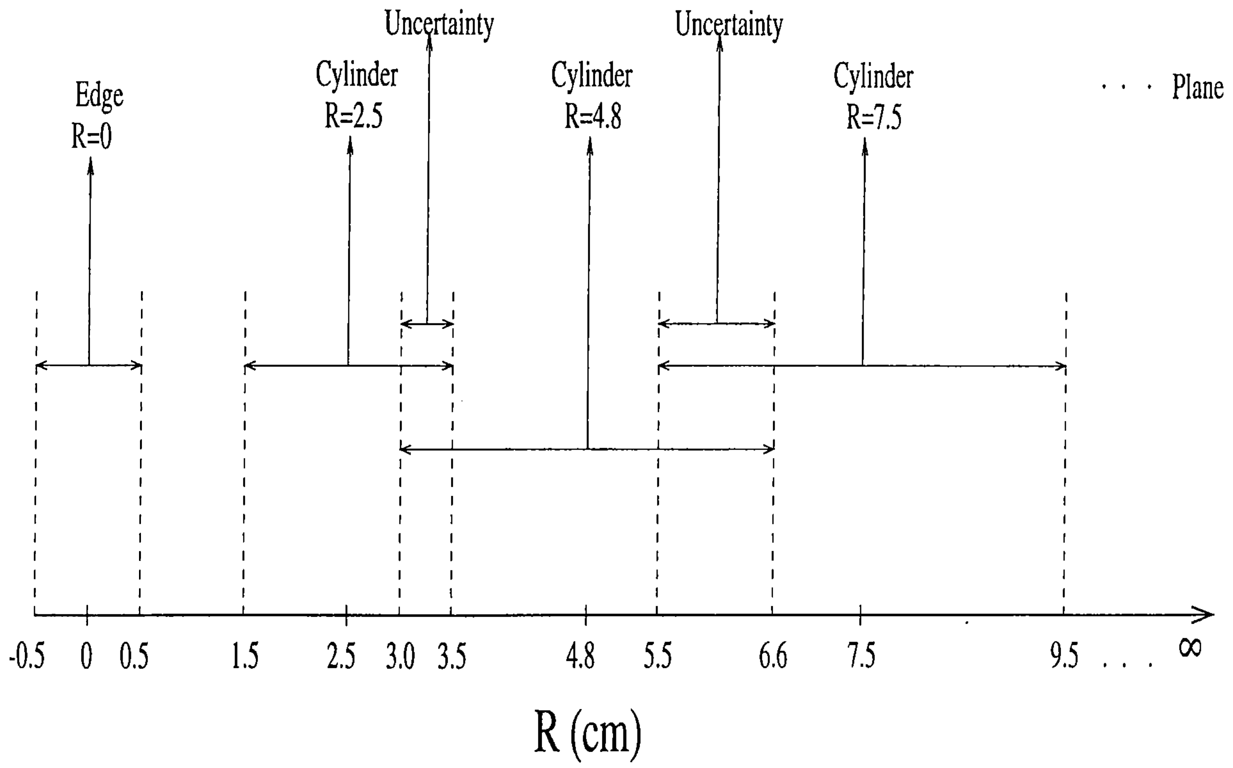


Figure 6.10: Target discrimination using radius of curvature estimation.

Chapter 7

CONCLUSION AND DIRECTIONS FOR FUTURE WORK

In this chapter, some conclusions are presented by summarizing the contents and contributions of the thesis and some suggestions for future work are made.

7.1 Summary of Thesis

In this study, the estimations of the location and radius of curvature of cylindrical objects are made by using the equations from geometry and the estimates are smoothed by using extended Kalman filtering. A tri-aural sensor consisting of three transmitting/receiving transducers is composed to perform the estimations. The main goal of the thesis is to improve radius of curvature estimation. This is done by obtaining sufficient data and rotating the transducers in order to make them perpendicular to the object. This way, SNR is increased and the estimation accuracy is improved. Two methods are used to rotate the transducers. First, the peripheral transducers are rotated around their joints. Second, the transducers are rotated around their centers. Also, the thresholding and curve-fitting methods are used to estimate TOF. In order to assess the performance of the estimator, the bias-variance combinations of both estimators are compared to the CRLB. Finally, in order to verify the simulation results, experimental data is collected using both Panasonic and Polaroid transducers.

The main conclusions of this thesis are as follows:

- The estimation with rotated configuration gives much better results than the estimation with linear configuration.
- The simulation results and the CRLB comparison test show that the curve-fitting method works better that the thresholding method.
- The experimental results illustrate that the thresholding and curve-fitting methods give comparable results.
- The extended Kalman filtering smoothes the estimates considerably.
- The radius of curvature estimation can be used to differentiate the different types of reflectors such as edges, cylinders and walls.
- The estimations are made for convex ($R > 0$) reflectors but they are equally applicable for concave ($R < 0$) reflectors.

7.2 Directions for Future Work

The following are the main directions of future research:

- Sensory information from different types of sensors such as sonar, infrared and laser can be fused to estimate the location and radius of curvature of cylindrical objects more accurately.
- Location and radius of curvature estimation can be extended to spherical targets in 3-D.
- The adjustable tri-aural sensor can be used on a mobile robot for map-building and robot navigation.
- The method can be generalized to extended surfaces with spatially varying curvature where the curvature can be both concave and convex.

Appendix A

CRAMÉR-RAO LOWER BOUNDS

In the following, the comparison of the standard deviation of the biased radius estimator with the Cramér-Rao lower bound (CRLB) is explained for different d, R and h_{o1}.

- In the measurements, zero-mean spatially uncorrelated additive white Gaussian noise is assumed:

$$\begin{aligned}\hat{h}_{o1} &= r - R + w_o(r,\theta,R)\\ \hat{h}_{or} &= \sqrt{r^2+d^2-2dr\sin\theta} - R + w_r(r,\theta,R)\\ \hat{h}_{ol} &= \sqrt{r^2+d^2+2dr\sin\theta} - R + w_l(r,\theta,R)\end{aligned} \qquad (A.1)$$

$$\hat{\mathbf{z}} \triangleq \begin{bmatrix}\hat{h}_{o1}\\ \hat{h}_{or}\\ \hat{h}_{ol}\end{bmatrix} \qquad \mathbf{h}(r,\theta,R) = \begin{bmatrix} r-R \\ \sqrt{r^2+d^2-2dr\sin\theta}-R \\ \sqrt{r^2+d^2+2dr\sin\theta}-R\end{bmatrix} \qquad (A.2)$$

- As explained in the text, the r, θ and R values maximizing Equation 4.5 are the maximum likelihood estimators which can be found by taking the inverse of $\hat{\mathbf{z}} = \mathbf{h}(\hat{r},\hat{\theta},\hat{R})$ as follows:

$$\hat{r} = \frac{2d^2 + 2(\hat{h}_{ol}+\hat{h}_{or})\hat{h}_{o1} - 2\hat{h}_{o1}^2 - \hat{h}_{ol}^2 - \hat{h}_{or}^2}{2\hat{h}_{or} + 2\hat{h}_{ol} - 4\hat{h}_{o1}} \qquad (A.3)$$

$$\hat{\theta} = \sin^{-1}\left[\frac{\hat{h}_{ol}^2 - \hat{h}_{or}^2 + 2(\hat{h}_{ol}-\hat{h}_{or})\hat{R}}{4d(\hat{h}_{o1}+\hat{R})}\right] \qquad (A.4)$$

$$\hat{R} = \frac{(\hat{h}_{o1}^2 + \hat{h}_{ol}^2) - 2(\hat{h}_{o1}^2 + d^2)}{4\hat{h}_{o1} - 2(\hat{h}_{or} + \hat{h}_{ol})} \tag{A.5}$$

- The biases of the three estimations are defined as:

$$\begin{aligned} b(r) &\triangleq E\{\hat{r}\} - r \\ b(\theta) &\triangleq E\{\hat{\theta}\} - \theta \\ b(R) &\triangleq E\{\hat{R}\} - R \end{aligned} \tag{A.6}$$

Here, $E\{.\}$ is the expectation operator.

- The variances of the estimators are found:

$$\begin{aligned} \sigma_{\hat{r}}^2 &= var\left[\hat{r}(\hat{\mathbf{z}}) - r\right] \\ \sigma_{\hat{\theta}}^2 &= var\left[\hat{\theta}(\hat{\mathbf{z}}) - \theta\right] \\ \sigma_{\hat{R}}^2 &= var\left[\hat{R}(\hat{\mathbf{z}}) - R\right] \end{aligned} \tag{A.7}$$

- CRLB for the variance of an unbiased estimator is given as follows:

$$\begin{aligned} \sigma_{\hat{r}}^2 &\geq J^{11} \\ \sigma_{\hat{\theta}}^2 &\geq J^{22} \\ \sigma_{\hat{R}}^2 &\geq J^{33} \end{aligned} \tag{A.8}$$

here J^{11}, J^{22} and J^{33} are the diagonal elements of the inverse of the Fisher information matrix.

