Capture and Location of Blobs in Digital Images

by

Sven Alexander Seelemann
B.A.Sc., University of Waterloo, 1990

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF APPLIED SCIENCE

in the
Department of Mechanical Engineering

We accept this thesis as conforming
to the required standard.

[redacted]
Dr. G. F. McLean, Supervisor (Dept. of Mechanical Engineering)

[redacted]
Dr. Y. Stepanenko, Department Member (Dept. of Mechanical Engineering)

[redacted]
Dr. W. Little, Outside Member (Dept. of Electrical and Computer Engineering)

[redacted]
Dr. M. Miller, External Examiner (Dept. of Computer Science)

Contents

List of Figures

List of Tables

Supervisor: Dr. Gerard F. McLean

Abstract

This thesis describes three methods of detecting the location of active targets using machine vision. In the field of visual servoing, the detection and location of targets is the first step in controlling the motion of a platform with respect to the target. In certain long range, outdoor settings, active targets are the only feasible target type. For each method (centroid, linear parameter estimation, and non–linear parameter estimation), a theoretical resolution and accuracy analysis is performed. This analysis is followed by an experimental suite that determines real resolution and accuracy of each method. The linear parameter estimation technique does not properly locate active targets. Both the centroid and non–linear parameter estimation can achieve accuracies of 0.25 pixels in locating active targets.

Examiners:

Dr. G. F. McLean, Supervisor (Dept. of Mechanical Engineering)

Dr. Y. Stepanenko, Department Member (Dept. of Mechanical Engineering)

Dr. W. Little, Outside Member (Dept. of Electrical and Computer Engineering)

Dr. M. Miller, External Examiner (Dept. of Computer Science)

Acknowledgements

I would like to thank Dr. Ged Mclean for his guidance in preparing this thesis and to all of my family for their support which made this thesis possible.

To my family

(and the letter G)

Chapter 1

Introduction

One of the current areas of interest in the fields of robotics and image processing is the application of visual methods to the position and trajectory control of robots. This problem, most often referred to as *visual servoing*, usually implies that some sort of object is located in an image and the motion of a robot (either a platform, manipulator, or both) is adjusted to achieve some goal based upon the estimated position of the object (target). This problem has application in both academic and industrial settings.

This thesis describes several methods to estimate the position of a class of objects known as active targets. For each method, a theoretical resolution and accuracy analysis is performed. This analysis is followed by an experimental suite that determines achievable resolution and accuracy of each method.

The introduction presented here is split into three sections. The first section outlines some of the interest in visual servoing in the literature. The second section briefly describes the type of targets that can be used for visual servoing. The third section outlines the remainder of this work.

1.1 Background

There is a large interest in the field of visual servoing, some of which is described in the following section.

Jaffe [14] present a discussion on the limits of current technology to underwater location and station keeping technology. Sonar and optical imaging, the two main methods of sub–sea active sensing, are compared. Sonar tend to work well over long distances, but suffers from a lack of resolution and speckle (spurious acoustic reflections caused by the water and the particles suspended it it) which degrade the signal quality. Optical methods have better resolution than sonar, but suffer from diffusion of light in water, which limits the distance that usable images can be taken over. Nevertheless, station keeping functions best when optical techniques are used.

Han, Rhee, and Hong [12] describe an indoor robotic navigation system. The system places a structured target on the ceiling of the room in which the robot operates. (It is assumed that the target can be imaged from anywhere in the room.) The camera, which is mounted on a pan and tilt head, is adjusted so that the target is in the center of the image. The orientation of the robot with respect to the target is then determined by finding the pan and tilt of the image from some reference position.

Hallset [11] describes a system to follow sub–sea pipes. The pipes are assumed to be laid on the sea floor, and an autonomous underwater vehicle (AUV) is used to inspect these pipes. The AUV is maneuvered so that the on-board camera is above and looking down onto the pipe. The images are processed to highlight the large rectangles in them. The camera position allows the assumption that the pipe is imaged as a large rectangle. These images are used to correct the orientation of the AUV with respect to the pipe. Other sensors (a sonar ranger and a compass) are

used to keep the AUV moving at the proper height and in the correct direction.

In [22], Negahdaripour *et al.* describe a system for sub–sea station keeping. Several systems for detecting position changes are considered, including using a satellite transponder, laying grids of transponders, and camera imaging of locally available features. The system described relies on camera imaging of local features (such as rocks). Such systems are the most general as well as possessing the best resolution in the sensor. Position changes are detected by measuring the optic flow field observed by the camera. Methods of correcting for illumination variation are also discussed.

Allen, Yoshimi, and Timcenko [2] discuss a manipulator control system that can track the 3–D position of an moving object and pick it up. A pair of calibrated but unregistered cameras is used to image the scene. (A calibrated camera is a camera where precise geometric and perspective parameters are known. A pair of registered cameras is a pair of cameras where the horizontal scan lines in each have been aligned to be parallel to each other.) The object is located using optic flow fields. The detected flow fields are then used to extract the 3–D position of the object. This calculated position is used to update the position of the manipulator arm. (The implementation requires a pipelined parallel computer to work in real time.)

Papanikolopoules, Khosla, and Kanade [26] present a system to track an arbitrary 3–D object's motion in two dimensions. The system combines the vision system into the manipulator's control logic to achieve the best possible results. A cross–correlation based optic flow technique known as "Sum—of—Squared Differences" (SSD) is used to locate the object. The SSD measure selects the displacement $\mathbf{d} = (u, v)$ that minimizes the measure

$$e(\mathbf{p_A}, \mathbf{d}) = \sum_{(m,n)\in N} [I_A(x_A + m, y_A + n) - I_B(x_A + m + u, y_A + n + v)]^2$$

where I_A represents the image intensity of the original image, and I_B represents the image intensity of the second image which was taken some time after the first, and N is some window about the point p_A. The derived motion parameters are used as control inputs to the manipulator control law. Three general control laws (a Proportional—Integral law, a Linear Quadratic Gaussian technique, and a Pole Assignment controller) have been implemented.

Verghese, Gale, and Dyer [36] describe a model based 3–D motion tracking system. For each object to be tracked, there exists a model of the object in an object database. This model uses the edges of the object as the feature it stores. Two slightly different approaches are described for tracking. The first method locates the object, extracts the edges, and fits these edges to the model stored in the database. The orientation and position of the object can be estimated from the configuration of the edges in the image. The second method builds on the first method; using the assumption that the changes between subsequent frames of images is small, it uses the last known pose and orientation to find what new configuration of the model best fits the current image. This second method dealt with rotation better than the first approach, as a rough rotation is already known. Both methods require implementation on parallel computers to work in real time.

Rehg and Witkin [29] describe a general tracking system that has been applied to fields ranging from part removal from a conveyer to cell edge motion tracking. The system is restricted to tracking objects in two dimensions. The object to be tracked is modeled as a patch, which is a 2–D sheet with the precise shape being parametrically defined. The patches in sequential images are related by two energy functions. The first is the SSD optic flow computation described above, and the second is a light/dark spot seeker. (The light/dark seeker tries to find areas in the

patch of brightest/darkest intensity; when evaluated over the pixels in the blob it yields a intensity signature for the blob. This signature can often be compared more efficiently than using correlations.) An object is tracked by finding the best patch in each image and then determining object motion by finding the minima of the energy functions.

A local company, Terra Surveys of Sidney, BC., described the need for a target tracking system for a water weed control device [28]. A large roto–tiller is used to dig up the water weed beds; to do this accurately, the range to a reference point must be known. Range is best measured using a time–of–flight laser, but a laser range finder does not track the target well, particularly in the presence of large disturbances in platform position such as large underwater rocks which are commonly encountered. Thus, more robust methods of keeping the range finder on target are required.

In summary, there is a large interest in locating, tracking, and servoing based on image analysis of specific objects. Most of the literature studied the use of passive targets, and several successful (short range) methods have been described. However, most real–time implementations require exotic hardware.

1.2 Target Selection

All visual servoing locates the camera in the frame of reference of a target, as illustrated schematically in Figure 1.1. There are two basic target types that can be used. The first type is a passive target, the second is active. Passive targets reflect ambient light into the camera. This imposes several restrictions on the target. First, the target must be clearly illuminated. In some applications, particularly in sub–sea work (see [14]), this is hard to guarantee. Second, the target must be clearly distin-

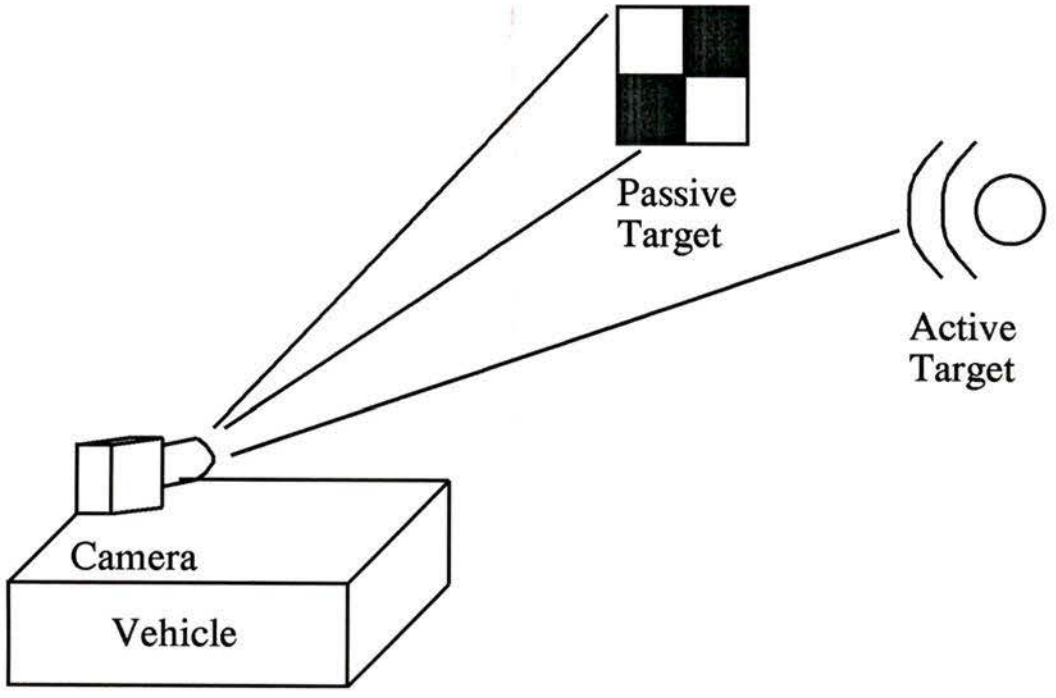


Figure 1.1: Camera and Target Configuration for Visual Servoing

guishable from any background clutter. This can lead to unobtainable target sizes; for example, Terra Surveys' weed destroyer must operate up to 1 km from the target, leading to the requirement for a 2×3m (minimum) target, which if nothing else would be subject to extreme wind loading. (The calculation for this figure is presented in Chapter 2.) Active targets, on the other hand, radiate their own light. This allows them to be structured specifically to remedy the shortcomings suffered by passive targets. However, they do require some power source. Also, active sources usually present very small targets [28] which do not have an easily identifiable edge structure which can be used for accurate positioning.

The goal is to investigate the location of active targets using low cost image processing systems.

1.3 Thesis Synopsis

There are many applications that require accurate location and tracking of targets using visual techniques. As has been outlined above, passive targets are inappropriate for many of these applications, leading to our concentration on active targets. This thesis will consider the problem of accurately locating the position of a small, active target using common, commercially available video cameras. We refer to the images produced by these as "blobs."

The thesis is divided into six chapters. Chapter 2 describes how commonly available cameras operate, the basic assumptions made about the target and how it is digitized, and the errors that are inherent in the acquisition of digital images. This discussion forms the basis of the theoretical analysis of each of the blob location techniques. Chapter 3 presents a literature review and a theoretical analysis of the method

of centroids as it applies to blob location. Chapter 4 does the same for two different methods of model parameter estimation techniques. Chapter 5 describes an experimental suite designed to implement and validate the methods described in Chapters 3 and 4. Finally, Chapter 6 will present a summary of the lessons learned during the course of the simulations and experiments. This will be followed by suggestions for future research in the area of visual servoing.

Chapter 2

Digital Imaging

This chapter describes the imaging process and its inherent errors. The chapter is divided into three sections. The first section describes the camera and how it converts the light falling on it into a digital image. As well, the initial assumptions about the blob will be presented. The second section describes the theoretical errors present in digital imaging. The third section introduces an error bound measure found in the literature.

2.1 Cameras and Image Acquisition

In this thesis, we assume that a Charged Coupled Device (CCD) video camera is used to obtain images of active targets. CCDs are set up as arrays of cells, each cell corresponding to a small segment of the picture (called *a picture element* or *pixel*). The charge stored by each cell depends on the amount of radiation (usually, this is light, but it need not be) falling onto it.

The geometric relationship between the object being imaged and the image formed by the camera lens is shown in Figure 2.1. (The half field of view angle θ is measured

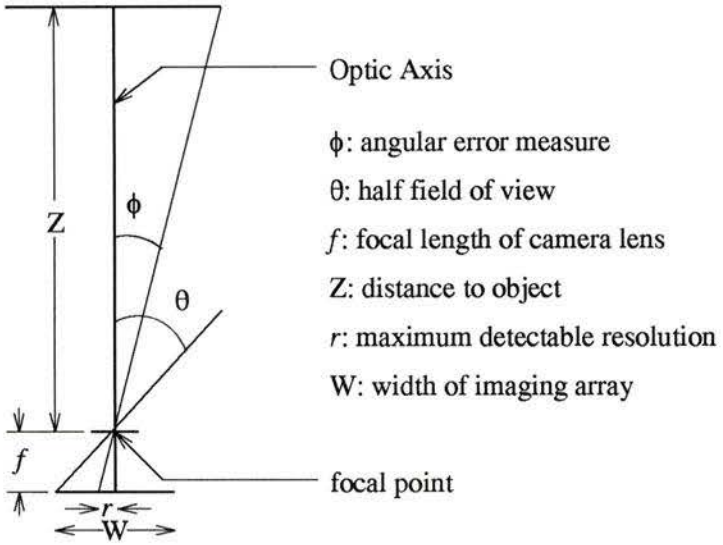


Figure 2.1: Geometric Relationship between Camera Image and Object

from the optic axis.) The image of the external scene is formed by light passing through the camera lens, intersecting at a point called the "focal point," and then falling on the image array (the CCD). (This is the pinhole model for a camera. It assumes that the lens introduces no distortion, which is usually a good assumption.) The relationship between the light entering the lens and where it falls on the imaging plane is modeled by the perspective transform. In one dimension (for example: the horizontal plane), this transform can be modeled as

$$u = \frac{fX}{Z} \tag{2.1}$$

where u is the pixel coordinate of a point that is located a distance Z from the camera and with a displacement X from the camera's optical axis. f is the focal length of the lens (typically, $f = 1.8, 3.5, 6, 10, 12.5, 16, 25, 50,$ or $100mm$).

Once the image has passed through the lens and onto the CCD, it is transmitted to a computer. Commonly, this transmission is done using coaxial cable and the RS–

170 black & white video standard. (RS–170 is compatible with the colour television standard NTSC.) The signal then is processed by a frame grabber which converts the RS–170 signal to a digital array in which the image data are spatially sampled and quantized in magnitude.

Two kinds of targets can be imaged for the purpose of visual servoing. Passive targets rely on reflected light hitting the CCD to allow the visual structure to be detected. Thus, the target must be of a size that is visible by the CCD. The minimum required target size can be determined from the perspective transform equation (equation 2.1), the size of the CCD array and the best detection resolution possible, the lens focal length, and the distance to the target. For the purposes of discussion, local visual servoing application discussed with Terra Surveys of Sidney, BC. will be used as a guide to the target size calculations. (This application involves moving a platform around with reference to a fixed point; no permanent constructions are allowed to support this task. Further, the camera must have a wide field of view so that it can localize the target over a large area.) For the purposes of discussion, it is assumed that the Pulnix TM–7 camera is to be used. It has a CCD array size of 7.95 (H) by 6.45 (V) millimeters organized as 641×489 cells. Common frame grabbers digitize NTSC signals to 640×480 pixels.

The distance to the target will be set to $1000m$, a distance that Terra Surveys had an interest in controlling a platform over. Equation 2.1 allows the calculation of the target size for a given focal length and detection resolution. The field of view can be calculated with

$$\theta = 2\tan^{-1}\left(\frac{0.003205}{f}\right) \tag{2.2}$$

where 0.003205 represents half the width of the CCD (distances assumed to be me-

f (mm)	Smallest Resolvable Angle (mrad)	Field of View (degrees)
1.8	5.56	121.36
3.5	2.86	84.96
6.0	1.67	56.22
10.0	1.00	35.54
12.5	0.80	28.76
16.0	0.63	22.65
25.0	0.40	14.61
100.0	0.10	3.67

Table 2.1: Trade–off Between Lens Focal Length, Smallest Angle Resolvable, and Field of View at a Distance of 1 km with 1 pixel Resolution

ters). Angular resolution ϕ is found by

$$\phi = \tan^{-1}\left(\frac{cr}{f}\right) \tag{2.3}$$

where c is a constant that converts pixels to meters, r is the location resolution, and f is the lens focal length. (Refer to Figure 2.1 to see this displayed schematically.) Table 2.1 summerizes the smallest angle that can be resolved, together with the field of view for the common focal lengths listed above assuming the image coordinates can be resolved to a single pixel. Thus, given a lens with focal length of $10mm$ and the ability to resolve images to 1 pixel, the smallest angle that can be measured over a kilometer is $1.00m$rad with a maximum field of view of 36 degrees. (For this application the single pixel resolution is appropriate. All sub–pixel resolution schemes rely on the fact that the object covers more that one or two pixels to interpolate the exact position.) If the target is centered with respect to the optic axis of the camera the minimum displacement (at 1 km) that can be localized is $1m$. This means that a specially patterned target, marked with a checker board, must be at least $2 \times 2m$ in size to be resolved by the camera. (Note that this is a very contrived case. An object

that is only 4 pixels in size (2×2 pixels) will be very difficult to find.) Such a size is too large, considering just the wind loading the target would have to sustain.

Active targets are not restricted to reflected light. Since the target emits light its apparent size decreases very slowly, allowing them to be imaged successfully over the 1 km distance without their spatial extents shrinking away to nothingness ([28]). This allows the target to have a large enough spatial extent to be able to estimate location to sub–pixel accuracy.

One of the primary specifications Terra Surveys had was that the camera have a large field of view to insure that it could find and track the target in the presence of vehicle disturbances. A minimum acceptable view field was determined to be 50 degrees. The second requirement was that the camera system be able to resolve the target position to at worst 1 milliradian, preferably 0.2 milliradians. Examining Table 2.1, we see the 1.8, 3.5, and $6.0mm$ lenses would meet the field of view requirement. For the $6.0mm$ lens and resolution $r = 1.0$ pixels, the angular resolution $\phi = 1.67$millirad. For a resolution $r = 0.10$ pixels, $\phi = 0.17$millirad. Using a $6mm$ lens would provide a compromise between field of view and resolution. With a $6mm$ lens, active targets must be located with an accuracy of 0.10 pixels. In order to insure that this goal was met, we decided to set a resolution of 0.05 pixels as our target accuracy.

In this thesis we will determine the accuracy with which we can locate blobs. As an guide, we desire to locate the blobs produced by active targets with an accuracy of 0.05 pixels. This is twice the accuracy reported in some of the literature (for example, [5]; see Chapters 3 and 4 for more details).

2.2 Theoretical Errors in Imaging

As has been indicated above, digital imaging does not capture a perfect likeness of the real world. Several errors are present at various stages in the imaging process. These error sources are:

1. Spatial Sampling. Due to the discrete nature of the imaging device (CCD) and the sampling inherent in the frame grabbing operation, the data exists only at sampled points in the 2–D image plane.

2. Amplitude Quantization. Images usually are obtained from a frame grabber or similar device. These devices have a limited resolution (in bits) with which to represent real (that is, continuous) functions. Quantization errors can be modelled as additive noise.

   It is most probable that a subset of the frame buffer's total resolution will be used for capturing Gaussian like pulses.

3. Imaging Noise. Any real sampling system adds some noise to the signal as it is being processed. This work will model this noise as an additive process.

4. Line Jitter. The transfer of image data from the camera to the frame grabber using the television standard for video information is subject to subtle, systemic errors.

The rest of this section will examine these error sources, beginning with spatial sampling. After that analysis, the effects of quantization and additive noise will be considered. Lastly, line jitter will be discussed.

2.2.1 Spatial Sampling

Real world data extends infinitely around an observer. The data in images does not. This is due to the fact that the CCD is made of a finite array of bins. The CCD is only able to image that part of the scene around it that is larger than its bin size and smaller than its total width. Thus, when an image of an object is taken, discrete parts of the continuous whole are stored. The well known Nyquist theorem states that in order to preserve all the information in the original signal, the signal must be sampled at twice the maximum frequency in it [24].

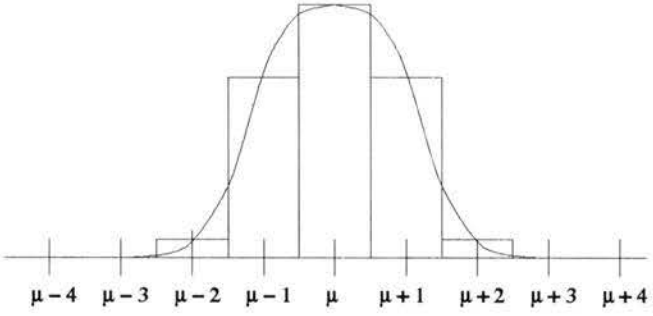
A diagram of the sampling process is found in Figure 2.2. It shows the entity function (a Gaussian pulse) being spatially quantized. The height of the boxes represents the magnitude of the pulse at the sample point (assumed to be the middle of the bin). Note that the sampled distribution can become asymmetric if the pulse mean is not aligned with the sampling interval.

We denote the unsampled intensity distribution of the image falling on the CCD by the continuous function $R(x, y)$. The sampling process then transforms this function using the sampling operation S to obtain the spatially sampled function $I(i, j)$:

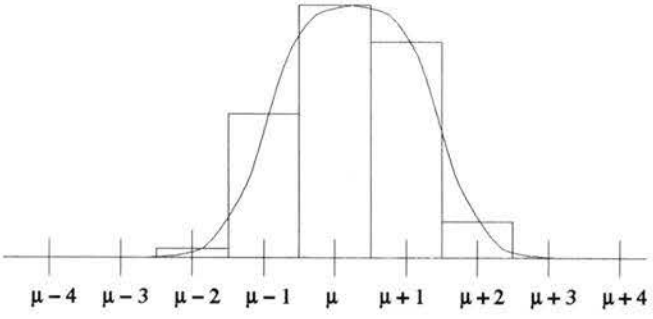
$$I(i, j) = S(R(x, y)) \tag{2.4}$$

2.2.2 Amplitude Quantization

The quantization process Q converts the real valued data $I(i, j)$ that is output from the sampling process S into a finite word representation $e(i, j)$. The finite word size is due to the fact that the computer hardware that this process is implemented on has a fixed number of symbols with which to represent the real quantity. (The number



(a) Pulse Mean Aligned with Sampling Interval



(b) Pulse Mean Shifted a Sub-Pixel Amount w.r.t. Sampling Interval

Figure 2.2: Sampling Process Diagram

of symbols in a digital computer is 2^n where n is the number of bits in the quantizer. Typical values of n are 8, 10, and 12 bits.)

$$e(i,j) = Q(I(i,j)) = Q(S[R(x,y)]) \tag{2.5}$$

This process rounds $I(i,j)$ to the closest integer.

This rounding introduces an error in the representation of $I(i,j)$. To develop a sense of what the error function looks like, a Gaussian function is evaluated at fixed intervals with the distribution mean being in the middle of the window. Several window sizes were used (15, 25, 33, and 50 steps wide), and in each window the Gaussian was evaluated with several variances. The variances were stepped from

0.5 to 9.5 in increments of 0.5. At each evaluation of the Gaussian, the generated real valued result was "quantized" by rounding to the nearest integer and the error between the real function value and the quantized version was noted. (Errors were tracked to 3 decimal places.) A normalized histogram of the error distribution is shown in Figure 2.3. This noise process has a magnitude strictly less than one. This

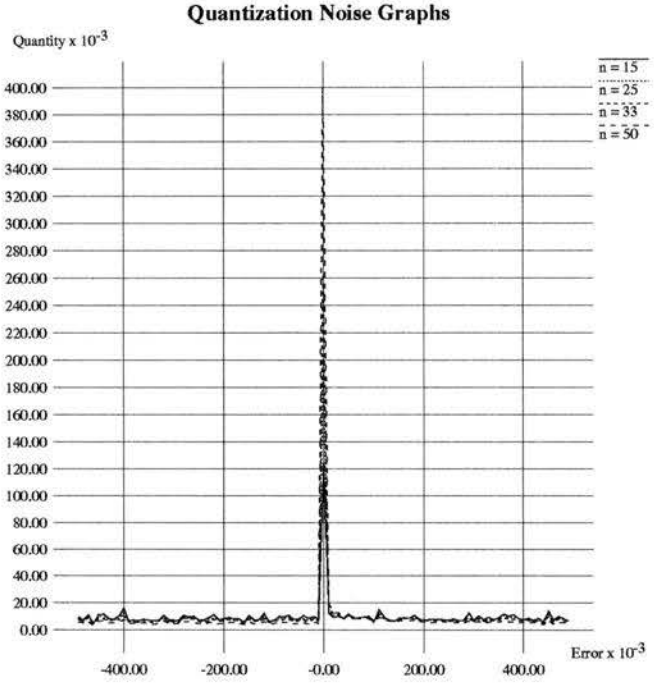


Figure 2.3: Quantization Noise Function

is visible in Figure 2.3 (where the error ϵ lies in the range $-0.5 < \epsilon < 0.5$, and can be deduced from the fact that the quantization function rounds the values to the nearest integer value. The noise looks like a uniform process with the exception of the spike at 0. This can be explained in terms of the generating function: the farther away a Gaussian is evaluated from its mean, the closer it is to 0 (a value it reaches only in the limit). The method used to create this graph did use narrow Gaussians, so that

a slight increase in the number of errors close to zero is to be expected. Examining the derivation in [15, Chapter 4], we see that the noise can be modeled as a uniform probability density function on the range (-0.5,0.5).

The above discussion suggests an additive, input independent process. This is not the case when processing images, however. The error in each sample of the blob is not independent of the rest of the samples; indeed, knowing the blob distribution and the error in one sample would allow perfect reconstruction of the blob. A better model for the quantization noise would be (as found in [15, page 163]) a gain followed by an additive component. Mathematically, this process is

$$e(i,j) = Q(I(i,j)) = \alpha I(i,j) + u(i,j) \tag{2.6}$$

where α is the gain component (which is less than unity) and $u(i,j)$ is the uniform, additive process with range $(-0.5, 0.5)$ intensity levels.

2.2.3 Additive Noise

Quantization is not the only noise source in this system. Other possible noise sources include: the cable and connector that join the camera to the frame buffer; non–linearities in the actual digitizing process; and subtle lens distortion. All these processes make it difficult to properly digitize the signal. These noise sources are hard to model, especially as their effects can be time varying. In order to get some idea of how the system responds to additional noise, a noise model of an additive, zero mean, single pixel standard deviation Gaussian process was selected. This is described as

$$f(i,j) = Q(I(i,j)) + n(i,j) = (\alpha I(i,j) + u(i,j)) + n(i,j) \tag{2.7}$$

were $n(i,j)$ is the zero mean, single pixel standard deviation Gaussian.

2.2.4 Line Jitter

Line jitter describes an error that is inherent to all imaging systems that use television type (RS–170) signals. The RS–170 signal defines the start and end of each horizontal line of data and the start of a new frame of data using pulses embedded into the data stream. Due to the electronic components used in the cameras, the timing of these synchronization signals drifts slightly through the time taken to transmit each frame. As can been seen in Figure 2.4,[1] each line appears to start somewhat before first line. (Figure 2.4 shows the line number along the vertical axis, with the line numbers being counted from top to bottom. This is the TV standard. The line shift is on the horizontal axis; the total shift from top to bottom of the field is ≈ 0.25 pixels.) This causes the signal on the line to appear shifted somewhat in space with respect to the preceeding line. As described in Lenz and Fritsch [19], the jitter is constant to within a smaller value ($\frac{1}{20}$ of a pixel in the camera system they report [19, page 97]) after the camera has warmed up.

In the one dimensional case, specifying that all measurements are to be constrained to be horizontal allows the jitter to be either be calibrated (as it is reasonably constant for each line), or be ignored (in those systems were relative displacement is more important than absolute displacement). Pixel jitter potentially plays a larger role in the two dimensional case, due to the fact that the jitter increases for each line from the top of the screen. However, the total jitter (top of the image to the bottom) is on the order of $\frac{1}{4}$ pixel. The most jitter occurs in the top 150 lines of a field.[2] The blob image can then be constrained to fall in the lower portion of the camera, reducing

[1] This figure is Figure 3 in [19]. Used without permission.

[2] Television (ie: RS 170) signals divide an image into two fields. Thus, for an image with 512 rows there will be two fields of at least 256 lines. The first field has the even rows, the second, the odd.

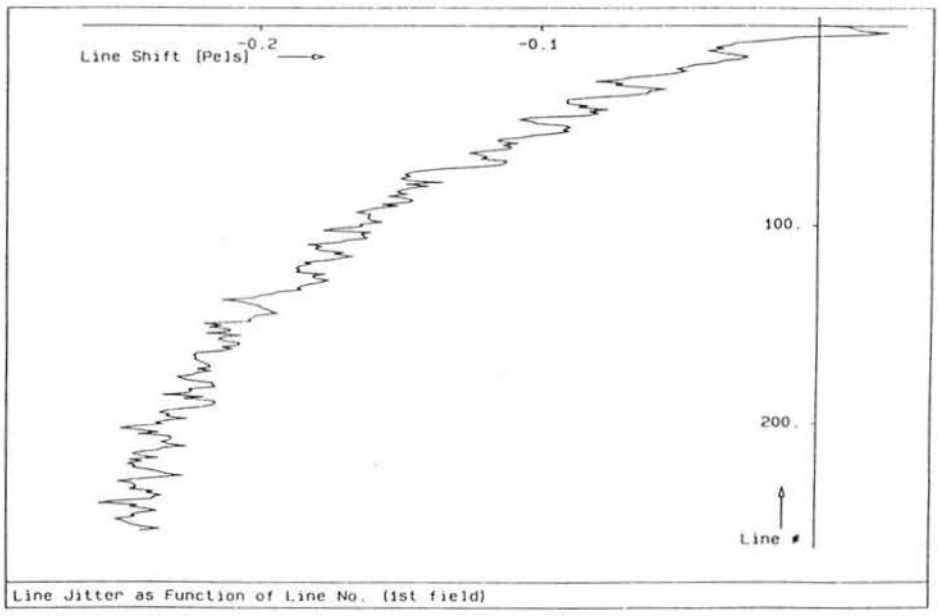


Figure 2.4: Pixel Jitter in RS 170 Signals as a Function of Line Number

the total effect of the jitter. If the spatial extent of the blob is small—on the order of 6 to 10 rows, then the total jitter down the blob image will be very small. Lastly, the precise jitter in a camera–image digitizer can be calibrated and so if need be removed from the system. (Removing the jitter requires spectral analysis of the image to find the phase of the *sel* clock signal. The exact details are in [18, pages 95—98].) Thus, pixel jitter can be eliminated as a significant noise source in both the one and two dimensional cases.

2.3 Absolute Performance Limits

The goal of this work is to determine the accuracy with which the location of active targets can be measured using low cost vision systems. The errors described in the previous section detract from achieving perfect accuracy. Given these errors, a

theoretical performance bound on blob location accuracy would be nice to have.

Several attempts to set performance bounds have been attempted. Havelock [13] considers a processing noise–free case. When signals (called entity functions) are quantized, some information is lost. In the case of the Gaussian entity function considered, the lost information relates to the precise position of the distribution. The visible result of this loss is the fact that several (small) changes in position (that is, small displacements of the mean) quantize to have the same digital representation. These common digital representations are called 'locales' by Havelock [13]. The size of the local determines how uncertain the position estimate is. This technique has two main deficiencies. First, the technique is very dependent on the parameters of the entity function used; locales generated for a Gaussian with a variance of 1 and amplitude 5 are completely different from the locales generated by a Gaussian with a variance of 1.5 and an amplitude of 5. Second, the effects of processing noise are not considered. This is due in large part to the difficulty in modelling such noise. Nevertheless, such noise might move certain entity function configurations from one locale into another.

Ping Wah Wong [37] describes how to find the mean absolute error and its probability distribution, building on a previous work by Kamgar–Parsi and Kamgar–Parsi [17]. The quantization error function is defined as

$$\epsilon = h\left(q(X_1), q(X_2), \ldots\right) - h\left(X_1, X_2, \ldots\right)$$

where h is the operator being studied, X_i are the quantities that h operates on, and $q(X_i)$ are the quantized quantities. A Taylor's series expansion is performed on this error function, where the high order terms (> 1) are neglected. Fourier transforms are used to evaluate the maximum error and its probability density function. (The use

of Fourier transforms is the primary difference between Wong and Kamgar–Parsi and Kamgar–Parsi; it makes Wong's work slightly more general.) The main stumbling block with this procedure is the requirement that the high order terms of the Taylor series be negligible. For the techniques considered in the thesis, the high order terms were not negligible, or the Taylor's series couldn't be formulated in the first place. (It is impossible to take the derivative of a linear least squares operation other than numerically.)

In summary, methods of determining performance bounds have been presented in the literature. These techniques, however, either do not include all noise sources or cannot be applied to the techniques under study. It is therefore impossible to predict absolute performance limits for the systems we are interested in.

This chapter has examined the process of imaging using common video equipment. Two possible target types (passive and active) were examined. Active targets were selected as the focus of this work as they were best suited to an interesting local application. This application set the desired location resolution, which is 0.05 pixels. An attempt was made to predict the performance limits, but proved unsuccessful.

Chapter 3

Blob Location Using Centroids

This chapter will describe the method of centroids for use as a blob location technique. The discussion is broken into six sections. The first section will outline how centroids have been used in the past. The second and third sections describe the 1–D formulation and examine the susceptibility to errors. The fourth and fifth sections do the same for the 2–D formulation. The last section summerizes the results.

3.1 Centroid Calculations

One popular method for locating the center of an entity function (a blob, shape, or edge) is to use some statistical moment (or combination of moments) of the image data. Spatial moments of image data are defined as

$$M_i = \sum_x x^i I(x)$$

where i represents the moment number, x the location of the data, and $I(x)$ the image data at the point. Some of the reports using moments are described below.

Alt [3] describes a character recognition scheme that uses stored prototypes and binary images. The system must be able to identify character maps that have been

translated, scaled, sheared, and/or had random noise added to them. A statistical pattern recognition scheme was created that used various moments of the data to define the classification features.

Abutaleb *et al.* [1] report on a system that tracks the passage of food through a person's upper intestinal system. The subject is fed some solid food that has a radio–active marker added to it. After enough time has passed to allow the food to begin digestion, the subject's abdomen is imaged. The progress of digestion can be monitored by tracking the spread and location of the marked food. The location of the bulk of the food is determined by finding the centroid of the imaged tracer food. The spread of the food through the stomach and upper intestine is determined using higher moments.

Bose and Amir [5] describe a system to position a printed circuit board (PCB) accurately. The system requires that the PCB have some easily recognizable shapes (called "fiducials") on the board. These fiducials are digitized to create a binary edge map. The precise location of the shapes is determined by finding their center using the first moment.

Bar–Shalom et al. [4] describe a system to track the position of a Gaussian plume (assumed to be a jet engine exhaust). The system finds the current location of the plume by finding its centroid. The tracking filter uses the the correlation between two consecutive image frames to find how the far the plume has shifted.

An automatic parts inspection system is described by Cleaver and Su [7]. In this system, a 'golden' image of a perfect part is taken. This image is registered with the image of a test piece by locating the centroid of each image. Once the difference in the centroids is removed, the test image can be compared directly to the golden

image.

Tabatabai and Mitchell [35] describe an edge detection scheme that uses intensity rather than spatial moments. One–dimensional intensity moments are defined as

$$\overline{m}_i = \frac{1}{n}\sum_{j=1}^{n} I(x_j)^i \tag{3.1}$$

where i refers to the degree (first, second, ...) of the moment, n refers to the number of elements considered, and $I(x_j)$ refers to the intensity at location x_j. This scheme assumes the step edge model for edges, and is presented in one and two dimensional forms. (The step edge model has two tails of constant but different intensity on either side of a discontinuous edge.) The operator works by preserving the first three moments of the data to locate the position of the edge. The main restriction on the data is that it be monotonic increasing (or decreasing). This means that the sample (or image) data is used to define the location of a step edge such that the three intensity moments for both the original image and the new step edge are the same. In the one dimensional case, this requires solving three equations. The two dimensional case is more complex, as the edge could take any orientation and position. A disk is used to implement the operator. The area covered by both tails of the edge are determined using the moment preservation technique. This provides the position of edge; the orientation is found by finding the center of gravity (of the intensities) of both tails of the edge.

Lyvers *et al.* [20] extend the edge detection work of Tabatabai and Mitchell [35]. Both a one and a 2–D case are presented. The 1–D case is solved by calculating the first three spatial moments of the line of data. The three moments allow accurate placement of the edge (to 0.1 pixels); moreover, the error is systemic so it can be removed through the use of a look–up table. The two dimensional case described is

somewhat more complex. Using several instances of the one dimensional operator proved to be poorly behaved, so two dimensional moments were used to derive a circular operator. To simplify the edge location, the higher moments are used to find the angle which the line makes with the x-axis and then the data set is rotated so that the resulting line is parallel to the y-axis. After the rotation takes place, the edge can be found in a manner similar to the one dimensional case. The location error can be corrected by use of a look-up table.

