A Comparison of Algorithms for Subpixel Peak Detection

R.B. Fisher and D.K. Naidu

Abstract. This paper compares the suitability and efficacy of five algorithms for determining the peak position of a line or light stripe to subpixel accuracy. The algorithms are compared in terms of accuracy, robustness and computational speed. In addition to empirical testing, a theoretical comparison is also presented to provide a framework for analysis of the empirical results.

1 Introduction

It is often necessary to make measurements that are outwith the precision of a visual measurement system which relies on locational accuracy to the nearest pixel. For example, in an imaging system which relies on accuracies to the nearest pixel while translating from 2-D camera coordinates to 3-D world coordinates, the accuracy of the estimated 3-D coordinates of a point in space will be limited by the image resolution. If a large spatial volume is projected onto the imaging surface, each single pixel on the imaging surface will record information from a range of positions. In our range sensor (working volume 20 cm on a side), each pixel images about 1 mm^2 of the scene. This limited resolution is not good enough for precision robotic image analysis. Therefore, algorithms that estimate feature positions to subpixel accuracy by interpolating the sensor response function (e.g., [4, 3, 5]) are useful. This paper compares five algorithms for determining the peak image position of a image line or stripe to subpixel accuracy.

To determine the stripe to subpixel accuracy, the image of the stripe width must be observed over more than one pixel. Here, we assume that the spread of intensity values across the width of the stripe is not simply random, but conforms to some kind of distribution and this pixel spread is exploited in the design of the subpixel interpolation algorithms. Some spread is almost always the case because, although it is possible optically to focus the stripe to less than a single pixel width, the operative response of individual sensor elements often leads to a measurement that is several pixels wide. If we did obtain an image of the stripe which was only a pixel wide, it would be impossible to determine where the peak of the stripe was located within the pixel because we would have data from only one pixel with which to interpolate. An example of a typical intensity response versus position is shown in Fig. 1.

J. L. C. Sanz (ed.), *Image Technology*

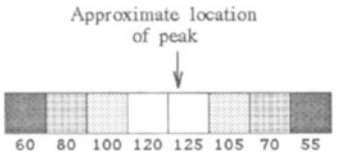

Fig. 1. Typical intensity values from contiguous pixels

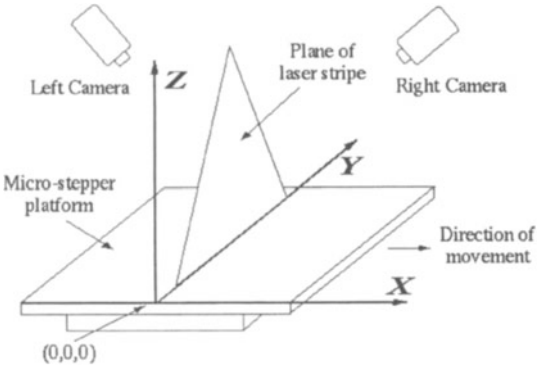

Fig. 2. Schematic of micro-stepper set-up

2 An Example Subpixel Stripe-Based Range Sensor

An example of where subpixel stripe detection methods are useful is in our laser stripe based triangulation sensor, which is used to acquire range images of objects (see Fig. 2). The target object is placed on a platform mounted on a linear micro-stepper which is activated under software control. The platform moves in small increments under a stationary laser stripe, and at each forward step of the platform a pair of digitized images of the laser stripe on the object is acquired using two cameras located on either side of the object. These images are then processed in software to derive a slice of range values of the object. Each successive step produces a fresh slice and these slices are accumulated to provide a complete range image of the object.

The digitized image from each camera is processed to determine the position of the stripe to subpixel precision. Because of the camera and stripe placement, the stripe is viewed in the image as a nearly vertical curve, the shape of which is determined by the shape of the object on which the stripe impinges. Since the curve is vertical, the scanning of the raster image is performed from left to right so as to process pixel values across the width of the perceived stripe. The y-coordinate of the pixel is determined by the vertical distance of the scan line from the top of the image. The x-coordinate is determined by the location of the pixel along a particular scan line. Therefore, when we refer to the subpixel position of the peak of the stripe, we are discussing the x-coordinate of the pixel.