- The Fisher information matrix $\mathbf{J}$ and its inverse $\mathbf{J}^{-1}$ are given as follows:

$$\mathbf{J} = \begin{bmatrix} J_{11} & J_{12} & J_{13} \\ J_{21} & J_{22} & J_{23} \\ J_{31} & J_{32} & J_{33} \end{bmatrix} \tag{A.9}$$

$$\mathbf{J}^{-1} = \begin{bmatrix} J^{11} & J^{12} & J^{13} \\ J^{21} & J^{22} & J^{23} \\ J^{31} & J^{32} & J^{33} \end{bmatrix} \tag{A.10}$$

$$\begin{aligned} J_{11} &= \frac{\partial \mathbf{h}^T(r,\theta,R)}{\partial r}\mathbf{C}^{-1}\frac{\partial \mathbf{h}(r,\theta,R)}{\partial r} \\ J_{22} &= \frac{\partial \mathbf{h}^T(r,\theta,R)}{\partial \theta}\mathbf{C}^{-1}\frac{\partial \mathbf{h}(r,\theta,R)}{\partial \theta} \\ J_{33} &= \frac{\partial \mathbf{h}^T(r,\theta,R)}{\partial R}\mathbf{C}^{-1}\frac{\partial \mathbf{h}(r,\theta,R)}{\partial R} \end{aligned}$$

$$\begin{aligned}
J_{12} &= J_{21} = \frac{\partial \mathbf{h}^T(r,\theta,R)}{\partial r}\mathbf{C}^{-1}\frac{\partial \mathbf{h}(r,\theta,R)}{\partial \theta} \\
J_{13} &= J_{31} = \frac{\partial \mathbf{h}^T(r,\theta,R)}{\partial r}\mathbf{C}^{-1}\frac{\partial \mathbf{h}(r,\theta,R)}{\partial R} \\
J_{23} &= J_{32} = \frac{\partial \mathbf{h}^T(r,\theta,R)}{\partial \theta}\mathbf{C}^{-1}\frac{\partial \mathbf{h}(r,\theta,R)}{\partial R}
\end{aligned}$$

where

$$\mathbf{C} = \begin{bmatrix} \sigma_{w_o}^2 & 0 & 0 \\ 0 & \sigma_{w_r}^2 & 0 \\ 0 & 0 & \sigma_{w_l}^2 \end{bmatrix} \tag{A.11}$$

$$\mathbf{C}^{-1} = \begin{bmatrix} \frac{1}{\sigma_{w_o}^2} & 0 & 0 \\ 0 & \frac{1}{\sigma_{w_r}^2} & 0 \\ 0 & 0 & \frac{1}{\sigma_{w_l}^2} \end{bmatrix} \tag{A.12}$$

Let us abbreviate some of the expressions as follows:

$$\begin{aligned}
A_1 &\triangleq \frac{\partial h_1}{\partial r} \quad B_1 \triangleq \frac{\partial h_2}{\partial r} \quad C_1 \triangleq \frac{\partial h_3}{\partial r} \\
A_2 &\triangleq \frac{\partial h_1}{\partial \theta} \quad B_2 \triangleq \frac{\partial h_2}{\partial \theta} \quad C_2 \triangleq \frac{\partial h_3}{\partial \theta} \\
A_3 &\triangleq \frac{\partial h_1}{\partial R} \quad B_3 \triangleq \frac{\partial h_2}{\partial R} \quad C_3 \triangleq \frac{\partial h_3}{\partial R}
\end{aligned} \tag{A.13}$$

By taking appropriate derivatives, above parameters can be found as follows:

$$\begin{aligned}
A_1 &= 1 \quad B_1 = \frac{r - d\sin\theta}{\sqrt{r^2 + d^2 - 2dr\sin\theta}} \quad C_1 = \frac{r + d\sin\theta}{\sqrt{r^2 + d^2 + 2dr\sin\theta}} \\
A_2 &= 0 \quad B_2 = \frac{-dr\cos\theta}{\sqrt{r^2 + d^2 - 2dr\sin\theta}} \quad C_2 = \frac{+dr\cos\theta}{\sqrt{r^2 + d^2 + 2dr\sin\theta}} \\
A_3 &= -1 \quad B_3 = -1 \quad C_3 = -1
\end{aligned} \tag{A.14}$$

With these derivations the elements of the Fisher information matrix can be found easily:

$$\begin{aligned}
J_{11} &= \frac{A_1^2}{\sigma_{w_o}^2} + \frac{B_1^2}{\sigma_{w_r}^2} + \frac{C_1^2}{\sigma_{w_l}^2} \\
J_{22} &= \frac{A_2^2}{\sigma_{w_o}^2} + \frac{B_2^2}{\sigma_{w_r}^2} + \frac{C_2^2}{\sigma_{w_l}^2} \\
J_{33} &= \frac{A_3^2}{\sigma_{w_o}^2} + \frac{B_3^2}{\sigma_{w_r}^2} + \frac{C_3^2}{\sigma_{w_l}^2}
\end{aligned}$$

$$\begin{aligned}
J_{12} &= J_{21} = \frac{A_1A_2}{\sigma_{w_o}^2} + \frac{B_1B_2}{\sigma_{w_r}^2} + \frac{C_1C_2}{\sigma_{w_l}^2} \\
J_{31} &= J_{13} = \frac{A_1A_3}{\sigma_{w_o}^2} + \frac{B_1B_3}{\sigma_{w_r}^2} + \frac{C_1C_3}{\sigma_{w_l}^2} \\
J_{23} &= J_{32} = \frac{A_2A_3}{\sigma_{w_o}^2} + \frac{B_2B_3}{\sigma_{w_r}^2} + \frac{C_2C_3}{\sigma_{w_l}^2} \qquad \text{(A.15)}
\end{aligned}$$

By using the elements of the Fisher information matrix, the diagonal elements of the inverse Fisher information matrix are evaluated:

$$\begin{aligned}
J^{11} &= \frac{J_{22}J_{33} - J_{23}^2}{|\mathbf{J}|} \\
J^{22} &= \frac{J_{11}J_{33} - J_{13}^2}{|\mathbf{J}|} \\
J^{33} &= \frac{J_{11}J_{22} - J_{12}^2}{|\mathbf{J}|} \qquad \text{(A.16)}
\end{aligned}$$

Appendix B

EXTENDED KALMAN FILTER ALGORITHM

The extended Kalman filter (EKF) is a nonlinear estimator which recursively calculates a minimum variance estimate for a state by using observations which are nonlinearly related to this state. The EKF has many application areas such as location estimation problems in mobile robot systems, aerospace navigation and process control [33, 34, 35, 36].

The notation of Bar-Shalom [32] is used to provide the details of the extended Kalman filtering. Also the paper by H. W. Sorenson [37] is very helpful.

Firstly, notation will be established and process and observation models will be given. In general, both the process model and the observation model can be nonlinear.

- The state or process model is

$$\mathrm{x}(k+1) = \mathbf{f}[k, \mathrm{x}(k)] + \mathrm{v}(k) \tag{B.1}$$

where $\mathrm{v}(k)$ is the zero-mean, additive, white Gaussian process noise with

$$E\left[\mathrm{v}(k)\mathrm{v}(j)\right] = \mathbf{Q}(k)\delta_{kj} \tag{B.2}$$

and δ_{kj} is the Kronecker-delta.

- The observation model is

$$\mathrm{z}(k) = \mathbf{h}\left[k, \mathrm{x}(k)\right] + \mathrm{w}(k) \tag{B.3}$$

where $\mathbf{w}(k)$ is zero-mean, additive, white Gaussian measurement noise with

$$E\left[\mathbf{w}(k)\mathbf{w}(j)\right] = \mathbf{R}(k)\delta_{kj} \tag{B.4}$$

- $\hat{\mathbf{x}}(j|k)$ is the estimate of the state vector $\mathbf{x}(j)$ at time step j given all observations up to time step k.
- $\mathbf{P}(j|k)$ is the conditional covariance matrix of $\mathbf{x}(j)$ given all observations up to time step k.