Reviewing these works indicates that the centroid method shows promise for our application. The centroid method has many advantages. First, it is template independent; it does not require that a sample or model of the object being sought be kept. Second, it is resolution independent; it will work as well on an image of size five pixels as on an image of size 50 pixels. Resolutions of 0.1 pixels have been reported in several different works (for example, [5, 35]).

The discrete, weighted centroid calculation as defined in [5] and [4] is[1]

$$C_x = \frac{\sum_{all(i,j)\in P} iP(i,j)}{\sum_{all(i,j)\in P} P(i,j)}, \quad C_y = \frac{\sum_{all(i,j)\in P} jP(i,j)}{\sum_{all(i,j)\in P} P(i,j)} \tag{3.2}$$

where P represents the set of pixels belonging to the object, and i, j are the column and row coordinates of the set P in the image.

3.2 One Dimensional Centroid Formulation

This section describes the calculation of the mean of a distribution using centroids. This discussion isolates a systemic error source, which leads into the error analysis of

[1]These formulae are from [5, page 1197]. Note that $P(i,j)$ represents the intensity of the pixel at the point (i,j), and that P is the set of all pixels belonging to the feature.

the next section.

Assume that the continuous real valued intensity function of a blob image is Gaussian, with the form

$$f(x) = Ae^{-\left(\frac{x-\mu}{\sigma}\right)^2}$$

It has infinite extent (spatially), amplitude A, mean μ, and variance σ^2. For a sampled Gaussian, an estimate of the mean is $\hat{\mu}$:

$$\hat{\mu} = \frac{\sum_{i=-\infty}^{\infty} iAe^{-\left(\frac{i}{\sigma}\right)^2}}{\sum_{i=-\infty}^{\infty} Ae^{-\left(\frac{i}{\sigma}\right)^2}} \tag{3.3}$$

This can be reduced as follows:

$$\hat{\mu} = \frac{\sum_{i=-\infty}^{\infty} ie^{-\left(\frac{i}{\sigma}\right)^2}}{\sum_{i=-\infty}^{\infty} e^{-\left(\frac{i}{\sigma}\right)^2}} \tag{3.4}$$

$$= \frac{\sum_{i=-\infty}^{0} ie^{-\left(\frac{i}{\sigma}\right)^2} + \sum_{i=0}^{\infty} ie^{-\left(\frac{i}{\sigma}\right)^2}}{\sum_{i=-\infty}^{\infty} e^{-\left(\frac{i}{\sigma}\right)^2}} \tag{3.5}$$

$$= \frac{\sum_{i=0}^{\infty} ie^{-\left(\frac{i}{\sigma}\right)^2} - \sum_{i=0}^{\infty} ie^{-\left(\frac{i}{\sigma}\right)^2}}{\sum_{i=-\infty}^{\infty} e^{-\left(\frac{i}{\sigma}\right)^2}} \tag{3.6}$$

$$= \frac{\sum_{i=0}^{\infty} (i-i)e^{-\left(\frac{i}{\sigma}\right)^2}}{\sum_{i=-\infty}^{\infty} e^{-\left(\frac{i}{\sigma}\right)^2}} \tag{3.7}$$

Equation 3.7 shows that $\hat{\mu} = 0$.

Generally, if the mean of the distribution is μ, μ an arbitrary value, the centroid

equation 3.2 becomes

$$\hat{\mu}' = \frac{\sum_{i=-\infty}^{\infty} i e^{-\left(\frac{i-\mu}{\sigma}\right)^2}}{\sum_{i=-\infty}^{\infty} e^{-\left(\frac{i-\mu}{\sigma}\right)^2}} \tag{3.8}$$

Introducing a change of variable $j = i - \mu$, therefore $i = j + \mu$, equation 3.8 becomes

$$\hat{\mu}' = \frac{\sum_{j=-\infty}^{\infty} (j+\mu) e^{-\left(\frac{j}{\sigma}\right)^2}}{\sum_{j=-\infty}^{\infty} e^{-\left(\frac{j}{\sigma}\right)^2}} \tag{3.9}$$

$$= \frac{\sum_{j=-\infty}^{\infty} j e^{-\left(\frac{j}{\sigma}\right)^2} + \sum_{j=-\infty}^{\infty} \mu e^{-\left(\frac{j}{\sigma}\right)^2}}{\sum_{j=-\infty}^{\infty} e^{-\left(\frac{j}{\sigma}\right)^2}} \tag{3.10}$$

The first term of the numerator in equation 3.10 can be reduced similarly as equation 3.7. Thus, the first term of equation 3.10 becomes zero, and equation 3.10 can be reduced to

$$\hat{\mu}' = \mu \frac{\sum_{j=-\infty}^{\infty} e^{-\left(\frac{j}{\sigma}\right)^2}}{\sum_{j=-\infty}^{\infty} e^{-\left(\frac{j}{\sigma}\right)^2}} \tag{3.11}$$

$$\hat{\mu}' = \mu \tag{3.12}$$

Clearly, the infinite sum correctly determines the mean (or centroid).

When calculating the centroid with image data, the signal is not infinite in spatial extent. Instead, the function is sampled with a window function that truncates it. For the purpose of this discussion, we assume that the function will select a total of either $2n + 1$ or $2n$ points centered about the maximum value in the distribution. A diagram of the sampling process is found in Figure 2.2.

In this context, equation 3.4 becomes

$$\hat{\mu}' = \frac{\sum_{i=-n}^{n} i e^{-\left(\frac{i}{\sigma}\right)^2}}{\sum_{i=-n}^{n} e^{-\left(\frac{i}{\sigma}\right)^2}} \tag{3.13}$$

(As the function is centered on a pixel, the left and right tails are mirror images of each other due to the symmetry of the Gaussian. Thus, the thresholding will always select $2n + 1$ terms for the calculations.)

The same argument that was presented before for shifting the mean to be μ, μ any integer, applies in this case. Thus, equation 3.13 becomes

$$\hat{\mu}' = \frac{\sum_{i=-n}^{n} i e^{-\left(\frac{i-\mu}{\sigma}\right)^2}}{\sum_{i=-n}^{n} e^{-\left(\frac{i-\mu}{\sigma}\right)^2}} \tag{3.14}$$

$$= \frac{\sum_{j=-n-\mu}^{n-\mu} (j+\mu) e^{-\left(\frac{j}{\sigma}\right)^2}}{\sum_{j=-n}^{n} e^{-\left(\frac{j}{\sigma}\right)^2}}, \; j = i - \mu \tag{3.15}$$

$$= \mu + \frac{\sum_{i=-n}^{n} i e^{-\left(\frac{i-\mu}{\sigma}\right)^2}}{\sum_{i=-n}^{n} e^{-\left(\frac{i-\mu}{\sigma}\right)^2}} \tag{3.16}$$

$$= \mu \tag{3.17}$$

The above discussion has centered on distributions where the mean falls on a pixel exactly. If the mean of the distribution is shifted by less than an integer quantity, then a subtly different process takes place.

When the mean is between two pixels, the tails are no longer completely balanced. Using equation 3.13 as a reference, and assuming a mean of δ, $0 < \delta < 0.5$, the

following derivation is obtained.[2]

$$\hat{\mu}' = \frac{\sum_{i=-n+1}^{n} i e^{-\left(\frac{i-\delta}{\sigma}\right)^2}}{\sum_{i=-n+1}^{n} e^{-\left(\frac{i-\delta}{\sigma}\right)^2}} \tag{3.18}$$

Making the substitution $j = i - \delta$, or $i = j + \delta$,

$$\hat{\mu}' = \frac{\sum_{j=-n+1-\delta}^{n-\delta} (j+\delta) e^{-\left(\frac{j}{\sigma}\right)^2}}{\sum_{j=-n+1-\delta}^{n-\delta} e^{-\left(\frac{j}{\sigma}\right)^2}} \tag{3.19}$$

$$= \frac{\sum_{j=-n+1-\delta}^{n-\delta} \delta e^{-\left(\frac{j}{\sigma}\right)^2} + \sum_{j=-n+1-\delta}^{n-\delta} j e^{-\left(\frac{j}{\sigma}\right)^2}}{\sum_{j=-n+1-\delta}^{n-\delta} e^{-\left(\frac{j}{\sigma}\right)^2}} \tag{3.20}$$

$$= \delta + \frac{\sum_{j=-n+1-\delta}^{n-\delta} j e^{-\left(\frac{j}{\sigma}\right)^2}}{\sum_{j=-n+1-\delta}^{n-\delta} e^{-\left(\frac{j}{\sigma}\right)^2}} \tag{3.21}$$

Replacing j with $i = j - \delta$,

$$\hat{\mu}' = \delta + \frac{\sum_{i=-n+1}^{n} (i-\delta) e^{-\left(\frac{i-\delta}{\sigma}\right)^2}}{\sum_{i=-n+1}^{n} e^{-\left(\frac{i-\delta}{\sigma}\right)^2}} \tag{3.22}$$

If the numerator of equation 3.22 is split into two parts, the first part consisting of the positive terms and the other, the negative, then

$$\hat{\mu}' = \delta + \frac{\sum_{i=1}^{n} (i-\delta) e^{-\left(\frac{i-\delta}{\sigma}\right)^2} + \sum_{i=-n+1}^{0} (i-\delta) e^{-\left(\frac{i-\delta}{\sigma}\right)^2}}{\sum_{i=-n+1}^{n} e^{-\left(\frac{i-\delta}{\sigma}\right)^2}} \tag{3.23}$$

[2]For simplicity, we assume that only $2n$ terms of the function have been selected by the windowing function.

results. The negative portion of equation 3.23 can be rewritten so that the summation uses positive indices, resulting in

$$\hat{\mu}' = \delta + \frac{\sum_{i=1}^{n}(i-\delta)e^{-\left(\frac{i-\delta}{\sigma}\right)^2} - \sum_{i=0}^{n-1}(i+\delta)e^{-\left(\frac{i+\delta}{\sigma}\right)^2}}{\sum_{i=-n+1}^{n} e^{-\left(\frac{i-\delta}{\sigma}\right)^2}} \tag{3.24}$$

$$= \delta + \frac{\sum_{i=1}^{n}(i-\delta)e^{-\left(\frac{i-\delta}{\sigma}\right)^2} - \sum_{j=1}^{n}(j+\delta-1)e^{-\left(\frac{j+\delta-1}{\sigma}\right)^2}}{\sum_{i=-n+1}^{n} e^{-\left(\frac{i-\delta}{\sigma}\right)^2}} \tag{3.25}$$

$$= \delta + \frac{\sum_{i=1}^{n}\left((i-\delta)e^{-\left(\frac{i-\delta}{\sigma}\right)^2} - (i+\delta-1)e^{-\left(\frac{i+\delta-1}{\sigma}\right)^2}\right)}{\sum_{i=-n+1}^{n} e^{-\left(\frac{i-\delta}{\sigma}\right)^2}} \tag{3.26}$$

Equation 3.26 can be rewritten as $\hat{\mu}' = \delta + \epsilon$, with

$$\epsilon = \frac{\sum_{i=1}^{n}\left((i-\delta)e^{-\left(\frac{i-\delta}{\sigma}\right)^2} - (i+\delta-1)e^{-\left(\frac{i+\delta-1}{\sigma}\right)^2}\right)}{\sum_{i=-n+1}^{n} e^{-\left(\frac{i-\delta}{\sigma}\right)^2}} \tag{3.27}$$

This error term ϵ represents the skew error (see Section 2.2). The skew error and the other errors will be analyzed below. The derivation for the case of $-0.5 < \delta < 0$ is only slightly different. The result is similar, with the precise details are shown in Appendix A, as is the derivation for the case where the windowing function selects $2n + 1$ elements instead of $2n$. Further, if the mean δ is shifted by some integer value η, it is easy to show that the centroid formula calculates the mean value to be $\hat{\mu}' = \eta + \delta + \epsilon$.

3.3 1–D Error Analysis

Each of the three remaining error sources from Section 2.2 is discussed under separate headings below. The discussion begins with spatial quantization, continues with intensity quantization, and finishes with the additive process noise.

Spatial Sampling

The error term (equation 3.27) clearly describes the effect of spatial quantization on infinite distributions. This can be seen by realizing that the centroid correctly calculates mean locations when the pulse is centered either on a pixel or half–pixel; in these two cases, the tails of the distribution balance on both sides of the mean. When portions of the distribution are spatially sampled in an unbalanced fashion (as in the subpixel shifts), the error appears. The error depends on three parameters: the window size used, the variance of the distribution, and the subpixel displacement.

To study the effect of spatial sampling in centroid calculations, real valued (unquantized) data is used. The theoretical error (equation 3.27) was plotted for various window sizes, variances, and means. A sample graph is shown below. The mean was shifted in steps of 0.02 pixels from -0.5 to 0.5 pixels about the center pixel (the window center) for standard deviations s, s = 1, 1.1, 2, 4.5, 9, and 18 pixels; with a constant window size ($2n + 1 = 9$ pixels). (The variances plotted were chosen so as to see the effects as the variance approached the window size. Also note that as the form of this graph is the same for other window sizes, other examples will not be given.) It is shown in two parts (Figures 3.1 and 3.2), due to the change in scale at the larger standard deviations.

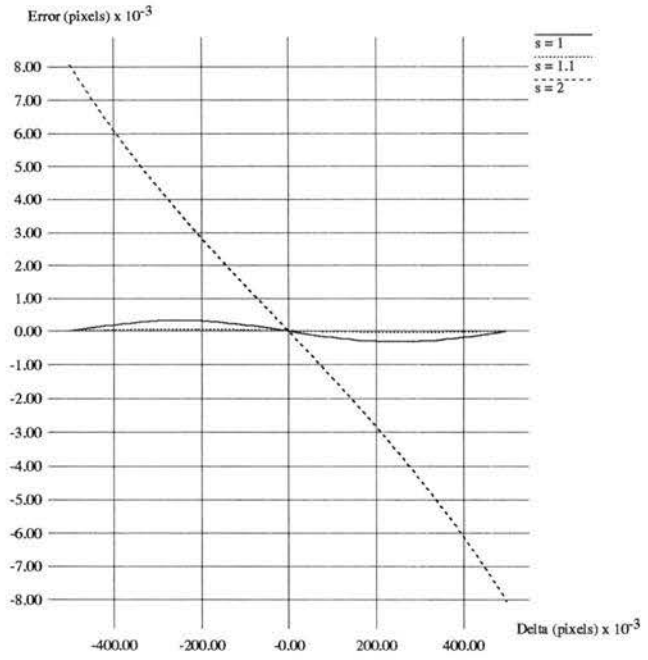


Figure 3.1: Skew Error for Constant Standard Deviation s and Window Size 9 pixels

In the first graph (Figure 3.1), with $s = 1, 1.1, 2$ pixels, we see that the small variances (s = 1 and 1.1) exhibit a very small systemic error. This is the skew error, due to the spatial sampling that has taken place. (Note that this error curve using equation 3.27 is identical to the error curve generated by the centroid equation 3.13.) We also note that the location error increases as the variances increase. When s = 2 pixels, the maximum error at a displacement of ± 0.5 pixels is less than 0.010 pixels. For s = 4.5 pixels, a standard deviation one half the window size, the maximum error is about 0.25 pixel. When s = 18 pixels, twice the window size, we see that the error is almost the sub–pixel displacement. To investigate this phenomenon further, other tests were run.

A series of tests where the subpixel displacement δ was held constant and the

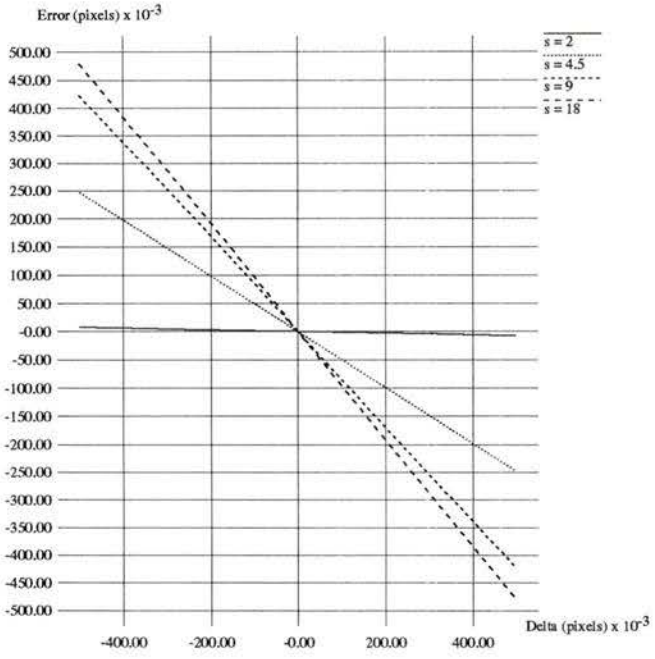


Figure 3.2: Skew Error for Constant Standard Deviation s and Window Size 9 pixels (Part 2)

standard deviations from 0.5 to 25 pixels were made. (These tests used the centroid calculation equation 3.13, not the skew error calculation equation 3.27.) Several tests with various window sizes were run. A sample graph (for $\delta = 0.3$) is shown in Figure 3.3 below.[3]

Here, the error decreases toward zero as the variance approaches 1. As the variance increases past 1, the accuracy begins to drop off until it reaches a asymptotic maximum for the given displacement. The effect of the sampling window size is to control the location where (in relation to the variance) the error curve drops away from zero.

[3]These graphs were created for $\delta = 0.1, 0.2, \ldots, 0.9$.

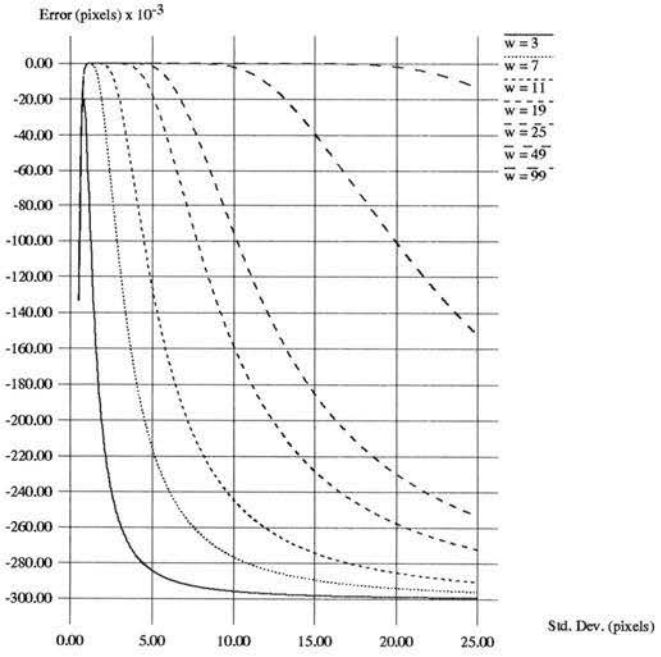


Figure 3.3: Skew Error for Constant Displacement $\delta = 0.3$ pixels and Window size w with Variable Variance

The behaviour demonstrated in these graphs is explained in terms of the Gaussian function. The standard deviation σ controls the width of the pulse. In fact, about 95% of the area under a Gaussian is found within $\pm 2\sigma$ of the mean. Thus, a variance of less than one constrains the sampled Gaussian to be very impulsive in nature.

As the variance increases, sufficient data is obtained to generate a good estimate, and the error decreases. However, as the variance increases with respect to the window size, the area covered by the centroid calculation decreases. Taken to the limit, the calculation becomes simply the average of $2n+1$ or $2n$ terms of the peak value. This reduces to simply the previous integer value, and so the error (in the limit) is the subpixel displacement δ.

These results suggest several important relationships. First, any data set generated should be constrained to $\sigma > 1$. Thus, impulse data is not acceptable. Second, the window size used to calculate the centroid should be set such that pulse data from $\pm 2\sigma$ about the mean is included (ie: $2n \geq 4\sigma$). This size insures that enough of the pulse is present to avoid significant averaging error.

Amplitude Quantization

The effects of amplitude quantization can be seen in the figure that shows the location errors for constant standard deviation s and window size 25 (Figure 3.4). This figure

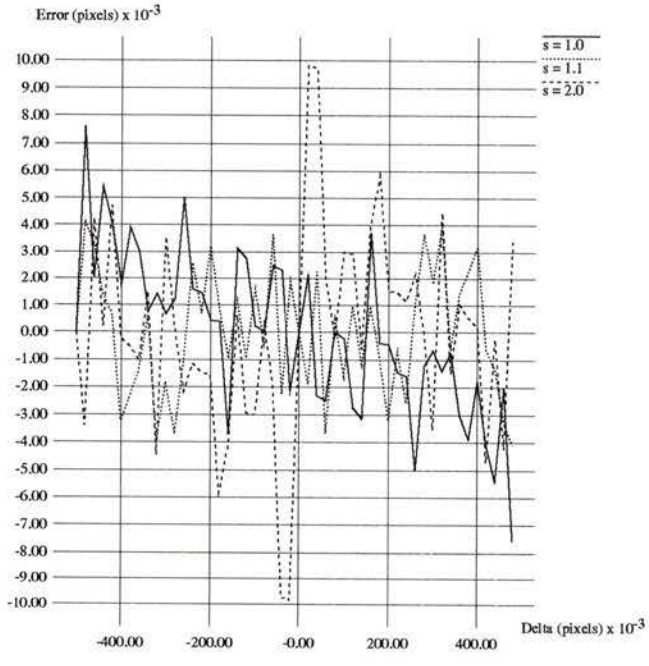


Figure 3.4: 1–D Centroid Location Errors for Quantized Data with Constant Variance s and Window Size 25 pixels

corresponds to Figure 3.1. The next example show the error curve generated with

a constant displacement, variable variance, and constant window size. The curve is in Figure 3.5. These plots show the same trend as the examples calculated without

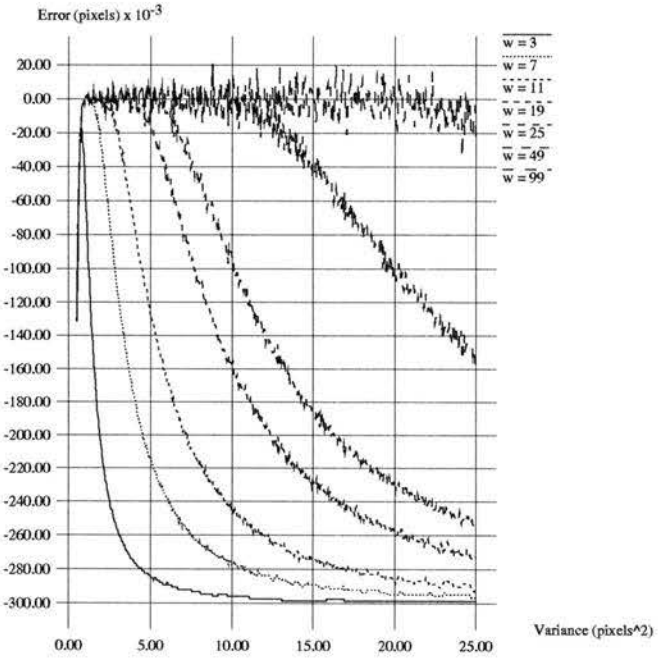


Figure 3.5: 1–D Centroid Location Errors for Quantized Data with Constant Displacement $\delta = 0.3$ pixels and Variable Window Size w

quantization. Note, however, while the error in Figure 3.4 is symmetric about the 0 sub-pixel shift, the simple skew error curve is replaced with a much more chaotic form. This is highlighted by Figure 3.5.

It is possible to calculate a Signal–to–Noise ratio (SNR) for these test. The definition used[4] is

$$SNR = \left(\frac{\sum_i \sum_j g^2(i,j)}{\sum_i \sum_j e^2(i,j)} \right)^{\frac{1}{2}}$$

[4]This definition is from [9, page 257].

Std. Dev. (pixels)	SNR
1.0	438.17
1.1	448.74
2.0	426.03

Table 3.1: Average SNR for each Standard Deviation s in Figure 3.4

where x and y are the image coordinates being processed, g is the noisy signal, and e is the noise added to the signal. For Figure 3.4, the mean of the SNR for the standard deviations are listed in Table 3.1. Note that the average SNR is quite high. The total error in these cases is larger than the error in the un-quantized case, but it is still within acceptable limits. (This table will be referred to below.)

Additive Noise

The centroid algorithm is sufficiently good to handle errors due to spatial and intensity quantization. This is not, however, the only noise source. Processing noise is added to the signal by the camera and frame grabber. A zero mean, single pixel standard deviation Gaussian process was chosen to model this. The effects of this noise were studied by running another set of trials as above, using quantized data with the Gaussian noise added.

In Figure 3.6, the test results for all possible subpixel displacements, small standard deviations (s = 1, 1.1, 2), and a window size of 7 pixels can be found. The noise adds enough error to the process to take the results out of the acceptable limits (that is, less than $\epsilon < 0.05$ pixels). Examining Table 3.2 shows that the SNR is about $\frac{1}{4}$ of the previous case. Clearly, large signal to noise ratios are required to insure that the

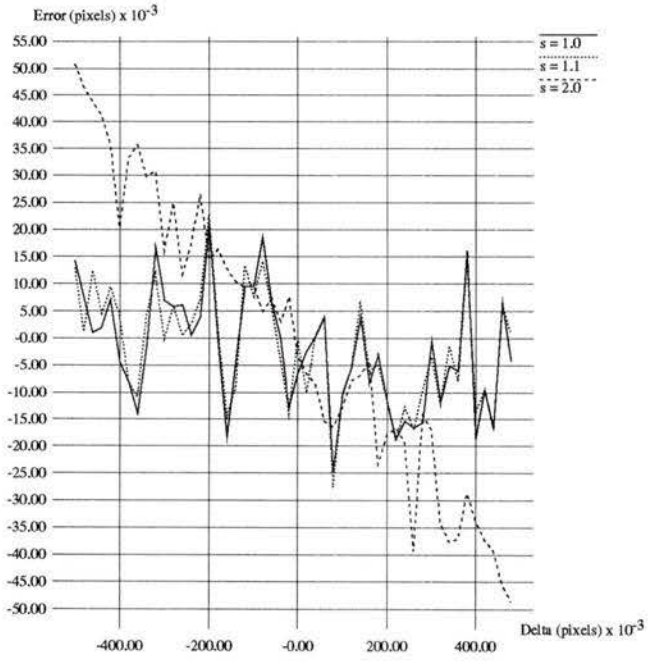


Figure 3.6: 1D Location Errors for Noisy, Quantized Data for Constant Standard Deviation s with Window Size 7 Pixels

Std. Dev. (pixels)	SNR
1.0	128.12
1.1	126.69
2.0	166.00

Table 3.2: Average SNR for each Standard Deviation s in Figure 3.6

centroid analysis produces results of sufficient accuracy. For Figure 3.6, the mean of the SNR for the standard deviations s are listed in Table 3.2.

This is further supported by the next graph, which shows the effects of changing the standard deviation for a constant displacement. (The graph is Figure 3.7.) The standard deviation for the pulse is varied from 0.5 pixels to 24.5 pixels. Three windows sizes are shown: 3, 9, and 19 pixels. The subpixel displacement is 0.3 pixels.

1D Centroid Calculation Summary

The noise free, un-quantized analysis suggested that a small, systematic error existed in the centroid calculation that could be easily modeled. Including just the intensity quantization effects, however, presented an error model that did have some symmetry about the origin, but was very sensitive to small perturbations in the estimated mean. Adding a process noise model with known statistics resulted in an error that is impossible to reasonably model. However, the simulations suggest that the total error is bounded. Further, through appropriate selection of the window size and a large SNR the error can be reduced to within acceptable limits (total error $\epsilon < 0.05$ pixels).

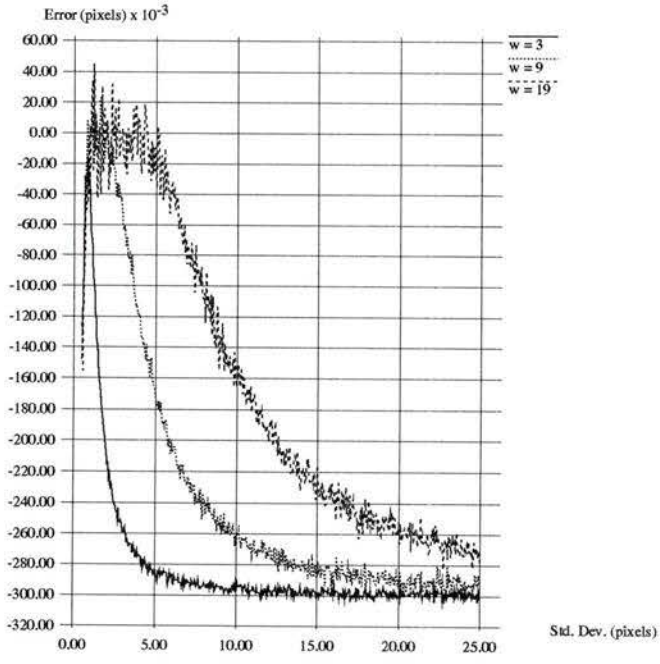


Figure 3.7: 1D Location Errors for Noisy, Quantized Data for Constant Displacement $\delta = 0.3$ with Window Size w

3.4 Two Dimensional Centroid Formulation

The two dimensional formulation follows the same lines as the one dimensional formulation. The two dimensional Gaussian has the form

$$\begin{aligned} f(x,y) &= Ae^{-\left(\left(\frac{x-\mu_x}{\sigma_x}\right)^2+\left(\frac{y-\mu_y}{\sigma_y}\right)^2\right)} \\ &= Ae^{-\left(\frac{x-\mu_x}{\sigma_x}\right)^2}e^{-\left(\frac{y-\mu_y}{\sigma_y}\right)^2} \end{aligned}$$

where μ_x, μ_y are the x, y means of the distribution and σ_x, σ_y are the standard deviations. (This is the circular symmetric case.) As the derivation starting from a zero mean ($\mu_x = 0, \mu_y = 0$) to an arbitrary position are the same as in the one dimensional case, it has not been repeated here.

In this context, the centroid equations become

$$\begin{aligned} \hat{\mu}_x &= \frac{\sum_{x=-\infty}^{\infty}\sum_{y=-\infty}^{\infty} xAe^{-\left(\frac{x-\mu_x}{\sigma}\right)^2}e^{-\left(\frac{y-\mu_y}{\sigma}\right)^2}}{\sum_{x=-\infty}^{\infty}\sum_{y=-\infty}^{\infty} Ae^{-\left(\frac{x-\mu_x}{\sigma}\right)^2}e^{-\left(\frac{y-\mu_y}{\sigma}\right)^2}}, \\ \hat{\mu}_y &= \frac{\sum_{x=-\infty}^{\infty}\sum_{y=-\infty}^{\infty} yAe^{-\left(\frac{x-\mu_x}{\sigma}\right)^2}e^{-\left(\frac{y-\mu_y}{\sigma}\right)^2}}{\sum_{x=-\infty}^{\infty}\sum_{y=-\infty}^{\infty} Ae^{-\left(\frac{x-\mu_x}{\sigma}\right)^2}e^{-\left(\frac{y-\mu_y}{\sigma}\right)^2}} \end{aligned} \tag{3.28}$$

For simplicity, the rest of the discussion will concentrate on the μ_x. The derivation for μ_y is the same.

The derivation of skew error continues as follows. The sampling process will select a window on the data; thus, assuming that the window function selects $(2n+1) \times$

$(2m+1)$ points, equation 3.28 becomes

$$\hat{\mu}'_x = \frac{\sum_{x=-n}^{n}\sum_{y=-m}^{m} xAe^{-\left(\frac{x-\mu_x}{\sigma}\right)^2}e^{-\left(\frac{y-\mu_y}{\sigma}\right)^2}}{\sum_{x=-n}^{n}\sum_{y=-m}^{m} Ae^{-\left(\frac{x-\mu_x}{\sigma}\right)^2}e^{-\left(\frac{y-\mu_y}{\sigma}\right)^2}} \tag{3.29}$$

where n, m are the number of points in the sampling window in the x and y direction (n need not equal m).

If the assumption that μ_x and μ_y are integer displacements, then the windowing process will select $2n+1$ and $2m+1$ points (as shown in equation 3.29). Noting that the tails of the distribution will be equal (in any direction), it can be seen that equation 3.29 will correctly estimate the mean; this is exactly the same as for the one dimensional case.

When subpixel displacements are made instead of integer displacements, the skew error returns. Using equation 3.29 as a basis, with μ_x and μ_y being replaced with the subpixel displacements δ and η (δ the displacement in x, η in y), the following is obtained:

$$\hat{\mu}'_x = \frac{\sum_{x=-n}^{n}\sum_{y=-m}^{m} xAe^{-\left(\frac{x-\delta}{\sigma}\right)^2}e^{-\left(\frac{y-\eta}{\sigma}\right)^2}}{\sum_{x=-n}^{n}\sum_{y=-m}^{m} Ae^{-\left(\frac{x-\delta}{\sigma}\right)^2}e^{-\left(\frac{y-\eta}{\sigma}\right)^2}} \tag{3.30}$$

Setting $i = x - \delta$, and $j = y - \eta$,

$$\hat{\mu}'_x = \frac{\sum_{i=-n-\delta}^{n-\delta}\sum_{j=-m-\eta}^{m-\eta} (i+\delta)Ae^{-\left(\frac{i}{\sigma}\right)^2}e^{-\left(\frac{j}{\sigma}\right)^2}}{\sum_{i=-n-\delta}^{n-\delta}\sum_{j=-m-\eta}^{m-\eta} Ae^{-\left(\frac{i}{\sigma}\right)^2}e^{-\left(\frac{j}{\sigma}\right)^2}} \tag{3.31}$$

$$= \delta + \frac{\sum_{i=-n-\delta}^{n-\delta}\sum_{j=-m-\eta}^{m-\eta} ie^{-\left(\frac{i}{\sigma}\right)^2}e^{-\left(\frac{j}{\sigma}\right)^2}}{\sum_{i=-n-\delta}^{n-\delta}\sum_{j=-m-\eta}^{m-\eta} e^{-\left(\frac{i}{\sigma}\right)^2}e^{-\left(\frac{j}{\sigma}\right)^2}} \tag{3.32}$$

Reversing the previous substitutions by setting $x = i + \delta$, and $y = j + \eta$,

$$\hat{\mu}'_x = \delta + \frac{\sum_{x=-n}^{n}\sum_{y=-m}^{m}(x-\delta)e^{-\left(\frac{x-\delta}{\sigma}\right)^2}e^{-\left(\frac{y-\eta}{\sigma}\right)^2}}{\sum_{x=-n}^{n}\sum_{y=-m}^{m}e^{-\left(\frac{x-\delta}{\sigma}\right)^2}e^{-\left(\frac{y-\eta}{\sigma}\right)^2}} \tag{3.33}$$

$$= \delta + \frac{\sum_{x=-n}^{n}(x-\delta)e^{-\left(\frac{x-\delta}{\sigma}\right)^2}\sum_{y=-m}^{m}e^{-\left(\frac{y-\eta}{\sigma}\right)^2}}{\sum_{x=-n}^{n}e^{-\left(\frac{x-\delta}{\sigma}\right)^2}\sum_{y=-m}^{m}e^{-\left(\frac{y-\eta}{\sigma}\right)^2}} \tag{3.34}$$

$$= \delta + \frac{\sum_{x=-n}^{n}(x-\delta)e^{-\left(\frac{x-\delta}{\sigma}\right)^2}}{\sum_{x=-n}^{n}e^{-\left(\frac{x-\delta}{\sigma}\right)^2}} \tag{3.35}$$

In other words, the skew error is expected to be independent of the second parameter y. This suggests that a simple correction factor scheme can be developed for the 2–D case as well. In order to determine the magnitude and form of the 2–D skew error, as well as the effects of other error sources, tests similar to the 1–D case are run.

3.5 2–D Error Analysis

Each of the three remaining error sources from Section 2.2 is discussed under a separate headings below.

Spatial Quantization

To study the effects of two dimensional spatial quantization (ie: the skew error), a test similar to the one dimensional spatial quantization test was run. This test consisted of generating a synthetic Gaussian pulse with shifted mean locations in x in y for several window sizes and pulse variances, and finding the error between the centroid

determined means μ_x and μ_y and the means used to generate the pulse. The subpixel displacements (both in x and y) were varied in steps of 0.02 pixels for a number of window sizes (the windows were $m \times m$ pixels, where $m = 5, 7, 15$ and a number of standard deviation s (s = 1, 3, 5, 7, 10, 15). The results are presented in Table 3.3. All quantities in the table are in units of pixels. The table lists smallest error, the mean error, the variance of the error, and the maximum error. (Note that the errors are only in locating μ_x; the corresponding table for μ_y is similar and so not shown here.)

This table suggests an important trend. The performance of the algorithm improves steadily as the window size was increased with respect to the variance of the pulse until the pulse becomes very impulsive in nature. Examining the raw data indicates that the algorithm in general tends to underestimate the mean.

Amplitude Quantization

The analysis is now refined by considering quantization noise. The same noise model as the one dimensional case was used. The noise process is the same as described before.

The noise effects were studied using the same test as was used for spatial quantization above; however, quantized values were substituted for the real valued Gaussian pulses. The result of this test is shown in Table 3.4. As before, all quantities in the table are in units of pixels. The table lists smallest error, the mean error, the variance of the error, the maximum error in locating μ_x, and the mean SNR for that test.