Once the peak of the stripe has been detected, the image coordinates of the peak are used to determine the 3D, real-world coordinates of the point by using the

known projective transform between the camera model and the real world in conjunction with the known 3D equation of the stripe plane. Greater accuracy in determining the peak position in 2D will automatically result in a more accurate determination of the location of the stripe in 3D coordinates, which in turn will produce more accurate estimates of object dimensions and location.

3 Description of Algorithms

All subpixel algorithms that we could locate in the literature plus a new one (Gaussian) are analyzed below.

In the analyses below, x is the pixel position of the observed peak sensor reading with value $f(x)$. $f(x-1)$ and $f(x+1)$ are the values of the adjacent pixels, etc. The true peak is at $x+\delta$ and we will estimate δ by $\hat{\delta}$. The calculations use intensity values that have had the background image intensity value subtracted.

3.1 Gaussian Approximation

This algorithm uses the three highest, contiguous intensity values around the observed peak of the stripe and assumes that the observed peak shape fits a Gaussian profile. This assumption is approximately true as the light incident on the scene is known to be nearly Gaussian distributed. The real distribution, of course, will not be Gaussian, because each pixel integrates light over its field of view, the physical sensor pads of the solid-state cameras we use have a gap between them, the sensor pads have internal structure that affects their sensitivity, and not all sensor pads are equally sensitive. None the less, while we do not know the exact form of the distribution, we assume that the composition of all these effects can be modeled by a Gaussian distribution. The subpixel offset ($\hat{\delta}$) of the peak is given by:

$$\hat{\delta} = \frac{1}{2} \frac{\ln(f(x-1)) - \ln(f(x+1))}{\ln(f(x-1)) - 2\ln(f(x)) + \ln(f(x+1))}$$

As the $f()$ are usually integers in the range 0–255, the log calculation can be performed by table lookup. We have not found any previous references to this form of peak detector in the literature.

3.2 Center of Mass

The center-of-mass algorithm also assumes that the spread of intensity values across the stripe conforms to a Gaussian distribution. Thus, the location of the

peak can be computed by a simple weighted-average method. The subpixel location of the peak is given by:

$$\hat{\delta} = \frac{f(x+1) - f(x-1)}{f(x-1) + f(x) + f(x+1)}$$

The above equation describes the method using only three points. However, we have compared the same algorithm using 3, 5 and 7 points (denoted CoM3, CoM5 and CoM7) to compute the center of mass. The extension of the algorithm for the latter two cases is:

$$\hat{\delta} = \frac{2f(x+2) + f(x+1) - f(x-1) - 2f(x-2)}{f(x-2) + f(x-1) + f(x) + f(x+1) + f(x+2)}$$

for the CoM5 algorithm and for the CoM7 algorithm:

$$\hat{\delta} = \frac{3f(x+3) + 2f(x+2) + f(x+1) - f(x-1) - 2f(x-2) - 3f(x-3)}{f(x-3) + f(x-2) + f(x-1) + f(x) + f(x+1) + f(x+2) + f(x+3)}$$

Algorithms to use all points along the raster scan also exist (e.g., as used in [6]).

3.3 Linear Interpolation

This method assumes that a simple, linear relationship defines the spread of intensity values before and after the peak. Thus, if the three highest intensity values are identified as before, then:

If $f(x+1) > f(x-1)$

$$\hat{\delta} = \frac{1}{2} \frac{f(x+1) - f(x-1)}{f(x) - f(x-1)}$$

else

$$\hat{\delta} = \frac{1}{2} \frac{f(x+1) - f(x-1)}{f(x) - f(x+1)}$$

3.4 Parabolic Estimator

A continuous version of the peak finder is derivable from the Taylor series expansion of the signal intensity near the peak. If the peak is at $f(x + \delta)$ and we observe the signal at $f(x)$, then we have:

$$f'(x + \delta) = 0 = f'(x) + \delta f''(x) + O(\delta^2)$$

Hence, neglecting the higher order terms,

$$\delta \doteq -\frac{f'(x)}{f''(x)}$$

We can estimate the derivatives discretely, resulting in:

$$\hat{\delta} = \frac{1}{2}\frac{f(x-1)-f(x+1)}{(f(x+1)-2f(x)+f(x-1))}$$

This estimator is also that found by fitting a parabolic (i.e., second-order) function to the points $f(x-1)$, $f(x)$ and $f(x+1)$. In the experiments below, we call this the **parabolic** estimator.