EXTENDED KALMAN FILTER ALGORITHM :

1. Estimate the state $\hat{\mathbf{x}}(k|k)$ at time t_k.
2. Predict the state at time t_{k+1} given the measurements up to time t_k:

$$\hat{\mathbf{x}}(k+1|k) = \mathbf{f}\left[k, \hat{\mathbf{x}}(k|k)\right] \tag{B.5}$$

3. Predict the measurement at time t_{k+1} given the measurements up to time t_k:

$$\hat{\mathbf{z}}(k+1|k) = \mathbf{h}\left[k+1, \hat{\mathbf{x}}(k+|k)\right] \tag{B.6}$$

4. Calculate the measurement residual or the innovation sequence:

$$\mathbf{r}(k+1) = \mathbf{z}(k+1) - \hat{\mathbf{z}}(k+1|k) \tag{B.7}$$

5. Estimate the state covariance $\mathbf{P}(k|k)$ at time t_k.
6. Calculate the Jacobian of $\mathbf{h}(k+1)$ at $\mathbf{x} = \hat{\mathbf{x}}(k+1|k)$ using steps 2 and 5:

$$\mathbf{H}(k+1) = \frac{\partial \mathbf{h}(k+1)}{\partial \mathbf{x}} \tag{B.8}$$

7. Predict the state covariance:

$$\mathbf{P}(k+1|k) = \mathbf{P}(k|k) + \mathbf{Q}(k) \tag{B.9}$$

8. Calculate the residual covariance:

$$\mathbf{S}(k+1) = \mathbf{H}(k+1)\mathbf{P}(k|k)\mathbf{H}'(k+1) + \mathbf{R}(k) \tag{B.10}$$

9. Calculate the filter gain:

$$\mathbf{W}(k+1) = \mathbf{P}(k+1|k)\mathbf{H}'(k+1)\mathbf{S}(k+1)^{-1} \tag{B.11}$$

10. Update the state covariance:

$$\mathbf{P}(k+1|k+1) = \mathbf{P}(k+1|k) - \mathbf{W}(k+1)\mathbf{S}(k+1)\mathbf{W}'(k+1) \tag{B.12}$$

11. Update the state using steps 4 and 9.

$$\hat{\mathbf{x}}(k+1|k+1) = \hat{\mathbf{x}}(k+1|k) + \mathbf{W}(k+1)\ \mathbf{r}\ (k+1) \tag{B.13}$$

12. Go to step 1.

Appendix C

CHI-SQUARE DISTRIBUTED RANDOM VARIABLES

In this appendix, the convergence test of the *extended Kalman filter* algorithm is provided. The notation of Bar-Shalom is used [38].

The following criteria are most commonly used for the consistency of a filter:

- The state errors should be zero-mean.
- The innovations should have the same property.
- The innovations should be white (uncorrelated in time).

The first criterion cannot be tested in real data applications. The last two are the consequences of the first one and can be tested in real data applications [38].

For a consistent filter, the normalized innovations squared, as given below,

$$\epsilon_v(k) \triangleq \mathbf{r}'(k)\mathbf{S}^{-1}(k)\mathbf{r}(k) \tag{C.1}$$

must have a chi-square distribution with n_z degrees of freedom, where n_z is the dimension of the measurement vector $\mathbf{z}$. In this study, since there are three measurements, $n_z = 3$.

Let $\mathbf{u}$ be n-dimensional Gaussian random vector with covariance matrix $\mathbf{C}$. When the following is calculated, a scalar quantity is obtained.

$$v = (\mathbf{u} - \bar{\mathbf{u}})'\mathbf{C}^{-1}(\mathbf{u} - \bar{\mathbf{u}}) \tag{C.2}$$

where $\bar{\mathbf{u}}$ is the mean vector.

v is the sum the squares of n independent, zero mean, unity variance Gaussian random variables. v is called a *chi-square* distributed random variable with n degrees of freedom.

The proof of this can be given as follows:

$$\mathbf{x} = \mathbf{C}^{-1/2}(\mathbf{u} - \bar{\mathbf{u}}) \tag{C.3}$$

$$\mathbf{E}[\mathbf{x}] = \mathbf{0} \tag{C.4}$$

$$\mathbf{E}[\mathbf{x}\mathbf{x}'] = \mathbf{I} \tag{C.5}$$

The components of $\mathbf{x}$ are independent since the covariance matrix is diagonal, and

$$v = \mathbf{x}'\mathbf{x} = \sum_{i=1}^{n} x_i^2 \tag{C.6}$$

where

$$x_i \sim \mathcal{N}(0,1) \tag{C.7}$$

In order to measure the performance of the estimator, the statistical table for chi-square distribution [38] is used as follows:

$$Q(x) \triangleq P(y > x) \tag{C.8}$$

where y is chi-square distributed random variable with m degrees of freedom:

$$y \sim \chi_m^2 \tag{C.9}$$

The confidence region of this random variable can be found by using the statistical table [38]. For example, the 95% confidence region for χ_2^2 is given as follows:

$$[\chi_2^2(0.025), \chi_2^2(0.975)] = [0.05, 7.38] \tag{C.10}$$

This means that when the value

$$\epsilon_v(k) \triangleq \mathbf{r}'(k)\mathbf{S}^{-1}(k)\mathbf{r}(k) \tag{C.11}$$

is calculated, it is likely to fall between 0.05 and 7.38 95% of the time. In other words, out of 100 samples of a chi-square random variable with two degrees-of-freedom, about five are expected to fall outside this interval.

Appendix D

COMPUTER PROGRAMS

In this chapter, the computer programs used for the simulations are presented. The programs are written in C programming language. There are basically two programs. First program simulates the location and radius of curvature estimations and CRLB with the thresholding and curve-fitting methods. Second program is for the extended Kalman filtering. The two programs produce MATLAB files to show the results graphically.