This trend in this table is similar to the trends noticed for the 1–D case. That is, the error is minimized when data from two standard deviations around the mean

Window Size	Standard Deviation	Min. Error	Mean Error	Error Variance	Max. Error
5	1	-0.000000	0.001163	0.000001	0.002726
	3	0.000000	0.155783	0.008134	0.311926
	5	0.000000	0.212230	0.015069	0.424483
	7	0.000000	0.230174	0.017724	0.460351
	10	0.000000	0.240140	0.019292	0.480280
	15	-0.000000	0.245583	0.020176	0.491167
7	1	0.000000	0.000209	0.000000	0.000326
	3	-0.000000	0.093916	0.002970	0.188595
	5	-0.000000	0.179298	0.010758	0.358669
	7	0.000000	0.211646	0.014986	0.423303
	10	-0.000000	0.230596	0.017789	0.461193
	15	-0.000000	0.241229	0.019467	0.482459
9	1	-0.000000	0.000207	0.000000	0.000324
	3	0.000000	0.045871	0.000715	0.092642
	5	0.000000	0.142188	0.006769	0.284530
	7	0.000000	0.188871	0.011935	0.377770
	10	-0.000000	0.218352	0.015950	0.436708
	15	-0.000000	0.235521	0.018557	0.471043
15	1	-0.000000	0.000207	0.000000	0.000325
	3	-0.000000	0.001304	0.000001	0.002724
	5	-0.000000	0.046061	0.000714	0.092441
	7	0.000000	0.110298	0.004073	0.220712
	10	0.000000	0.169693	0.009634	0.339409
	15	-0.000000	0.211174	0.014918	0.422350
25	1	0.000000	0.000207	0.000000	0.000325
	3	0.000000	0.000000	0.000000	0.000001
	5	-0.000000	0.001341	0.000001	0.002724
	7	-0.000000	0.020949	0.000148	0.042029
	10	0.000000	0.080094	0.002148	0.160261
	15	-0.000000	0.154235	0.007959	0.308484

Table 3.3: 2D Centroid Spatial Sampling Error Summary

Window Size	Standard Deviation	Min. Error	Mean Error	Error Variance	Max. Error	Mean SNR
5	1	0.000000	0.002002	0.000002	0.008130	234.13
	3	0.000000	0.155814	0.008136	0.312500	614.60
	5	0.000000	0.212240	0.015072	0.425139	767.24
	7	0.000000	0.230175	0.017722	0.460684	825.07
	10	0.000000	0.240130	0.019292	0.481005	857.97
	15	0.000000	0.245582	0.020182	0.491679	877.34
7	1	0.000000	0.002002	0.000002	0.008130	227.27
	3	0.000000	0.093942	0.002970	0.189631	469.71
	5	0.000000	0.179298	0.010762	0.359141	668.66
	7	0.000000	0.211644	0.014984	0.424132	761.44
	10	0.000000	0.230592	0.017790	0.461886	820.70
	15	0.000000	0.241229	0.019469	0.482809	856.87
9	1	0.000000	0.002002	0.000002	0.008130	227.27
	3	0.000000	0.045897	0.000717	0.093344	369.98
	5	0.000000	0.142193	0.006771	0.285008	573.80
	7	0.000000	0.188873	0.011933	0.378235	693.39
	10	0.000000	0.218347	0.015953	0.437348	780.44
	15	0.000000	0.235517	0.018561	0.471504	836.23
15	1	0.000000	0.002002	0.000002	0.008130	227.27
	3	0.000000	0.002009	0.000002	0.007288	230.29
	5	0.000000	0.046082	0.000714	0.093278	368.52
	7	0.000000	0.110287	0.004071	0.221035	500.71
	10	0.000000	0.169694	0.009637	0.339730	640.32
	15	0.000000	0.211178	0.014919	0.422685	756.94
25	1	0.000000	0.002002	0.000002	0.008130	227.28
	3	0.000000	0.002009	0.000002	0.007288	223.62
	5	0.000000	0.001490	0.000001	0.006546	229.91
	7	0.000000	0.021036	0.000150	0.042724	310.20
	10	0.000000	0.080107	0.002148	0.160762	437.58
	15	0.000000	0.154231	0.007959	0.308663	601.04

Table 3.4: 2D Centroid Error Summary—Quantized Signal

is present. Indeed, the error is within the desired limits ($\epsilon < 0.05$pixels) only when this condition is met. Adding more data beyond that point increases the error. This can explained in terms of the SNR for the patch. For example, for the 25 pixel case the SNR for variances of 1, 3, and 5 pixels are all about 226; however, the error is smallest for the 5 pixel case, which has the least amount of extra data in the tails of the distribution.

Imaging Noise

The algorithms susceptibleness to additive noise was tested by adding zero mean, single pixel variance Gaussian noise to the function before quantization. The results of this test are summarized in Table 3.5 below. Again, all quantities in the table are in units of pixels. The table lists smallest error, the mean error, the variance of the error, the maximum error in locating μ_x, and the mean SNR for the test.

This table shows that for the given additive noise model, data from two standard deviations around the mean is required to get errors that are acceptably low.

3.6 Summary

The centroid algorithm has been carefully examined and its performance under varying circumstances has been studied. Based on this study we conclude the following:

1. The centroid algorithm is a general purpose blob location technique. It works for any radially symmetric function.

2. Several errors have been postulated in the literature, including skew error, intensity quantization, additive noise, and line jitter.

Window Size	Standard Deviation	Min. Error	Mean Error	Error Variance	Max. Error	Mean SNR
5	1	0.000000	0.006902	0.000027	0.035118	67.78
	3	0.000000	0.155799	0.008123	0.315910	171.60
	5	0.000000	0.212199	0.015073	0.427829	216.15
	7	0.000000	0.230185	0.017719	0.463052	234.01
	10	0.000000	0.240132	0.019303	0.484140	241.11
	15	0.000000	0.245596	0.020164	0.493602	247.42
7	1	0.000000	0.011713	0.000083	0.048380	52.36
	3	0.000171	0.093945	0.002965	0.194883	131.17
	5	0.000108	0.179268	0.010749	0.362224	187.12
	7	0.000000	0.211700	0.014979	0.427245	212.82
	10	0.000000	0.230639	0.017783	0.464760	229.51
	15	0.000000	0.241261	0.019460	0.484217	240.27
9	1	0.000000	0.018231	0.000197	0.100099	42.16
	3	0.000000	0.046049	0.000722	0.102302	103.13
	5	0.000078	0.142259	0.006759	0.290061	159.49
	7	0.000063	0.188888	0.011916	0.380357	193.07
	10	0.000000	0.218347	0.015929	0.440007	218.34
	15	0.000000	0.235538	0.018551	0.473719	233.09
15	1	-0.000000	0.048059	0.001327	0.212584	26.28
	3	0.000005	0.007137	0.000029	0.034677	65.87
	5	0.000084	0.045989	0.000718	0.100182	102.35
	7	0.000034	0.110367	0.004071	0.226397	138.89
	10	0.000000	0.169742	0.009626	0.344493	178.22
	15	0.000102	0.211185	0.014921	0.425204	210.38
25	1	0.000062	0.117755	0.007999	0.569254	15.98
	3	0.000010	0.016284	0.000148	0.068112	44.45
	5	0.000019	0.007297	0.000029	0.028276	65.57
	7	0.000006	0.021372	0.000163	0.053703	86.28
	10	0.000044	0.080106	0.002154	0.168426	121.44
	15	0.000000	0.154233	0.007952	0.312018	166.82

Table 3.5: 2D Centroid Error Summary—Quantized Signal with Added Noise

A model for the skew error has been developed for the case of a known entity function. Compared to quantization and noise errors, skew errors are insignificant.

3. The centroid algorithm is capable of producing results with accuracies of 0.05 pixels provided that two standard deviations of data are available on either side of the mean. If this condition is met, then even low SNR (assuming Gaussian noise) can be tolerated.

The method of centroids shows a great deal of promise for blob location.

Chapter 4

Blob Location using Parameter Estimation

An alternative formulation for blob location can be developed using the assumption that the light observed has a known distribution. This assumption provides a model for the data that can be expected in the sensed image. This model can then be used to drive the mean estimation process.

This chapter will present the two main methods of parameter estimation considered. It is divided into four main sections. The first section presents a brief literature review of parameter estimation. The second section describes a simple linear least squares model. The third section presents a more general non–linear model. The fourth section discusses the results of simulating the various parameter estimation techniques.

4.1 Parameter Estimation Literature Review

Fitting image data to a model is a generally accepted technique. Khotanzad and Chen [18] describe a technique to locate the boundaries of textured regions. The

textured region is considered to be a 2D intensity variation which is modeled as a 2D noncausal, stochastic random field known as a simultaneous autoregressive (SAR) model. A total of two such SAR models are used to define six parameters describing the texture fields. The image is split into overlapping windows for which the SAR parameters are determined; the boundary is determined by finding the place where the SAR parameters change.

Another work by Nalwa and Binford [21] finds edges in an image by modelling the edge as a sequence of one dimensional surfaces with direction and position called "edgels." These edgels are short, linear segments which can be modeled by either the tanh or a cubic function. (These two functions are not completely independent; however, they can be used in combination to insure that no spurious edges are detected.) The procedure used divides the image into small windows or patches. A planar surface is fit to the patch to estimate the orientation of the edgel if one exists. (The gradient of the patch is used for the initial orientation estimate; the estimate is refined using a cubic surface model.) Once a valid edgel orientation is found, the image data is used to fit the tanh model using the least squares estimation technique. (Other processing is done to insure that the edgel found does exist.)

Li, Young, and Magrel (in [32]) discuss a sub–pixel edge detection scheme for a line scan camera. The edge is modeled as a truncated signum function. A two stage procedure is used to locate the edge. First, the nearest pixel is located by finding the point of maximum correlation between the data and the template. Second, the exact position of the edge is located by one of three methods: 1) three point interpolation, 2) interpolation with Chebyshev polynomials, and 3) fitting an edge model using least squares. (It should be noted that the work assumes that the data is smoothly varying.)

The procedure for model based image processing is as follows. First, an appropriate model for the intensity distribution is determined. Second, an estimation procedure is developed to find the model parameters given observed image data. This procedure makes use of as much data as possible to minimize the error. Third, the model parameters must be found using available data.

Two models are used to describe the pulse of light. The first is a simple, linear model that models the pulse as a Gaussian function. The second model uses the sum of several Gaussian pulses, each having the same mean but different amplitudes and variances. These two models and their associated errors are discussed in two sections below.

4.2 Single Pulse Model

In this section, the intensity distribution is modeled as having a Gaussian form. This was done for three main reasons: first, the model is mathematically compact; second, it was assumed that Gaussian light could be obtained for experimental purposes, even if only from a laser; third, there was some evidence in the literature that this model was acceptable (see [13]). A Gaussian pulse is described by three parameters: amplitude, mean location, and variance. (Note that the number of mean and possibly variance parameters is defined by the dimensionality of the system under consideration.) These are the parameters that must be estimated from the image data.

As we have a model for the data and a set of parameters that must be estimated from the data, the next step is the estimation of the data. A common estimation technique is the method of minimum squared error. This procedure is described in

[27, Chapter 14] and in [16, Chapter 13]. The imaging system provides a series of coordinates w_i and corresponding measurements z_i. These measurements are subject to error. There also exists a model that relates the points to the measurements. An error function can be defined as

$$\eta = \sum_{i=1}^{N} \left[z_i - \sum_{j=1}^{M} a_j \theta_j(w_i) \right]^2$$

where N is the number of points w_i, M is the number of basis functions θ_j, a_j are the parameters of the function, and therefore $\sum_{j=1}^{M} a_j \theta_j(w_i)$ is the model relating the points w_i and measurements y_i. This error function has the feature that, given normally distributed errors, the maximum likelihood estimate for the parameters a_j can be found by minimizing η. This is done by setting $\frac{d\eta}{da_k} = 0 \quad \forall\ k$ and solving the resulting system of equations. (This resulting system of equations is called the system of *normal equations* in the parameter estimation literature.)

The development of the single pulse model based blob finder will be split into one and two dimensional sections.

4.2.1 1–D Single Pulse Parameter Estimation Formulation

The intensity distribution is modeled as a Gaussian pulse, with the well known definition which is repeated here.

$$f(x) = Ae^{-\left(\frac{x-\mu}{\sigma}\right)^2}$$

There are a total of three unknowns in this case: the amplitude A, the mean location μ, and the standard deviation σ. In order to use linear least squares parameter estimation, the defining equation must be linearized. This is done by taking the

logarithm of both sides, resulting in

$$\ln f(x) = \ln A - \left(\frac{x-\mu}{\sigma}\right)^2 \tag{4.1}$$

$$\ln f(x) = \ln A - \frac{\mu^2}{\sigma^2} + \frac{2\mu x}{\sigma^2} - \frac{x^2}{\sigma^2} \tag{4.2}$$

This can be recast into matrix form by noting that there are three functions of x on the right hand side of the equation. These functions are

$$\begin{aligned} a_0 &= \ln A - \frac{\mu^2}{\sigma^2} \\ a_1 &= \frac{2\mu}{\sigma^2} \\ a_2 &= -\frac{1}{\sigma^2} \end{aligned}$$

with $\ln f(x_i) = a_0 + a_1 x_i + a_2 x_i^2$. In matrix form this can be expressed as

$$A\vec{a} = \vec{b} \tag{4.3}$$

$$A = \begin{bmatrix} 1 & x_0 & x_0^2 \\ 1 & x_1 & x_1^2 \\ & \cdots & \end{bmatrix} \tag{4.4}$$

$$\vec{a} = \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} \tag{4.5}$$

$$\vec{b} = \begin{bmatrix} \ln f(x_0) \\ \ln f(x_1) \\ \vdots \end{bmatrix} \tag{4.6}$$

x_i represents the location of the data points, and the $f(x_i)$ are the function values. In order to solve this system, pre–multiply by A^T. The resulting equations are:

$$A^T A\vec{a} = A^T\vec{b} \tag{4.7}$$

$$\begin{bmatrix} n & \sum x_i & \sum x_i^2 \\ \sum x_i & \sum x_i^2 & \sum x_i^3 \\ \sum x_i^2 & \sum x_i^3 & \sum x_i^4 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} \sum \ln f(x_i) \\ \sum x_i \ln f(x_i) \\ \sum x_i^2 \ln f(x_i) \end{bmatrix} \tag{4.8}$$

where n is the number of image points included in the calculation.

There are several observations to be made about the system of equations defined by equation 4.8. The first is that the matrix A^TA can be made constant for several images provided that the images have the same size and same position relative to some datum. Such an arrangement would allow the precomputation of the matrix A^TA which would make the method computationally less expensive. Second, the solution vector $a^T\vec{b}$ is a vector of higher moments of the logarithm of the image intensity data. Now the logarithm function tends to compress the range of the input data; this leads to an improved SNR of the signal as the (presumed additive) noise is (mostly) compressed out of the data. Thus, the single pulse parameter estimation is in effect performing a series of moment based calculations on the data. Lastly, the first matrix is square and the solution vector is the same size as the unknown vector. This means that the system can be solved using any standard matrix system solution method. (The solution method selected for the trials conducted below used LU decomposition, due primarily to its availability.) As image data are used to create the matrices, the system is guaranteed to be non–singular. This is due to the fact that the square matrix consists of combinations of the locations of data points (such as $\sum x_i$, or $\sum x_i^3$); there is no conceivable real set of numbers that would result in any of the rows in the matrix being a multiple of another. There are two restrictions on the process: the first is that none of the data values may be 0 (as $\ln 0$ is undefined). The second restriction is that at least 3 points must be used to keep the solution valid. (If only one or two data points are used, then the original system of equations 4.3 is under-determined and there is insufficient data to solve for the three unknowns.)

4.2.2 1–D Error Analysis

A series of tests was run to examine the performance of the 1–D model when exposed to the various error sources. These tests simulated pulses of data by using a Gaussian function evaluated at discrete points to create a virtual image. The first set of tests used unquantized (or real valued) data. The sub–pixel displacement was varied from -0.5 to 0.5 pixels from the center pixel in the window over a range of standard deviation (s = 1, 3, 5, 7, 10, 15). This test was run for several values of n—the number of data points included in the parameter estimation ($n = 4, 5, 6, 7, 10, 15, 20$). Two sets of results are shown in the two figures below. Figure 4.1 shows the error in locating the mean of the Gaussian with a window size of 5 for several standard deviation s, and Figure 4.2 shows the same graph for a window size of 25.

There are two points of interest in these figures. First, the systemic skew error noted in the centroid calculation (Chapter 3) has been replaced with a very chaotic error function that cannot be predicted. Second, for a small window size the large standard deviations generate the largest errors; for the large window size, the reverse is true. In neither case is the error significant. To further study this effect, a series of tests that determined the error for a fixed displacement and over a range of standard deviations ($0.9 < s < 25$) for same set of window sizes as above ($n = 4, 5, 6, 7, 10, 15, 20$). Figure 4.3 shows the graph that resulted for a window size of 15. (The other curves are similar in shape, except that the smaller window sizes do not have as large an initial large error at the standard deviations close to 1.) Once the data moved past the region of impulsive data (standard deviation $s < 1$), the error dropped to zero until the standard deviation starts to approach half of the window size. The conclusion is that the window size should be large enough to capture data

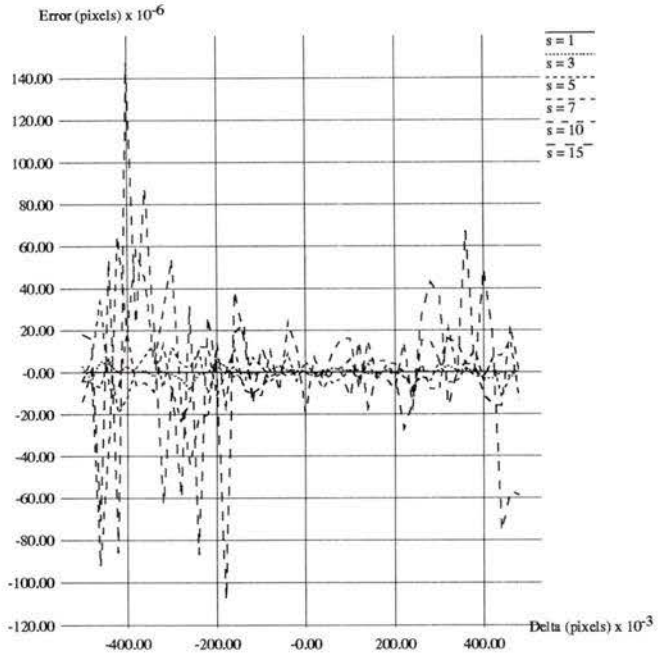


Figure 4.1: Error Location Single Pulse Spatially Sampled Data Constant Standard Deviation s with Window Size 5 pixels

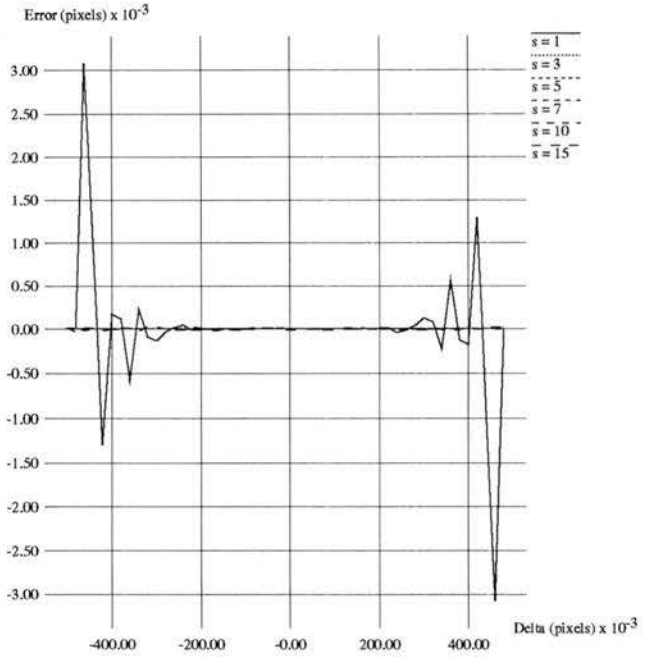


Figure 4.2: Error Location Single Pulse Spatially Sampled Data Constant Standard Deviation s with Window Size 25

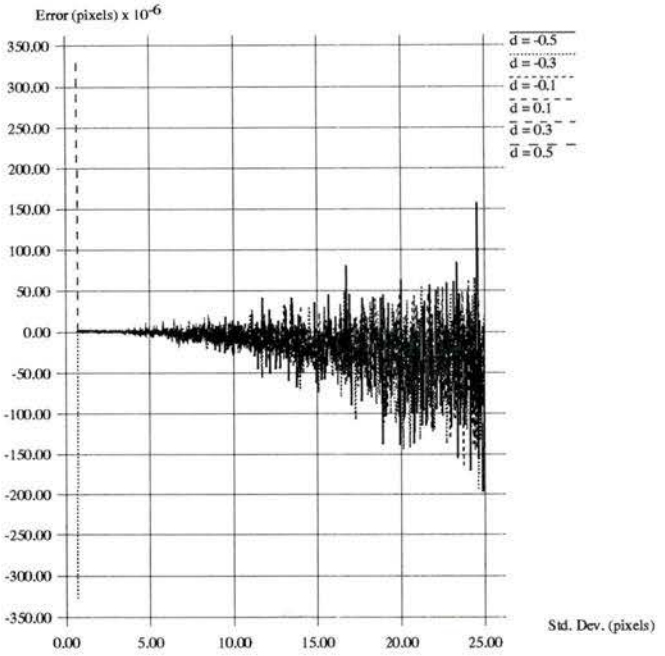


Figure 4.3: Error Locating Single Pulse Spatially Sampled Data with Constant Displacement d and Window Size 15 with Variable Standard Deviation

from two standard deviations about the mean.

Two more sets of validation were performed. The first set repeated the constant standard deviation, constant window size test, substituting quantized data for the real data. The quantization was performed by rounding the function values to the nearest integer. The second set added zero mean, single pixel standard deviation Gaussian noise to the quantized intensity values. As anticipated, the errors in these test sets were larger. Figure 4.4 shows the result of the constant standard deviation ($s = 1, 3, 5, 7, 10, 15$) and window (9 pixels) test on quantized data. As standard deviation increased beyond the limits suggested above, the error increased beyond the acceptable limits ($\epsilon < 0.05$ pixels).

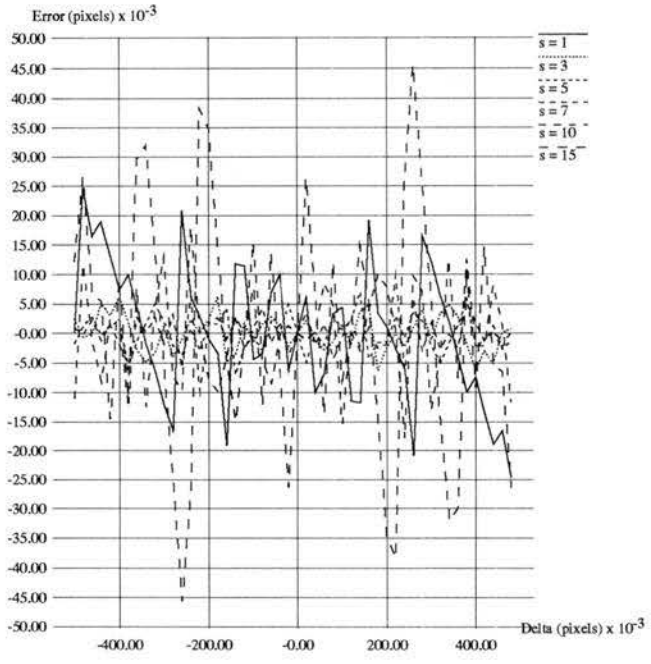


Figure 4.4: 1–D Location Error for Quantized Data, Window Size 9 pixels, and Constant Standard Deviations

Std. Dev. (pixels)	SNR
1	468.79
3	594.48
5	751.09
7	858.51
10	886.13
15	921.22

Table 4.1: Average SNR for each Standard Deviation s in Figure 4.4

Std. Dev. (pixels)	SNR
2	145.63
3	174.44
5	221.08
7	229.64
10	259.85
15	259.71

Table 4.2: Average SNR for each Standard Deviation s in Figure 4.5

The mean SNR for each of the curves in Figure 4.4 is found in Table 4.1.

Figure 4.5 shows the same test except that zero mean, single pixel standard deviation Gaussian noise has been added. The graph shows the mean location error for a window of 9 pixels and a set of standard deviations $s, s = 2, 3, 5, 7, 10, 15$.

The mean SNR for each of the curves in Figure 4.5 is found in Table 4.2. The mean SNR is about $\frac{1}{4}$ of the previous case (Table 4.1).

Clearly, the noise added pushes the error beyond the acceptable boundaries. The pattern that was observed from cases that used real valued data is repeated here. Small standard deviations yield small error with small window sizes, and large errors

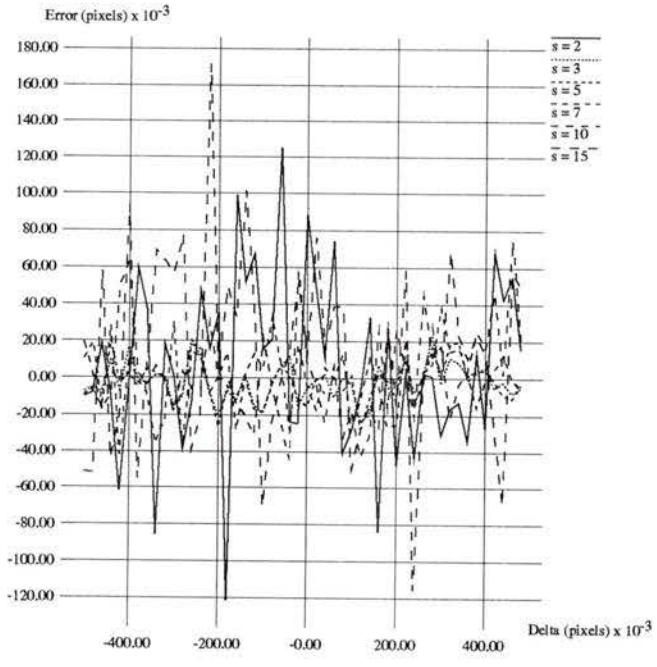


Figure 4.5: 1D Location Error for Quantized Data with Noise Added and Constant Standard Deviation s and Window Size 9 pixels

with large window sizes; large variances have large errors with small window sizes and small errors with large window sizes.

In summary, 1–D parameter estimation accuracies of 0.05 pixels are feasible subject to the following conditions. First, a minimum of three points must be used in the calculations. This insures that the solution remains valid. Second, none of the image data points used may have a value of zero. Third, the number of points used in determining the matrices should be about twice the standard deviation of the generating curve. Fourth, the processing noise level must be controlled.

4.2.3 2–D Single Pulse Parameter Estimation Formulation

There are two possible pulse models in the 2–D case. The first model assumes that the pulse is circularly symmetric. The second, slightly more general model, does not. In analysis following the model derivation, the circularly symmetric model was used.

The intensity distribution is modeled as a Gaussian pulse, which is defined (assuming circular symmetry) as

$$f(x,y) = Ae^{-\left(\frac{(x-\mu_x)^2+(y-\mu_y)^2}{\sigma^2}\right)} \tag{4.9}$$

Without circular symmetry, this model is

$$f(x,y) = Ae^{-\left(\frac{(x-\mu_x)^2}{\sigma_x^2}+\frac{(y-\mu_y)^2}{\sigma_y^2}\right)} \tag{4.10}$$

This is linearized using logarithms, and is split into functions of x and y. Thus, equation 4.9 becomes

$$\ln f(x,y) = \ln A - \frac{(x-\mu_x)^2+(y-\mu_y)^2}{\sigma^2} \tag{4.11}$$

$$= \ln A - \frac{\mu_x^2+\mu_y^2}{\sigma^2} + \frac{2\mu_x}{\sigma^2}x - \frac{x^2+y^2}{\sigma^2} + \frac{2\mu_y}{\sigma^2}y \tag{4.12}$$

The parameters are $a_0 = \ln A - \frac{\mu_x^2+\mu_y^2}{\sigma^2}$, $a_1 = \frac{2\mu_x}{\sigma^2}$, $a_2 = \frac{1}{\sigma^2}$, $a_3 = \frac{2\mu_y}{\sigma^2}$. Thus, equation 4.12 becomes $\ln f(x,y) = a_0 + a_1 x + a_2(x^2+y^2) + a_3 y$, which in matrix form is (as before, and pre-multiplying with A^T)

$$\begin{bmatrix} 1 & x_0 & x_0^2+y_0^2 & y_0 \\ 1 & x_1 & x_1^2+y_1^2 & y_1 \\ & \cdots & \cdots & \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} \ln f(x_0) \\ \ln f(x_1) \\ \vdots \end{bmatrix} \tag{4.13}$$

$$\begin{bmatrix} n & \sum x_i & \sum x_i^2+y_i^2 & \sum y_i \\ \sum x_i & \sum x_i^2 & \sum x_i(x_i^2+y_i^2) & \sum x_i y_i \\ \sum x_i^2+y_i^2 & \sum x_i(x_i^2+y_i^2) & \sum (x_i^2+y_i^2)^2 & \sum y_i(x_i^2+y_i^2) \\ \sum y_i & \sum x_i y_i & \sum y_i(x_i^2+y_i^2) & \sum y_i^2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} =$$

$$\begin{bmatrix} \sum \ln f(x_i,y_i) \\ \sum x_i \ln f(x_i,y_i) \\ \sum (x_i^2+y_i^2) \ln f(x_i,y_i) \\ \sum y_i \ln f(x_i,y_i) \end{bmatrix} \tag{4.14}$$

where n is the number of image points included in the calculation.

The restriction here (aside from the exclusion of zero valued data points) is that at least 4 points must be used to keep the solution valid.

For the non-circular symmetric case, the model is

$$\begin{bmatrix} n & \sum x_i & \sum x_i^2 & \sum y_i & \sum y_i^2 \\ \sum x_i & \sum x_i^2 & \sum x_i^3 & \sum x_i y_i & \sum x_i y_i^2 \\ \sum x_i^2 & \sum x_i^3 & \sum x_i^4 & \sum x_i^2 y_i & \sum x_i^2 y_i^2 \\ \sum y_i & \sum x_i y_i & \sum x_i^2 y_i & \sum y_i^2 & \sum y_i^3 \\ \sum y_i^2 & \sum x_i y_i^2 & \sum x_i^2 y_i^2 & \sum y_i^3 & \sum y_i^4 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix} =$$

$$\begin{bmatrix} \sum \ln f(x_i,y_i) \\ \sum x_i \ln f(x_i,y_i) \\ \sum x_i^2 \ln f(x_i,y_i) \\ \sum y_i \ln f(x_i,y_i) \\ \sum y_i^2 \ln f(x_i,y_i) \end{bmatrix} \tag{4.15}$$

where n is the number of image points included in the calculation, $a_0 = \ln A - \frac{\mu_x^2}{\sigma_x^2} - \frac{\mu_y^2}{\sigma_y^2}$, $a_1 = \frac{2\mu_x}{\sigma_x^2}$, ; $a_2 = \frac{1}{\sigma_x^2}$, ; $a_3 = \frac{2\mu_y}{\sigma^2}$, and $a_4 = \frac{1}{\sigma_y^2}$. In this case, a minimum of five data points is required for mathematical validity.

4.2.4 2–D Single Pulse Error Analysis

A series of tests was run to examine how the two dimensional model performed when exposed to the various error sources. These tests were performed using the circularly symmetric model. The first test consisted of varying the sub–pixel displacements (both in x and y) from -0.5 to 0.5 pixels about the window's center pixel over a range of standard deviations (s = 1, 3, 5, 7, 10, 15). The size of the support window n was given values $n = 5, 7, 9, 15, 25$. (The window used was an $n \times n$ square.) Only one result graph is shown; a summary table appears after it. Figure 4.6 is a surface plot of the error in locating the x mean over the entire sub–pixel range. The standard deviation in this case was 3 pixels, and the window size was 5 pixels (for a 5×5 window). The maximum error is 0.000016 pixels. Note that the errors do not fall in a neat pattern. Similar curves result for the rest of the displacements, standard deviations, and window size.

Table 4.3 shows the minimum error, mean error, variance of the error, and maximum error (all in units of pixels) for various window size cases.

There are two key conclusions to draw from this figure and table (and its companions which are not shown). First, there is no neat, systemic error surface (as in the case with the centroid algorithm). Thus, it is impossible to post process the estimated mean location to improve the location estimate. Second, for a small window size, the large standard deviations generate the largest errors, and for the larger window size, the reverse is true. The same conclusion is arrived at as for the single dimensional case: the window should be about twice the standard deviation.

The errors in locating the y mean show similar results as those of the x mean location errors. The maximum y error did not, however, occur in the same place as

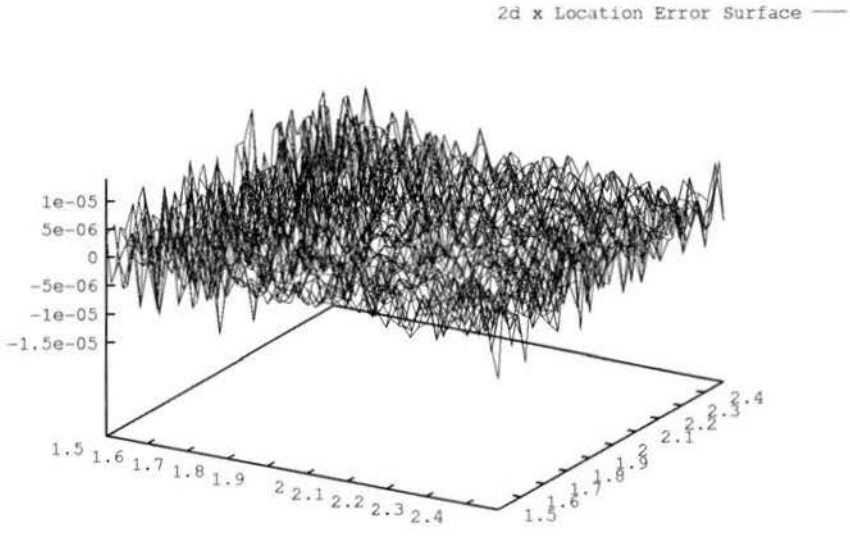


Figure 4.6: 2D Location Error for Spatially Sampled Data with Window Width 5 pixels, Standard Deviation of 1 pixel

Window Size	Standard Deviation	Min. Error	Mean Error	Error Variance	Max. Error
5	1	0.000000	0.000000	0.000000	0.000001
	3	-0.000000	0.000002	0.000000	0.000016
	5	0.000000	0.000007	0.000000	0.000042
	7	-0.000000	0.000017	0.000000	0.000105
	10	0.000000	0.000032	0.000000	0.000235
	15	0.000000	0.000068	0.000000	0.000479
7	1	0.000000	0.000000	0.000000	0.000001
	3	0.000000	0.000002	0.000000	0.000012
	5	0.000000	0.000006	0.000000	0.000040
	7	0.000000	0.000013	0.000000	0.000085
	10	0.000000	0.000026	0.000000	0.000146
	15	0.000000	0.000059	0.000000	0.000371
9	1	0.000000	0.000000	0.000000	0.000003
	3	0.000000	0.000002	0.000000	0.000010
	5	-0.000000	0.000006	0.000000	0.000041
	7	-0.000000	0.000013	0.000000	0.000071
	10	0.000000	0.000025	0.000000	0.000149
	15	0.000000	0.000057	0.000000	0.000329
15	1	0.000000	0.000019	0.000000	0.000302
	3	-0.000000	0.000001	0.000000	0.000005
	5	0.000000	0.000005	0.000000	0.000025
	7	-0.000000	0.000010	0.000000	0.000061
	10	-0.000000	0.000023	0.000000	0.000119
	15	0.000000	0.000054	0.000000	0.000254
25	1	0.000000	0.000208	0.000000	0.001030
	3	0.000000	0.000019	0.000000	0.000046
	5	0.000000	0.000019	0.000000	0.000045
	7	0.000000	0.000020	0.000000	0.000072
	10	0.000000	0.000027	0.000000	0.000123
	15	0.000000	0.000052	0.000000	0.000258

Table 4.3: 2D Single Pulse Location Error by Window Size Using Spatially Sampled Data

the maximum x mean location error. (No reason for this result is known.)

The second test set substituted quantized data for the real valued data. The third test set added zero mean, single pixel standard deviation noise to the quantized data. As anticipated, the errors in the test sets were larger. Table 4.4 shows the results for quantized data, while Table 4.5 shows the results for noise added to the quantized data.

Examining Tables 4.4 and 4.5, the fact that the 2–D parameter estimation works well under slightly less stringent criteria than the centroid algorithm is obvious. For example, in Table 4.5, the error for standard deviations of 3 and 5 with a window size of 5 pixels is acceptable. Examining the window size of 25 pixels, the standard deviations of 10 and 15 yield good results. The conclusion to draw is that the method generates acceptable results (less than 0.05 pixels error) when the standard deviation is between about $\frac{1}{3}$ and 1 times the window size.

In summary, the following restrictions are placed on the two dimensional parameter estimation technique. First, a minimum of six points must be used in the calculations. Second, no zero data points may be used. Third, accurate results are possible when the standard deviation s is related to the window size w by $\frac{1}{3}w < s < w$. As in the one dimensional case, these restrictions will be met by the image segmentation algorithm.

4.3 Multiple Pulse Model

In this section, a more general model for the intensity distribution is presented. The intensity distribution is assumed to be the sum of k Gaussian pulses, each with its own variance and amplitude, but sharing a common mean. For the remainder of this

Window Size	Standard Deviation	Min. Error	Mean Error	Error Variance	Max. Error	Mean SNR
5	1	0.0000	0.0132	0.0001	0.0461	254.09
	3	0.0000	0.0011	0.0000	0.0049	614.61
	5	0.0000	0.0020	0.0000	0.0086	767.25
	7	0.0000	0.0035	0.0000	0.0195	825.08
	10	0.0000	0.0071	0.0000	0.0298	857.98
	15	0.0000	0.0149	0.0001	0.0593	877.36
7	1	0.0000	0.0132	0.0001	0.0461	254.09
	3	0.0000	0.0013	0.0000	0.0055	469.71
	5	0.0000	0.0013	0.0000	0.0056	668.67
	7	0.0000	0.0020	0.0000	0.0099	761.45
	10	0.0000	0.0036	0.0000	0.0179	820.71
	15	-0.0000	0.0077	0.0000	0.0295	856.88
9	1	0.0000	0.0132	0.0001	0.0461	254.09
	3	0.0000	0.0027	0.0000	0.0124	369.98
	5	0.0000	0.0011	0.0000	0.0056	573.81
	7	0.0000	0.0013	0.0000	0.0057	693.40
	10	-0.0000	0.0024	0.0000	0.0091	780.45
	15	0.0000	0.0045	0.0000	0.0201	836.23
15	1	0.0000	0.0132	0.0001	0.0461	254.09
	3	0.0000	0.0156	0.0002	0.0501	248.81
	5	0.0000	0.0032	0.0000	0.0147	368.53
	7	0.0000	0.0012	0.0000	0.0054	500.72
	10	-0.0000	0.0012	0.0000	0.0046	640.33
	15	0.0000	0.0020	0.0000	0.0100	756.94
25	1	0.0000	0.0132	0.0001	0.0461	254.09
	3	0.0000	0.0156	0.0002	0.0501	248.81
	5	0.0000	0.0082	0.0000	0.0335	248.34
	7	0.0000	0.0069	0.0000	0.0233	310.32
	10	0.0000	0.0017	0.0000	0.0068	437.58
	15	0.0000	0.0011	0.0000	0.0063	601.04

Table 4.4: 2D Single Pulse Error Location by Window Size Using Quantized Data

Window Size	Standard Deviation	Min. Error	Mean Error	Error Variance	Max. Error	Mean SNR
5	1	0.0000	0.0633	0.0025	0.2836	71.46
	3	0.0000	0.0041	0.0000	0.0190	171.30
	5	0.0000	0.0077	0.0000	0.0340	216.87
	7	-0.0000	0.0132	0.0001	0.0612	232.73
	10	0.0000	0.0258	0.0004	0.1127	242.63
	15	0.0001	0.0569	0.0020	0.3491	247.80
7	1	0.0000	0.1597	0.0143	0.6703	55.58
	3	0.0000	0.0047	0.0000	0.0224	130.71
	5	0.0000	0.0047	0.0000	0.0197	186.07
	7	0.0000	0.0076	0.0000	0.0304	212.96
	10	0.0000	0.0135	0.0001	0.0651	229.55
	15	0.0000	0.0276	0.0004	0.1279	239.03
9	1	0.0001	0.2163	0.0256	1.0872	45.10
	3	0.0000	0.0100	0.0001	0.0532	103.12
	5	0.0000	0.0039	0.0000	0.0158	160.02
	7	0.0000	0.0052	0.0000	0.0222	193.27
	10	0.0000	0.0085	0.0000	0.0354	217.87
	15	0.0000	0.0175	0.0002	0.0915	233.68
15	1	0.0002	0.2706	0.0418	1.1479	28.33
	3	0.0000	0.0813	0.0034	0.3278	68.84
	5	0.0000	0.0107	0.0001	0.0486	102.40
	7	0.0000	0.0043	0.0000	0.0219	139.03
	10	-0.0000	0.0045	0.0000	0.0200	178.01
	15	0.0000	0.0071	0.0000	0.0313	210.37
25	1	0.0000	0.3590	0.0757	1.5757	17.24
	3	0.0001	0.2132	0.0243	0.9061	47.31
	5	0.0001	0.0824	0.0036	0.3505	68.66
	7	0.0000	0.0220	0.0003	0.0952	86.89
	10	0.0000	0.0057	0.0000	0.0254	121.45
	15	0.0000	0.0041	0.0000	0.0169	166.72

Table 4.5: 2D Single Pulse Error Location by Window Size Using Quantized Data with Noise Added

work, it is assumed that $k = 3$. (The is value was chosen as it was experimentally determined to match an image of a specific blob.) A general light model for the intensity at the point $\vec{x_i}$ is presented below; the explicit model used for the simulations will be presented in the appropriate section below.

$$f(\vec{x_i}) = \sum_{m=1}^{k} A_m e^{-\left(\frac{\vec{x_i}-\mu}{\vec{\sigma_m}}\right)^2}$$

The parameters which may be modifies include the mean μ, the amplitudes A_m, and standard deviations σ_m. These parameters are collected in a vector denoted by θ.

This multi–pulse model cannot be linearized as was done in the preceeding section to arrive at a simple, single step solution. Instead, non–linear solution techniques must be applied. Two different approaches will be considered. The first approach is to consider the problem as a non–linear minimization problem. The specific approach is the simplex method, which is described in [23]. The second approach is the Levenberg–Marquardt non–linear parameter estimation, described in [31, 30, 34]. It must be noted that time efficiency of the solution method was not important in this study; rather, the method merely had to show promise to perform the task in a finite amount of time.

For the discussions below it assumed that the model is given by $f(\theta)$, which operates on image data z_i which has coordinates $\vec{x}_i$.

Simplex Optimization

The simplex method is an approach to function minimization that avoids the calculation of Hessian (second partial derivatives) matrices in order to complete the task.

The process begins with a function that is to be minimized:

$$c = \min_{\mathbf{X}} f(\mathbf{X}) \tag{4.16}$$

where f returns a scalar value, and $\mathbf{X}$ is a vector of n parameters that f depends on. The main restriction on f is that it have a minimum and be continuous in the area of the minimum, and that the minimum be "close" to the initial value $\mathbf{X}$. (Close need not mean that the function has no other minima between the current point X and the global minimum; the technique usually is robust enough to find the global minimum [23, page 311].) This represents (most generally) an unconstrained minimization; however it is possible to place some constraints on the values of X by driving f to a very large value if a constraint is violated.

The function f can be thought of as existing in an n–dimensional space. In this space a general simplex or configuration of space defined by $(n + 1)$ points can be defined over a set of points $P_0, P_1, \ldots, P_n$. The minimization method adjusts the current simplex to move down the local slopes of the function, contracting down to the minimum function value.

To explain the minimization process, the following notation is introduced. The value of $f(P_i)$ is denoted by c_i. The index h indicates which point P_i has the largest function value $P_h = \max_i c_i$, and the index l the point with the smallest value $P_l = \min_i c_i$. The point $\overline{P}$ is the centroid of all points in the simplex excluding P_h. The distance between two points P_i and P_j is denoted by $[P_i P_j]$.

The minimization proceeds by iteratively adjusting the point P_h using three different operations—*reflection, contraction,* and *expansion.* The reflection of P_h is P^*, and is defined by

$$P^* = (1 + \alpha)\overline{P} - \alpha P_h \tag{4.17}$$

where α is the positive *reflection constant.* Geometrically, P^* is on the line connecting P_h and $\overline{P}$, on the far side of $\overline{P}$ from P_h with $[P^*\overline{P}] = \alpha[P_h\overline{P}]$. If $c^* < c_h$, then P_h is replaced by P^* and the process iterates with the new simplex.

If $c^* < c_l$ (a new minimum), then P^* is expanded to P^{**}:

$$P^{**} = \gamma P^* + (1 - \gamma)\overline{P} \tag{4.18}$$

where the expansion coefficient γ, $\gamma > 1$, is the ratio of the distances $[P^{**}\overline{P}]$ to $[P^*\overline{P}]$. If $c^{**} < c_l$ then P_h is replaced with P^{**} and the the method iterates; if $c^{**} > c_l$ the expansion has failed and P_h is replaced with P^* before iterating.

Now if reflecting P_h to P^* results in $c^* > c_i \; \forall i \neq h$, then a contraction is attempted. The contraction is calculated setting P_h to be either P_h or P^*, depending on which has the smaller function value c; P^{**} is then found by

$$P^{**} = \beta P_h + (1 - \beta)\overline{P} \tag{4.19}$$

with the contraction coefficient β (the ratio of the distance $[P^{**}\overline{P}]$ to $[P_h\overline{P}]$) constrained such that $0 < \beta < 1$. P_h is replaced by P^{**} except if $c^{**} > \min(c_h, c^*)$, in which case all points P_i are replaced with $(P_i + P_l)/2$. After these calculations are complete, the method iterates.

The failed expansion may be viewed in geometric terms as the case where the algorithm tried to slide down a valley but wound up higher than it previously was on the opposite slope. Similarly, the failed contraction may be view in terms of one point in the simplex being much farther away from the valley bottom than the rest; a contraction would simply moves the point farther away. The coefficient α, γ, and β control the rate at which the simplex volume contracts onto the minimum.

The algorithm ends when the "standard error," defined as $(\sum(y_i - \overline{y})^2/n)^{\frac{1}{2}}$, falls below a preset value. (In implementation, the algorithm can also end when a preset maximum iteration count is exceeded.) This is designed to prevent the simplex algorithm from trying to improve estimates in a region of very small curvature while continuing to correct estimates in areas with large curvature.

In summary, the simplex method provides a means to iteratively minimize a non–linear function using only linear combinations of points in the hyperspace of the functions range. For successful convergence to a global maximum, the algorithm must start with an initial parameter set that is reasonably close to minimum.

Levenberg–Marquardt Parameter Estimation

The Levenberg–Marquardt parameter estimation algorithm is a refinement of the Gauss–Newton non–linear least squares estimation algorithm. As in the linear least squares case, the development begins with the definition of an error that is to be minimized. This error function is the error between the observed data z_i and the modeled observation $f(\vec{x}_i;\theta)$. The residual function $r(\theta)$ is defined as the difference between the observed data and the modeled data:

$$r(\theta) = \sum_{i=\vec{x}_0}^{\vec{x}_n} [z_i - f(\vec{x}_i;\theta)] \tag{4.20}$$

For ease of manipulation, the residual r is written in matrix form as $\mathbf{r}(\theta)$:

$$\mathbf{r}(\theta) = \mathbf{z} - \mathbf{f}(\vec{x}_i;\theta) \tag{4.21}$$

where $\mathbf{z}$ is the vector of observed data, and $\mathbf{f}(\vec{x}_i;\theta)$ is the vector of modeled data. With this definition, the desired error function is the sum squared error $S(\theta)$.

$$S(\theta) = \mathbf{r}^T\mathbf{r} \tag{4.22}$$

(The sum squared error is a scalar term.) Due to the non–linear nature of the problem, this equation 4.22 can have many local minima in addition to the global minimum.

To find the parameters $\hat{\theta}$ which minimizes equation 4.22, the first derivative with respect to θ is set to 0.

$$\left.\frac{\partial S(\theta)}{\partial \theta}\right|_{\hat{\theta}} = 0 \tag{4.23}$$

$$\mathbf{r}(\hat{\theta})^T \frac{\partial \mathbf{r}(\hat{\theta})}{\partial \theta} = 0 \tag{4.24}$$

As non–linear models cannot be solved analytically, equation 4.24 cannot be used to directly determine the value of $\hat{\theta}$.

The solution technique involves iteratively improving the estimate of θ to $\theta^{(a)}$. (This implies that an initial estimate of the parameters must be supplied.) If $\theta^{(a)}$ is close to θ, then a linear Taylor's series expansion can be used to approximate $\theta^{(a)}$.

$$f(\theta) \approx f(\theta^{(a)}) + \frac{\partial f(\theta^{(a)})}{\partial \theta}(\theta - \theta^{(a)}) \tag{4.25}$$

This is applied to the error function $S(\theta)$ (equation 4.22). Defining a gradient vector $g(\theta) = \frac{\partial S(\theta)}{\partial \theta}$ and a Hessian matrix $H(\theta) = \frac{\partial^2 S(\theta)}{\partial \theta \partial \theta^T}$ the approximation of the sum squared error is

$$S(\theta) \approx S(\theta^{(a)}) + g^T(\theta^{(a)})(\theta - \theta^{(a)}) + \frac{1}{2}(\theta - \theta^{(a)})^T H(\theta^{(a)})(\theta - \theta^{(a)}) \tag{4.26}$$

The minimum of this approximated error function (with respect to θ) occurs when

$$\theta - \theta^{(a)} = -[H(\theta^{(a)})]^{-1} g(\theta^{(a)}) \tag{4.27}$$

$$\delta^{(a)} = -[H(\theta^{(a)})]^{-1} g(\theta^{(a)}) \tag{4.28}$$

Thus, the improved estimate $\theta^{(a+1)}$ is

$$\theta^{(a+1)} = \theta^{(a)} + \delta^{(a)} \tag{4.29}$$

Unfortunately, the numerical calculation of the Hessian matrix is generally unstable. This is the point where the Levenberg–Marquardt modifications take place. The update step $\delta^{(a)}$ is modified to be expressed in terms of the Jacobian J of $S(\theta)$, which is defined by

$$J = \frac{\partial \mathbf{r}}{\partial \theta^T} \tag{4.30}$$

$\delta^{(a)}$ becomes

$$\delta^{(a)} = -(J^{(a)T}J^{(a)} + \eta^{(a)}D^{(a)})^{-1}J^{(a)T}\mathbf{r}^{(a)} \tag{4.31}$$

where $D^{(a)}$ is a diagonal matrix with positive diagonal elements, and η is an adjustment factor. When $D^{(a)}$ is the identity matrix I, the value of η determines how much the algorithm interpolate the Gauss—Newton algorithm ($\eta \rightarrow 0$) and the steepest descent direction ($\eta \rightarrow \infty$). (This essentially controls the rate of convergence of the method.) This approximation does not involve second derivatives, and has been proven to be numerically stable.

In summary, the Levenberg–Marquardt provides an algorithm to iteratively improve an initial estimate of a parameter set θ. The algorithm uses only first derivatives of the residual function $\mathbf{r}(\theta)$, and so is numerically stable.

The MatLab [10] implementations of both the simplex and the Levenberg–Marquardt estimation routines were used for this work.

4.3.1 1–D Multi Pulse Formulation

In the 1–D case, the intensity distribution is modeled as

$$f(x) = \sum_{l=1}^{3} A_l e^{-\left(\frac{x-\mu}{\sigma_l}\right)^2} \tag{4.32}$$

In order to perform a simplex minimization, a minimizable, continuous function is required. Given the above model (equation 4.32), a natural candidate function is the sum squared error between the imaged data and the model:

$$s(\theta) = \sum_{i=0}^{n} \left(z_i - \sum_{l=1}^{3} A_l e^{-\left(\frac{x_i - \mu}{\sigma_l}\right)^2} \right)^2 \tag{4.33}$$

where θ is the vector of parameter to solve; $\theta = [\theta_k]^T = [\mu\ A_1\ \sigma_1\ A_2\ \sigma_2\ A_3\ \sigma_3]^T$. The A_l and σ_l represent the amplitude and standard deviation of the lth Gaussian, μ is the mean of the Gaussians, z_i represents the ith image data point. The total number of image data points (and therefore coordinates) is n.

Equation 4.32 also leads directly to the least squares error function. The Levenberg—Marquardt algorithm requires the residual vector $\mathbf{r}$:

$$\mathbf{r}(\theta) = \mathbf{z} - \sum_{l=1}^{3} A_l e^{-\left(\frac{\mathbf{x} - \mu}{\sigma_l}\right)^2} \tag{4.34}$$

where θ is the vector of parameter to solve; $\theta = [\theta_k]^T = [\mu\ A_1\ \sigma_1\ A_2\ \sigma_2\ A_3\ \sigma_3]^T$, $\mathbf{z}$ is the vector of observed data points, and $\mathbf{x}$ is the vector of coordinates corresponding to the observed data points. The A_l and σ_l represent the amplitude and standard deviation of the lth pulse. The Jacobian $J^{(a)}$ is directly obtainable (numerically) from equation 4.34.

4.3.2 1–D Multiple Pulse Error Analysis

Both of the non-linear estimation techniques were tested to see how they responded to the three noise sources outlined in Chapter 2. Each of these tests (spatial quantization, intensity quantization, and additive noise) is described below. The test signal was a multiple Gaussian pulse as defined by equation 4.32. The sample window size w and the "base" standard deviation σ_1 were specified, and the rest of the function

parameters were as follows: $\mu = w/2$, $\sigma_2 = 2.57\sigma_1$, $\sigma_3 = 4.97\sigma_1$, $A_1 = 154.25$, $A_2 = 77.68$, and $A_3 = 22.07$.

Spatial Sampling

The first set of tests evaluated how the algorithms reacted to spatial sampling. The trials consisted of varying the sub–pixel displacement off the mean of the function from -0.5 to 0.5 pixels about the middle of a set of sampling windows for several standard deviation s, $s = 1, 3, 5, 7, 10, 15$. The windows were 7, 9, 15, and 25 pixels wide. Figure 4.7 shows a graph of the error in locating the mean for a window of 9 pixels, using the simplex method for calculations. This figure shows that the magnitude of the error depends on the magnitude of the standard deviation in relation to the window size. The errors for $s = 1$ and $s = 15$ are relatively large, while the errors for others are somewhat smaller.

A further study held the sub-pixel displacement constant and varied the windows size and standard deviation. A sample, using the simplex method for calculations, is Figure 4.8. This shows the error in locating the mean for sub–pixel displacements of -0.5, -0.3, -0.1, 0.1, and 0.3 from the center pixel in a window of 9 pixels for standard deviations from 0.5 to 25. The error all but disappears as enough data is gathered to fill the window.

The graphs for the least squares tests showed the same trends as those presented above. The main difference between them was the fact that the error was in general three orders of magnitude smaller.

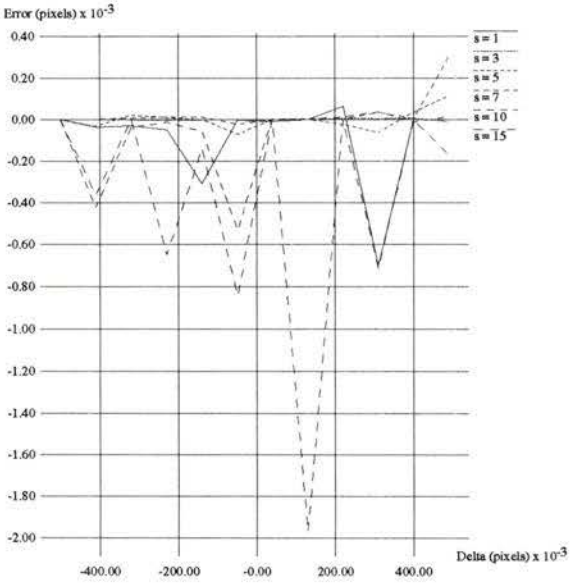


Figure 4.7: 1–D Multi Pulse Location Error for Constant Standard Deviation s and Window Size 9 pixels

Amplitude Quantization

The next set of trials substituted quantized data for the calculations. The trial set up was the same as for the previous trial. The results are presented from the trials using the simplex algorithm; the Levenberg–Marquardt least squares performance was essentially the same. Figure 4.9 shows a graph of the error in locating the mean for a window of 7 pixels for mean displacements from -0.5 to 0.5 pixels from the center pixel for several standard deviations. This figure shows that the trend noticed in the previous methods (the centroid and single pulse) have reappeared. The smallest error occurs when data from two standard deviations around the mean is used. Indeed, the increase in error for all standard deviations larger than 1 is larger than is acceptable (in Figure 4.9 all other curves do not fit the criteria).

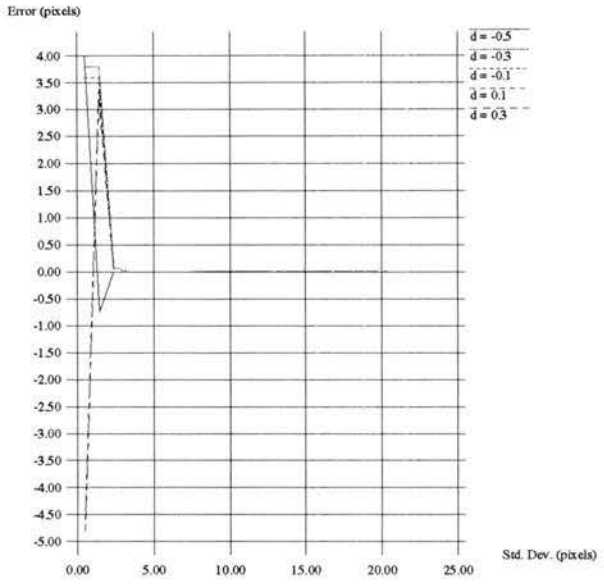


Figure 4.8: 1–D Multi Pulse Location Error for Constant Displacement d and Window Size 9 pixels

The next study held the sub-pixel displacement constant and varied the windows size and standard deviation. Figure 4.10 shows the error in locating the mean for a sub–pixel displacements of -0.5, -0.3, -0.1, 0.1, and 0.3 from the center pixel in a window of 7 pixels for standard deviations from 0.5 to 25. For very small, impulsive data, the error is huge. As the standard deviation increases from 0.5 to 3 pixels, the location error decrease to less than 0.05 pixels. The location error begins to increase again until at standard deviations greater than 5 pixels the error is larger than 0.10 pixels. The SNR for this system can also be examined; Table 4.6 shows the mean of the SNR for the standard deviations for Figure 4.9. The table shows that the SNR improved for every increase in standard deviation; as has been noted, however, this did not necessarily improve the mean estimate. The proceeding clearly reinforces the

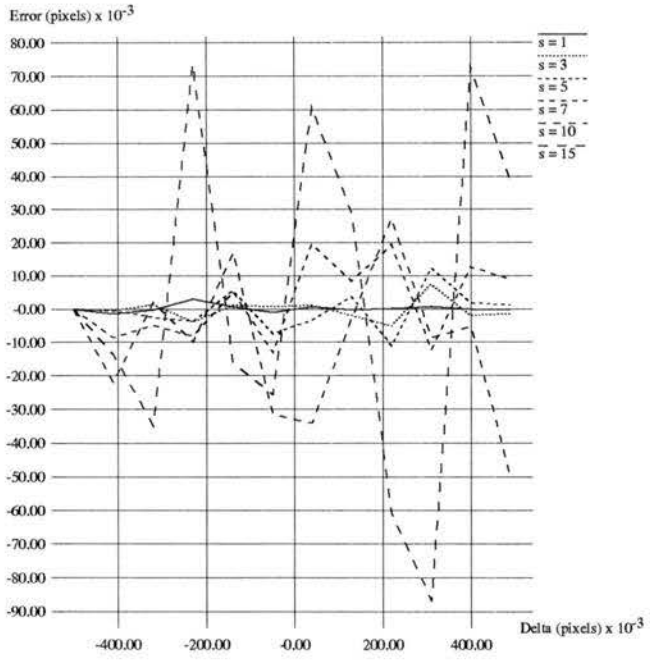


Figure 4.9: 1–D Multi Pulse Location Error for Quantized Data with Constant Standard Deviation s with Window Size 7 pixels

Std. Dev. (pixels)	SNR
1	454.77
3	719.82
5	812.65
7	955.16
10	895.01
15	933.72

Table 4.6: Average SNR for each Standard Deviation s in Figure 4.10

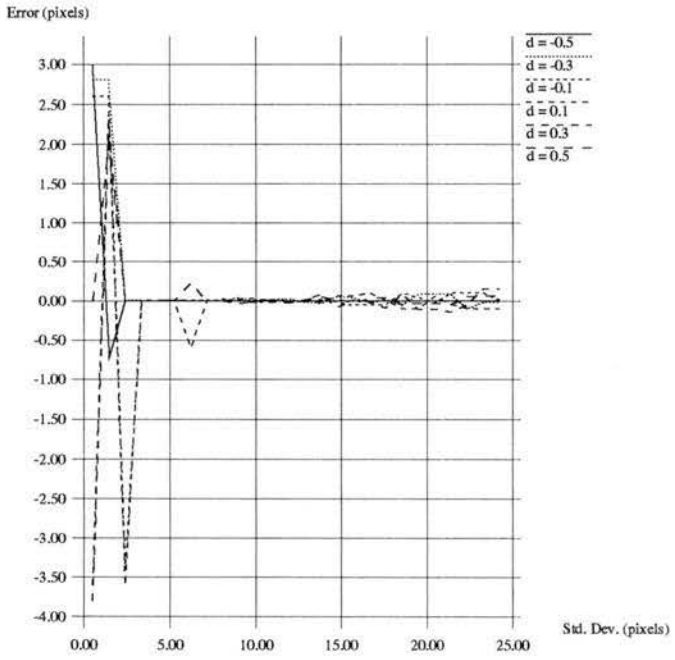


Figure 4.10: 1–D Multi Pulse Location Errors with Constant Displacement d and Window Size 7 pixels using Quantized Data

requirement that data be from within two standard deviations from the mean.

Additive Noise

The effects of processing noise were simulated by running another set of trials as above, using quantized data that had Gaussian noise added.

In Figure 4.11, the test results for all possible subpixel displacements, standard deviations of 2, 3, 5, 7, 10, and 15 with a window size of 7 pixels can be found. (The standard deviation of 1 pixel is not shown as neither method would converge on a solution in a reasonable amount of time.) The noise adds enough error to the process

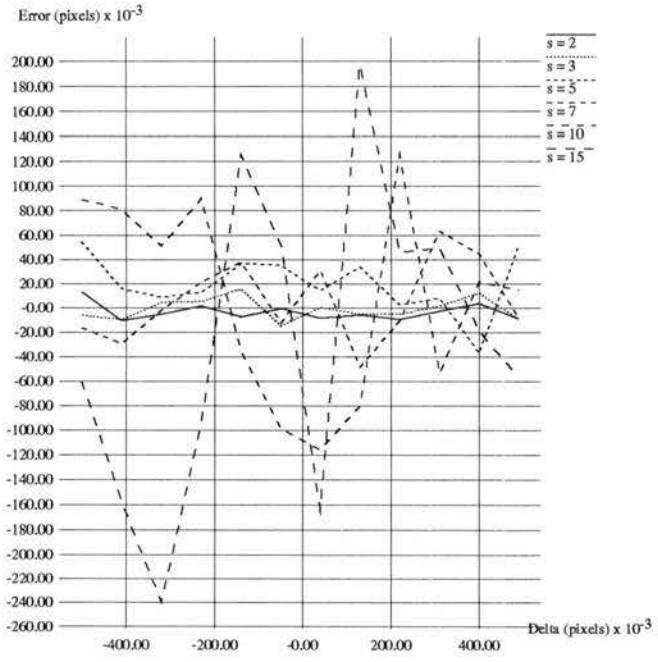


Figure 4.11: 1–D Multi Pulse Location Errors with Constant Standard Deviation s and Window Size 7 pixels using Noisy, Quantized Data

to take the results out of the acceptable limits (that is, less than $\epsilon < 0.05$ pixels).

Std. Dev. (pixels)	SNR
1.0	128.12
1.1	126.69
2.0	166.00

Table 4.7: Average SNR for each Standard Deviation s in Figure 4.11

Table 4.7 shows that the mean SNR is about $\frac{1}{3}$ of the previous case. Clearly, large signal to noise ratios are required to insure that the non–linear parameter estimation analysis produces results of sufficient accuracy. For Figure 4.11, the mean of the SNR for the standard deviations are listed in Table 4.7.

This is further supported by the next graph, which shows the effects of changing the standard deviation for a constant displacement. (The graph is Figure 4.12.) The standard deviation for the pulse is varied from 0.5 pixels to 24.5 pixels. Several displacements (-0.5, -0.3, -0.1, 0.1, 0.3, 0.5) are shown.

The results for the least squares methods are similar to the ones presented above; the errors are in general slightly less.

In summary, the one dimensional multiple pulse analysis methods do provide and acceptable means of locating the mean of a blob provided that the blob is not too impulsive, the window used is about two standard deviations on either side of the mean, and the SNR is reasonably large.

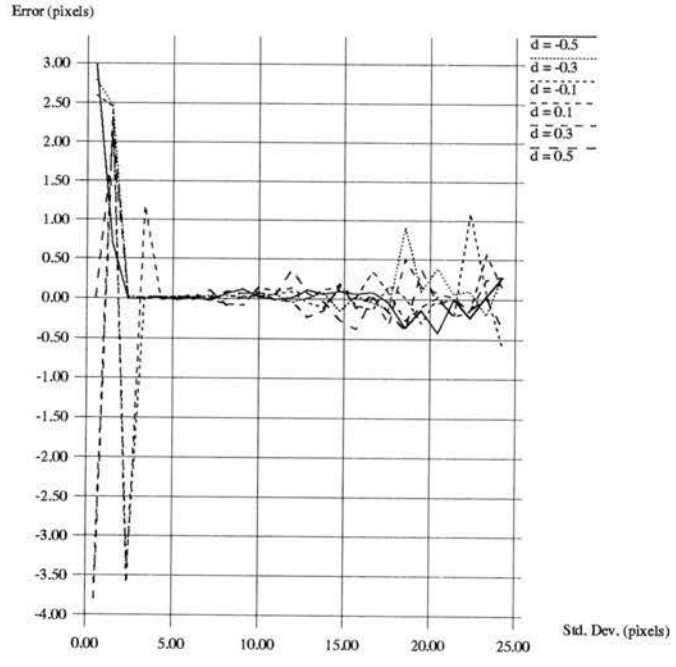


Figure 4.12: 1–D Multi Pulse Location Errors with Constant Displacement d and Window Size 7 pixels using Noisy, Quantized Data

4.3.3 2–D Multi Pulse Formulation

In two dimensions, there are again two possible light models, depending on the pulse type. In the circularly symmetric case, the light is modeled as

$$f(x,y) = \sum_{l=1}^{3} A_l e^{-\left(\frac{(x-\mu_x)^2+(y-\mu_y)}{\sigma_l}\right)^2} \tag{4.35}$$

In the non–symmetric case, the light is modeled as

$$f(x,y) = \sum_{l=1}^{3} A_l e^{-\left(\left(\frac{x-\mu_x}{\sigma_{x,l}}\right)^2+\left(\frac{y-\mu_y}{\sigma_{y,l}}\right)^2\right)} \tag{4.36}$$

As in the 1–D case, these models provide the basis for the simplex and least squares functions. The function for the circularly symmetric simplex case is again the sum squared error between the imaged data and the model:

$$s(\theta) = \sum_{i=1}^{n}\sum_{j=1}^{m}\left(z_{i,j} - \sum_{l=1}^{3} A_l e^{-\left(\frac{(x_i-\mu_x)^2+(y_j-\mu_y)^2}{\sigma_l^2}\right)}\right)^2 \tag{4.37}$$

where θ is the vector of parameter to solve; $\theta = [\theta_k]^T = [\mu_x\ \mu_y\ A_1\ \sigma_1\ A_2\ \sigma_2\ A_3\ \sigma_3].^T$ The A_l and σ_l represent the amplitude and standard deviation of the lth Gaussian, μ_x, μ_y are the column and row means of the Gaussians, and $z_{i,j}$ represents the (i,j)th image data point. x_i is the ith column coordinate, and y_j is the jth row coordinate. A total number of image data points (and therefore coordinates) is $n \times m$. The more general non–symmetric case is given by

$$s(\theta) = \sum_{i=1}^{n}\sum_{j=1}^{m}\left(z_{i,j} - \sum_{l=1}^{3} A_l e^{-\left(\left(\frac{(x_i-\mu_x)}{\sigma_{x,l}}\right)^2+\left(\frac{(y_j-\mu_y)}{\sigma_{y,l}}\right)^2\right)}\right)^2 \tag{4.38}$$

where θ is the vector of parameters to solve:

$$\theta = [\theta_k]^T = [\mu_x\ \mu_y\ A_1\ \sigma_{x,1}\ \sigma_{y,1}\ A_2\ \sigma_{x,2}\ \sigma_{y,2}\ A_3\ \sigma_{x,3}\ \sigma_{y,3}]^T$$

The $\sigma_{x,l}$ and $\sigma_{y,l}$ represent the column wise and row wise standard deviations, respectively. The rest of the symbols have the same definitions as in the circularly symmetric case.

Equations 4.35 and 4.36 also lead directly to the least squares residual vectors. The standard implementations of the non–linear least squares algorithms expect a vector of residuals, not an array of residuals. This is dealt with by recasting the 2–D array of image data points $Z_{n,m}$ and the corresponding coordinates as vectors. That is, the original matrix $Z_{n,m}$ with n rows and m columns is stored as the vector z, which is $n \times m$ places long. Each of the n rows is stored sequentially in z, so that the first row's data starts at $z(0)$, the second row's data at $z(m)$, and so on. The row and column coordinates of the image data points Z are also stored as vectors, called x and y; these vectors are also of size $n \times m$. These two vectors contain essentially repeated data. This process allows the 2–D problem to be cast in a 1–D form.

With these adjustments made, the 2–D circularly symmetric least squares residual vector $\mathbf{r}$ becomes:

$$\mathbf{r}(\theta) = \mathbf{z} - \sum_{l=1}^{3} A_l e^{-\left(\frac{(\mathbf{x}-\mu_x)^2+(\mathbf{y}-\mu_y)^2}{\sigma_l^2}\right)} \tag{4.39}$$

where θ is the vector of parameter to solve; $\theta = [\theta_k]^T = [\mu_x\ \mu_y\ A_1\ \sigma_1\ A_2\ \sigma_2\ A_3\ \sigma_3]^T$, and $\mathbf{z}$ is the vector of observed data points cast in 1–D form, and $\mathbf{x}$ and $\mathbf{y}$ are the vectors of coordinates corresponding to the observed data points. The A_l and σ_l represent the amplitude and standard deviation of the lth Gaussian, μ_x, μ_y are the column and row means of the Gaussians. The Jacobian $J^{(a)}$ is directly obtainable (numerically) from equation 4.39. The more general non–symmetric residual vector $\mathbf{r}$ is given by

$$\mathbf{r}(\theta) = \mathbf{z} - \sum_{l=1}^{3} A_l e^{-\left(\left(\frac{(\mathbf{x}-\mu_x)}{\sigma_{x,l}}\right)^2+\left(\frac{(\mathbf{y}-\mu_y)}{\sigma_{y,l}}\right)^2\right)} \tag{4.40}$$

where θ is the vector of parameter to solve:

$$\theta = [\theta_k]^T = [\mu_x \ \mu_y \ A_1 \ \sigma_{x,1} \ \sigma_{y,1} \ A_2 \ \sigma_{x,2} \ \sigma_{y,2} \ A_3 \ \sigma_{x,3} \ \sigma_{y,3}]^T$$

The $\sigma_{x,l}$ and $\sigma_{y,l}$ represent the column wise and row wise standard deviations, respectively. The rest of the symbols have the same definitions as in the circularly symmetric case. The Jacobian $J^{(a)}$ is directly obtainable (numerically) from equation 4.40. The form of these residuals is almost identical to the 1–D case.

4.3.4 2–D Multiple Pulse Error Analysis

The two dimensional models were tested using simulated data to examine how they performed when exposed to the various error sources. These tests were performed using the circularly symmetric model. The test pulse was built up in the same way as the 1–D case. The window size w and the "base" standard deviation σ_1 were specified. From these two parameters the rest of the pulse parameters were derived: $\mu_x = w/2$, $\mu_y = w/2$, $\sigma_2 = 2.57\sigma_1$, $\sigma_3 = 4.97\sigma_1$, $A_1 = 154.25$, $A_2 = 77.68$, and $A_3 = 22.07$. The data presented is from the trials using the least squares technique; the simplex data was similar.

Spatial Sampling

The first test set examined the effects of spatial sampling. The trials consisted of varying the sub–pixel displacements (both in x and y) from -0.5 to 0.5 pixels over a range of standard deviations (s = 1, 3, 5, 7, 10, 15). The size of the support window n was given values $n = 5, 7, 9, 15, 25$. (The window used was an $n \times n$ square.) Table 4.8 shows the summary of the result.

Window Size	Standard Deviation	Min. Error	Mean Error	Error Variance	Max. Error
5	1	-0.000000	0.000003	0.000000	0.000017
	7	0.000000	0.000004	0.000000	0.000016
15	1	-0.000000	0.000000	0.000000	-0.000000
	7	-0.000000	0.000000	0.000000	0.000016

Table 4.8: 2–D Multi Pulse Location Error by Window Size using Spatial Quantization

From this table (Table 4.8) it is obvious that spatial quantization does not introduce significant error into the process.

Quantization Noise

The second trial set consisted of substituting quantized data for the real valued data used in the previous trial; the rest of the experimental configuration is the same. Table 4.9 shows the summary of the result.

Window Size	Standard Deviation	Min. Error	Mean Error	Error Variance	Max. Error	Mean SNR
5	1	0.0000	0.0009	0.0000	0.0034	334.27
	7	0.0000	0.0060	0.0000	0.0285	845.06
15	1	0.0000	0.0009	0.0000	0.0027	117.37
	7	0.0000	0.0011	0.0000	0.0042	603.92

Table 4.9: 2–D Multi Pulse Error by Window Size using Intensity Quantized Data

The trends noted already reappear here. As long as data from two standard deviations from the mean are present, the algorithms perform well. When insufficient data is available, unacceptable errors result. However, having too large a window

does not necessarily result in excessive errors.

Additive Noise

The last trial set added the zero mean, single pixel standard deviation Gaussian noise to the quantized data of the previous trial. Table 4.10 shows the summary of the result.

Window Size	Standard Deviation	Min. Error	Mean Error	Error Variance	Max. Error	Mean SNR
5	1	-0.0000	0.0033	0.0000	0.0153	94.42
	7	0.0001	0.0222	0.0004	0.0857	238.21
15	1	0.0001	0.0031	0.0000	0.0128	33.48
	7	0.0001	0.0041	0.0000	0.0139	169.87

Table 4.10: 2–D Multi Pulse Error by Window Size using Intensity Quantized Data with Noise Added

The results of the noise trial indicate that the noise can cause the location errors to be too large. However, when compared with the quantization only experiment, it is clear that moderate amounts of noise can be tolerated.

In summary, the 2–D non–linear estimation techniques both show promise for locating the mean of blobs provided that several conditions are met. First, the data for the blobs must come from an area two standard deviations about the mean of the distribution. Second, the SNR must be relatively high.

4.4 Conclusions for Blob Location Using Parameter Estimation

There are several points that the preceeding analysis has shown. First, two different approaches to blob location through parameter estimation (linear and non–linear) were studied. Both techniques show promise for successfully locating blobs. We see no reason why the process cannot be adapted to other blob models. Second, parameter estimation techniques can localize the mean of a blob successfully if the window size is matched to the blob size. Third, the SNR of the signal plays a minor role in determining how well the function will locate means; moderate amounts of imaging noise can be tolerated.