PROGRAM 1:

```
 #include <stdio.h>
 #include <math.h>
 #include <stdlib.h>
 #include<stddef.h>
 #define RAND_MAX (2147483647)
 #define Random_Seed 0    /* 0 = randomize */

double c,Amax,rmin,pc,fo,var,R;



double N ( double m, double std)

/* calculates two normal (mean m, variance std**2) values, returns one */
{
   double fac, u1, u2, v1, v2, s;
   int u;
   static int iset = 0;
   static double gset;

   if (iset == 0)
   {
      s = 0;
      do
      {
 u = rand ();
 u1 = (double) u/ RAND_MAX;
 u = rand ();
 u2 = (double) u/ RAND_MAX;
 v1 = 2 * u1 - 1;
 v2 = 2 * u2 - 1;
 s = v1 * v1 + v2 * v2;
      } while (s > 1);

      fac = sqrt ((-2 * log (s)) / s);

      gset = (v1 * fac);
      iset = 1;
```

```
      v2 = v2 * fac * std + m;
      return v2;
   }
   else
   {
      iset = 0;
      return (gset * std) + m;
   }
}

void set_seed (void)
/* routine to set seed of random number generator in a 'random' fashion by
  using the time */
{
   srand ((unsigned int) time(NULL));
}


/* This Function Calculates The  Signal  Value At  A  Specific Time */

double f( double p, double A, double rm, double r, double tetha, double t,
 double to, double foo, double devT, double dev)
{
return(p*A*pow(rm,3/2)/pow(r,3/2)*exp(-pow(tetha/devT,2))
*exp(-pow((t-to-3/foo)/dev,2))*sin(2*M_PI*foo*(t-to)));
}


/* This  Function  Finds Noisy Time  Of  Flight */

int   f1(double truet, double tidiv, double trh, double tet1,double hg )
{

int i,j;
double k,h[50000];
k=floor(truet*tidiv+0.5);
i=0; j=0;
do{
if(i<=k)
  h[i]=0 + N(0,var);
```

```
else
h[i]=f(pc,Amax,rmin,hg,tet1,i/tidiv,k/tidiv,fo,6.5*M_PI/180,1/fo)+N(0,var);
if(h[i]>trh)
       {j=i;
return(j);}
i++;
}while(i<=50000 && j==0);


}

/*This  function is  for  curve  fitting */
double function(double time, double distance, double angle, double tdiv,
 int coun)
{
double a,b,c,fun[5],cur[5],delta,j;
int  i;
delta = -1;
j=coun;
do
{
for(i=0;i<3;i++)
{fun[i]=f(pc,Amax,rmin,distance,angle,(j-2.5*i)/tdiv,time,
fo,6.5*M_PI/180,1/fo)+N(0,var);
 cur[i]=(j-2.5*i)/tdiv;}
a=((fun[0]-fun[1])*(cur[0]-cur[2])-(fun[0]-fun[2])*(cur[0]-cur[1]))
/((cur[0]-cur[1])*(cur[1]-cur[2])*(cur[0]-cur[2]));
b=((fun[0]-fun[1])-a*(cur[0]*cur[0]-cur[1]*cur[1]))
/(cur[0]-cur[1]);
c=fun[0]-a*cur[0]*cur[0]-b*cur[0];
delta=b*b-4*a*c;
j=j-1;
}while(delta < 0 );
return((-b+sqrt(delta))/(2*a));
}


/* This Function  is  For  Drawing  The  Signal */

void   draw(double truet, double tidiv, double trh, double tet1 ,
```

```
char ofile[10],int lenght,double hg2)
{
FILE *fout;
int i,j;
double k,ho[50000];
fout=fopen(ofile,"w");
fprintf(fout,"A=[ \n");
k=floor(truet*tidiv);
for(i=0;i<=lenght;i++){

if(i<=k)
  ho[i]=0 + N(0,var);
else
ho[i]=f(pc,Amax,rmin,hg2,tet1,i/tidiv,k/tidiv,
fo,6.5*M_PI/180,1/fo)+N(0,var);
fprintf(fout,"%f \n",ho[i]);}
fprintf(fout,"] ; \n plot(A); \n grid; \n");
fclose(fout);
}



main()
{



int  i,j,l,v,count,count1,method,rot,count2,count3,count4,
count5,count6;
int  lent,lent1;
double  B,k,hl,hr,ho,ro,tet,d,R1[1000],R2[1000],r[200],r2[200],
angle[200],angle2[200],alp,x,rl,rr,a,b,hl2,hot,hlt,hrt,hr2,
tr2,hr2n,tr2n,ho1,tl2,tl2n,hl2n,phi,c1,c3,R21,rr2,z1,beta,
c2,R2r,tetg,rg;
double  z,a1,b1,a2,a3,a4,alp1,too,tl1,trr,tl12,variable[200],sum,
sum1,sum2,sum3,sum4,sum5,sum6,tog,hr1,tetn,alphan,etan,
ho2,to2,to2n,ho2n,rl2g,rl2,rr2g;
double  hoa[50000],hla[50000],hra[50000],hl2a[50000],trhold,timediv,
Radius,Radius2,distance,distance2,ang,ang2,mean,mean2,mean3,
mean4,mean5,mean6,std,std1,std2,std3,std4,std5,std6,hog,toog,
hlg,hrg,bias,bias3,bias5,stb,stb3,stb5,A,A1,CRb,to1n,ho1n,to1,
ro1,hl1,tl1,rl1,alpha,eta,hl1n,hr1n,tl1n,tr1n,tr1,rr1,A2,B1,B2,
C,C1,C2,k1,k2,k3,k4,k5,k6,CRb2,CRb3,detJ,e1[100],e2[100],
e3[100],me1,me2,me3,ve1,ve2,ve3,vtotal1,vtotal2,vtotal3;

char   letter,input[10];
FILE *fou;
if (!Random_Seed) set_seed();
        else srand(Random_Seed);


/* Takes The True Values */
printf("Choose The  Variable  d, R, r, t (theta), s (std) : ");
scanf("%c",&letter);
printf(" enter  the  file name  :");
scanf("%s",input);
printf(" 1 (thresholding) or  2 (curvefitting) :");
scanf("%d",&method);
printf("1 (rotation around joint) or 2 (rotation  around center) :");
scanf("%d",&rot);
printf("enter the lenght  of  the  variable : ");
scanf("%d",&lent);
printf("enter the iteration  number : ");
scanf("%d",&lent1);
switch(letter){
case 'd' :
printf("enter ho1:");
scanf("%lf", &ho1);
printf(" enter R:");
scanf("%lf", &R);
printf("enter noise std :");
scanf("%lf",&var);
printf("enter  theta: ");
scanf("%lf",&tet);
printf(" enter the  start point  for d :  ");
scanf("%lf",&d);
for(i=0;i<lent;i++)
variable[i]=d+i;
break;

case 'R' :
```

```
printf("enter ho1:");
scanf("%lf", &ho1);
printf("enter d:");
scanf("%lf", &d);
printf("enter noise std :");
scanf("%lf",&var);
printf("enter  theta: ");
scanf("%lf",&tet);
printf("enter the start point  for the  radius :");
scanf("%lf",&R);
for(i=0;i<lent;i++)
variable[i]=R+i;
break;

case 'r' :
printf(" enter R:");
scanf("%lf", &R);
printf("enter d:");
scanf("%lf", &d);
printf("enter noise std :");
scanf("%lf",&var);
printf("enter  theta: ");
scanf("%lf",&tet);
printf(" enter the  start point  for  r : ");
scanf("%lf",&ho1);
for(i=0;i<lent;i++)
variable[i]=ho1+5*i;
break;

case 's' :
printf("enter ho1:");
scanf("%lf", &ho1);
printf(" enter R:");
scanf("%lf", &R);
printf("enter d:");
scanf("%lf", &d);
  printf("enter  theta: ");
scanf("%lf",&tet);
for(i=0;i<lent;i++)
variable[i]=0.00001*i;
```

```
break;
case 't':
printf(" enter R:");
scanf("%lf", &R);
printf("enter d:");
scanf("%lf", &d);
printf("enter noise std :");
scanf("%lf",&var);
printf("enter ho1:");
scanf("%lf", &ho1);
printf(" enter the  start point  for  theta : ");
scanf("%lf",&tet);
for(i=0;i<lent;i++)
variable[i]=tet+i;
break;
       }