Chapter 5

Experimental Results

The previous chapters outlined several possible methods of locating the center of a roughly Gaussian blob such as we would expect from the image of an active target. The next step is to experimentally determine if the techniques work as well as the simulations suggest. This was done by collecting a suite of images of active targets in which the position of the target is mechanically displaced by known increments. The means of each blob are located, and the difference between them and the expected difference can be compared. In this chapter we report on the results of these tests.

The rest of this chapter is divided into six main sections. The first section describes the experimental apparatus used. The calibration procedure used to calibrate the lab cameras is described in the second section. The third section discusses the process of extracting the parts of the image that belong to the blob (this is called segmentation). The fourth section presents a detailed description of the experiments that were run. The fifth section discusses the experimental results. The last section summarizes the results presented in previous section.

5.1 Experimental Apparatus

The experiments consist of moving a camera a known displacement and determining what the change in the blob position is. For simplicity, only horizontal motion is considered.

The apparatus to set up and measure such displacements consists of the following items:

1. A Pulnix TM–7 NTSC black & white camera with 6mm lens.
2. A computer with frame grabber (SUN VideoPix 24 bit frame grabber).
3. A translation stage and tripod with pan and tilt mount.
4. A vernier caliper.
5. A passive target, with known functional form.
6. An active target.
7. A camera calibration system.

The camera is mounted on the translation stage, which in turn is mounted on the tripod. The translation stage allows the camera's horizontal position to be adjusted easily. The pan and tilt mount on the tripod allow the camera to be aligned with the rest of the system. The calipers, which have an accuracy of $\frac{1}{1000}$ of an inch, are used to precisely measure the translations the camera makes along the translation stage.

As mentioned above, the experiments seek to verify that the location methods are capable of detecting minute changes in position. This requires that an expected

change in position be known in advance. Only a calibrated camera can provide such information. A camera is calibrated when the parameters of its perspective transform are known (that is, its focal length, array width, etc.). The calibration procedure, used to estimate these parameters, is explained in the following section.

The 'calibration box' is used to facilitate calibration and to hold the target to be imaged. Two different targets are used. The first target is passive, consisting of a printed Gaussian pulse. This target has the advantage of possessing a known functional form. The second target is active, consisting of a model 1488 14V light bulb mounted behind a screen made of $\frac{1}{8}$th plex–glass with one side sand blasted. The plexi–glass is required to insure that the light intensity does not saturate the camera, yielding an unusable pulse.

A picture of the experimental apparatus is shown in Figure 5.1. (Imaging the active target is shown.) In the foreground the camera on its translation stage and tripod is visible. The calibration box is seen in the background. The outline of the screen is visible below the middle of the box, as is the blob created by the light bulb. The passive target is imaged by taping the paper with the target to the box so that the bottom edge of the paper and box line up.

Images are digitized using the commercial program "VideoPix." To insure that images collected had good contrast, the gamma setting on the camera is set to 1.0 (instead of 0.45) indicating a linear mapping of imaged intensity to digital representation. The camera's electronic shutter was set to slowest possible speed, and the aperture was set to insure that the image did not saturate the CCD array (blooming). The camera observes the targets from distances between 8 and 16 feet. (The following section will describe why the exact distance is not required.)

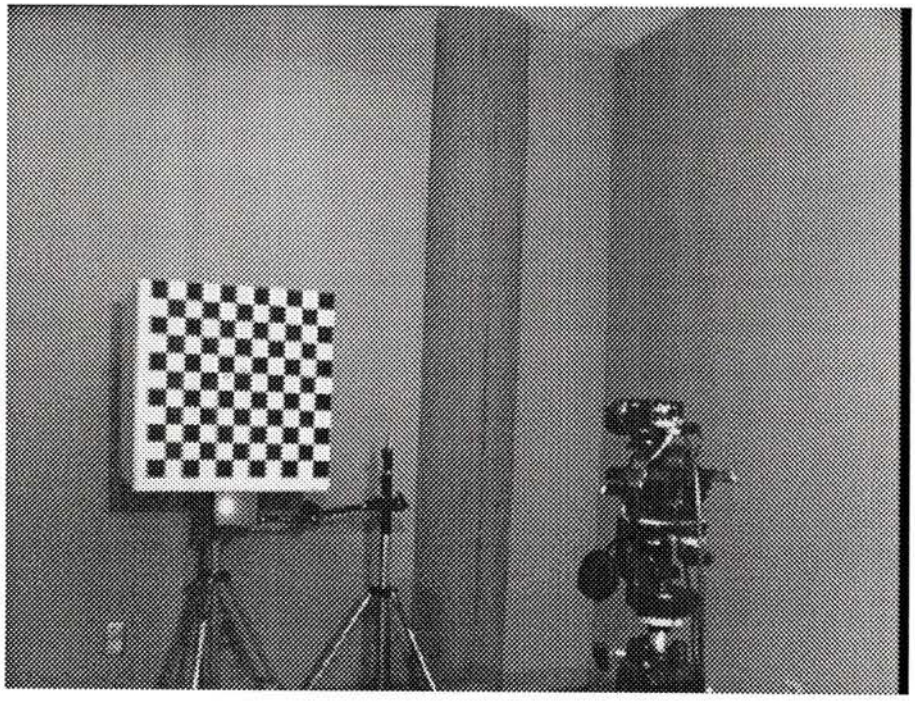


Figure 5.1: Lab Setup for Experiments

5.2 Calibration

As mentioned above, the parameters of the camera's perspective transform must be known to be able to determine in advance what the expected image position change for a given real world displacement is. However, no such camera was easily available. This is due in part to the fact that a complete camera calibration is a difficult operation to perform. The procedure described below, however, permits sufficient information about the imaging geometry to be obtained to allow the experiments to continue.

The surface of the calibration box is covered with squares. (This is shown in Figure 5.1.) The tripod and translation stage are set up so that the camera can be rotated on axes parallel to its optic axis and the vertical axis, and that it can be

translated along its image plane. This arrangement allows the camera to be oriented so that its image plane is parallel to the surface of the box. Once the image plane and the calibration box are parallel, the camera can be translated along the image plane without introducing perspective distortion as an additional locating error provided that the blobs are in the same plane as the front of the box. A 2–D schematic is shown in Figure 5.2.

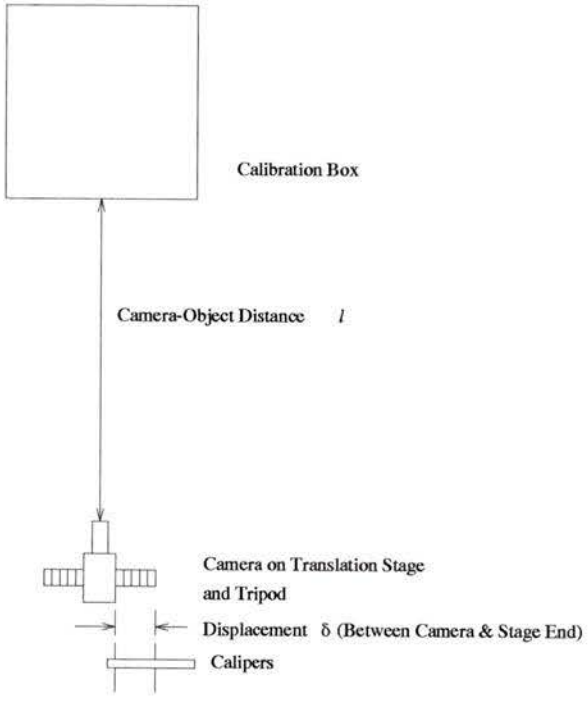


Figure 5.2: Schematic Layout of Experimental Apparatus

The image plane is made to be parallel by using the known geometry of the calibration box. All of the rectangles on that box (the rectangles measure 51.0 $\times$ 50.5mm) have the same orientation and size. Thus, when the image plane is parallel to the box, the imaged rectangles will have the same size, and the horizontal and vertical lines formed by the edges of the rectangles will also be horizontal and vertical. These

relationships can be verified visually on a video monitor, and the camera adjusted appropriately to compensate for any deviations. In order for these relationships to be most effectively exploited, the camera should be set up with the calibration box roughly in the middle of the image. This arrangement prevents the subtle errors due to the fact that one might be working on one side of the image center.

Once the camera and calibration box are aligned, the "pseudo-calibration" can begin. The camera is displaced many horizontal displacements, and images of the box are taken at each displacement. From the images, the location of a control point on the box can be determined for each displacement. The relative displacements between the images and the relative displacements of the control point can be used to determine what real world displacement provides a single pixel shift in the image using

$$1pixel = \left| \frac{du}{dreal} \right| \tag{5.1}$$

where du is the change in image coordinates and $dreal$ is the camera's displacement measured by the calipers. Once this calibration is complete, the trails can be run, using the derived single pixel displacement as a guide.

As noted in chapter 2, this procedure is carried out when the camera is warm (that is, powered up for at least one half hour) to minimize any pixel jitter errors. Camera displacements for the calibration and experiments were measured using a Mitutoyo Imperial caliper, which provides a $\frac{1}{1000}$ of an inch accuracy. A table of camera displacements against expected pixel displacements is found in Table 5.1, in Section 5.4.

5.3 Segmentation

One very important area of image processing is the selection of areas of interest in an image. This selection process is called segmentation.

The purpose of the work is to locate small blobs of light accurately. As discussed before, the blobs have a roughly Gaussian form. These restrictions limit the ability of standard isolation techniques to locate the area in question. As an example, consider the optimal matched detector described by Castleman [6]. The matched detector consists of a filter whose output is maximum in the presence of the spectrum of the signal it is designed to detect. Now a small, Gaussian object will have a Gaussian spectrum; since the object is spatially local the spectrum will be broad. Thus, the detector will have trouble finding the blob; moreover, filtering with such a detector is the same as low–pass filtering.

An alternate approach is to add some structure to the blob that makes it stand out. This can be done by causing the target to flash. The imaging system takes two images of the scene; the one has the target illuminated, the other doesn't. (The time between the image exposures must be small for this to work.) Subtracting the two images will result in one area—the target—being visible. The images that result from this process consist of a single, bright blob on a dark background. The maximum image intensity is near the center of the blob. This technique has been used experimentally; however, the subsequent image processing should use data from an image that has not had the constant background removed. Using "pure" data insures that the least amount of processing noise has been added.

Once the approximate blob location has been determined, the pixels belonging to the blob must be extracted. In our analysis of blob location methods it was

determined that blob location techniques performed best when presented with data that came from within two standard deviations of the mean. In fact, adding more data tended to decrease the accuracy of the calculation.

Two main methods of segmenting the image have been developed for this work. The first method is based on the optimal threshold detection scheme described by Shio [33]. The second scheme uses a heuristic developed by examining sample images.

Shio develops an automatic thresholding scheme for segmenting arbitrary images. The image is arbitrarily divided into a number of equal sized regions, each of which are processed individually. (It is assumed that any illumination changes occur slowly.) For each region, an initial threshold and a contrast measure are found. The initial threshold estimate is performed using the discriminant analysis method of Otsu [25]. This technique considers the image region to consist of two classes, the background C_0 and the object of interest C_1, each with distinct pixel gray levels. Gray level values are from $1 \ldots k$ for C_0 and $k+1 \ldots L$ for C_1, where k is the threshold between the classes and L is the maximum gray level value. The probability of each gray level is $p_i = \frac{n_i}{N}$, where n_i is the number of pixels with gray level i, and N is the total number of pixels being considered. The probability of a class occurring and its mean are given by

$$\omega_0 = \sum_{i=1}^{k} p_i = \omega(k) \tag{5.2}$$

$$\omega_1 = \sum_{i=k+1}^{L} p_i = 1 - \omega(k) \tag{5.3}$$

$$\mu_0 = \sum_{i=1}^{k} i\Pr(i|C_0) = \sum_{i=1}^{k} \frac{ip_i}{\omega_0} = \frac{\mu(k)}{\omega(k)} \tag{5.4}$$

$$\mu_1 = \sum_{i=k+1}^{L} i\Pr(i|C_1) = \sum_{i=k+1}^{L} \frac{ip_i}{\omega_1} = \frac{\mu_T - \mu(k)}{\omega(k)} \tag{5.5}$$

where ω_i are the class probabilities, μ_i are the probabilities of the class means, and $\mu_T = \mu(L) = \sum_{i=1}^{L} ip_i$ is the image mean gray level. These definitions can be used to find the class and image variances

$$\sigma_0^2 = \sum_{i=1}^{k}(i-\mu_0)^2 \Pr(i|C_0) = \sum_{i=1}^{k}(i-\mu_0)^2 \frac{p_i}{\omega_0} \tag{5.6}$$

$$\sigma_1^2 = \sum_{i=k+1}^{L}(i-\mu_1)^2 \Pr(i|C_1) = \sum_{i=k+1}^{L}(i-\mu_1)^2 \frac{p_i}{\omega_1} \tag{5.7}$$

$$\sigma_T^2 = \sum_{i=1}^{L}(i-\mu_T)^2 p_i \tag{5.8}$$

The discriminant criterion then measures the separation of the two classes by comparing the ratio of the class variances. The threshold value selected is the gray level that maximizes this ratio. The criterion can be written as the ratio of the background class variance and total image variance:

$$\eta = \frac{\sigma_0^2}{\sigma_T^2} \tag{5.9}$$

This can be simplified to finding the gray level k' that maximizes the variance of the background class

$$\sigma_{C_0}^2(k') = \max_{1 \le k < L} \sigma_{C_0}^2(k) \tag{5.10}$$

Shio modifies this threshold based on the thresholds and contrast measures in the regions surrounding the current region. This approach makes the technique illumination independent.

This technique was modified for use in this work. There is only one area that requires segmentation, and the rough position of the blob is known. The exact size of the blob is unspecified. An arbitrary window is selected around the blob maximum, and the optimal threshold for this window is determined. The size of the window is then increased, and the new optimal threshold found. If this threshold is the same as

the previous threshold, then the best threshold has been deemed found. If not, then the entire process iterates again.

Once the threshold is found, the window consisting of all pixels greater than the threshold is determined. This window is used by the blob location code.

The heuristic segmentation technique also uses the image histogram to compute the best threshold. However, no attempt at optimal threshold selection is made. Instead, the threshold is found relative to the most common intensity in the image. The image consists predominately of background with a few relatively very bright pixels. It is reasonable, therefore, that the most common intensity be assumed to be part of the background. The heuristic sets the threshold to be the first intensity that is followed by 5 empty (or nearly empty) intensity bins. The window around the blob is selected as before, by tracing down the contours of the blob until pixels with intensities at or below the threshold are found.

Two variations on the histogram heuristic are also implemented. First, the window determined by the segmentation scheme can be arbitrarily extended by a (small) number of pixels. This has the effect of including more of the blob distribution in the window; having the tails of the distribution included can improve the performance of the location schemes.

When digitizing small, bright blobs (width of 10 pixels of so), the image obtained is not very uniform. (See Figure 5.3 for a graphical display of a typical image row.) Along any row of the image the intensities fall off very rapidly (compared to the rising edge of the blob). This does not occur if the blob is relatively (80 pixels or more) wide. This phenomenon has been observed using two different frame buffers, and as such seems to be a camera artifact. To prevent this from affecting the results, the

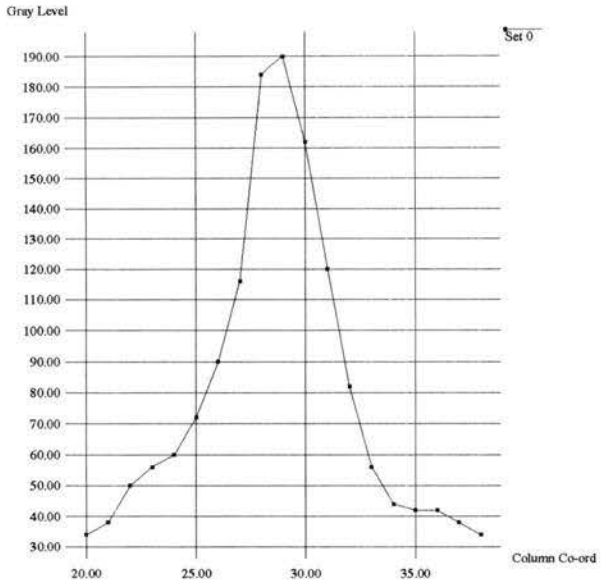


Figure 5.3: Typical Image Row with Rapid Intensity Decay on Right Side

trailing edge of the distribution can be discarded. Thus, the second variation on the histogram heuristic used performs the histogram segmentation as before, but limits the right side of the window to be one pixel beyond the maximum intensity value.

5.4 Experiments

As discussed above, two sets of trials were run. The first used a passive, Gaussian target, and the the second used a masked light bulb. The trials tried to verify that the three techniques discussed in Chapters 3 and 4 would perform as expected when presented with real image data.

Each trial was divided into three parts. These parts consisted of varying levels of subpixel shift. The first part shifted the target in 1.00 pixel increments; this was the

Distance to Camera	Image Displacement (Pixels)	Req'd Camera Shift (inches)
$\approx$ 8 ft	1.00	0.1694
$\approx$ 16 ft	0.10	0.0294
$\approx$ 16 ft	0.05	0.0147

Table 5.1: Calibrated Camera Shifts for Expected Displacement

minimum accuracy each general technique had to have. The second part shifted the target in 0.10 pixel increments; this tested the techniques to the limits suggested by the literature (for example, see [5]). The third part shifted the target in 0.05 pixel increments; this was designed to test the techniques to their theoretical best accuracies and the design constraints. Note that such tests have never been done before.

For each displacement set, the camera was set up some distance away from the calibration box. The camera was aligned (to the limits of the imaging accuracy) so that the image plane was parallel to the front of the box and the horizontal axis of the camera was parallel to the horizontal lines on the calibration box. Once this alignment was complete, the system was calibrated as described previously, obtaining the required camera displacement for the given pixel displacement. The distance from the camera to the box was selected primarily on the basis of how easy it was to displace the camera the required amount. Table 5.1 shows the calibrated displacements for the expected sub–pixel shifts. Note that the accuracy presented in the table is unmeasurable; however, keeping the extra accuracy allowed for better displacement calculations over the entire series of images.

Each experiment used a total of 11 displacements. Thus, for the 1.00 pixel shifts the total shift from first to last image was 10 pixels; for 0.10 pixel shifts, 1 pixel;

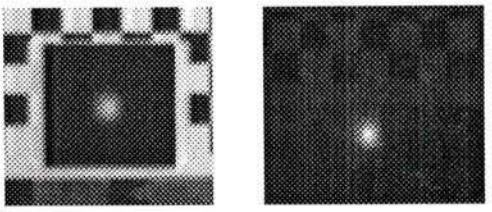
for 0.05 pixel shifts, 0.5 pixels. A total of five images $I_a, \ldots, I_e$ were taken at each displacement of the camera. The five images were averaged together (using floating point arithmetic) such that

$$I_m = (I_a + I_b + I_c + I_d + I_e)/5.0$$

to provide two test sets (I_a and I_m) in each trial. The averaged images were collected to determine if imaging noise could be removed from data by simply averaging as was noted in [26]. The first test set consisted of one of the original five images for each displacement; the second test set consisted of the averaged images. The alignment of the camera in the calibration stage allowed the assumption that the camera motion was strictly horizontal.

For the passive target tests, the blob image was taped to the bottom of the calibration box. (The target was attached so that it was completely supported by the box. The paper was not free to sway with available breeze.) The room lighting was turned off and the Image Lab's photo floodlights were turned on to insure that the target was evenly illuminated. The camera's aperture was adjusted so that the maximum image intensity value was less than 250. (The frame grabber used quantizes the image into eight bit words. Keeping the maximum intensity to less than 250 insured that the frame buffer was never saturated. Saturated images of the blob would undermine the theoretical analysis preformed in the preceeding chapters.) The background on the paper the target was printed on insured that a region in the image had a bright spot with a dark background.

In the case of the active target, the screen was attached to the bottom of the calibration box and the light bulb mounted on a small distance (about 3 cm) behind it. All room lighting (including the photo floods) was turned off, and the camera's



(a) Passive Sample (b) Active Sample

Figure 5.4: Sample Passive and Active Images

aperture was adjusted to make the maximum imaged intensity value less than 250. The result was another set of bright dots on a dark background. Sample images for the passive and active cases are found in Figure 5.4.

The images were cropped to be between 50 and 60 pixels square from the 640×480 pixel image provided by the frame buffer. The cropping was performed to simplify the image collecting process.

In summary, three different displacement trials were run on two types of images. The two image types are passive (the printed blob) and active (the light bulb). For each image type two data sets are collected: a single look image set and a multiple image averaged data set.

5.5 Experimental Results

The results of the experiments are presented several tables below. They are grouped by location method: centroid, single pulse (linear) parameter estimation, and multiple pulse (non–linear) parameter estimation. First, however, are some comments on the various segmentation techniques.

The Shio/Otsu based optimal threshold measure performed the task of segmenting reasonably well. However, it tended to avoid including much of the tails of the pulses in the window about the blob maximum. This behaviour was to be expected, as the blob decays to the background noise level. Thus, the segmented region includes only the very bright portions of the blob. The performance of the locators improves as more of the tails of the pulses are included.

The heuristic histogram segmentation and extended histogram segmentation did include more of the pulse than did the optimal thresholding described above. This indicates that for the image types segmented, the heuristic was good. The shortened window was designed to remove the rapidly decaying right hand side of the images; this also correctly occurred.

All techniques did provide a reasonable segmentation; the heuristic based methods tended to include more of the tails of the pulse.

The results for the centroid locater experiments are found in table 5.2 (for passive data) and table 5.3. The table is partitioned into three parts, one for each expected pixel displacement. In each part the mean displacement μ and the variance σ^2 in mean displacement between each of the test images is shown against the thresholding technique used. All units are in pixels. Note that the shortened histogram window was not considered.

The table clearly shows that the centroid algorithm can properly detect changes of one pixel. Changes of 0.05 pixels, however, do not get tracked at all. The 0.10 pixel displacements were not quite properly tracked. Table 5.3 shows that the displacements in the active data set are overestimated by about 20%. Figure 5.5 shows the trend for the active dataset; the graph does display the expected total shift of one

Threshold Technique	Image	Passive	
		Mean Displacement	Variance of Mean
Expected Displacement: 0.05 pixels			
Shio & Otsu	single	0.1707	0.0089
	multi	0.1453	0.0093
Histogram	single	0.1561	0.0148
	multi	0.1734	0.0103
Extended Histogram	single	0.1578	0.0149
	multi	0.1734	0.0103
Expected Displacement: 0.10 pixels			
Shio & Otsu	single	0.1856	0.0501
	multi	0.1749	0.0299
Histogram	single	0.1853	0.0426
	multi	0.1717	0.0377
Extended Histogram	single	0.1853	0.0426
	multi	0.1717	0.0377
Expected Displacement: 1.00 pixels			
Shio & Otsu	single	1.0084	0.0457
	multi	1.0093	0.0333
Histogram	single	0.9956	0.0274
	multi	0.9886	0.0218
Extended Histogram	single	0.9956	0.0274
	multi	0.9886	0.0218

Table 5.2: Centroid: Average Change in Estimated Blob Location for Passive Data

Threshold Technique	Image	Active	
		Mean Displacement	Variance of Mean
Expected Displacement: 0.05 pixels			
Shio & Otsu	single	0.0476	0.0018
	multi	0.0285	0.0012
Histogram	single	0.0658	0.0032
	multi	0.0623	0.0045
Extended Histogram	single	0.0762	0.0034
	multi	0.0623	0.0045
Expected Displacement: 0.10 pixels			
Shio & Otsu	single	0.1246	0.0126
	multi	0.1271	0.0119
Histogram	single	0.1402	0.0158
	multi	0.1331	0.0114
Extended Histogram	single	0.1402	0.0158
	multi	0.1331	0.0114
Expected Displacement: 1.00 pixels			
Shio & Otsu	single	0.9857	0.0196
	multi	0.9783	0.0194
Histogram	single	0.9998	0.0160
	multi	0.9994	0.0260
Extended Histogram	single	0.9998	0.0160
	multi	0.9994	0.0260

Table 5.3: Centroid: Average Change in Estimated Blob Location for Active Data

pixel. However, the individual steps to get from the first image to the last image are variable in size. This indicates that the centroid method is accurate to about 0.25 pixels. The passive data set shows much greater error that the active data set, but examining a graph of the calculated displacements indicates that the possibility of an error in collecting the passive data cannot be ruled out. Using the averaged image improved the variance of the determined means, but it did not significantly improve the actual estimate. (Note that in Figure 5.5 the horizontal axis is labeled 'Disp;' the displacement referred to is that of the translation stage when the corresponding image was obtained.)

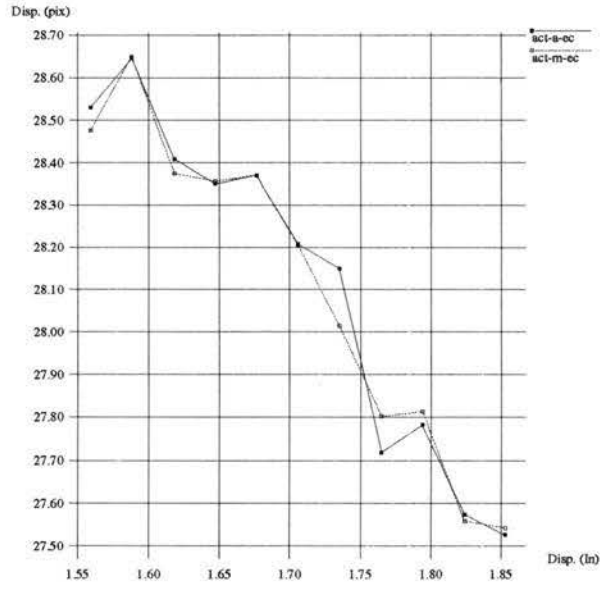


Figure 5.5: Centroid Results: 0.10 Pixel Expected Displacements for the Active Data. Single look is denoted by 'act-a-ec,' multi look by 'act-m-ec'

The results for the single pulse, linear parameter estimation experiments are presented in Tables 5.4 (the passive data set) and 5.5 (the active data set). These tables

Threshold Technique	Image	Circular Symmetric		Non–Symmetric	
		Mean Displacement	Variance of Mean	Mean Displacement	Variance of Mean
Expected Displacement: 0.05 pixels					
Shio & Otsu	single	0.0402	0.0010	0.0397	0.0011
	multi	0.0394	0.0009	0.0390	0.0010
Histogram	single	0.0339	0.0005	0.0326	0.0005
	multi	0.0460	0.0015	0.0456	0.0014
Extended Histogram	single	0.0341	0.0007	0.0333	0.0007
	multi	0.0453	0.0015	0.0450	0.0014
Shortened Histogram	single	0.0346	0.0005	0.0349	0.0006
	multi	0.0471	0.0019	0.0468	0.0019
Expected Displacement: 0.10 pixels					
Shio & Otsu	single	0.0585	0.0012	0.0580	0.0011
	multi	0.0545	0.0016	0.0540	0.0015
Histogram	single	0.3565	0.1080	0.3565	0.1080
	multi	0.0886	0.0023	0.0886	0.0023
Extended Histogram	single	0.0579	0.0021	0.0562	0.0021
	multi	0.0693	0.0020	0.0696	0.0018
Shortened Histogram	single	0.0510	0.0019	0.0514	0.0021
	multi	0.0725	0.0026	0.0724	0.0025
Expected Displacement: 1.00 pixels					
Shio & Otsu	single	0.1696	0.0797	0.1704	0.0795
	multi	0.1677	0.0805	0.1667	0.0805
Histogram	single	0.2275	0.0780	0.2275	0.0780
	multi	0.1655	0.0640	0.1655	0.0640
Extended Histogram	single	0.1742	0.0836	0.1737	0.0847
	multi	0.1688	0.0835	0.1695	0.0836
Shortened Histogram	single	0.1750	0.0868	0.1749	0.0871
	multi	0.1654	0.0813	0.1658	0.0825

Table 5.4: Single Pulse Parameter Estimation: Average Change in Estimated Blob Location for Passive Data

Threshold Technique	Image	Circular Symmetric		Non–Symmetric	
		Mean Displacement	Variance of Mean	Mean Displacement	Variance of Mean
Expected Displacement: 0.05 pixels					
Shio & Otsu	single	0.0402	0.0031	0.0424	0.0042
	multi	0.0400	0.0033	0.0434	0.0044
Histogram	single	0.0331	0.0015	0.0334	0.0018
	multi	0.0423	0.0050	0.0407	0.0044
Extended Histogram	single	0.0370	0.0014	0.0344	0.0018
	multi	0.0420	0.0049	0.0403	0.0042
Shortened Histogram	single	0.0365	0.0015	0.0420	0.0015
	multi	0.0433	0.0041	0.0456	0.0044
Expected Displacement: 0.10 pixels					
Shio & Otsu	single	0.0352	0.0034	0.0384	0.0026
	multi	0.0380	0.0031	0.0425	0.0023
Histogram	single	0.6136	0.1558	0.6136	0.1558
	multi	0.5091	0.0700	0.5091	0.0700
Extended Histogram	single	0.0520	0.0031	0.0459	0.0031
	multi	0.0417	0.0038	0.0403	0.0038
Shortened Histogram	single	0.0522	0.0022	0.0520	0.0020
	multi	0.0434	0.0040	0.0462	0.0036
Expected Displacement: 1.00 pixels					
Shio & Otsu	single	0.0736	0.0054	0.0804	0.0056
	multi	0.0734	0.0046	0.0814	0.0049
Histogram	single	0.3382	0.0769	0.3382	0.0769
	multi	0.2741	0.0303	0.2741	0.0303
Extended Histogram	single	0.0821	0.0076	0.0785	0.0069
	multi	0.0813	0.0047	0.0812	0.0049
Shortened Histogram	single	0.0794	0.0054	0.0813	0.0056
	multi	0.0845	0.0048	0.0835	0.0048

Table 5.5: Single Pulse Parameter Estimation: Average Change in Estimated Blob Location for Active Data

present the results for the two models (circularly symmetric and non–symmetric) developed in Chapter 4 in the same format as the preceding tables (Tables 5.2 and 5.3).

The tables show that the algorithm cannot track the 1.00 and 0.10 pixel displacements. In fact, looking at Figure 5.6 it is obvious that the algorithm is generating a lot of noise. For no combination of data set (active or passive), segmentation scheme

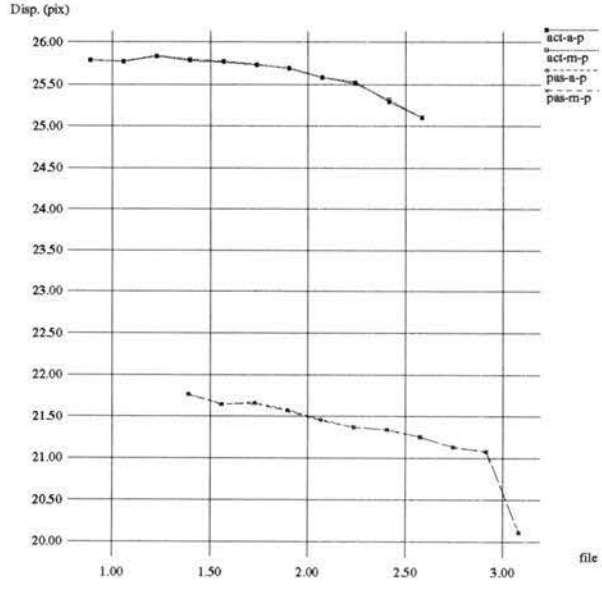


Figure 5.6: Single Pulse Parameter Estimation Results: 1.00 Pixel Expected Displacement Between Estimates

(Shio & Otsu, histogram, extended histogram, or shortened histogram), or imaging approach (single look versus multi look) did this algorithm produce worthwhile results. This leads directly to the conclusion that the single Gaussian pulse model does not describe the observed intensity distribution.

The results for the multiple pulse, non–linear parameter estimation experiments are presented in Tables 5.6 (the passive data set) and 5.7 (the active data set).

Threshold Technique	Image	Circular Symmetric		Non–Symmetric	
		Mean Displacement	Variance of Mean	Mean Displacement	Variance of Mean
Expected Displacement: 0.05 pixels					
Shio & Otsu	single	0.1569	0.0094	0.1591	0.0101
	multi	0.1678	0.0106	0.1677	0.0100
Least Squares	single	0.1551	0.0108	0.1591	0.0101
	multi	0.1661	0.0103	0.1685	0.0099
Expected Displacement: 0.10 pixels					
Shio & Otsu	single	0.1753	0.0330	0.1753	0.0338
	multi	0.1591	0.0241	0.1613	0.0254
Least Squares	single	0.1753	0.0330	0.1753	0.0338
	multi	0.1587	0.0239	0.1519	0.0261
Expected Displacement: 1.00 pixels					
Shio & Otsu	single	1.0083	0.0383	1.0099	0.0354
	multi	1.0107	0.0341	1.0120	0.0327
Least Squares	single	1.0083	0.0383	1.0099	0.0354
	multi	1.0106	0.0347	1.0097	0.0330

Table 5.6: Multi Pulse Parameter Estimation: Average Change in Estimated Blob Location for Passive Data

Threshold Technique	Image	Circular Symmetric		Non–Symmetric	
		Mean Displacement	Variance of Mean	Mean Displacement	Variance of Mean
Expected Displacement: 0.05 pixels					
Shio & Otsu	single	0.0568	0.0034	0.0556	0.0033
	multi	0.0523	0.0029	0.0799	0.0061
Least Squares	single	0.0571	0.0034	0.0556	0.0033
	multi	0.0513	0.0029	0.0786	0.0131
Expected Displacement: 0.10 pixels					
Shio & Otsu	single	0.1107	0.0086	0.1110	0.0085
	multi	0.1212	0.0065	0.1188	0.0068
Least Squares	single	0.1107	0.0086	0.1110	0.0085
	multi	0.1176	0.0072	0.1169	0.0195
Expected Displacement: 1.00 pixels					
Shio & Otsu	single	0.9905	0.0140	0.9891	0.0124
	multi	0.9886	0.0128	0.9863	0.0115
Least Squares	single	0.9905	0.0140	0.9891	0.0124
	multi	0.9884	0.0144	0.9853	0.0108

Table 5.7: Multi Pulse Parameter Estimation: Average Change in Estimated Blob Location for Active Data

As before, the format for these tables is similar to the preceding ones. Note that two different approaches to the non–linear parameter estimation are presented. The first four entries in each group report on using simplex optimization for the blob location; the thresholding heading describes which function was used with them. The last entry, least squares, reports on the use of Levenberg—Marquardt non–linear least squares parameter estimation; just the Shio/Otsu segmentation technique was applied to select the window.

Note that the tables seem to show that the methods resolved (at least on the active data set) to an accuracy of 0.05 pixels. This is not the case; examining the graph of the results (Figure 5.7) shows that the individual displacements are not properly tracked. Indeed, even in the 0.10 case the tracking is only good to 0.25 pixels.

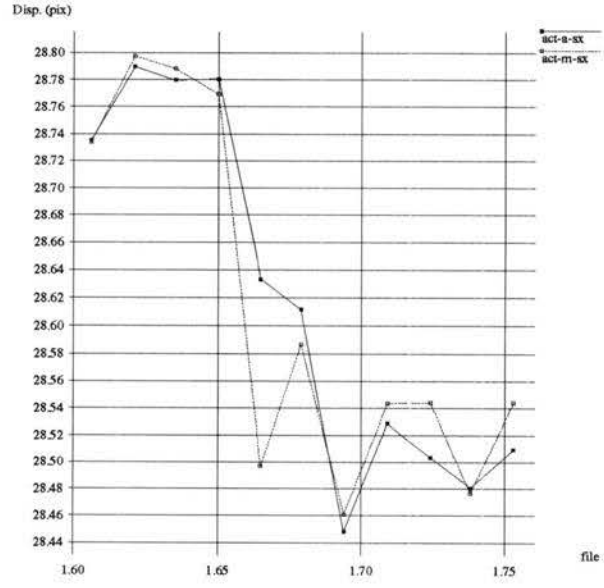


Figure 5.7: Multi Pulse Parameter Estimation Results: 0.05 Pixel Expected Displacement Between Estimates

The same model was used for both estimation techniques; the conclusion is that neither is more accurate. Lastly, the three pulse model is a reasonable approximation to the data.

5.6 Results Summary

The results detailed above can be summarized as follows.

1. The single Gaussian pulse model does not match what the camera images. Therefore, this model is inappropriate for blob location.

2. No method can provide sub–pixel location that is accurate to ± 0.05pixels.

3. The centroid method's best accuracy is about 0.25 pixel.

4. The non–linear methods' best accuracy is about 0.25 pixel. This is what the simulations predicted for a very noisy case.

5. The multiple Gaussian pulse did model the image data well;

6. The centroid method performed as well as the non–linear optimizations in the 1.00 pixel displacement case. In the 0.10 pixel displacement case, the non–linear methods performed slightly better on the active data set.

Chapter 6

Conclusions and Future Work

In this work, the problem of blob location has been considered. This function is an important part in the field of visual servoing, the process of controlling platforms or manipulators through the analysis of image data. Much of the visual servoing work in the literature has concentrated on passive targets. (These targets are what the device is to track.) Little work has been reported on active targets; these targets were best for certain other applications. The thesis concentrated on locating small, active targets which were modeled by a family of Gaussian blobs.

6.1 Future Work

Several areas in the imaging process require further study. The first of these areas is the camera digitization process. As noted in Section 5.3, the camera doesn't cleanly digitize small blobs well. This seems to be systemic; all images showed this problem. This noise source adds significantly to error introduced into the data stream. Ways of modeling this phenomenon should be investigated with a view to being able to remove it.