/* Assigns Constants */



c=34350;
fo=50000;
Amax=1;
rmin=10;
sum=0;
sum1=0;
sum2=0;
sum3=0;
sum4=0;
sum5=0;
sum6=0;
count=0;
count1=0;
count2=0;
count3=0;
count4=0;
count5=0;
count6=0;
```

```
timediv=1000000;
fou=fopen(input,"w");
fprintf(fou,"A=[ \n ");

vtotal1=0;
vtotal2=0;
vtotal3=0;
for(l=0;l<lent;l++)
{
switch (letter) {
case 'd' :
d=variable[l];
fprintf(fou,"%lf %lf %lf ",R,ho1+R,tet);
tet=tet*M_PI/180;
break;
case 'R' :
R=variable[l];
fprintf(fou,"%lf %lf %lf ",R,ho1+R,tet);
tet=tet*M_PI/180;
break;
case 'r' :
ho1=variable[l];
fprintf(fou,"%lf %lf %lf ",R,ho1+R,tet);
tet=tet*M_PI/180;
break;
case 's' :
var=variable[l];
fprintf(fou,"%lf %lf %lf ",R,ho1+R,tet);
tet=tet*M_PI/180;
break;
case 't' :
tet=variable[l]*M_PI/180;
fprintf(fou,"%lf %lf %lf ",R,ho1+R,tet*180/M_PI);
break;


}
hog=ho1;
rg=ho1+R;
tetg=tet;
```

```
 pc=0.44946*R-0.022471;
 to1=2*ho1/c;
 tog=to1;
 Radius=0;
 Radius2=0;
 ang=0;
 ang2=0;
 distance=0;
 distance2=0;
 std=0;
 std1=0;
 std3=0;
 std4=0;
 std5=0;
 std6=0;

 if(var>0)
 trhold=6*var;
 else
 trhold=1e-9;



 /* Monte Carlo*/

 me1=0; me2=0; me3=0;
 for(v=0;v<lent1;v++)
 {
 j=f1(to1,timediv,trhold,tet,ho1+R);
 draw(to1,timediv,trhold,0.0,"out1.m",10000,ho1+R);
 if(method==1)
 {to1n=j/timediv;
 ho1n=c*to1n/2;}
 else
 {to1n=function(to1,ho1+R,0.0,timediv,j);
 ho1n=c*to1n/2;}

 e1[v]=ho1-ho1n;
```

```
 ro1=ho1+R;
 /*printf("ho1n  :%lf \n",ho1n);*/


 /* Calculates the  true hl1 and tl1 */
 hl1=sqrt(pow(ro1,2.0)+pow(d,2.0)+2*d*ro1*sin(tet))-R;
 tl1=2*hl1/c;
 rl1=hl1+R;

 /* Calculates the  true hr1 and tr1 */
 hr1=sqrt(pow(ro1,2.0)+pow(d,2.0)-2*d*ro1*sin(tet))-R;
 tr1=2*hr1/c;
 rr1=hr1+R;

 /* calculates the true alpha and eta */
 alpha=asin((rl1*rl1+d*d-ro1*ro1)/(2*d*rl1));
 eta=asin((rr1*rr1+d*d-ro1*ro1)/(2*d*rr1));


 /* Calculates Noisy hl1 and tl1 */
 j=f1(tl1,timediv,trhold,alpha,hl1+R);
 if (method==1)
 {tl1n=j/timediv;
 hl1n=c*tl1n/2;}
 else
 {tl1n=function(tl1,hl1+R,alpha,timediv,j);
 hl1n=c*tl1n/2;}
 /*printf("hl1n  :%lf \n",hl1n);*/


 /* Calculates Noisy hr1 and tr1 */
 j=f1(tr1,timediv,trhold,eta,hr1+R);
 if (method==1)
 {tr1n=j/timediv;
 hr1n=c*tr1n/2;}
 else
 {tr1n=function(tr1,hr1+R,eta,timediv,j);
 hr1n=c*tr1n/2;}
```

```
e2[v]=hr1-hr1n;
e3[v]=hl1-hl1n;


/*First  calculation  of  the estimated radius, distance  and angle */
R1[v]=((hr1n*hr1n+hl1n*hl1n)-2*(ho1n*ho1n+d*d))
/(4*ho1n-2*(hr1n+hl1n));
r[v]=(2*d*d+2*(hl1n+hr1n)*ho1n-2*ho1n*ho1n-hl1n*
hl1n-hr1n*hr1n)/(2*hr1n+2*hl1n-4*ho1n);
angle[v]=180/M_PI*asin((hl1n*hl1n-hr1n*hr1n+2*
(hl1n-hr1n)*R1[v])/(4*d*(ho1n+R1[v])));

Radius=Radius+R1[v];
  distance=distance+r[v];
ang=ang+angle[v];



/* calculates noisy */

ro1=ho1n+R1[v];
rl1=hl1n+R1[v];
rr1=hl1n+R1[v];

/* calculates noisy theta,alpha end eta*/

tetn=asin((rl1*rl1-ro1*ro1-d*d)/(2*d*ro1));
alphan=asin((rl1*rl1+d*d-ro1*ro1)/(2*d*rl1));
etan=asin((rr1*rr1+d*d-ro1*ro1)/(2*d*rr1));

/*The value  of  ho2 and to2 in  the  presence  of  white  noise */
if (rot==2)
{ho2=ho1;
to2=2*ho2/c;
j=f1(to2,timediv,trhold,0.0,ho2+R);
if(method==1)
{to2n=j/timediv;
ho2n=c*to2n/2;}
else
{to2n=function(to2,ho2+R,0.0,timediv,j);
```

```
ho2n=c*to2n/2;}
}


/*The Value Of hl2 And tl2  In  The  Absence  Of Noise*/
if (rot==2)
{hl2=hl1;
tl2=2*hl2/c;}
else
{hl2=sqrt(rl1*rl1+6*rl1*sin(alphan))-R1[v];
rl2g=hl2+R1[v];
hl2=sqrt((hl1+R)*(hl1+R)+6*(hl1+R)*sin(alpha))-R;
tl2=2*hl2/c;}


/*printf("hl2  :%lf \n",hl2);*/

/*Calculation  Of  hl2  In  The  Presence  Of  White  Noise */

j=f1(tl2,timediv,trhold,0.0,hl2+R);
if (method==1)
{tl2n=j/timediv;
hl2n=c*tl2n/2;}
else
{tl2n=function(tl2,hl2+R,0.0,timediv,j);
hl2n=c*tl2n/2;}
/*printf("hl2n  :%lf  \n",hl2n);*/


/* The  Second  Calculation  Of  The  Radius  Of   Curvature
 by Using Left sensor */

if(rot==1)
{rl2=hl2n+R1[v];
a=(rl2g*rl2g-rl1*rl1)/(2*rl2g);
b=(6+2*rl1*sin(alphan))/(2*rl2g);
z=(3-a*b)/(b*b-1)-sqrt(a*a-6*a*b+9)/(b*b-1);
/*printf("z  :%lf \n",z);*/
phi=acos(3/(3+z));
/*printf("phi  :%lf  \n",phi);*/
```

```
c1=sqrt(pow(3+z,2.0)-3*3);
c3=d-z;
R21=sqrt((rl2-c1)*(rl2-c1)+c3*c3-2*rl2*c3*sin(phi))-ho1n;
}


/* The Value Of hr2 And tr2  In  The  Absence  Of Noise*/
if (rot==1)
{hr2=sqrt(rr1*rr1+6*rr1*sin(etan))-R1[v];
rr2g=hr2+R1[v];
hr2=sqrt((hr1+R)*(hr1+R)+6*(hr1+R)*sin(eta))-R;



tr2=2*hr2/c;}
else
{hr2=hr1;
tr2=2*hr2/c;}

/*Calculation  Of  hr2  In  The  Presence  Of  White  Noise */

j=f1(tr2,timediv,trhold,0.0,hr2+R);
if (method==1)
{tr2n=j/timediv;
hr2n=c*tr2n/2;}
else
{tr2n=function(tr2,hr2+R,0.0,timediv,j);
hr2n=c*tr2n/2;}

/* The  Second  Calculation  Of  The  Radius  Of
  Curvature by Using Right sensor */

if(rot==1)
{rr2=hr2n+R1[v];
  a=(rr2g*rr2g-rr1*rl1)/(2*rr2g);
b=(6+2*rr1*sin(eta))/(2*rr2g);
   z1=(3-a*b)/(b*b-1)-sqrt(a*a-6*a*b+9)/(b*b-1);
beta=acos(3/(3+z1));
c1=sqrt(pow(3+z1,2.0)-3*3);
c2=hr2n-c1;
```

```
c3=d-z1;
R2r=sqrt((rr2-c1)*(rr2-c1)+c3*c3-2*(rr2-c1)*c3*sin(beta))-ho1n;
}
      /* The  second  calculation  of  radius, distance and angle */
if (rot==2)
     {R2[v]=((hr2n*hr2n+hl2n*hl2n)-2*(ho2n*ho2n+d*d))
/(4*ho2n-2*(hr2n+hl2n));
      r2[v]=(2*d*d+2*(hl2n+hr2n)*ho2n-2*ho2n*ho2n-hl2n*
hl2n-hr2n*hr2n)/(2*hr2n+2*hl2n-4*ho2n);
      angle2[v]=180/M_PI*asin((hl2n*hl2n-hr2n*hr2n+2*
(hl2n-hr2n)*R2[v])/(4*d*(ho2n+R2[v])));}
else
{R2[v]=(R2l+R2r)/2;
 r2[v]=0;
         angle2[v]=0;}

Radius2=Radius2+R2[v];
distance2=distance2+r2[v];
       ang2=ang2+angle2[v];
me1=me1+e1[v];
me2=me2+e2[v];
me3=me3+e3[v];

                 }


mean=Radius/lent1;
mean2=Radius2/lent1;
mean3=distance/lent1;
mean4=distance2/lent1;
mean5=ang/lent1;
mean6=ang2/lent1;
me1=me1/lent1;
me2=me2/lent1;
me3=me3/lent1;
if(mean>=0)
{ sum=sum+mean;
count++;}
if(mean2>=0)
{sum1=sum1+mean2;
```

```
count1++;}
if(mean3>=0)
{sum3=sum3+mean3;
 count3++;}
if(mean4>=0)
{sum4=sum4+mean4;
 count4++;}
if(mean5>=0)