The second area of study is the work on locales by Havelock [13]. The technique would be to completely map the possible locales used by an entity function (preferably not the Gaussian considered) and store it either a neural net or a (large) table. Either approach would change the blob localization process into a look–up function. Such a modification would vastly increase the servoing speed possible.

Another area of promise is an investigation of the effects of perspective size change, as considered by [8]. This work showed that once an object has been recognized, it can be effectively tracked even as it changes scale. This was one area not touched on in this work.

6.2 Conclusions

Two main techniques of blob location have been considered, namely centroid detection and parameter estimation.

The use of centroids was analyzed theoretically with the result that location accuracies of 0.05 pixels were expected, provided that the process noise could be controlled. The experimental test suite indicated that the method of centroid can produce results accurate to 0.25 of a pixel.

Parameter Estimation was divided into two parts: simple linear estimation, and the more general non–linear estimation. The simple linear estimation indicated that accuracies of 0.05 pixels were possible, even in the presence of rather large noise. The experimental suite proved that the single Gaussian pulse model was a poor representation of reality.

Two non–linear parameter estimation techniques were considered: simplex function minimization and Levenberg—Marquardt least squares. The model used in both

cases was a sum of three Gaussian pulses, with common mean. This model was arrived at experimentally. The theoretical analysis indicated that both methods could locate a blob to an accuracy of 0.05 pixels if the amount of process noise could be minimized. The experimental suite indicated that accuracies of 0.25 pixels are achievable.

Location accuracies of 0.25 pixels are sufficiently good enough for the application presented by Terra Surveys. The use of the centroid algorithm would allow the implementation of the tracking system on a low–cost PC type computer, resulting in a feasible solution to Terra's tracking problem.

Bibliography

[1] A. Abutaleb, Z.J. Delalic, R. Ech, and J.A. Siegel. Automated analysis for scintigraphic evaluation of gastric emptying using invariant moments. *IEEE Transactions on Medical Imaging*, 8(4):364–370, December 1989.

[2] Peter K. Allen, Billibon Yoshimi, and Aleksandar Timecenko. Real–time visual servoing. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 851–856, 1991.

[3] Frank L. Alt. Digital pattern recognition by moments. *Journal of the Association for Computing Machinery*, 9:240–258, 1962.

[4] Yaakov Bar-Shalom, Hemchandra M. Shertukde, and Krishna R. Pattipati. Use of measurements from an imaging sensor for precision target tracking. *IEEE Transactions on Aerospace and Electronic Systems*, 25(6), November 1989.

[5] Chinmoy B. Bose and Isreal Amir. Design of fiducials for accurate registration using machine vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(12), December 1990.

[6] Kenneth R. Castleman. *Digital Image Processing.* Prentice-Hall, 1979.

[7] T.G. Cleaver and C.-L. Su. Registration techniques for image subtraction. In *Proceedings 1988 IEEE SOUTHEASTCON*, pages 205–210. IEEE, April 1988.

[8] Karen A. Dzialo and Robert J. Schalkoff. Control implications in tracking moving objects using time–varying perspective–projection imagery. *IEEE Transactions on Industrial Electronics*, IE–33(3):247–253, Aug 1986.

[9] Rafael C. Gonzalez and Paul Wintz. *Digital Image Processing.* Addison Wesley, Don Mills, Ontario, Canada, 2 edition, 1987.

[10] Andrew Grace. *Optimization Toolbox User's Guide.* Cochituate PLace, 24 Prime Park Way, Natick, MA, USA, 01760, 1990.

[11] Jan O. Hallset. Simple visual tracking of pipelines for an autonomous underwater vehicle. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2767–2772, 1990.

[12] Min-Hong Han, Sang-Yong Rhee, and Mun-Seoung Hong. Navigation control for a mobile robot. Technical Report of the Dept. of Industrial Engineering, Pohang Institute of Science and Technology, Pohang, KOREA, March 1991.

[13] David I. Havelock. Geometric precision in noise-free digital images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(10):1065–1075, October 1989.

[14] Jules S. Jaffe. Sensors for underwater robotic vision: Status and prospects. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2759–2766, 1991.

[15] N.S. Jayant and Peter Noll. *Digital Coding of Waveforms: Principles and Applications to Speech and Video.* Prentice–Hall, Inc., Englewood Cliffs, NJ., 1984.

[16] J.G. Kalbfleisch. *Probability and Statistical Inference*, volume 2: Statistical Inference. Springer-Verlag, New York, 2 edition, 1985.

[17] Behrooz Kamgar-Parsi and Behzad Kamgar-Parsi. Evaluation of quantization error in computer vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(9):929–939, September 1989.

[18] Alireza Khotanzad and Jong-Yih Chen. Unsupervised segmentation of textured edges by edge detection in multidimensional features. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(4):414–421, April 1989.

[19] Reimar Lenz and Dieter Fritsch. Accuracy of videometry with CCD sensors. *ISPRS Journal of Photogrammetry and Remote Sensing*, 45:90–110, 1990.

[20] Edward P. Lyvers, Owen Robert Mitchell, Mark L. Akey, and Antony P. Reeves. Subpixel measurements using a moment–based edge operator. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(12):1293–1309, December 1989.

[21] V. S. Nalwa and Thomas O. Binford. On detecting edges. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI–8(6):699–714, 1986.

[22] S. Negahdaripour, A. Shokrollahi, J. Fox, and S. Arora. Improved methods for undersea optical stationkeeping. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2752–2758, 1991.

[23] J.A. Nelder and R. Mead. A simplex method for function minimization. *The Computer Journal*, 7:308–313, 1965.

[24] Alan V. Oppenheim and Alan S. Willsky. *Signals and Systems.* Prentice–Hall, Inc., Englewood Cliffs, NJ., 1983.

[25] Nobuyuki Otsu. A threshold selection method from gray–level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC–9(1):62–66, January 1979.

[26] N. Papanikolopoulos, P. K. Khosla, and T. Kanade. Vision and control techniques for robotic visual tracking. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 857–864, 1991.

[27] William H. Press, Brian P. Flanner, Saul A. Teukolsky, and William T. Vetterling. *Numerical Recipies in C.* Press Syndicate of the University of Cambridge, Cambridge, England, 1988.

[28] Rick Quinn. Discussion concerning visual servoing for a water weed control device on 25 November 1991. Terra Surveys Ltd, Sidney, BC.

[29] James M. Rehg and Andrew P. Witkin. Visual tracking with deformation models. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 844–850, 1991.

[30] L.E. Scales. *Introduction to Non–Linear Optimization.* Macmillan, London, 1985.

[31] G.A.F. Seber and C.J. Wild. *Nonlinear Regression.* Wiley, New York, 1989.

[32] Yu shan Li, Tzay Y. Young, and Joseph A. Magrel. Subpixel edge detection and estimation with a microprocessor–controlled line scan camera. *IEEE Transactions on Industrial Electronics*, 35(1):105–112, Feb 1988.

[33] Akio Shio. An automatic thresholding algorithm based on an illumination–independent contrast measure. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 632–637. IEEE, June 1989.

[34] Harold W. Sorenson. *Parameter Estimation: Principles and Problems.* Marcel Dekker, Inc., New York, 1980.

[35] A. J. Tabatabai and O. R. Mitchell. Edge location to subpixel values in digital imagery. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI–6(2):188–200, 1984.

[36] Gilbert Verghese, Karey Lynch Gale, and Charles R. Dyer. Real–time motion tracking of three–dimensional objects. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2759–2766, May 1990.

[37] Ping Wah Wong. On quantization errors in computer vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(9):951–956, September 1991.

Appendix A

Alternate Centroid Calculations

This appendix develops the error present in the centroid calculation when the mean of the distribution doesn't fall on an integer pixel value, but where the displacement δ is negative. (Again, the restriction $-0.5 < \delta < 0$ is required.) We start with equation 3.13. As the distribution is to the left of the current point, the right side of the window will decay slightly sooner than the left. Thus, the left side is one pixel 'longer' than the right, leading to the following equations.

$$\hat{\mu}' = \frac{\sum_{i=-n}^{n-1} i e^{-\left(\frac{i-\delta}{\sigma}\right)^2}}{\sum_{i=-n}^{n-1} e^{-\left(\frac{i-\delta}{\sigma}\right)^2}} \tag{A.1}$$

Making the substitution $j = i - \delta$, or $i = j + \delta$,

$$\hat{\mu}' = \frac{\sum_{j=-n-\delta}^{n-\delta-1} (j+\delta) e^{-\left(\frac{j}{\sigma}\right)^2}}{\sum_{j=-n-\delta}^{n-\delta-1} e^{-\left(\frac{j}{\sigma}\right)^2}} \tag{A.2}$$

$$= \frac{\sum_{j=-n-\delta}^{n-\delta-1} \delta e^{-\left(\frac{j}{\sigma}\right)^2} + \sum_{j=-n-\delta}^{n-\delta-1} j e^{-\left(\frac{j}{\sigma}\right)^2}}{\sum_{j=-n-\delta}^{n-\delta-1} e^{-\left(\frac{j}{\sigma}\right)^2}} \tag{A.3}$$

$$= \delta + \frac{\sum_{j=-n-\delta}^{n-\delta-1} j e^{-\left(\frac{j}{\sigma}\right)^2}}{\sum_{j=-n-\delta}^{n-\delta-1} e^{-\left(\frac{j}{\sigma}\right)^2}} \tag{A.4}$$

Replacing j with $i = j - \delta$,

$$\hat{\mu}' = \delta + \frac{\sum_{i=-n}^{n-1} (i-\delta) e^{-\left(\frac{i-\delta}{\sigma}\right)^2}}{\sum_{i=-n}^{n-1} e^{-\left(\frac{i-\delta}{\sigma}\right)^2}} \tag{A.5}$$

The numerator of equation A.5 is split into two parts, the first part consisting of the positive terms and the other, the negative. Note that since δ is constrained to be negative, the term for $i = 0$ is positive (zero less a negative number is positive).

$$\hat{\mu}' = \delta + \frac{\sum_{i=0}^{n-1}(i-\delta) e^{-\left(\frac{i-\delta}{\sigma}\right)^2} + \sum_{i=-n}^{-1} (i-\delta) e^{-\left(\frac{i-\delta}{\sigma}\right)^2}}{\sum_{i=-n}^{n-1} e^{-\left(\frac{i-\delta}{\sigma}\right)^2}} \tag{A.6}$$

results. The negative portion of equation A.6 can be rewritten so that the summation uses positive indices, resulting in

$$\hat{\mu}' = \delta + \frac{\sum_{i=0}^{n-1}(i-\delta) e^{-\left(\frac{i-\delta}{\sigma}\right)^2} - \sum_{i=1}^{n}(i+\delta) e^{-\left(\frac{i+\delta}{\sigma}\right)^2}}{\sum_{i=-n}^{n-1} e^{-\left(\frac{i-\delta}{\sigma}\right)^2}} \tag{A.7}$$

$$= \delta + \frac{\sum_{i=0}^{n-1}(i-\delta) e^{-\left(\frac{i-\delta}{\sigma}\right)^2} - \sum_{j=0}^{n-1}(j+\delta+1) e^{-\left(\frac{j+\delta+1}{\sigma}\right)^2}}{\sum_{i=-n}^{n-1} e^{-\left(\frac{i-\delta}{\sigma}\right)^2}} \tag{A.8}$$

$$= \delta + \frac{\sum_{i=1}^{n-1}\left((i-\delta) e^{-\left(\frac{i-\delta}{\sigma}\right)^2} - (i+\delta+1) e^{-\left(\frac{i+\delta+1}{\sigma}\right)^2}\right)}{\sum_{i=-n}^{n-1} e^{-\left(\frac{i-\delta}{\sigma}\right)^2}} \tag{A.9}$$

Equation A.9 corresponds to equation 3.26.

Continuing with the example of $-0.5 < \delta < 0$, we now consider the case where $2n + 1$ terms are selected. The derivation is similar, again using equation 3.13 as a starting point:

$$\hat{\mu}' = \frac{\sum_{i=-n}^{n} i e^{-\left(\frac{i-\delta}{\sigma}\right)^2}}{\sum_{i=-n}^{n} e^{-\left(\frac{i-\delta}{\sigma}\right)^2}} \tag{A.10}$$

Making the substitution $j = i - \delta$, or $i = j + \delta$,

$$\hat{\mu}' = \frac{\sum_{j=-n-\delta}^{n-\delta} (j+\delta) e^{-\left(\frac{j}{\sigma}\right)^2}}{\sum_{j=-n-\delta}^{n-\delta} e^{-\left(\frac{j}{\sigma}\right)^2}} \tag{A.11}$$

$$= \delta + \frac{\sum_{j=-n-\delta}^{n-\delta} j e^{-\left(\frac{j}{\sigma}\right)^2}}{\sum_{j=-n-\delta}^{n-\delta} e^{-\left(\frac{j}{\sigma}\right)^2}} \tag{A.12}$$

Replacing j with $i = j - \delta$,

$$\hat{\mu}' = \delta + \frac{\sum_{i=-n}^{n} (i-\delta) e^{-\left(\frac{i-\delta}{\sigma}\right)^2}}{\sum_{i=-n}^{n} e^{-\left(\frac{i-\delta}{\sigma}\right)^2}} \tag{A.13}$$

Splitting the numerator into positive and negative components, and rewriting the summation for the negative portion to use positive indices, we obtain

$$\hat{\mu}' = \delta + \frac{\sum_{i=0}^{n} (i-\delta) e^{-\left(\frac{i-\delta}{\sigma}\right)^2} + \sum_{i=-n}^{-1} (i-\delta) e^{-\left(\frac{i-\delta}{\sigma}\right)^2}}{\sum_{i=-n}^{n} e^{-\left(\frac{i-\delta}{\sigma}\right)^2}} \tag{A.14}$$

$$= \delta + \frac{\sum_{i=0}^{n}(i-\delta)e^{-\left(\frac{i-\delta}{\sigma}\right)^2} - \sum_{i=1}^{n}(i+\delta)e^{-\left(\frac{i+\delta}{\sigma}\right)^2}}{\sum_{i=-n}^{n} e^{-\left(\frac{i-\delta}{\sigma}\right)^2}} \tag{A.15}$$

$$= \delta + \frac{\sum_{i=0}^{n}(i-\delta)e^{-\left(\frac{i-\delta}{\sigma}\right)^2} - \sum_{j=0}^{n-1}(j+\delta+1)e^{-\left(\frac{j+\delta+1}{\sigma}\right)^2}}{\sum_{i=-n}^{n} e^{-\left(\frac{i-\delta}{\sigma}\right)^2}} \tag{A.16}$$

$$= \delta + \frac{(n-\delta)e^{-\left(\frac{n-\delta}{\sigma}\right)^2} + \sum_{i=1}^{n-1}\left((i-\delta)e^{-\left(\frac{i-\delta}{\sigma}\right)^2} - (i+\delta+1)e^{-\left(\frac{i+\delta+1}{\sigma}\right)^2}\right)}{\sum_{i=-n}^{n} e^{-\left(\frac{i-\delta}{\sigma}\right)^2}} \tag{A.17}$$

Equation A.17 also corresponds to equation 3.26. The case for $0 < \delta < 0.5$ is evaluated similarly.

VITA

Surname:	Seelemann	Given Names:	Sven Alexander
Place of Birth:	Toronto, Ontario	Date of Birth:	15 June 1966

Educational Institutions Attended:

University of Victoria	1990 to 1993
University of Waterloo	1985 to 1990

Degrees Awarded:

B.A.Sc. (Honours)	University of Waterloo	1990

Honours and Awards:

JOBTRAC/SUPPLEMENT AWARD	1990
University of Waterloo Entrance Scholarship	1985

Publications:

Sven A. Seelemann and Gerard F. McLean. *Sub–pixel Detection of Active Targets.* IEEE Pacific Rim Conference on Computers, Communications, and Signal Processing. May 18–19, 1993 Victoria BC.

PARTIAL COPYRIGHT LICENSE

Title of Thesis: Capture and Location of Blobs in Digital Images.

Author [redacted]

Sven Alexander Seelemann

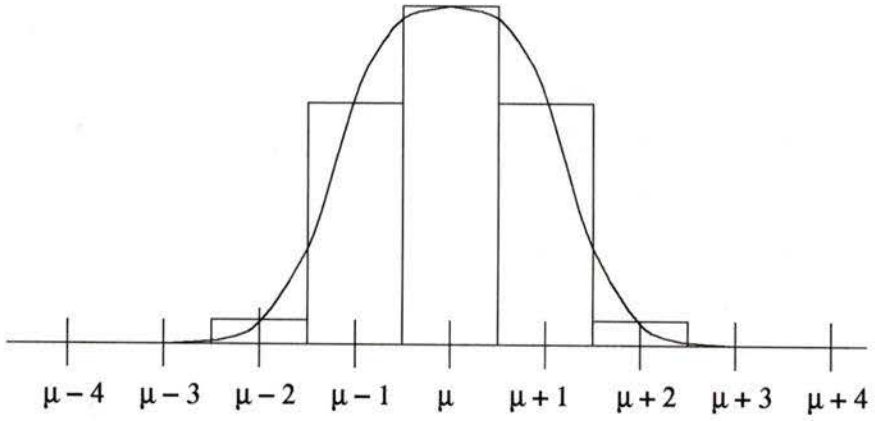
29 April 1993

Large Versions of Figures for

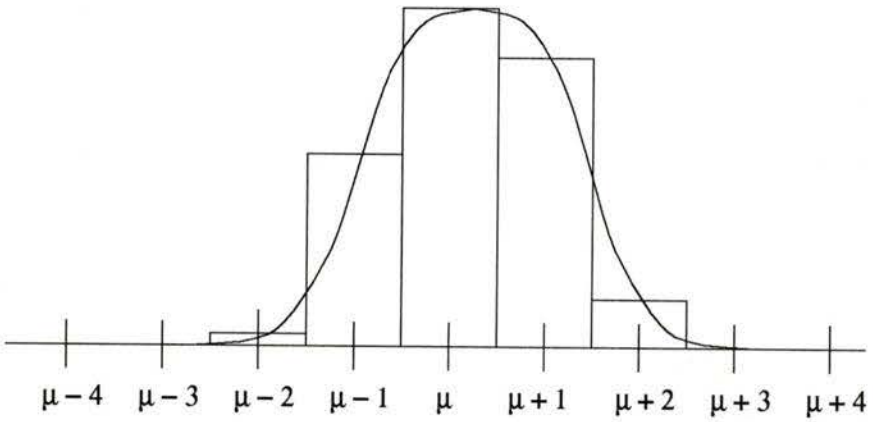
Capture and Location of Blobs in Digital Images

by

Sven Alexander Seelemann



(a) Pulse Mean Aligned with Sampling Interval



(b) Pulse Mean Shifted a Sub-Pixel Amount w.r.t. Sampling Interval

Figure 2.2: Sampling Process Diagram

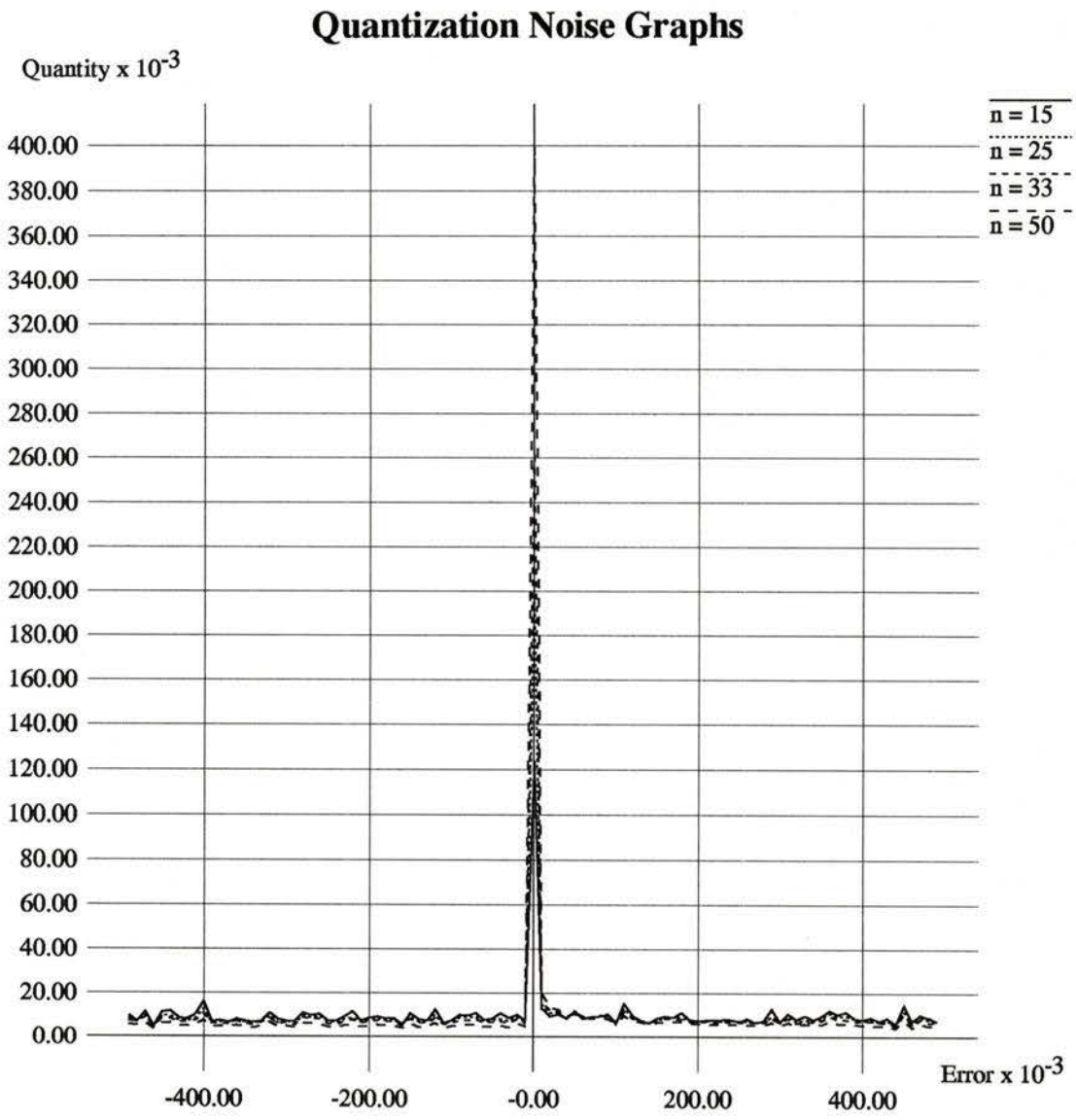


Figure 2.3: Quantization Noise Function

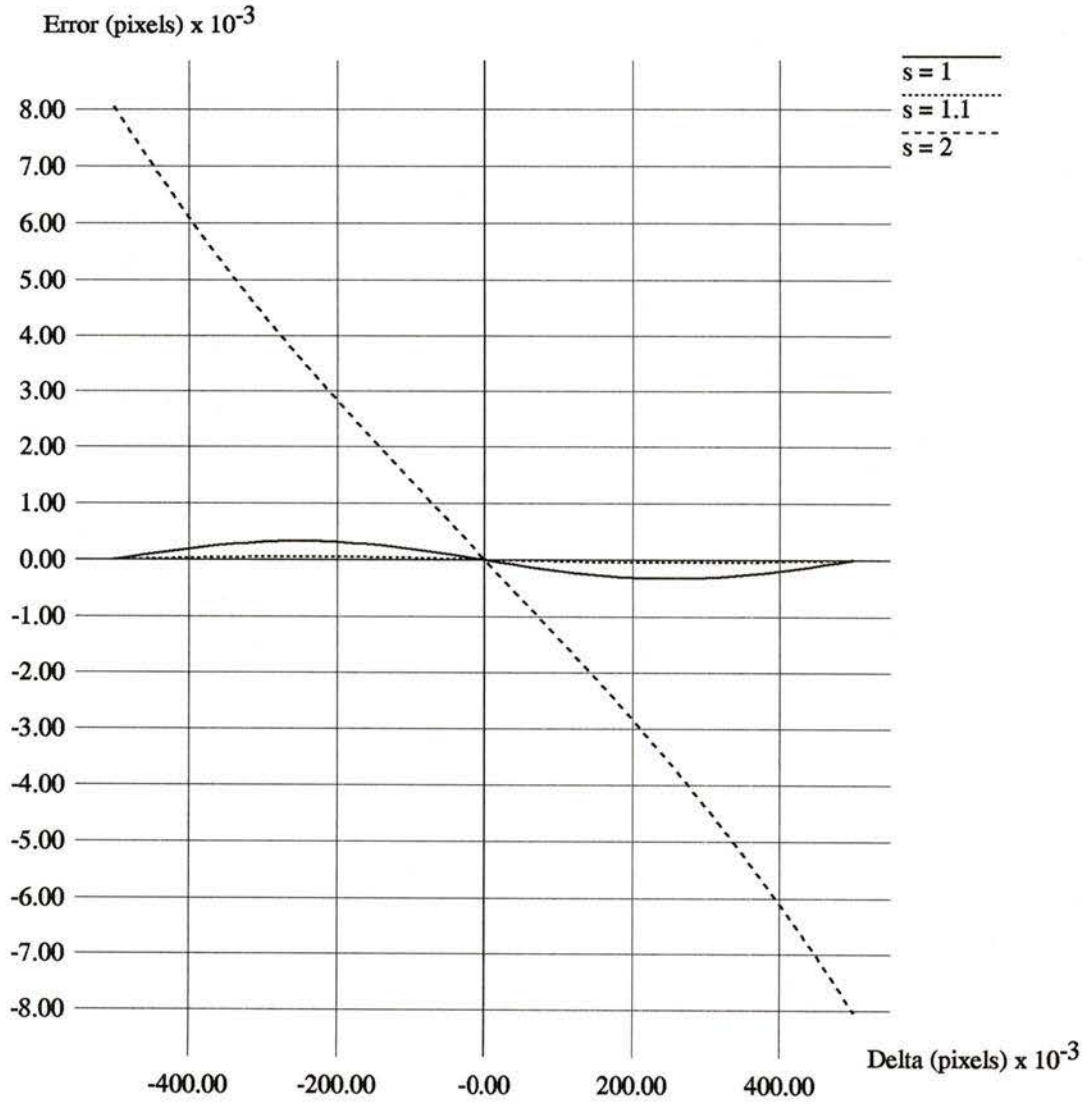


Figure 3.1: Skew Error for Constant Standard Deviation s and Window Size 9 pixels

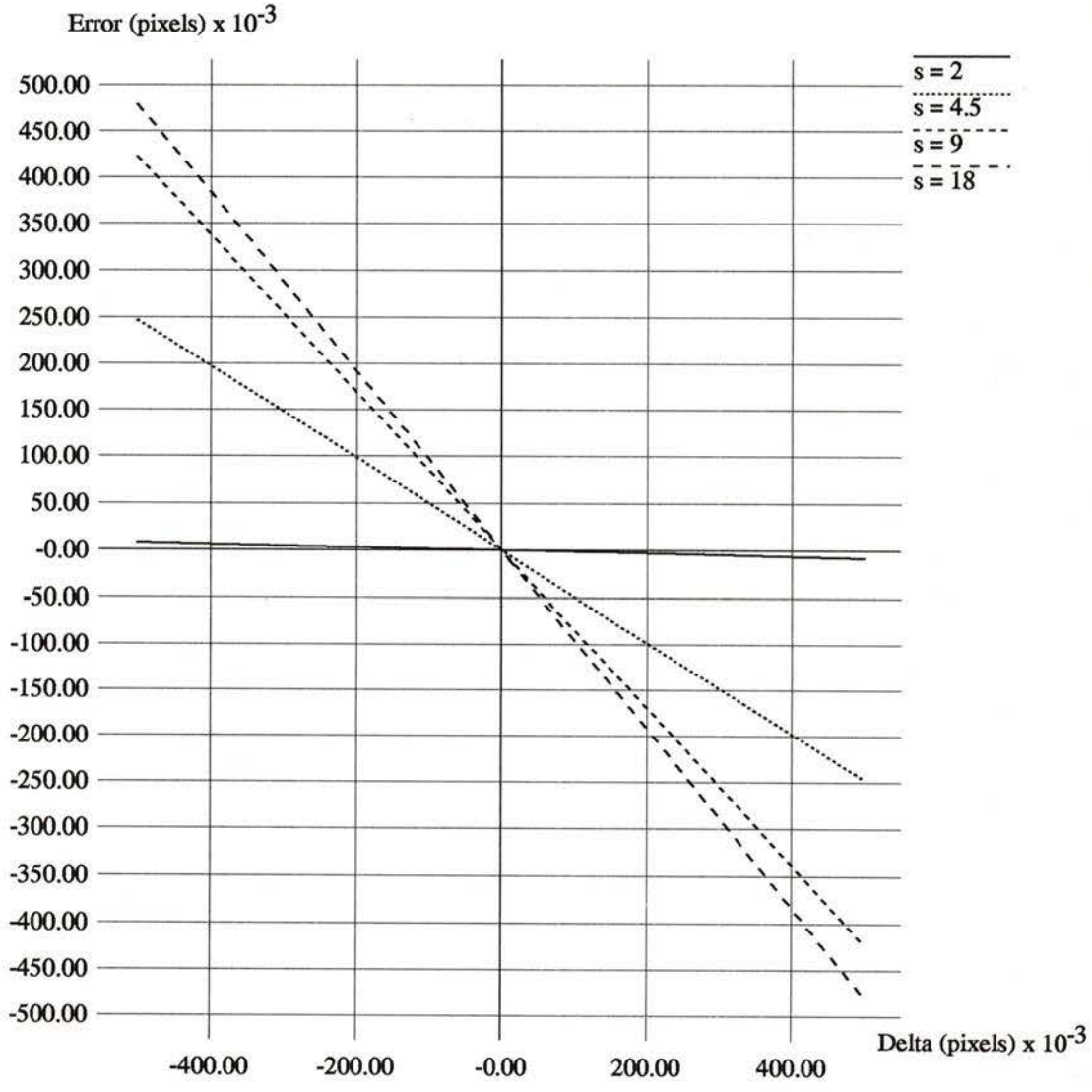


Figure 3.2: Skew Error for Constant Standard Deviation s and Window Size 9 pixels (Part 2)

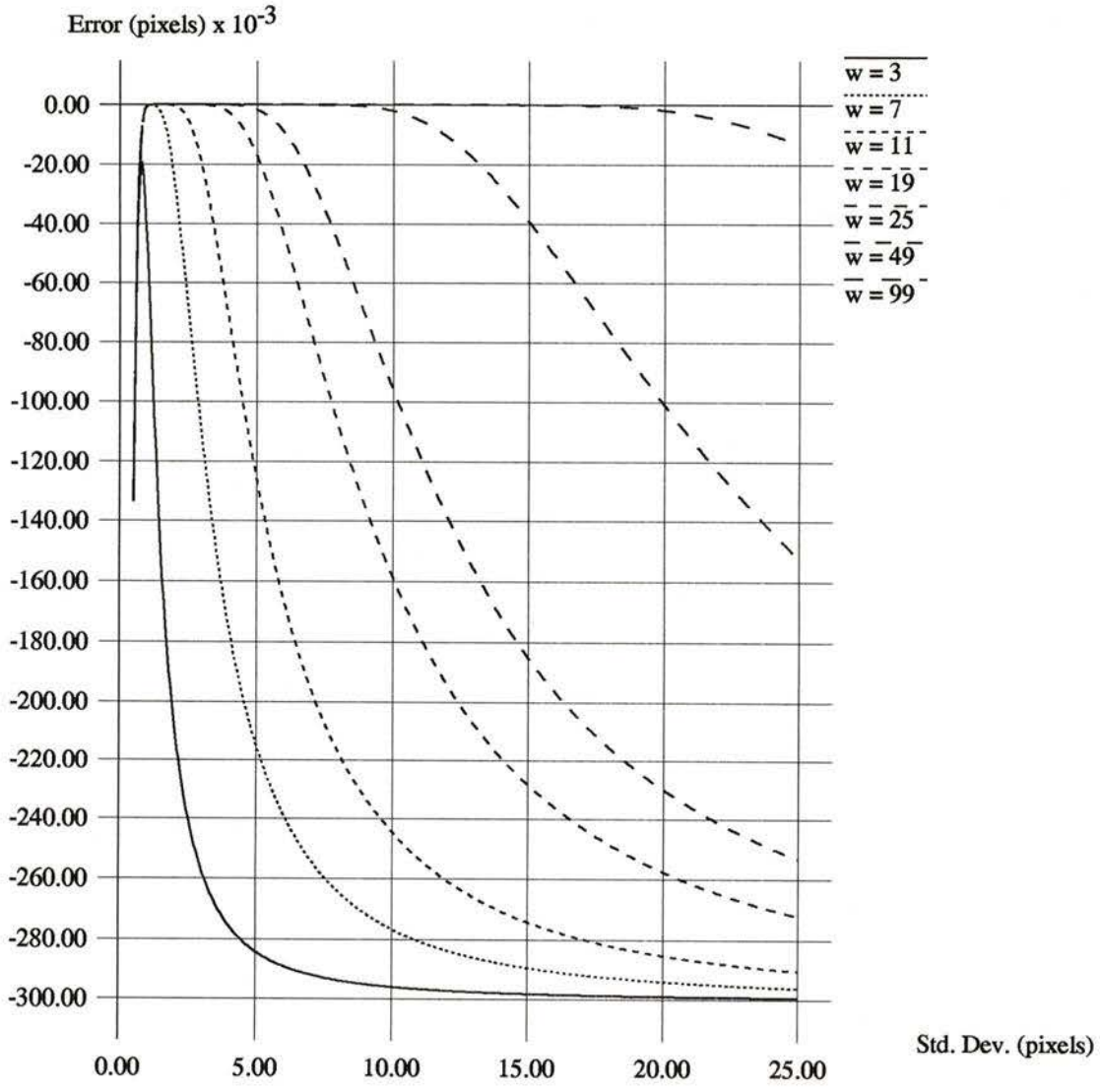


Figure 3.3: Skew Error for Constant Displacement $\delta = 0.3$ pixels and Window size w with Variable Variance

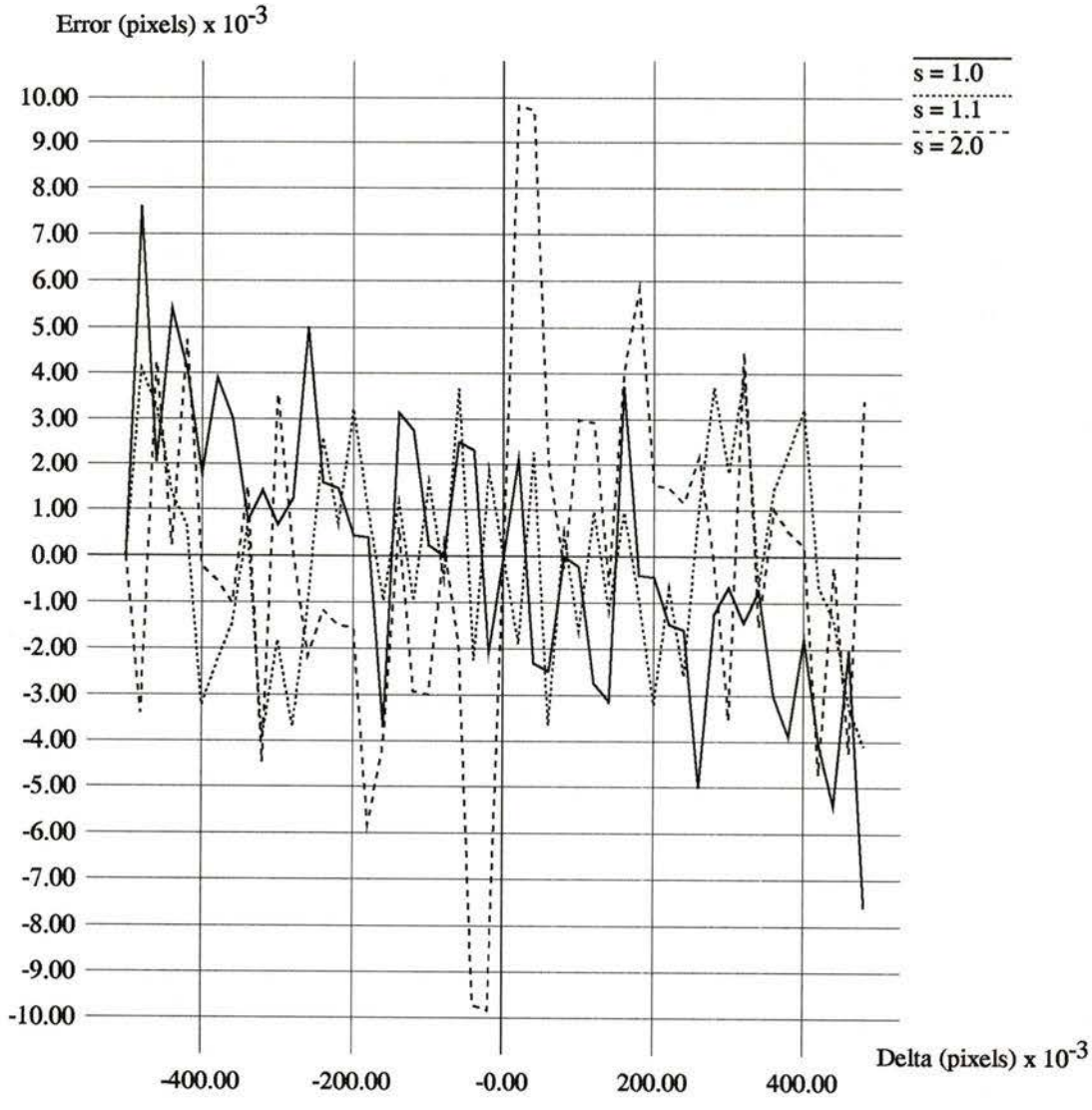


Figure 3.4: 1–D Centroid Location Errors for Quantized Data with Constant Variance s and Window Size 25 pixels