{sum5=sum5+mean5;
 count5++;}
if(mean6>=0)
{sum6=sum6+mean6;
 count6++;}
ve1=0;
ve2=0;
ve3=0;
for(v=0;v<lent1;v++)
{std=std+pow(R1[v]-mean,2);
 std1=std1+pow(R2[v]-mean2,2);
 std3=std3+pow(r[v]-mean3,2);
 std4=std4+pow(r2[v]-mean4,2);
 std5=std5+pow(angle[v]-mean5,2);
 std6=std6+pow(angle2[v]-mean6,2);
 ve1=ve1+pow(e1[v]-me1,2);
 ve2=ve2+pow(e2[v]-me2,2);
 ve3=ve3+pow(e3[v]-me3,2);}
ve1=ve1/lent1;
ve2=ve2/lent1;
ve3=ve3/lent1;
if (ve1 < 0.0000002)
ve1=0.0000002;
if (ve2 < 0.0000002)
ve2=0.0000002;
if (ve3 < 0.0000002)
ve3=0.0000002;

std=sqrt(std/lent1);
std1=sqrt(std1/lent1);
std3=sqrt(std3/lent1);
std4=sqrt(std4/lent1);
```

```
std5=sqrt(std5/lent1);
std6=sqrt(std6/lent1);
bias=mean-R;
bias3=mean3-(hog+R);
bias5=mean5-tetg*180/M_PI;
stb=sqrt(std*std+bias*bias);
stb3=sqrt(std3*std3+bias3*bias3);
stb5=sqrt(std5*std5+bias5*bias5);

vtotal1=vtotal1+ve1;
vtotal2=vtotal2+ve2;
vtotal3=vtotal3+ve3;

A=1;
B=((hog+R)-d*sin(tetg))/(sqrt((hog+R)*rg+d*d-2*d*rg*sin(tetg)));
C=(rg+d*sin(tetg))/(sqrt(rg*rg+d*d+2*d*sin(tetg)));
B1=-d*rg*cos(tetg)/(sqrt(rg*rg+d*d-2*d*sin(tetg)));
C1=d*rg*cos(tetg)/(sqrt(rg*rg+d*d+2*d*sin(tetg)));
A2=-1;
B2=-1;
C2=-1;
k1=A*A/ve1+B*B/ve2+C*C/ve3;
k2=B1*B1/ve2+C1*C1/ve3;
k3=A2*A2/ve1+B2*B2/ve2+C2*C2/ve3;
k4=B*B1/ve2+C*C1/ve3;
k5=A*A2/ve1+B*B2/ve2+C*C2/ve3;
k6=B1*B2/ve2+C1*C2/ve3;
detJ=k1*(k2*k3-k6*k6)-k4*(k3*k4-k5*k6)+k5*(k4*k6-k2*k5);
CRb=(k2*k3-k6*k6)/(detJ);
CRb2=(k1*k3-k5*k5)/(detJ)*180/M_PI;
CRb3=(k1*k2-k4*k4)/(detJ);
CRb=sqrt(CRb);
CRb2=sqrt(CRb2);
CRb3=sqrt(CRb3);
tet=tetg*180/M_PI;
printf("me1=%lf \n me2=%lf \n me3=%lf \n ve1=%lf \n ve2=%lf\n  ve3=%lf
 \n detJ=%lf\n CRb1=%lf \n CRb2=%lf \n CRb3=%lf \n k1=%lf \n k2=%lf \n k
 \n k4=%lf \n k5=%lf \n k6=%lf \n ",me1,me2,me3,ve1,ve2,ve3,detJ,CRb,CRb
k1,k2,k3,k4,k5,k6);
```

```
fprintf(fou," %lf %lf %lf %lf %lf %lf %lf %lf %lf %lf %lf %lf %lf %lf
 %lf %lf %lf %lf %lf %lf %lf %lf  %lf %lf %lf %lf %lf  ",variable[l],mean,
mean-std,mean+std,std,mean2,mean2-std1,mean2+std1,std1,mean3,mean3-std3,
mean3+std3,std3,mean4,mean4-std4,mean4+std4,std4,mean5,mean5-std5,mean5+std5,
std5,mean6,mean6-std6,mean6+std6,std6,bias,stb);
printf("deg   :%lf \n",variable[l]);
printf("R1 :%lf \n  R2  :%lf \n  r1 :%lf \n  r2  :%lf \n  angle1 :%lf
\n  angle2  :%lf \n  ",mean,mean2,mean3,mean4,mean5,mean6);

fprintf(fou,"%lf %lf %lf %lf %lf %lf %lf %lf %lf %lf %lf %lf \n",CRb,
CRb2,CRb3,stb,stb3,stb5,bias,bias3,bias5,std,std3,std5);

}

vtotal1=vtotal1/lent;
vtotal2=vtotal2/lent;
vtotal3=vtotal3/lent;
fprintf(fou,"] \n B=[ \n");
for(l=0;l<lent;l++)
{
switch (letter) {
case 'd' :
d=variable[l];
tet=tet*M_PI/180;
break;
case 'R' :
R=variable[l];
tet=tet*M_PI/180;
break;
case 'r' :
ho1=variable[l];
tet=tet*M_PI/180;
break;
case 's' :
var=variable[l];
tet=tet*M_PI/180;
break;
case 't' :
tet=variable[l]*M_PI/180;
fprintf(fou,"%lf %lf %lf ",R,ho1+R,tet*180/M_PI);
```

```
break;

}
hog=ho1;
rg=ho1+R;
tetg=tet;
A=1;
B=((hog+R)-d*sin(tetg))/(sqrt((hog+R)*rg+d*d-2*d*rg*sin(tetg)));
C=(rg+d*sin(tetg))/(sqrt(rg*rg+d*d+2*d*sin(tetg)));
B1=-d*rg*cos(tetg)/(sqrt(rg*rg+d*d-2*d*sin(tetg)));
C1=d*rg*cos(tetg)/(sqrt(rg*rg+d*d+2*d*sin(tetg)));
A2=-1;
B2=-1;
C2=-1;
k1=A*A/vtotal1+B*B/vtotal2+C*C/vtotal3;
k2=B1*B1/vtotal2+C1*C1/vtotal3;
k3=A2*A2/vtotal1+B2*B2/vtotal2+C2*C2/vtotal3;
k4=B*B1/vtotal2+C*C1/vtotal3;
k5=A*A2/vtotal1+B*B2/vtotal2+C*C2/vtotal3;
k6=B1*B2/vtotal2+C1*C2/vtotal3;
detJ=k1*(k2*k3-k6*k6)-k4*(k3*k4-k5*k6)+k5*(k4*k6-k2*k5);
CRb=(k2*k3-k6*k6)/(detJ);
CRb2=(k1*k3-k5*k5)/(detJ)*180/M_PI;
CRb3=(k1*k2-k4*k4)/(detJ);
CRb=sqrt(CRb);
CRb2=sqrt(CRb2);
CRb3=sqrt(CRb3);
fprintf(fou,"%lf %lf %lf %lf \n",variable[l],CRb,CRb2,CRb3);
printf(fou,"%lf %lf %lf %lf \n",variable[l],CRb,CRb2,CRb3);

}

sum=sum/count;
sum1=sum1/count1;
sum3=sum3/count3;
sum4=sum4/count4;
sum5=sum5/count5;
sum6=sum6/count6;
printf("Average Radius1  : %lf \n Average Radius2 : %lf \n ",sum,sum1);
printf("Average distance1  : %lf \n Average distance2 : %lf \n ",sum3,sum
```

```
printf("Average angle1  : %lf \n Average angle2 : %lf \n ",sum5,sum6);


/*fprintf(fou,"] \n hold on; plot(A(:,4),A(:,1));  plot(A(:,4),A(:,5),'c--');
 plot(A(:,4),A(:,6),'y-',A(:,4),A(:,6),'yo');  plot(A(:,4),A(:,7),'y-',A(:,4),
A(:,7),'yo'); plot(A(:,4),A(:,9),'y-.'); grid;  plot(A(:,4),A(:,10),'y-.',A(:,
,A(:,10),'y+');  plot(A(:,4),A(:,11),'y-.',A(:,4),A(:,11),'y+');  hold off;
 pause;   figure(2);  hold on;   plot(A(:,4),A(:,2));
  plot(A(:,4),A(:,13),'c--');  plot(A(:,4),A(:,14),'y-',A(:,4),A(:,14),'yo');
 plot(A(:,4),A(:,15),'y-',A(:,4),A(:,15),'yo');
 plot(A(:,4),A(:,18),'y-.',A(:,4),A(:,18),'y+');
  plot(A(:,4),A(:,19),'y-.',A(:,4),A(:,19),'y+');  plot(A(:,4),A(:,17),'y-.');
 grid;  hold off;  pause;   figure(3); hold on;     plot(A(:,4),A(:,3));
  plot(A(:,4),A(:,21),'c--');  plot(A(:,4),A(:,22),'y-',A(:,4),A(:,22),'yo');
 plot(A(:,4),A(:,23),'y-',A(:,4),A(:,23),'yo');
 plot(A(:,4),A(:,26),'y-.',A(:,4),A(:,26),'y+');
 plot(A(:,4),A(:,27),'y-.',A(:,4),A(:,27),'y+');
  plot(A(:,4),A(:,25),'y-.');    grid;  hold off;  "); */

/*  This  part  will  be  used  only  for  CR */
fprintf(fou,"] \n hold on; plot(A(:,4),A(:,31)); plot(B(:,1),B(:,2),'-.');
hold off;  grid; pause; figure(2); hold on; plot(A(:,4),A(:,32));
 plot(B(:,1),B(:,3),'-.'); hold off; grid; pause; figure(3);
 hold on; plot(A(:,4),A(:,33)); plot(B(:,1),B(:,4),'-.'); hold off; grid;
 pause; figure(4); hold on; plot(A(:,4),A(:,34)); plot(A(:,4),A(:,37),'c-.');
plot(A(:,4),A(:,40),'y-',A(:,4),A(:,40),'yo'); grid; hold off;  pause;
  figure(5); hold on:    plot(A(:,4),A(:,35)); plot(A(:,4),A(:,38),'c-.');
 plot(A(:,4),A(:,41),'y-',A(:,4),A(:,41),'yo');  grid; hold off;
 pause; figure(6); hold on; plot(A(:,4),A(:,36)); plot(A(:,4),A(:,39),'c-.');
plot(A(:,4),A(:,42),'y-',A(:,4),A(:,42),'yo'); grid; ");




fclose(fou);

}
```

PROGRAM 2:

```
#include "../cbmat/cbmat.h"
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include<stddef.h>
#define RAND_MAX (2147483647)
#define Random_Seed 0    /* 0 = randomize */

double c,Amax,rmin,pc,fo,var,R;


main()
{


matrix_t  F,H,x_est,x_pred,z_pred,P_est,P_pred,v_res,S_res,
W_gain,QQ,RR,zz,F_T,H_T,W_T,S_I,tXX1,tXX2,tX,tZX,tXZ,
tZZ,tmpmsr;
int  i,j,l,v,count,count1,method,rot,count2,count3,count4,
count5,count6,method2;
int   lent,lent1,XDIM,ZDIM,steps,dumm1,dumm2,dumm3;
double  B,k,hl,hr,ho,ro,tet,d,R1[1000],R2[1000],r[200],r2[200],
angle[200],angle2[200],alp,x,rl,rr,a,b,hl2,hot,hlt,hrt,
hr2,tr2,hr2n,tr2n,ho1,tl2,tl2n,hl2n,phi,c1,c3,R2l,rr2,
z1,beta,c2,R2r,tetg,rg,nu;
double  z,a1,b1,a2,a3,a4,alp1,too,tll,trr,tll2,variable[200],
sum,sum1,sum2,sum3,sum4,sum5,sum6,tog,hr1,tetn,alphan,
etan,ho2,to2,to2n,ho2n,rl2g,rl2,rr2g;
double  Rk[100],rk[100],tetk[100],hoa[50000],hla[50000],hra[50000],
hl2a[50000],trhold,timediv,Radius,Radius2,distance,distance2,
ang,ang2,mean,mean2,mean3,mean4,mean5,mean6,std,std1,std2,
std3,std4,std5,std6,hog,toog,hlg,hrg,bias,bias3,bias5,stb,
stb3,stb5,A,A1,CRb,to1n,ho1n,to1,ro1,hl1,tl1,rl1,alpha,eta,
hl1n,hr1n,tl1n,tr1n,tr1,rr1,A2,B1,B2,C,C1,C2,k1,k2,k3,k4,k5,
k6,CRb2,CRb3,detJ,radius_noise,dist_noise,angle_noise,qx,d1,
d2,d3,d4,kstd1,kstd2,kstd3,km1,km2,km3,dumm4,dumm5;
char   letter,input[20],input1[20],input2[20];
FILE *fou, *fou1, *fou2;
if (!Random_Seed) set_seed();
       else srand(Random_Seed);
```

```
/* Takes The True Values */


printf("Choose The  Variable  d, R, r, t (theta), s (std) : ");
scanf("%c",&letter);
printf("enter the filename  to  read  the  data : ");
scanf("%s",input2);
printf("enter the filename  to  write the results : ");
scanf("%s",input1);
printf("enter the iteration  number : ");
scanf("%d",&lent1);
printf("enter d: ");
scanf("%lf",&d);
printf("enter R : ");
scanf("%lf",&R);
printf("enter ho1 :");
scanf("%lf",&ho1);
printf("enter theta:");
scanf("%lf",&tet);
ro1=ho1+R;

/* Assigns Constants */
c=34350;
fo=50000;
Amax=1;
rmin=10;
sum=0;
sum1=0;
sum2=0;
sum3=0;
sum4=0;
sum5=0;
sum6=0;
count=0;
count1=0;
count2=0;
count3=0;
count4=0;
```

```
count5=0;
count6=0;
timediv=1000000;
fou2=fopen(input2,"r");
fou1=fopen(input1,"w");
fprintf(fou1,"B=[ \n ");

hog=ho1;
rg=ho1+R;
tetg=tet;
pc=0.44946*R-0.022471;
to1=2*ho1/c;
tog=to1;
Radius=0;
Radius2=0;
ang=0;
ang2=0;
distance=0;
distance2=0;
std=0;
std1=0;
std3=0;
std4=0;
std5=0;
std6=0;

/*  For  Kalman  Filtering */

tmpmsr=mat_new(lent1+5,3);

XDIM=3;  ZDIM=3;
nu=10;
x_est=mat_new(XDIM,1);
x_pred=mat_new(XDIM,1);
F=mat_new(XDIM,XDIM);
H=mat_new(ZDIM,XDIM);
P_est=mat_new_diag(0.0,XDIM);
zz=mat_new(ZDIM,1);
z_pred=mat_new(ZDIM,1);
v_res=mat_new(ZDIM,1);
```

```
P_pred=mat_new_diag(0.0,XDIM);
S_res=mat_new(ZDIM,ZDIM);
W_gain=mat_new(XDIM,ZDIM);
QQ=mat_new(XDIM,XDIM);
RR=mat_new(ZDIM,ZDIM);
/* Temporary stuff */
F_T = mat_new(XDIM, XDIM);
H_T = mat_new(XDIM, ZDIM);
W_T = mat_new(ZDIM, XDIM);
S_I = mat_new(ZDIM, ZDIM);
tXX1 = mat_new(XDIM, XDIM);
tXX2 = mat_new(XDIM, XDIM);
tX = mat_new(XDIM, 1);
tZX = mat_new(ZDIM, XDIM);
tXZ = mat_new(XDIM, ZDIM);
tZZ = mat_new(ZDIM, ZDIM);
/* Process  Noise */

for (i = 0; i < XDIM; ++i)
     for (j = 0; j < XDIM; ++j)
e(QQ, i, j) = 0.0;

e(QQ, 0, 0) = 0.001;
e(QQ, 1, 1) = 0.001;
e(QQ, 2, 2) = 0.0001;


/*  Initialization  of state prediction  covariance */

for (i = 0; i < XDIM; ++i)
    for (j = 0; j < XDIM; ++j)
e(P_est,i,j)=nu*e(QQ,i,j);


/*  Some  Parts  of  Jacobian  Matrices */

F=mat_new_diag( 1.0 , 3);

e(H, 0, 0)=-1; e(H, 0, 1)=1; e(H, 0, 2)=0; e(H, 1 ,0)=-1; e(H,2,0)=-1;
```

```
km1=0; km2=0; km3=0; kstd1=0; kstd2=0; kstd3=0;
for(v=0;v<(lent1+1);v++)
{

fscanf(fou2,"%d %d %d %d %lf %lf %lf %lf %lf
  \n ",&steps,&dumm1,&dumm2,&dumm3,&ho1n,&hr1n,
&hl1n,&dumm4,&dumm5);

ho1n=ho1n/10;
hr1n=hr1n/10;
hl1n=hl1n/10;


/* Calculates the  true hl1 and tl1 */
hl1=sqrt(pow(ro1,2.0)+pow(d,2.0)+2*d*ro1*
sin(tet))-R;
tl1=2*hl1/c;
rl1=hl1+R;

/* Calculates the  true hr1 and tr1 */
hr1=sqrt(pow(ro1,2.0)+pow(d,2.0)-2*d*ro1*
sin(tet))-R;
tr1=2*hr1/c;
rr1=hr1+R;

e(tmpmsr,v,0)=ho1n; e(tmpmsr,v,1)=hr1n;
e(tmpmsr,v,2)=hl1n;


km1=km1+ho1n; km2=km2+hr1n; km3=km3+hl1n;
       }
km1=km1/(lent1+1); km2=km2/(lent1+1); km3=km3/(lent1+1);


for(v=0;v < (lent1+1) ;v++)
{

 kstd1=kstd1+pow((e(tmpmsr,v,0)-ho1),2.0);
 kstd2=kstd2+pow((e(tmpmsr,v,1)-hr1),2.0);
```

```
   kstd2=kstd2+pow((e(tmpmsr,v,2)-hl1),2.0);
 }
 kstd1=sqrt(kstd1/(lent1+1));
 kstd2=sqrt(kstd2/(lent1+1));
 kstd3=sqrt(kstd3/(lent1+1));

 for (i = 0; i < ZDIM; ++i)
       for (j = 0; j < ZDIM; ++j)
 e(RR, i, j) = 0.0;
 e(RR, 0, 0) = kstd1*kstd1;
 e(RR, 1, 1) =kstd2*kstd2;
 e(RR, 2, 2) =kstd3*kstd3;

 for(v=0;v < lent1;v++)
 {


 holn=e(tmpmsr,v,0); hrln=e(tmpmsr,v,1);
 hlln=e(tmpmsr,v,2);
 printf("holn : %lf \n hrln : %lf  hlln : %lf",
 e(tmpmsr,v,0),e(tmpmsr,v,1),e(tmpmsr,v,2));

 if(v==0)
 {
 e(x_est,0,0)=4.9;
 e(x_est,1,0)=55.1;
   e(x_est,2,0)=0.01;
 }

 /* Noisy Measurements */
 e(zz,0,0)=e(tmpmsr,v+1,0);
 e(zz,1,0)=e(tmpmsr,v+1,1);
 e(zz,2,0)=e(tmpmsr,v+1,2);

 /* State Prediction */

     e(x_pred,0,0)= e(x_est,0,0) ;
 e(x_pred,1,0)= e(x_est,1,0) ;
 e(x_pred,2,0)= e(x_est,2,0) ;
```

```
/* Measurement  Prediction */

   e(z_pred,0,0)=e(x_pred,1,0)-e(x_pred,0,0);
e(z_pred,1,0)=sqrt(pow(e(x_pred,1,0),2.0)+d*d-2*
                d*e(x_pred,1,0)*sin(e(x_pred,2,0)))-
e(x_pred,0,0);
e(z_pred,2,0)=sqrt(pow(e(x_pred,1,0),2.0)+d*d+
2*d*e(x_pred,1,0)*sin(e(x_pred,2,0)))-
e(x_pred,0,0);

/* Measurement  Residual */

qmat_sub(v_res,zz,z_pred);


/* Evaluation  of  Jacobians */

e(H,1,1)=(e(x_pred,1,0)-d*sin(e(x_pred,2,0)))
/sqrt(pow(e(x_pred,1,0),2.0)+d*d-2*d*
e(x_pred,1,0)*sin(e(x_pred,2,0)));
e(H,1,2)=(-d*e(x_pred,1,0)*cos(e(x_pred,2,0)))
/sqrt(pow(e(x_pred,1,0),2.0)+d*d-2*d*
e(x_pred,1,0)*sin(e(x_pred,2,0)));
e(H,2,1)=(e(x_pred,1,0)+d*sin(e(x_pred,2,0)))
/sqrt(pow(e(x_pred,1,0),2.0)+d*d+
2*d*e(x_pred,1,0)*sin(e(x_pred,2,0)));
e(H,2,2)=(d*e(x_pred,1,0)*cos(e(x_pred,2,0)))
/sqrt(pow(e(x_pred,1,0),2.0)+d*d+
2*d*e(x_pred,1,0)*sin(e(x_pred,2,0)));
       /*  State  Prediction   Covariance */

qmat_trans(F_T,F);
qmat_mpy(tXX1,F,P_est);
qmat_mpy(tXX2, tXX1,F_T);
qmat_add(P_pred,tXX2,QQ);

/* Residual  Covariance */

qmat_trans(H_T, H);
qmat_mpy(tZX, H, P_est);
```

```
qmat_mpy(tZZ, tZX, H_T);
qmat_add(S_res, tZZ, RR);

/* Filter  gain */

qmat_mpy(tXZ, P_pred, H_T);
if (qmat_inv(S_I, S_res  ) == SINGULAR)
            fprintf(stderr, "Singular Inversion of S\n");
qmat_mpy(W_gain, tXZ, S_I);

/* Updated  State   Estimate */

qmat_mpy(tX, W_gain, v_res);
qmat_add(x_est, x_pred, tX);

/*  Updated  State  Covariance */

qmat_trans(W_T, W_gain);
qmat_mpy(tXZ, W_gain, S_res);
qmat_mpy(tXX1, tXZ, W_T);
qmat_sub(P_est, P_pred, tXX1);
fprintf(fou1,"%d %g %g %g %g %g %g %g %g %g
 %g %g %g %g %g   \n",v,e(v_res,0,0),
e(v_res,1,0),e(v_res,2,0),e(x_est,0,0),
e(x_est,1,0),180/M_PI*e(x_est,2,0),
 e(x_est,1,0)-R,holn,e(x_pred,0,0),
e(x_pred,1,0),e(x_pred,2,0)*180/M_PI,
 R, ro1, tet*180/M_PI);
printf("%d %g %g %g %g %g %g \n",v,e(v_res,0,0),
e(v_res,1,0),e(v_res,2,0),e(x_est,0,0),
e(x_est,1,0),180/M_PI*e(x_est,2,0));


                }

    fprintf(fou1,"]; figure(1);  plot(B(:,1),B(:,2));  grid; pause;
 figure(2); plot(B(:,1),B(:,3)); grid; pause; figure(3); plot(B(:,1),B(:,4));
 grid; pause; figure(4); hold on;  plot(B(:,1),B(:,5),'y-');
 plot(B(:,1),B(:,10),'r-.'); grid; hold off; pause; figure(5); hold on;
 plot(B(:,1),B(:,6),'y-'); plot(B(:,1),B(:,11),'c-.');  grid;hold off; pause;
```

```
 figure(6); hold on; plot(B(:,1),B(:,7),'y-'); plot(B(:,1),B(:,12),'c-.');
 hold off; grid;");
fclose(fou1);
fclose(fou2);

}
```