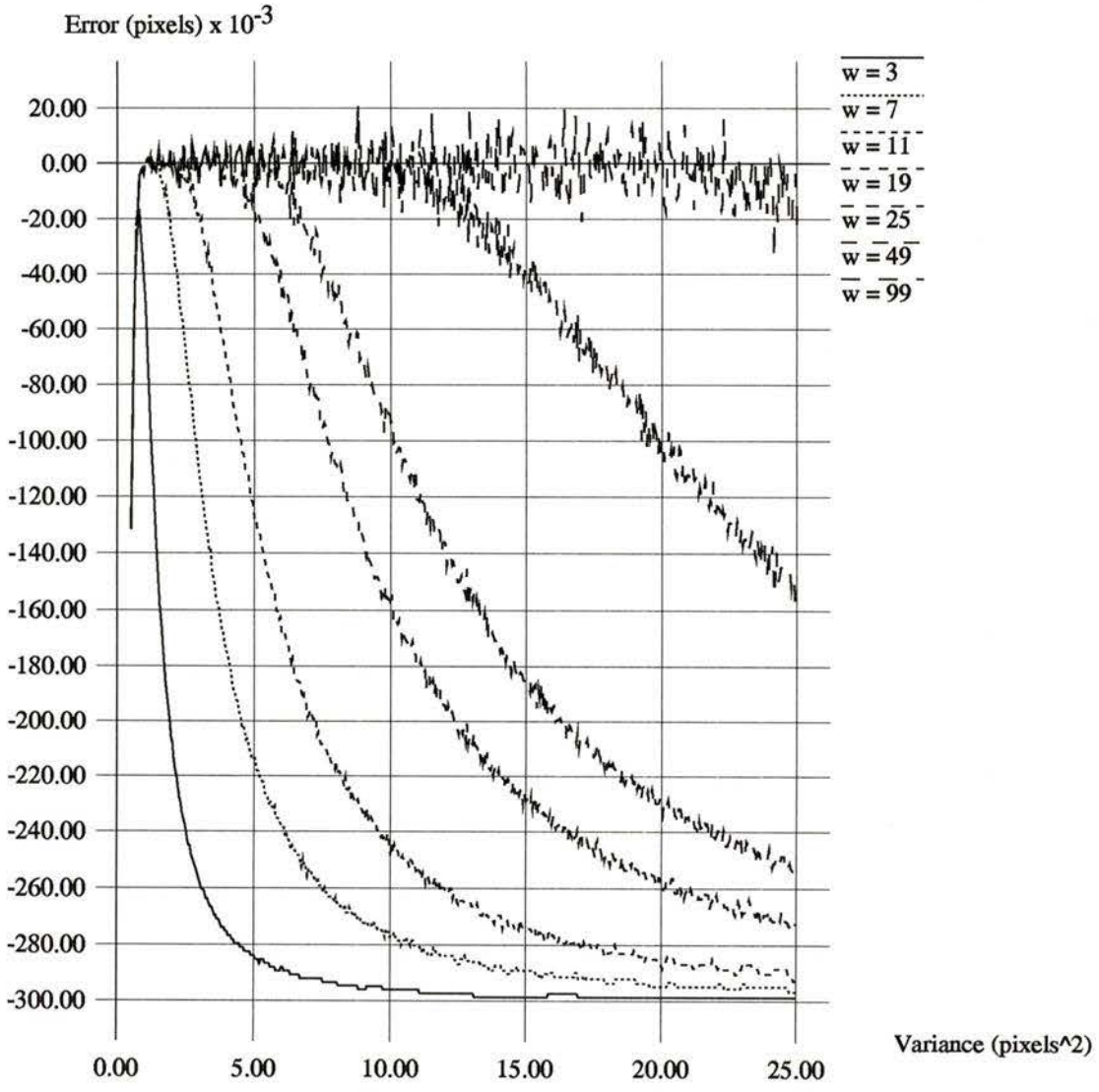


Figure 3.5: 1–D Centroid Location Errors for Quantized Data with Constant Displacement $\delta = 0.3$ pixels and Variable Window Size w

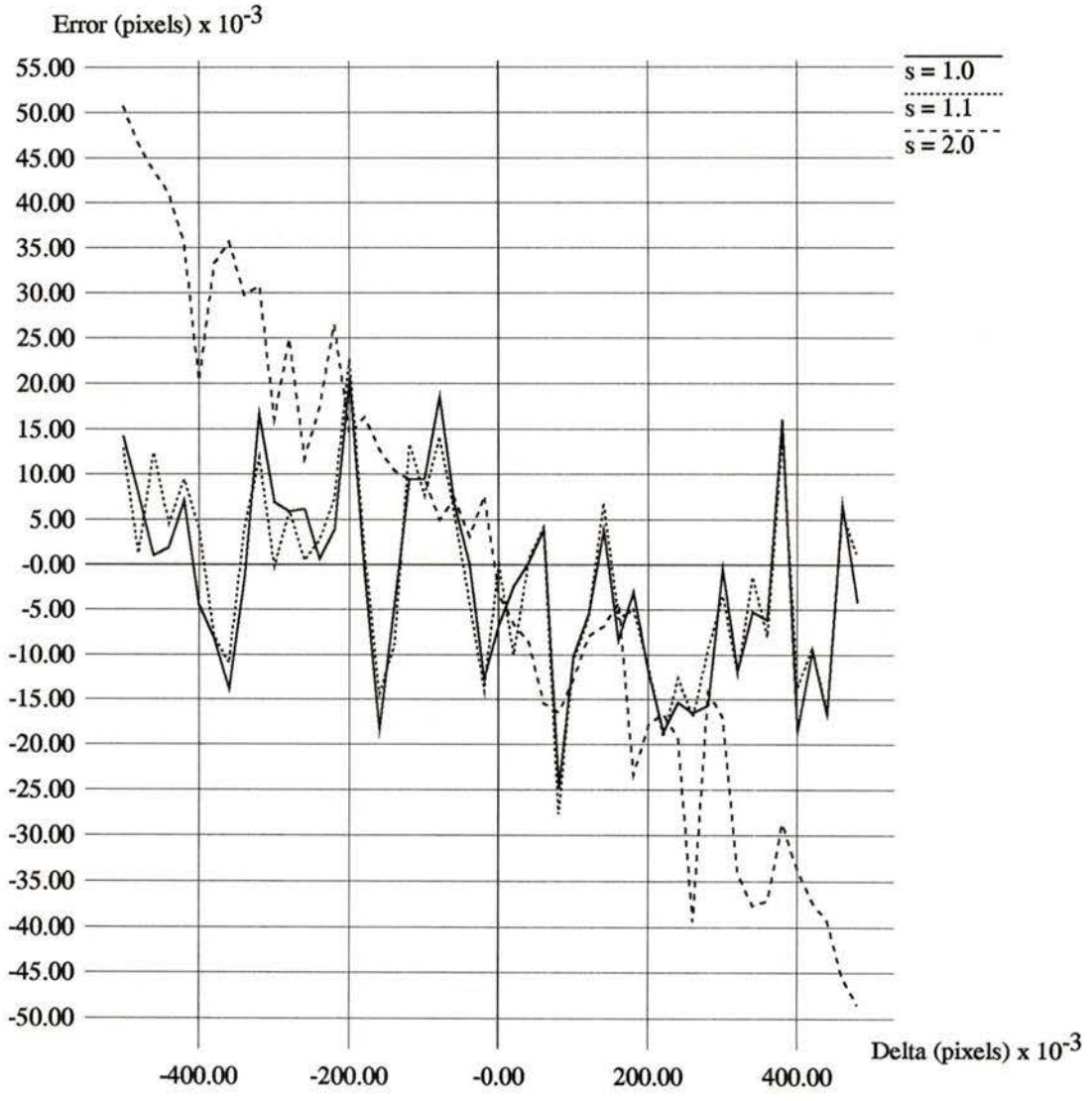


Figure 3.6: 1D Location Errors for Noisy, Quantized Data for Constant Standard Deviation s with Window Size 7 Pixels

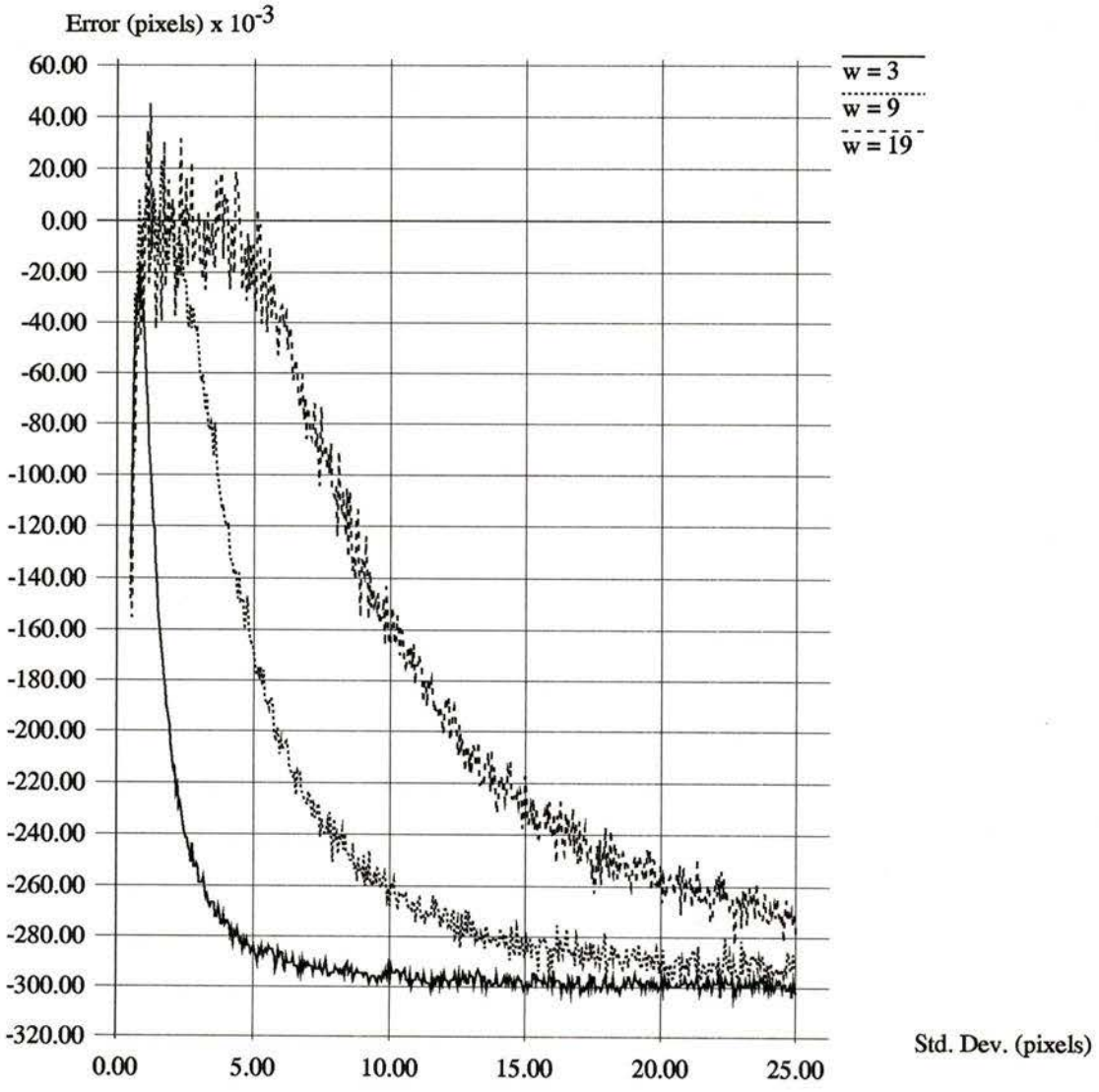


Figure 3.7: 1D Location Errors for Noisy, Quantized Data for Constant Displacement $\delta = 0.3$ with Window Size w

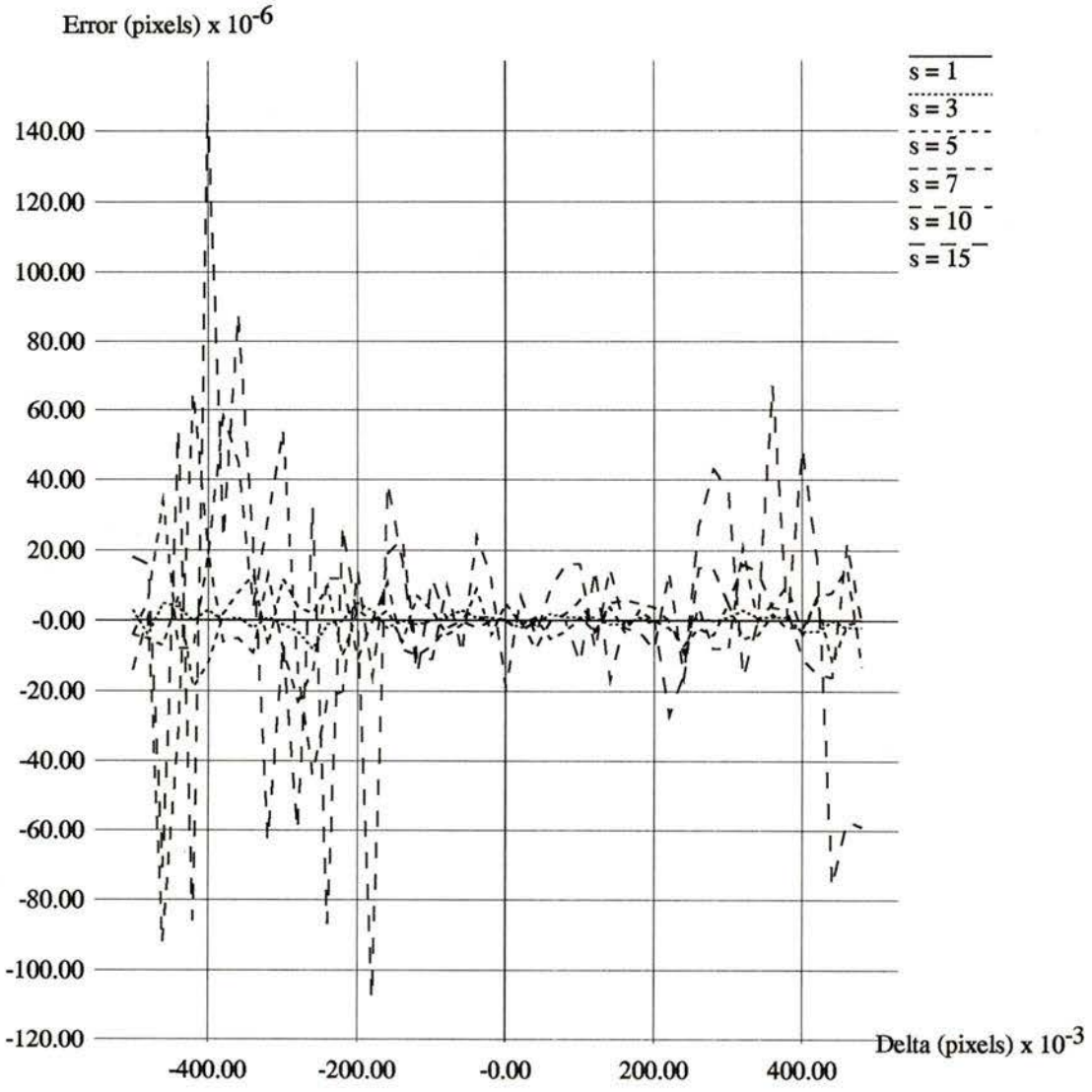


Figure 4.1: Error Location Single Pulse Spatially Sampled Data Constant Standard Deviation s with Window Size 5 pixels

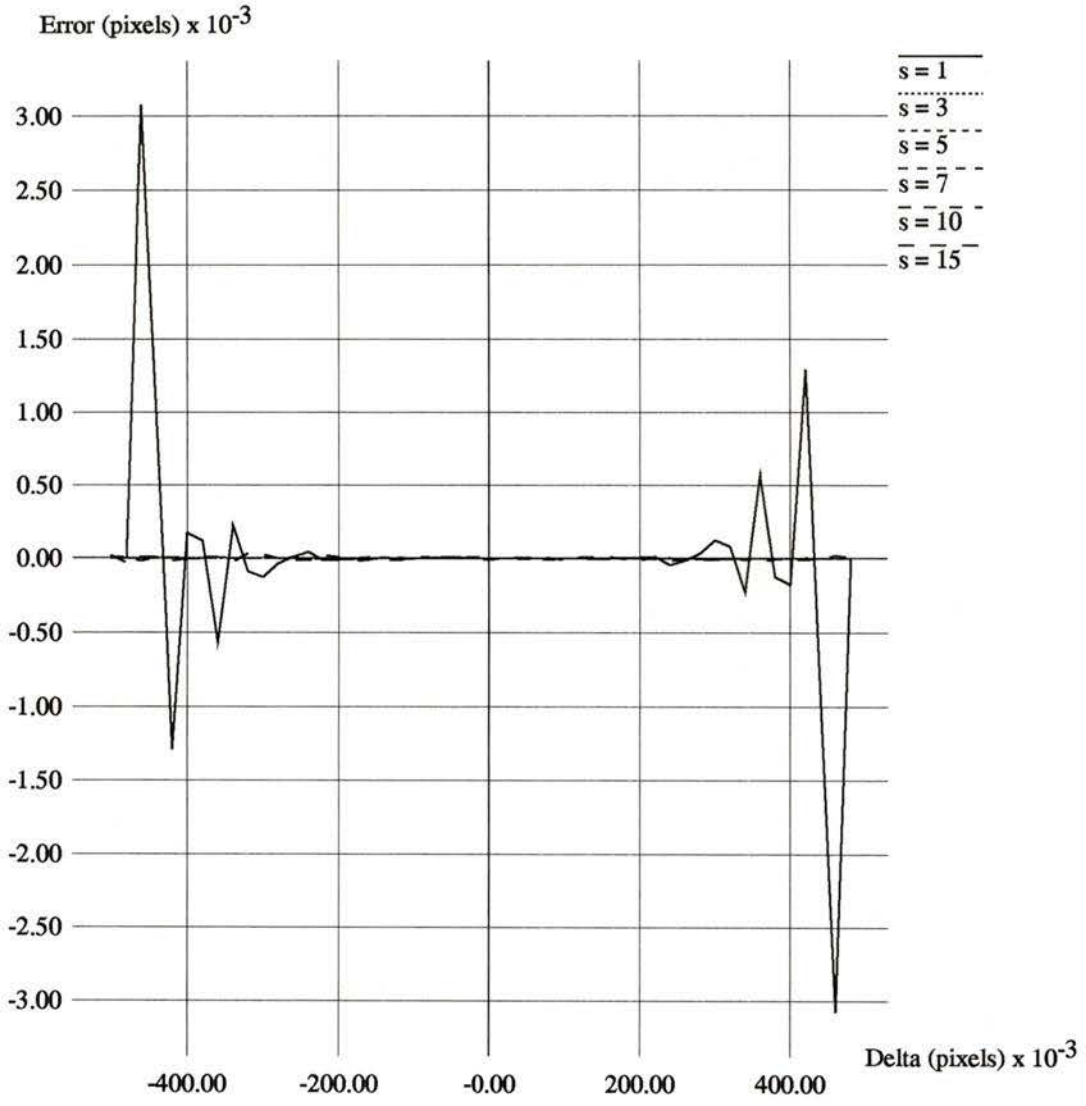


Figure 4.2: Error Location Single Pulse Spatially Sampled Data Constant Standard Deviation s with Window Size 25

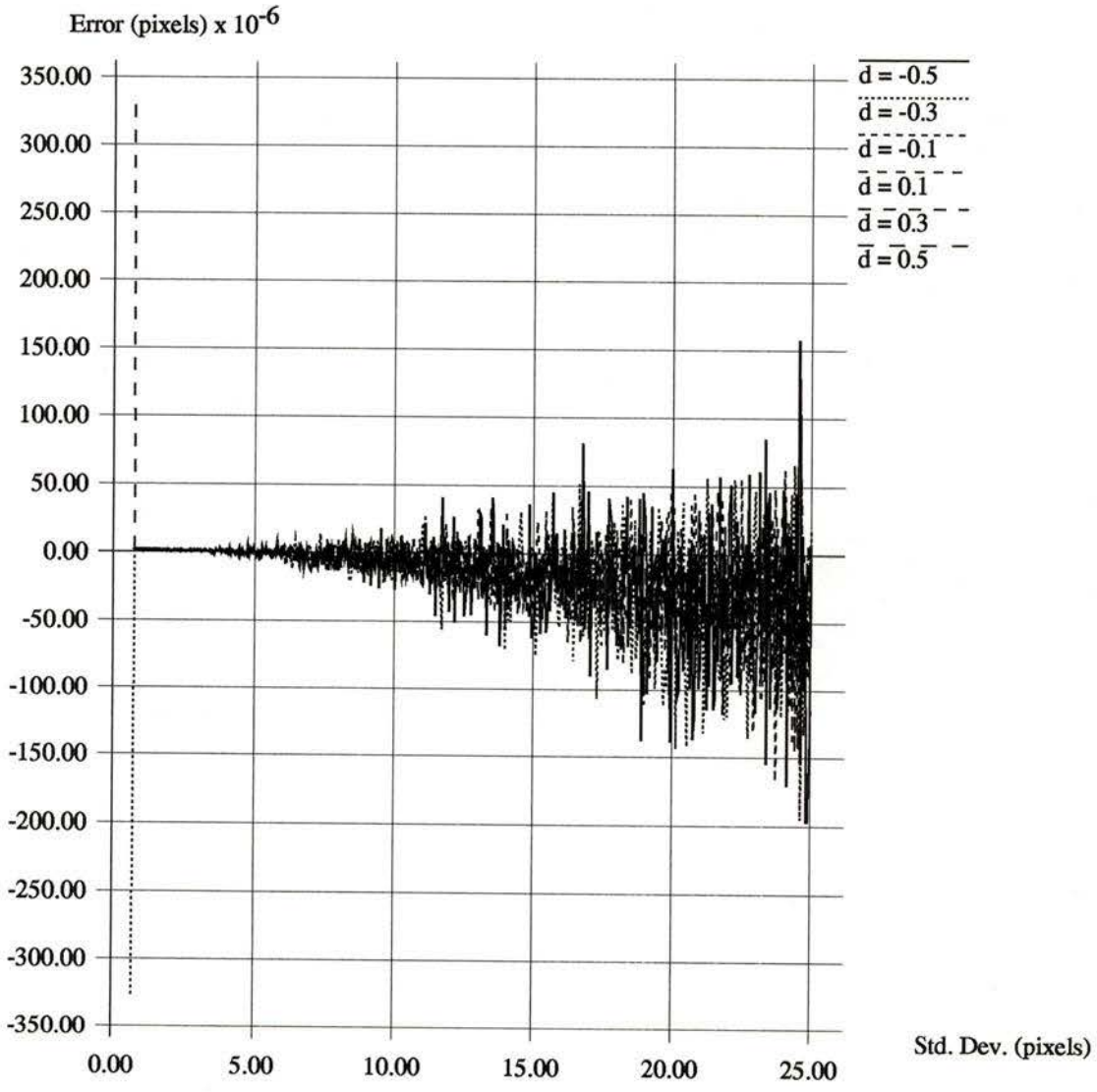


Figure 4.3: Error Locating Single Pulse Spatially Sampled Data with Constant Displacement d and Window Size 15 with Variable Standard Deviation

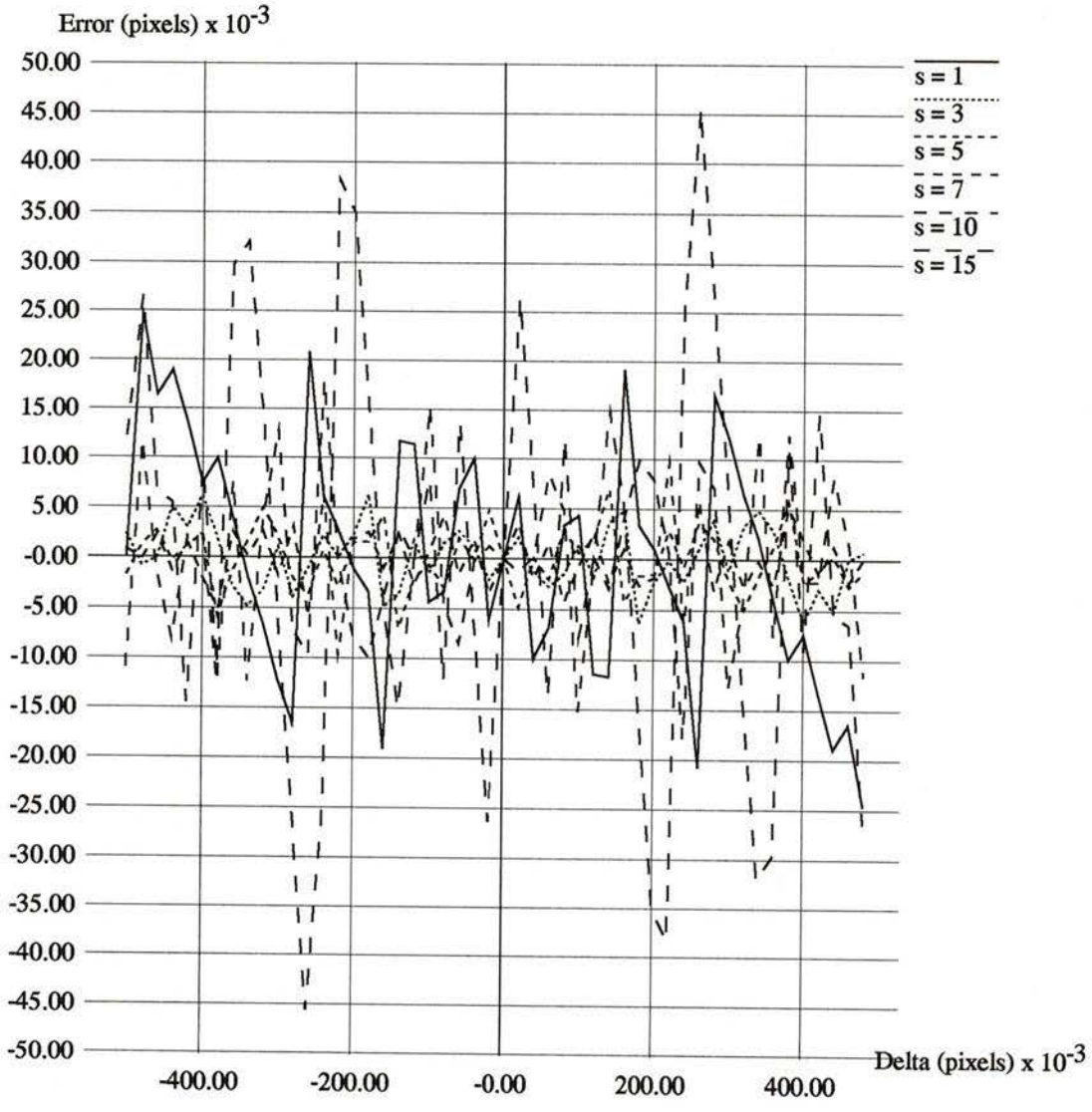


Figure 4.4: 1–D Location Error for Quantized Data, Window Size 9 pixels, and Constant Standard Deviations

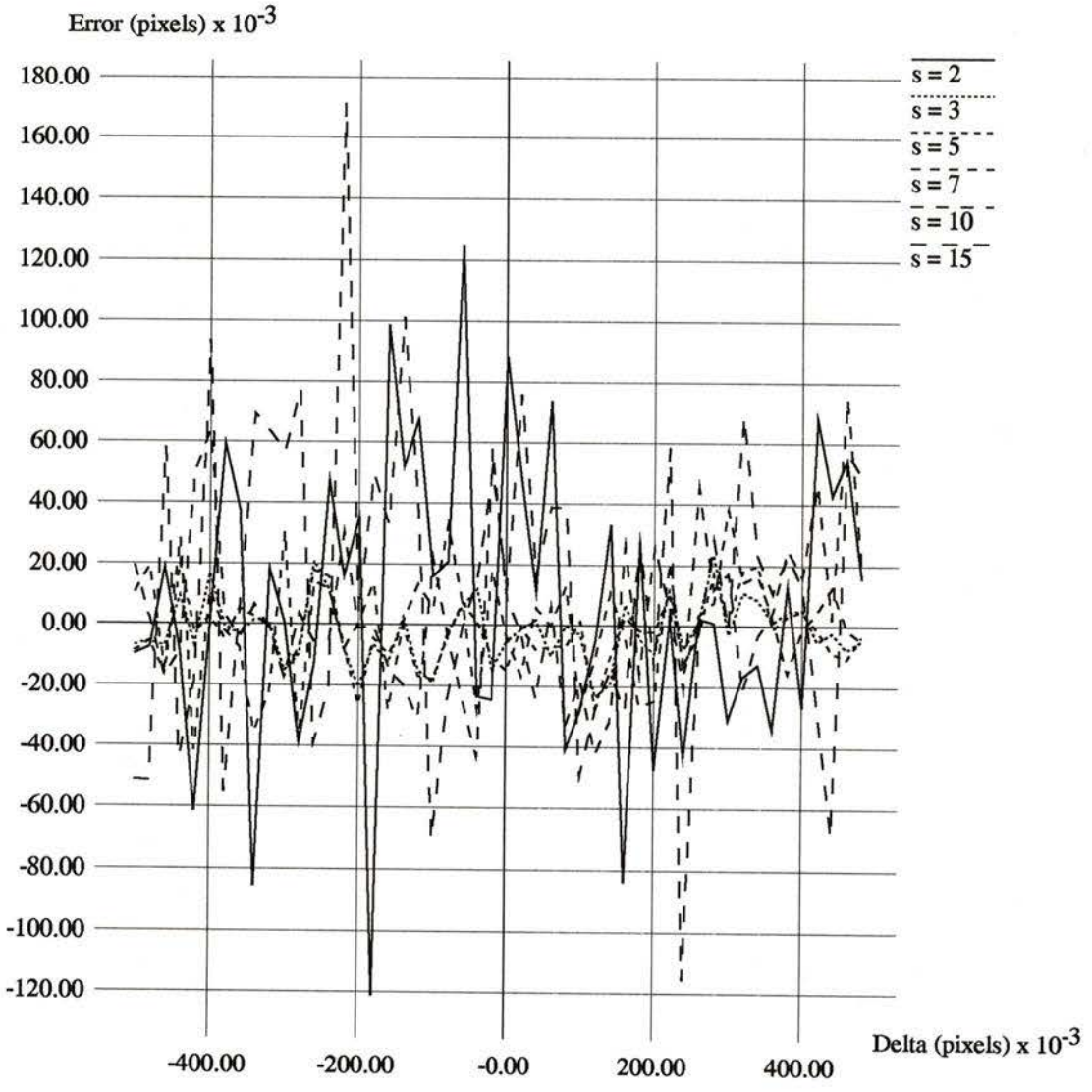


Figure 4.5: 1D Location Error for Quantized Data with Noise Added and Constant Standard Deviation s and Window Size 9 pixels

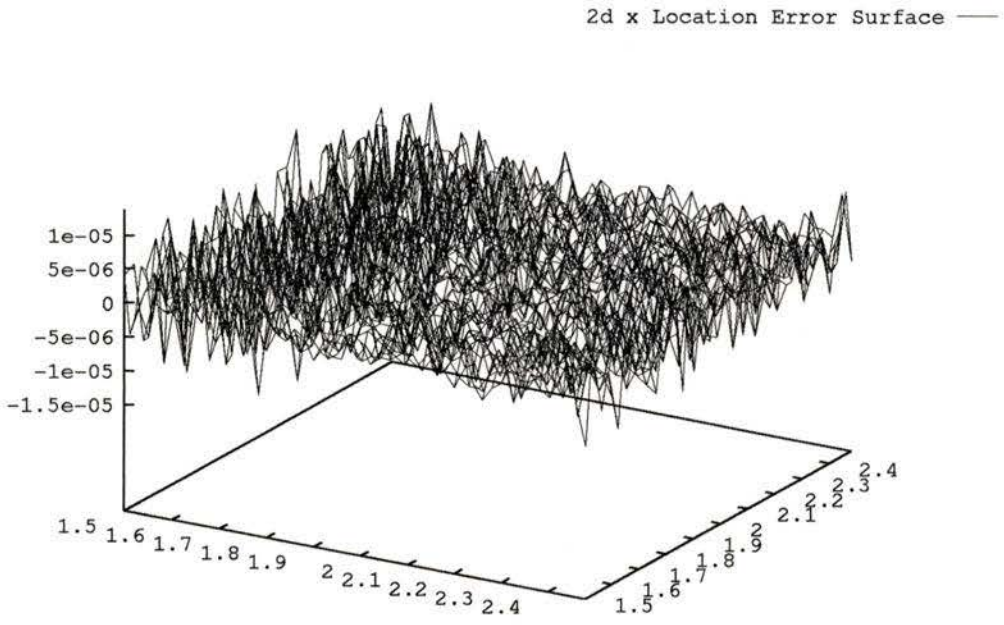


Figure 4.6: 2D Location Error for Spatially Sampled Data with Window Width 5 pixels, Standard Deviation of 1 pixel

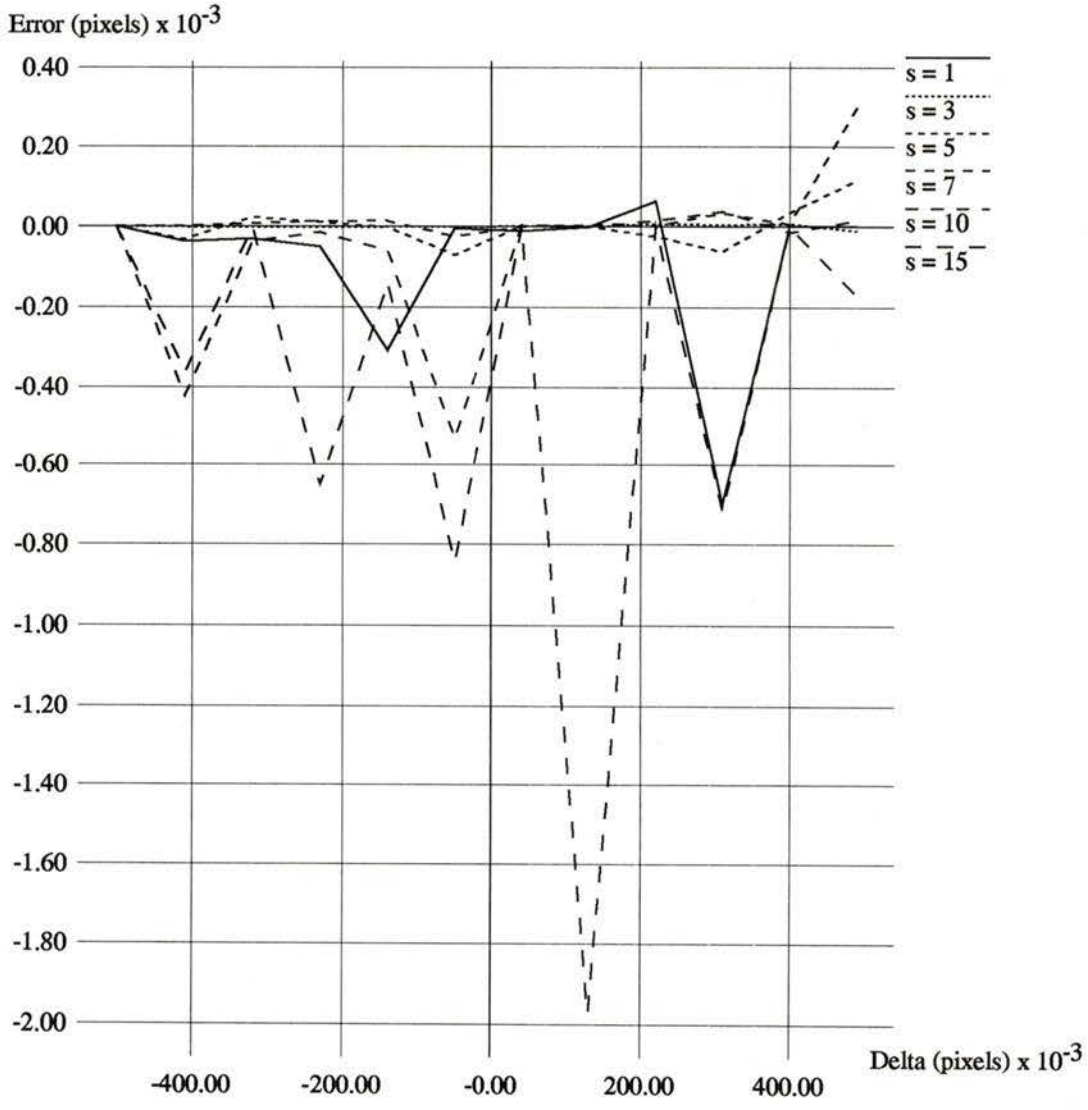


Figure 4.7: 1–D Multi Pulse Location Error for Constant Standard Deviation s and Window Size 9 pixels