REFERENCES

[1] H. Peremans, K. Audenaert and J. M. Van Campenhout, "A high-resolution sensor based on tri-aural perception," *IEEE Transactions on Robotics and Automation*, vol. 9, pp. 36–48, February 1993.

[2] B. Barshan and R. Kuc, "Differentiating sonar reflections from corners and planes by employing an intelligent sensor," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, pp. 560–569, June 1990.

[3] R. Kuc, "Fusing binaural sonar information for object recognition," in *Proceedings of IEEE/SICE/RSJ International Conference on Multisensor Fusion and Integration for Intelligent Systems*, pp. 727–735, 1996.

[4] L. Kleeman and R. Kuc, "Mobile robot sonar for target localization and classification," *International Journal of Robotics Research*, vol. 14, pp. 295–318, August 1995.

[5] K. Sasaki and M. Takano, "Classification of object's surface by acoustic transfer function," in *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 821–828, Raleigh, NC, July 7–10, 1992.

[6] A. M. Flynn, "Combining sonar and infrared sensors for mobile robot navigation," *The International Journal of Robotics Research*, vol. 7, pp. 5–14, December 1988.

[7] J. L. Crowley, "Navigation for an intelligent mobile robot," *IEEE Transactions on Robotics and Automation*, vol. RA-1, pp. 31–41, March 1985.

[8] A. Elfes, "Sonar based real-world mapping and navigation," *IEEE Transactions on Robotics and Automation*, vol. RA-3, pp. 249–265, June 1987.

[9] J. J. Leonard and H. F. Durrant-Whyte, "Mobile robot localization by tracking geometric beacons," *IEEE Transactions on Robotics and Automation*, vol. 7, pp. 376–382, 1991.

[10] A. Ohya, T. Ohya, S. Yuta, "Obstacle detectibility of ultrasonic ranging system and sonar map understanding," *Robotics and Autonomous Systems*, vol. 18, pp. 251–257, 1996.

[11] A. M. Sabatini, "Adaptive target tracking algorithms for airborne ultrosonic rangefinders," *IEE Proceedings-Radar Sonar and Navigation*, vol. 142, pp. 81–87, 1995.

[12] A. M. Sabatini, "A digital signal processing technique for compensating ultrosonic sensors," *IEEE Transactions on Instrumentation and Measurement*, vol. 44, pp. 869–874, 1995.

[13] A. Curran and K. J. Kyriakopoulos, "Sensor-based self-localization for wheeled mobile robots," *Journal of Robotic Systems*, vol. 12, pp. 163–176, 1995.

[14] I. Gibson P. Webb and C. Wykes, "Robot guidance using ultrasonic arrays," *Journal of Robotic Systems*, vol. 11, pp. 681–692, 1994.

[15] J. Borenstein and Y. Koren, "Obstacle avoidance with ultrasonic sensors," *IEEE Transactions on Robotics and Automation*, vol. RA-4, pp. 213–218, April 1988.

[16] Ö. I. Bozma. *A Physical Model-Based Approach to Analysis of Environments using Sonar.* PhD thesis, Yale University, New Haven, CT, May 1992.

[17] R. Kuc and B. V. Viard, "A physically-based navigation strategy for sonar-guided vehicles," *The International Journal of Robotics Research*, vol. 10, pp. 75–87, April 1991.

[18] J. H. Ko, W. J. Kim and M. J. Chung, "A method of acoustic landmark extraction for mobile robot navigation," *IEEE Transcations on Robotics and Automation*, vol. 12, pp. 478–485, 1996.

[19] C. C. Chang and K. T. Song, "Ultrasonic sensor data integration and its application to environment perception," *Journal of Robotic Systems*, vol. 13, pp. 663–677, 1996.

[20] M. L. Hong and L. Kleeman, "A low sample rate 3-D sonar sensor for mobile robots," in *Proceedings IEEE International Conference on Robotics and Automation*, pp. 3015–3020, Nagoya, Japan, May 21–27, 1995.

[21] M. L. Hong and L. Kleeman, "Analysis of ultrasonic differentiation of three-dimensional corners, edges and planes," in *Proceedings IEEE International Conference on Robotics and Automation*, pp. 580–584, Nice, France, May 12–14, 1992.

[22] L. Kleeman and H. Akbarally, "A sonar sensor for accurate 3-D target localization and classification," in *Proceedings IEEE International Conference on Robotics and Automation*, pp. 3003–3008, Nagoya, Japan, May 21–27, 1995.

[23] A. Hilton, J. Illingworth and T. Windeatt, "Statistics of surface curvature estimates," *Pattern Recognition*, vol. 28, pp. 1201–1221, 1995.

[24] B. Barshan and R. Kuc, "A bat-like sonar system for obstacle localization," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 22, pp. 636–646, July/August 1992.

[25] J. Zemanek, "Beam behaviour within the nearfield of a vibrating piston," *The Journal of the Acoustical Society of America*, vol. 49, pp. 181–191, January 1971.

[26] A. D. Pierce. *Acoustics, An Introduction to Its Physical Principles and Applications.* McGraw-Hill, New York, 1981.

[27] B. Barshan. *A Sonar-Based Mobile Robot for Bat-Like Prey Capture.* PhD thesis, Yale University, New Haven, CT, December 1991. University of Michigan Microfilms, order number 9224325.

[28] L. W. Camp. *Underwater Acoustics*, chapter 7, p. 166. Wiley-Interscience, New York, 1970.

[29] B. Ayrulu. "Classifications of Target Primitives with Sonar using two Nonparametric Data Fusion Methods," Master's thesis, Bilkent University, Ankara, Turkey, July 1996.

[30] B. Barshan, "Location and curvature estimation of spherical targets using a flexible sonar configuration," in *Proceedings 1996 IEEE International Conference on Robotics and Automation*, pp. 1218–1223, Minneapolis, MN, April 22-28, 1996.

[31] H. L. Van Trees. *Detection, Estimation, and Modulation Theory, Part I.* John Wiley & Sons, New York, 1968.

[32] Y. Bar-Shalom and Xiao-Rong Li. *Estimation and Tracking: Principles, Techniques, and Software.* Artech House, Boston, 1993.

[33] H. W. Sorenson, ed. *Kalman Filtering: Theory and Application.* IEEE Press, New York, 1985.

[34] N. Ayache and O. D. Faugeras, "Maintaining representations of the environment of a mobile robot," *IEEE Transactions on Robotics and Automation*, vol. 5, pp. 804–819, December 1989.

[35] L. Matthies and S. Shafer, "Error modelling in stereo navigation," *IEEE Journal of Robotics and Automation*, vol. RA-3, pp. 239–248, June 1987.

[36] D. J. Kriegman, E. Triendl and T. O. Binford, "Stereo vision and navigation in building mobile robots," *IEEE Transactions on Robotics and Automation*, vol. 5, pp. 792–803, December 1989.

[37] H. W. Sorenson, "Least-squares estimation: from Gauss to Kalman," *IEEE Spectrum*, vol. 7, pp. 63–68, July 1970.

[38] Y. Bar-Shalom and T. E. Fortmann. *Tracking and Data Association.* Academic Press, New York, NY, 1988.