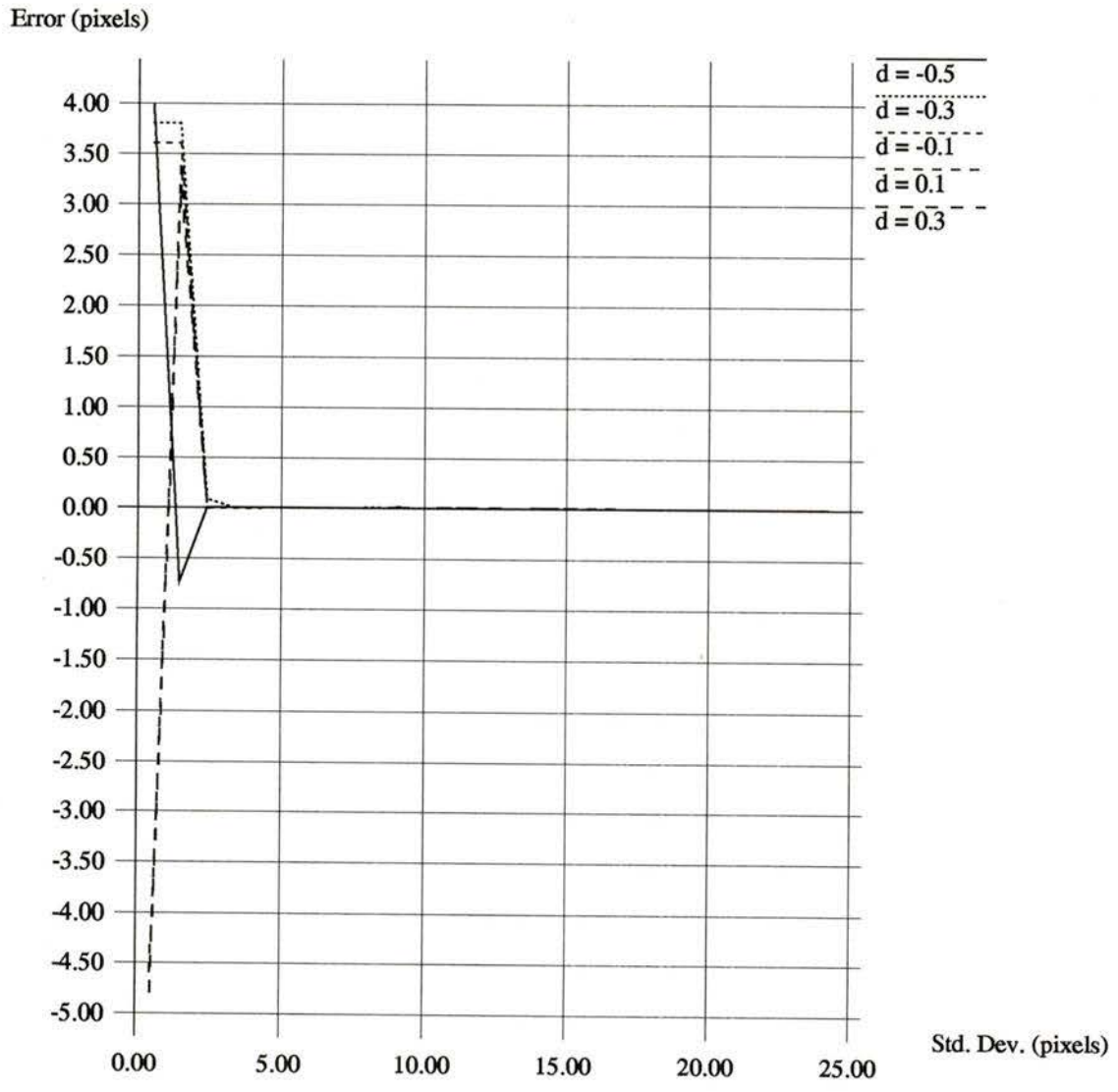


Figure 4.8: 1–D Multi Pulse Location Error for Constant Displacement d and Window Size 9 pixels

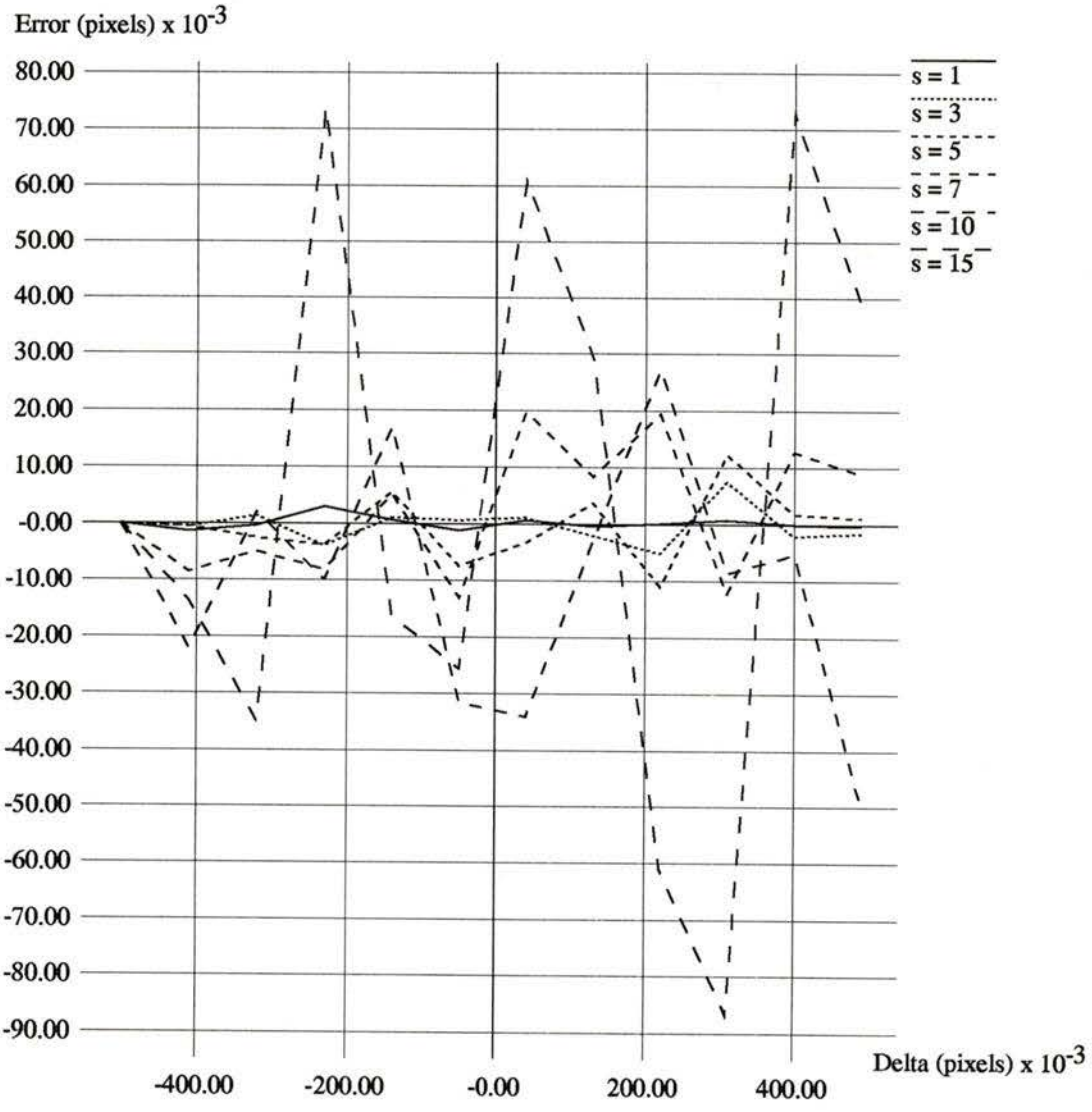


Figure 4.9: 1–D Multi Pulse Location Error for Quantized Data with Constant Standard Deviation s with Window Size 7 pixels

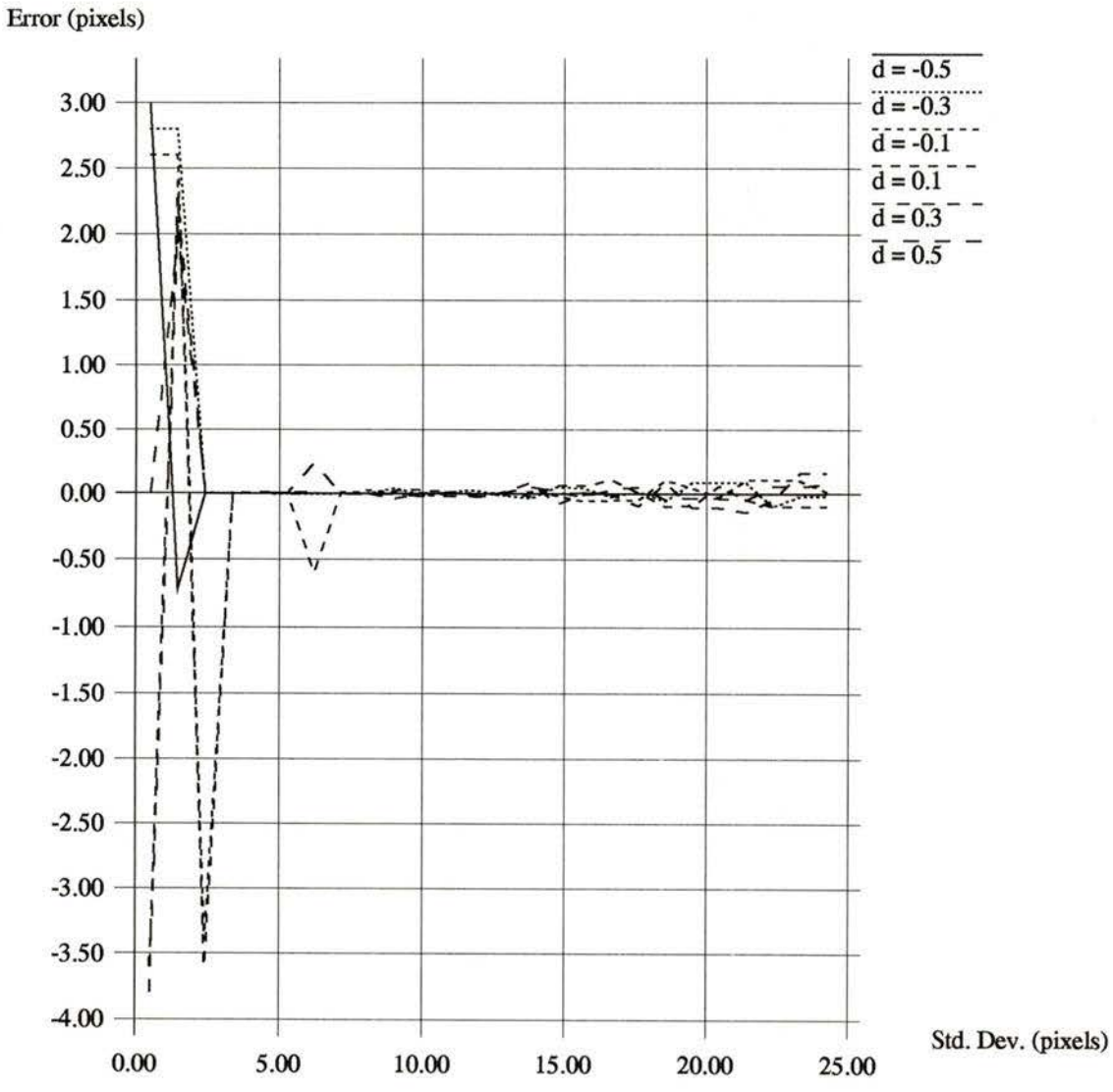


Figure 4.10: 1–D Multi Pulse Location Errors with Constant Displacement d and Window Size 7 pixels using Quantized Data

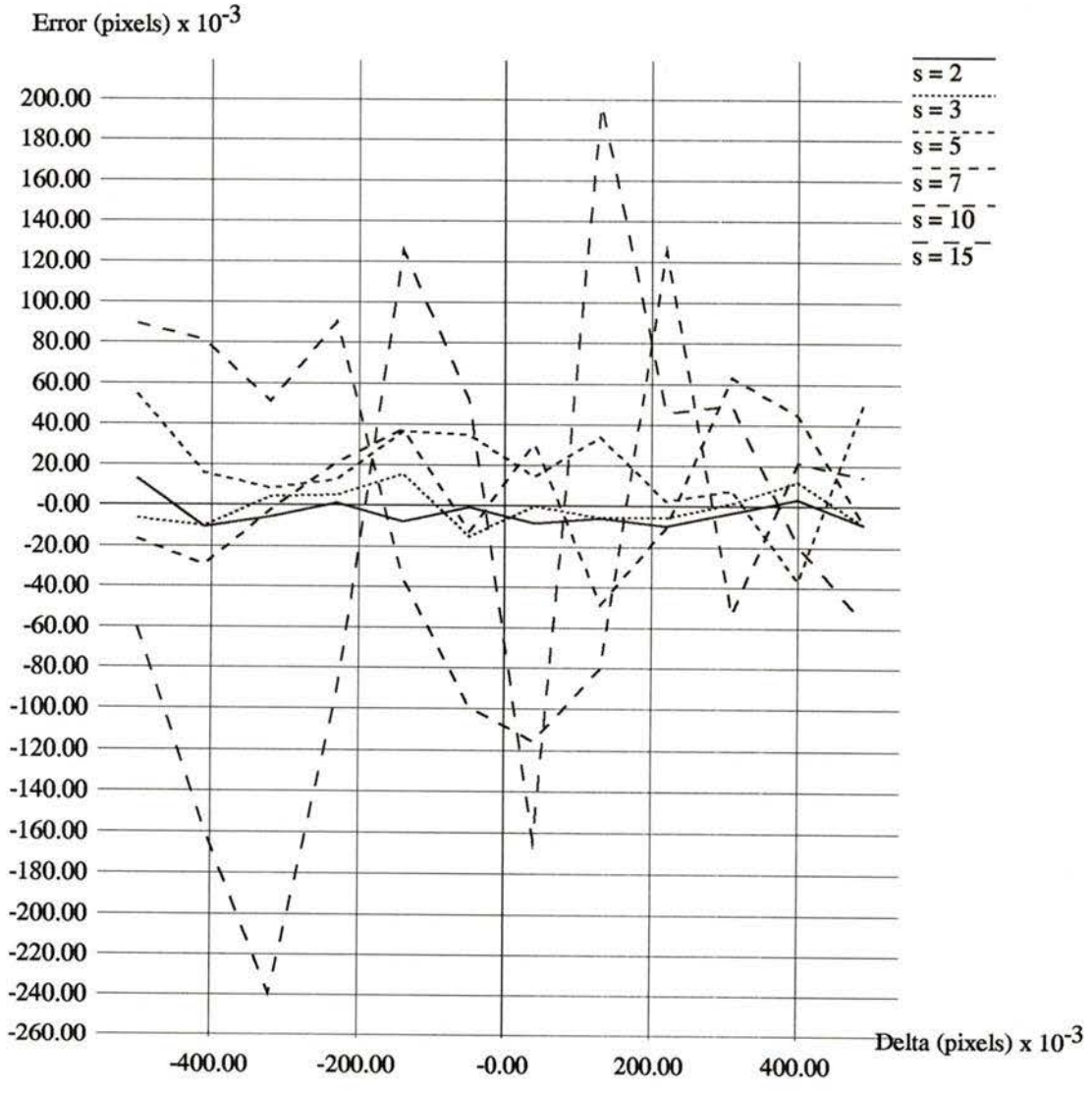


Figure 4.11: 1–D Multi Pulse Location Errors with Constant Standard Deviation s and Window Size 7 pixels using Noisy, Quantized Data

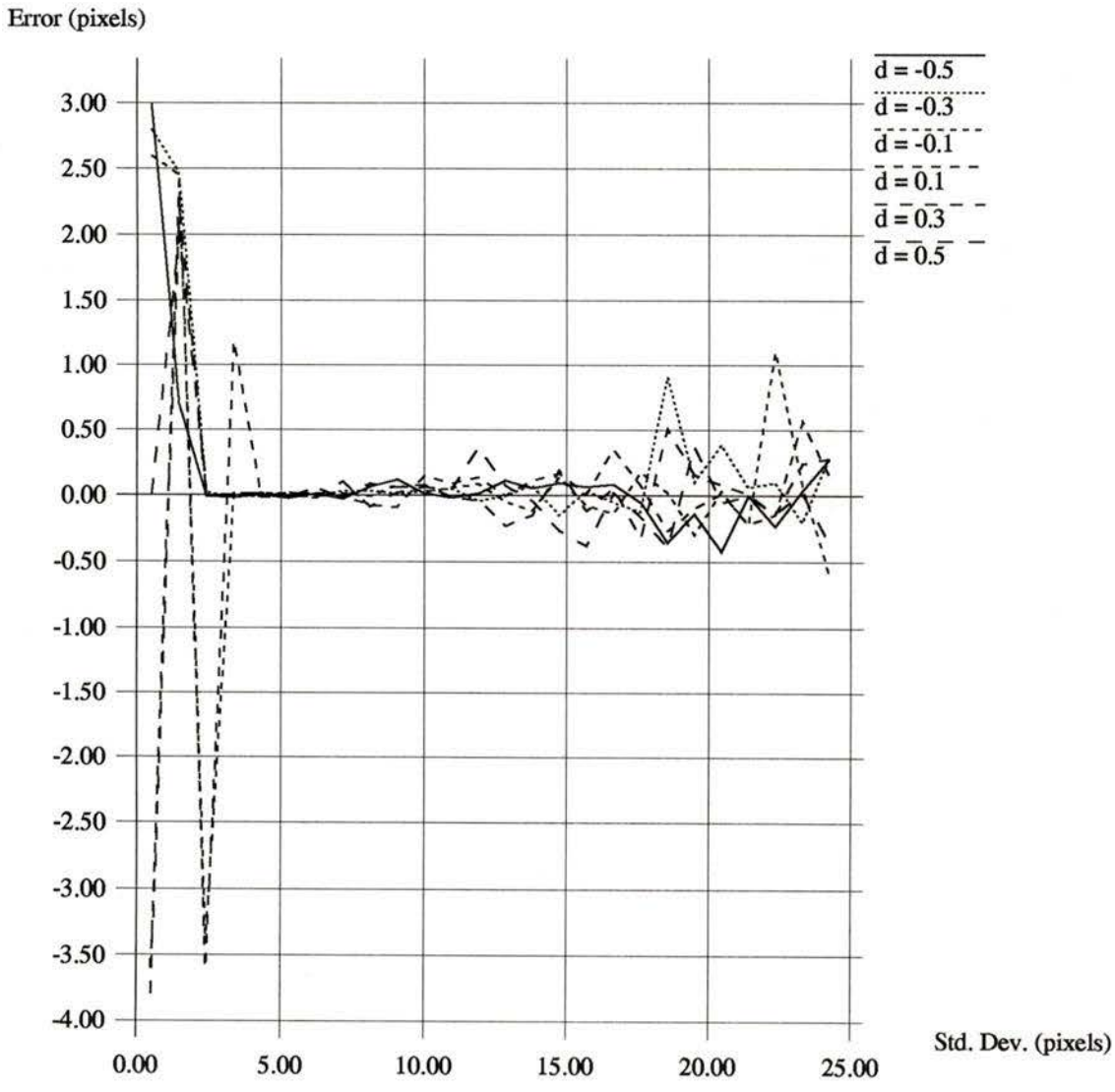


Figure 4.12: 1–D Multi Pulse Location Errors with Constant Displacement d and Window Size 7 pixels using Noisy, Quantized Data

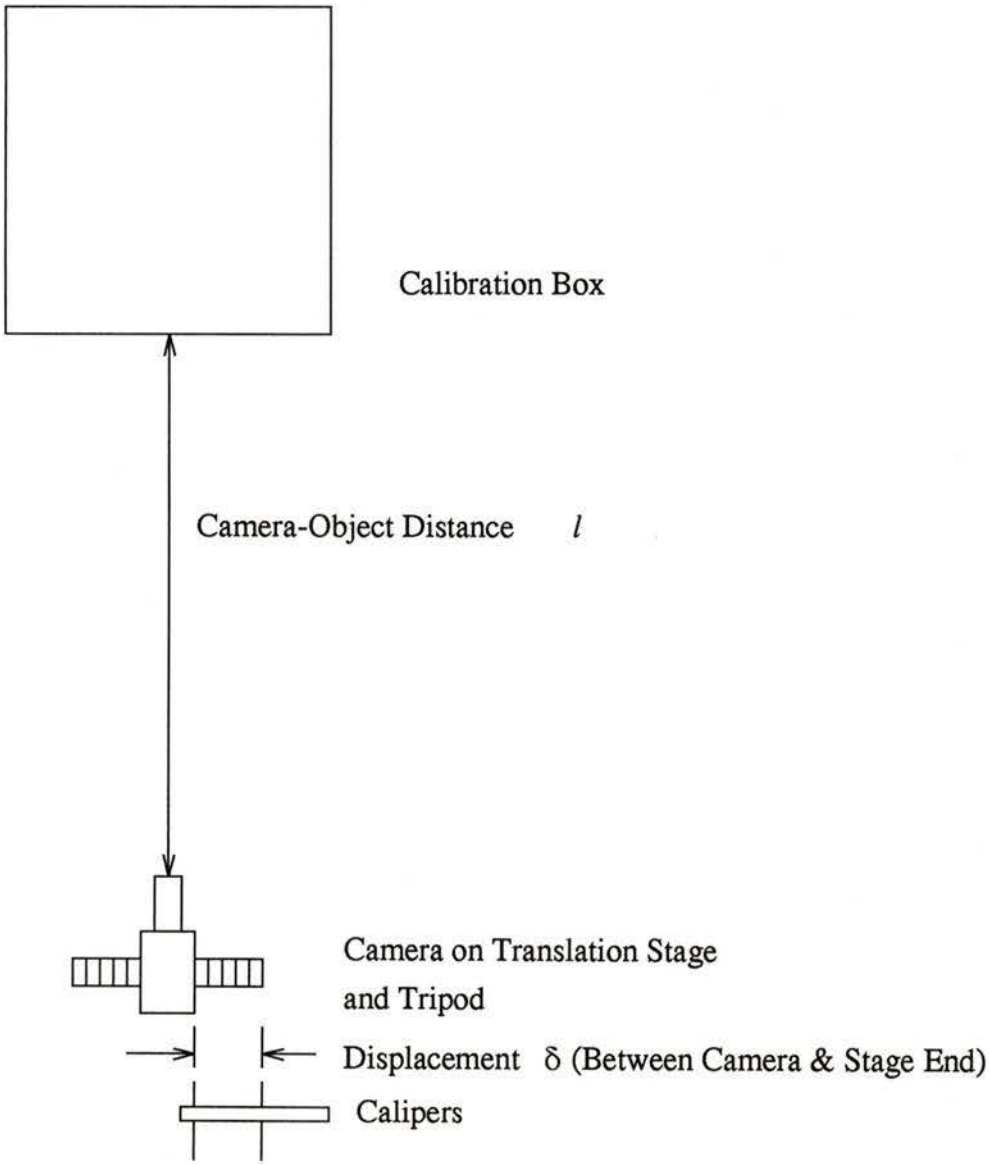


Figure 5.2: Schematic Layout of Experimental Apparatus

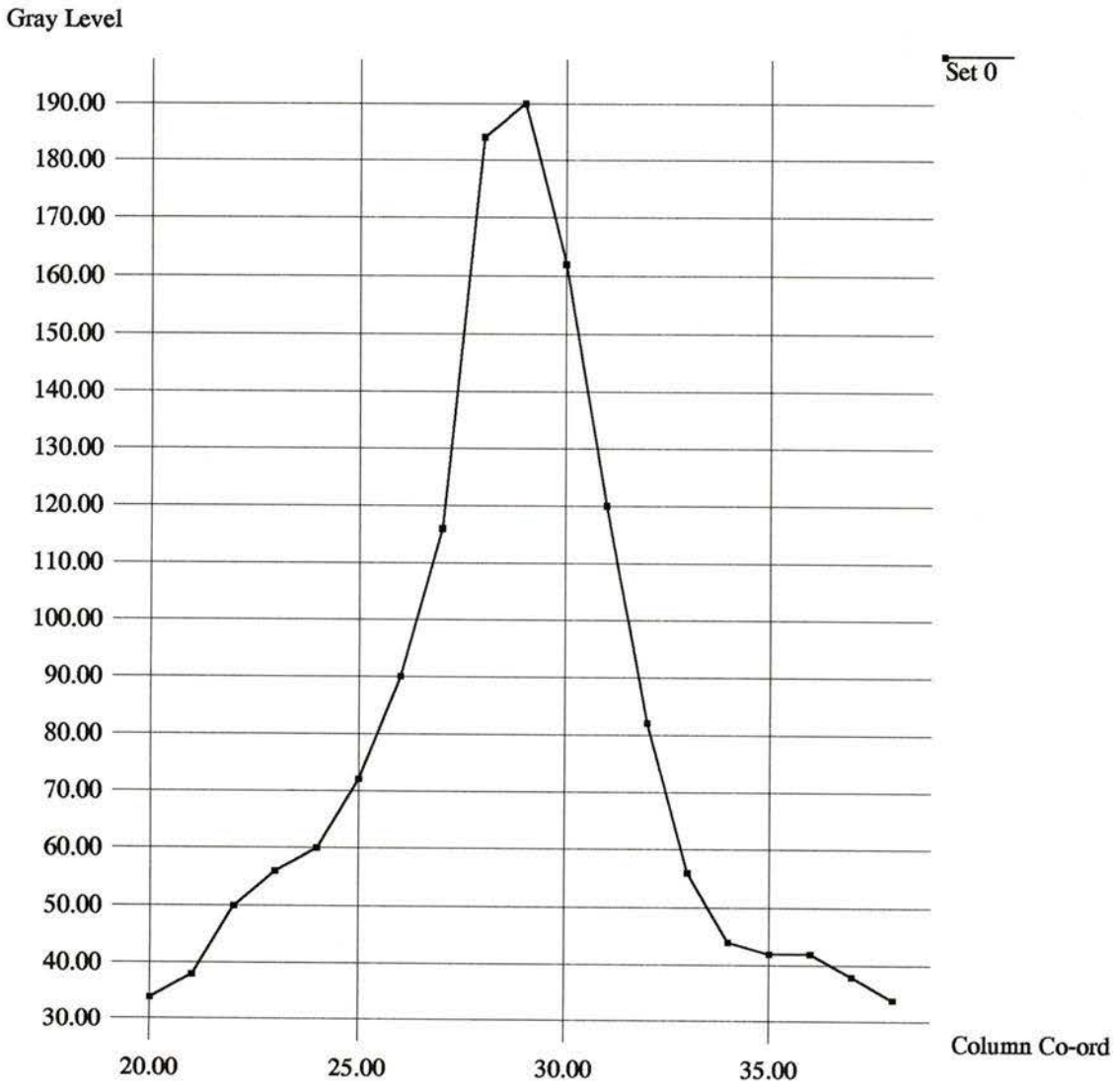


Figure 5.3: Typical Image Row with Rapid Intensity Decay on Right Side

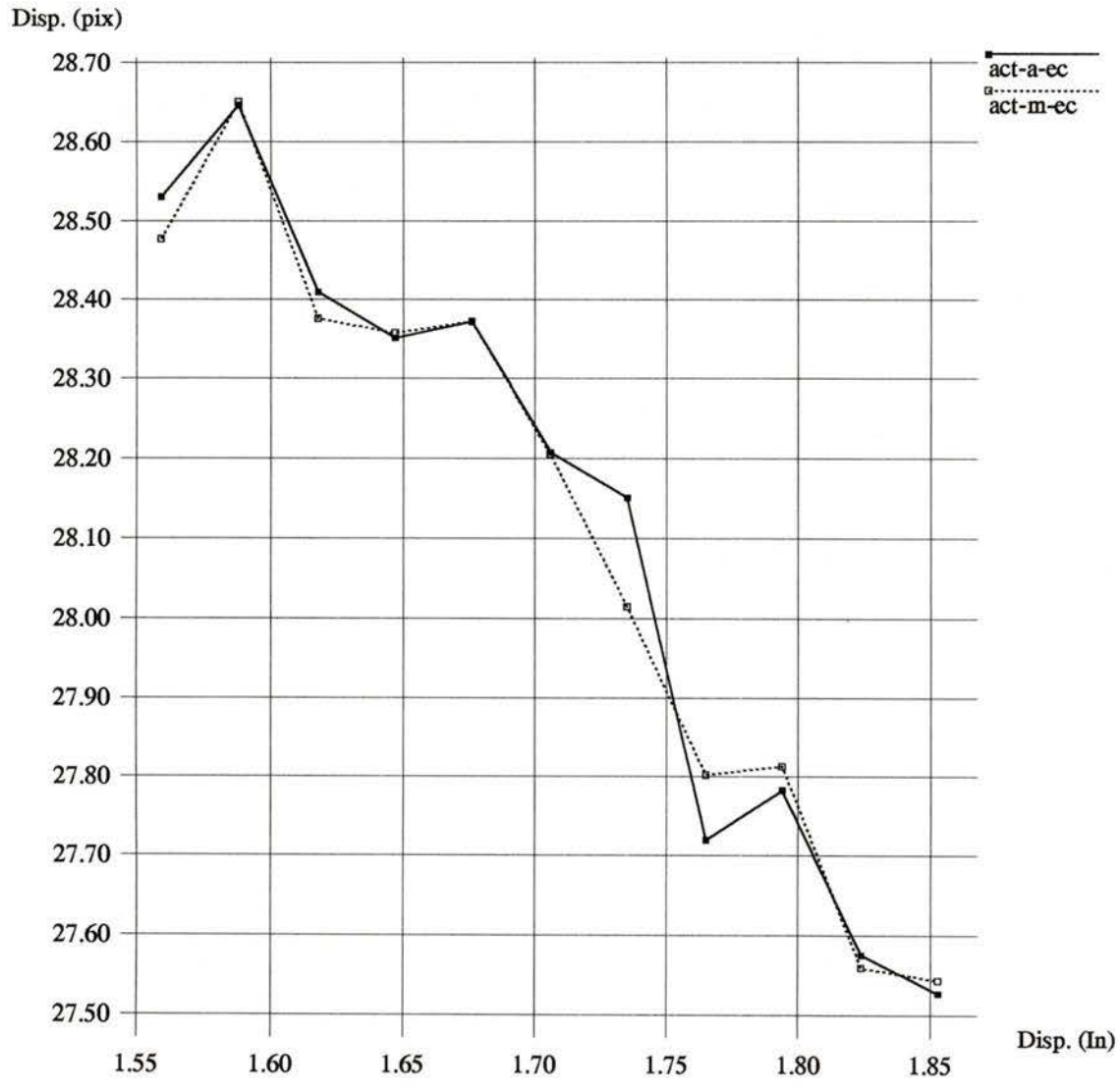


Figure 5.5: Centroid Results: 0.10 Pixel Expected Displacements for the Active Data. Single look is denoted by 'act-a-ec,' multi look by 'act-m-ec'

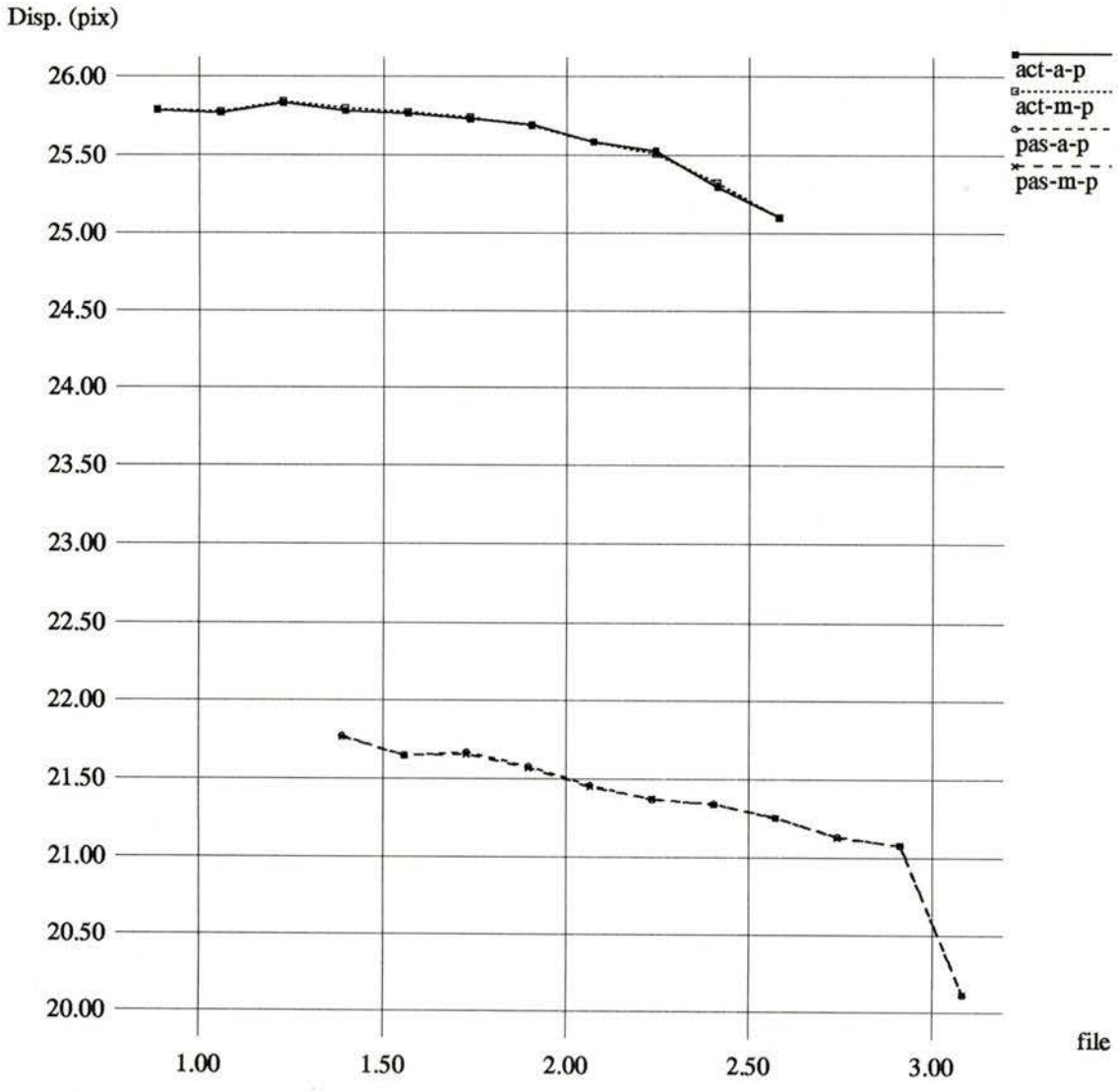


Figure 5.6: Single Pulse Parameter Estimation Results: 1.00 Pixel Expected Displacement Between Estimates

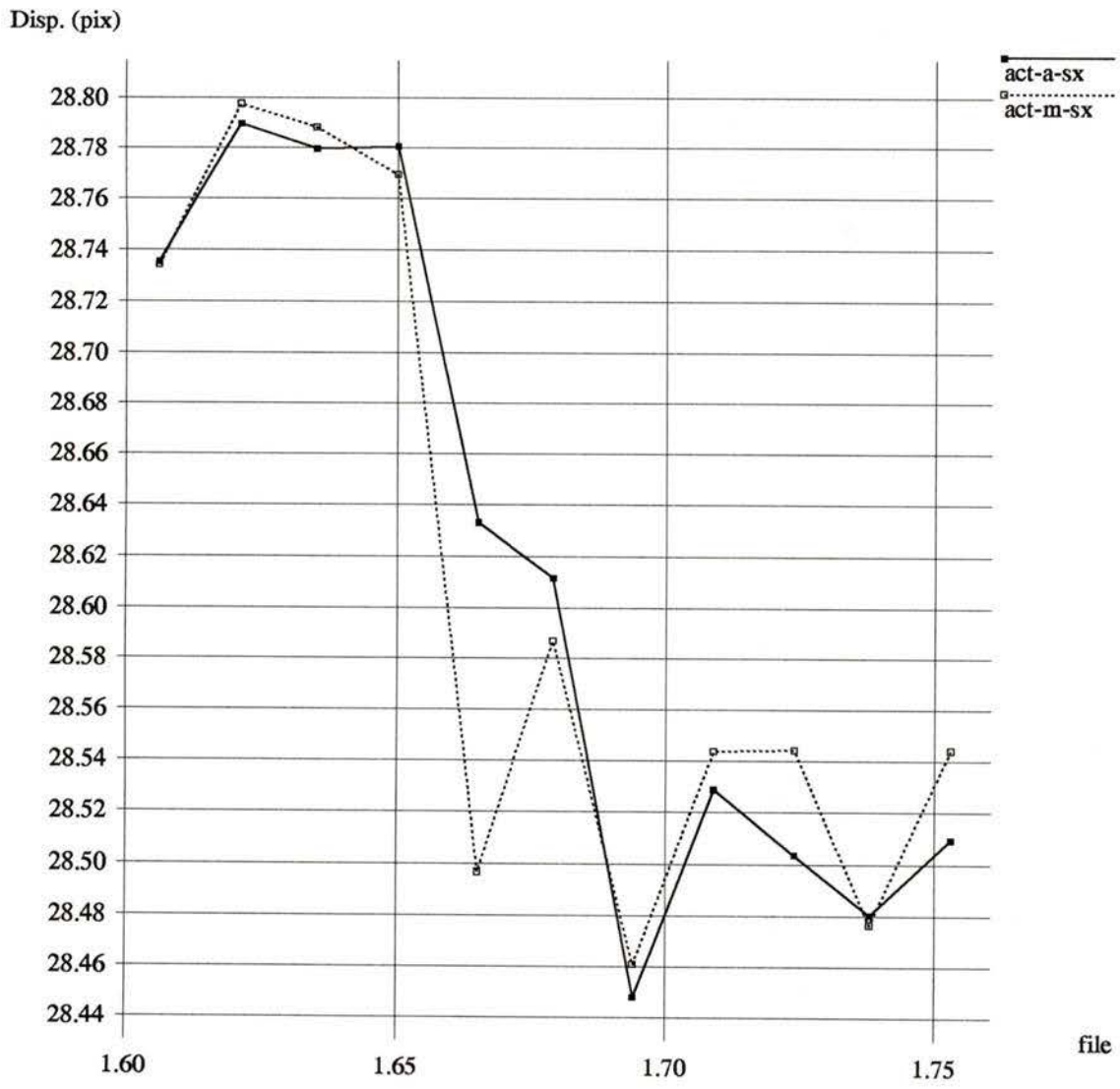


Figure 5.7: Multi Pulse Parameter Estimation Results: 0.05 Pixel Expected Displacement Between Estimates