3.5 Blais and Rioux Detectors

Blais and Rioux [2] introduced fourth and eighth order linear filters:

$$g_4(x) = f(x-2) + f(x-1) - f(x+1) - f(x+2)$$
$$g_8(x) = f(x-4) + f(x-3) + f(x-2) + f(x-1) - f(x+1) - f(x+2) - f(x+3) - f(x+4)$$

to which we also add a second order filter:

$$g_2(x) = f(x-1) - f(x+1)$$

These operators act like a form of numerical derivative operator. The peak position is estimated as above by:

$$\hat{\delta} = \frac{g(x)}{g(x)-g(x+1)}$$

The results of Blais and Rioux showed that the 4^{th} order operator had better performance than the 8^{th} order operator over the stripe widths that we are interested in here, so we only analyze it (called **BR4** below) and the simplified 2^{nd} order operator (called **BR2** below). The 8^{th} order operator has better performance for stripe widths with Gaussian width parameter larger than 2 pixels. Note that this operator is only applied in the given form for $f(x+1) > f(x-1)$. If $f(x+1) < f(x-1)$, then:

$$\delta = \frac{g(x-1)}{g(x-1)-g(x)} - 1$$

4 Maximum Error of Estimators

Assuming that the observed stripe has Gaussian form and the true peak position is near to an observed pixel, we determine the relationship between the estimated

and true peak positions (i.e., offsets from that pixel), for each of the peak detectors. Assume that the continuous stripe is modeled by:

$$f(n) = e^{-\frac{(n-\delta)^2}{2\sigma^2}}$$

where $-\frac{1}{2} \leq \delta \leq \frac{1}{2}$ is the true peak position and f is sampled at $n = -2, -1, 0, 1, 2, \ldots$ We ignore the problems of pixels integrating their inputs over their spatial extent, as well as any shaping functions the camera and digitizer may apply.

We might ask what is the maximum deviation $|\delta - \hat{\delta}|$ over the range $-\frac{1}{2} \leq \delta \leq \frac{1}{2}$ for each estimator. We generated sampled stripes for values of δ over this interval and calculated the estimated $\hat{\delta}$. For three values of σ the maximum errors are:

σ	Gaussian	CoM3	CoM5	CoM7	Linear	Parabolic	BR2	BR4
0.5	0.0	0.026	0.023	0.023	0.087	0.169	0.009	0.015
1.0	0.0	0.223	0.042	0.003	0.043	0.047	0.034	0.018
1.5	0.0	0.350	0.178	0.060	0.067	0.021	0.019	0.014

Figure 3 (left) shows the error versus δ for the CoM7 estimator for $\sigma = 1.0$.

By weighting the estimator ($\hat{\delta}' = \alpha_{\text{estimator}}\hat{\delta}$) we can, for a given σ, reduce the maximum error by spreading the error across the full range. Figure 3 (right) shows the error for the resulting CoM7 estimator when $\alpha = 1.006$. This shows that the maximum error has been reduced by almost a factor of 10. By choosing an appropriate value of α for each algorithm and the expected value of σ, we can minimize the maximum error. Here, we choose the α that minimizes the error for stripe width $\sigma = 1.0$ pixel, and examine the maximum error for the same three real stripe widths:

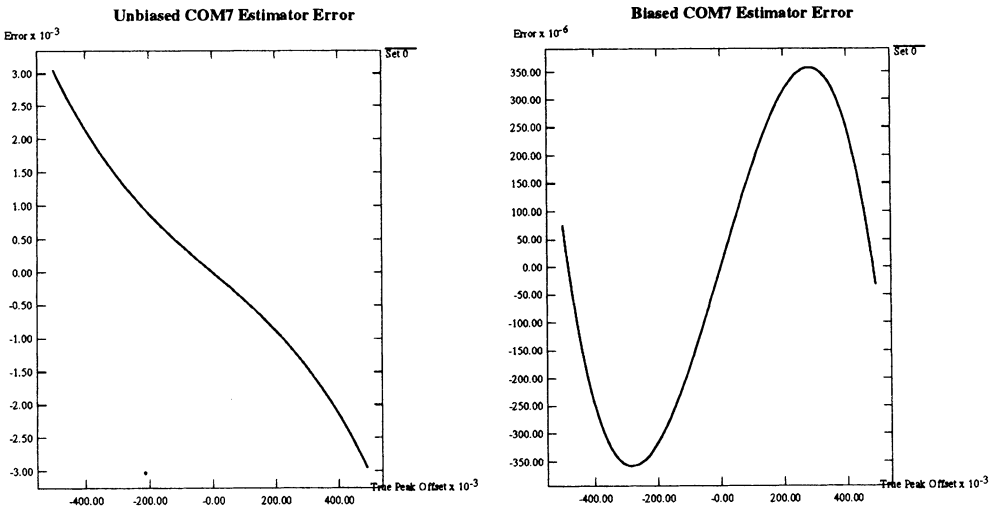

Fig. 3. Error versus δ for unbiased (*left*) and biased (*right*) COM7 estimator

σ	Gaussian	CoM3	CoM5	CoM7	Linear	Parabolic	BR2	BR4
0.5	0.0	0.380	0.041	0.021	0.103	0.156	0.026	0.023
1.0	0.0	0.005	0.002	0.000	0.030	0.029	0.024	0.013
1.5	0.0	0.239	0.150	0.057	0.049	0.034	0.022	0.011
α	1.0	1.85	1.093	1.006	0.93	1.08	0.95	0.975

This shows that, in at least the case of $\sigma = 1.0$, we can tune the estimator to have a very low error; however, setting the α values for one σ may produce reduced performance at other σs.

5 Bias of Estimators

Using the Gaussian stripe model in Sect. 4, we can determine an analytical model of the estimated peak offset $\hat{\delta}$ for a small, real offset, δ, Our analysis assumes first-order approximations:

$$e^x \doteq 1 + x$$

$$log(1+x) \doteq x$$

So:

$$f(n) \doteq \left(1 + \frac{n\delta}{\sigma^2}\right) e^{-\frac{n^2}{2\sigma^2}}$$

We can now determine the form of $\hat{\delta}$ for each peak estimator. For the **Gaussian** estimator:

$$\begin{aligned}
\hat{\delta} &= \frac{1}{2} \frac{log(f(-1)) - log(f(1))}{log(f(-1)) + log(f(1)) - 2log(f(0))} \\
&\doteq \frac{1}{2} \frac{log(e^{-\frac{1}{2\sigma^2}}(1-\frac{\delta}{\sigma^2})) - log(e^{-\frac{1}{2\sigma^2}}(1+\frac{\delta}{\sigma^2}))}{log(e^{-\frac{1}{2\sigma^2}}(1-\frac{\delta}{\sigma^2})) + log(e^{-\frac{1}{2\sigma^2}}(1+\frac{\delta}{\sigma^2})) - 2log(1)} \\
&= \frac{1}{2} \frac{log(1-\frac{\delta}{\sigma^2}) - log(1+\frac{\delta}{\sigma^2})}{2log(e^{-\frac{1}{2\sigma^2}}) + log(1-\frac{\delta}{\sigma^2}) + log(1+\frac{\delta}{\sigma^2})} \\
&\doteq \frac{1}{2} \frac{-\frac{\delta}{\sigma^2} - \frac{\delta}{\sigma^2}}{2(-\frac{1}{2\sigma^2}) - \frac{\delta}{\sigma^2} + \frac{\delta}{\sigma^2}} \\
&= \frac{1}{2} \frac{-2\frac{\delta}{\sigma^2}}{-\frac{1}{\sigma^2}} \\
&= \delta
\end{aligned}$$

Hence, the **Gaussian** estimator has the ideal form for small δ. For the **Linear** estimator:

$$\begin{aligned}\hat{\delta} &= \frac{f(1)-f(-1)}{2(f(0)-f(-1))} \\ &\doteq \frac{e^{-\frac{1}{2\sigma^2}}(1+\frac{\delta}{\sigma^2})-e^{-\frac{1}{2\sigma^2}}(1-\frac{\delta}{\sigma^2})}{2(1-e^{-\frac{1}{2\sigma^2}}(1-\frac{\delta}{\sigma^2}))} \\ &\doteq \frac{e^{-\frac{1}{2\sigma^2}}(2\frac{\delta}{\sigma^2})}{2(1-e^{-\frac{1}{2\sigma^2}})} \\ &= \frac{\delta}{\sigma^2}\frac{e^{-\frac{1}{2\sigma^2}}}{(1-e^{-\frac{1}{2\sigma^2}})}\end{aligned}$$

We skip the derivations for the other cases and summarize their results:

Estimator	Local Estimate	Estimator	Local Estimate
Gaussian	δ	CoM3	$\frac{2\delta}{\sigma^2}\frac{e^{-\frac{1}{2\sigma^2}}}{(1+2e^{-\frac{4}{2\sigma^2}})}$
Linear	$\frac{\delta}{\sigma^2}\frac{e^{-\frac{1}{2\sigma^2}}}{(1-e^{-\frac{1}{2\sigma^2}})}$	CoM5	$\frac{2\delta}{\sigma^2}\frac{e^{-\frac{1}{2\sigma^2}}+4e^{-\frac{4}{2\sigma^2}}}{(1+2e^{-\frac{4}{2\sigma^2}}+2e^{-\frac{4}{2\sigma^2}})}$
Parabolic	$\frac{\delta}{2\sigma^2}\frac{e^{-\frac{1}{2\sigma^2}}}{(1-e^{-\frac{1}{2\sigma^2}})}$	CoM7	$\frac{2\delta}{\sigma^2}\frac{e^{-\frac{1}{2\sigma^2}}+4e^{-\frac{4}{2\sigma^2}}+9e^{-\frac{9}{2\sigma^2}}}{(1+2e^{-\frac{1}{2\sigma^2}}+2e^{-\frac{4}{2\sigma^2}}+2e^{-\frac{9}{2\sigma^2}})}$
BR2	$\frac{2\delta}{\sigma^2}\frac{e^{-\frac{1}{2\sigma^2}}}{(1-e^{-\frac{4}{2\sigma^2}})}$	BR4	$\frac{2\delta}{\sigma^2}\frac{e^{-\frac{1}{2\sigma^2}}+2e^{-\frac{4}{2\sigma^2}}}{(1+e^{-\frac{1}{2\sigma^2}}-e^{-\frac{4}{2\sigma^2}}-e^{-\frac{9}{2\sigma^2}})}$

From these results, we see that the **Parabolic** operator gives one half the estimate of the **Linear** operator. When $\sigma = 1.0$ (as is approximately our case), the estimators are now:

Estimator	Gauss	CoM3	CoM5	CoM7	Linear	Parabolic	BR2	BR4
Local Estimate	1.00δ	0.55δ	0.92δ	0.99δ	1.54δ	0.77δ	1.40δ	1.20δ

However, in light of the results from Sect. 4, we use the α estimator bias to change the overall bias according to the algorithm. When $\sigma = 1.0$ (as approximately in our case), the resulting $\hat{\delta}$ is:

Estimator	Gauss	CoM3	CoM5	CoM7	Linear	Parabolic	BR2	BR4
$\hat{\delta}$	1.00δ	1.01δ	1.00δ	1.00δ	1.40δ	0.83δ	1.33δ	1.17δ

Hence, all but the **Linear** and **BR2** estimators are reasonably unbiased. Overall, this noise-free theoretical and empirical analysis suggests that the **Linear**, and **BR2** estimators are not particularly good. However, given typical sensor substructure, pixel spatial integration and cross-talk, non-gaussian stripe formation and non-linear sensor transfer functions, errors of less than 5% seem unlikely in any case. Hence, the **Gauss**, **CoM5**, **CoM7**, **Parabolic** and **BR4** estimators still seem like good candidates.

6 Errors in the Presence of Noise

In line with the experiments of Blais and Rioux [2], we investigated how error in the stripe data affected the estimated stripe position. These experiments were conducted by generating stripe data with a known, but randomly chosen stripe offset about an exact pixel position, and then corrupting the observed stripe intensity with noise. The main controlled variable was the stripe width. Uniform noise was added (following the model of Blais and Rioux). Point measurements were generated by:

$$f(n, \delta, \sigma, \beta) = e^{-\frac{(n-\delta)^2}{2\sigma^2}} + \beta\varepsilon$$

where:

$\delta \in U\,[-0.5, +0.5]$ is the stripe position.
$n \in \{-3, -2, -1, 0, 1, 2, 3\}$ are the measured pixel positions.
$\varepsilon \in U\,[0, 1]$ is the noise variable.
σ is the stripe width parameter (range 0.8 to 1.8).
β was the magnitude of the noise, and was considered for $\beta = 0.0, 0.1, 0.25$, which bounded our observed noise level (i.e., $\beta < 0.1$).

We measured both RMS error ($\sqrt{\frac{1}{N}\Sigma(\delta_i - \hat{\delta}_i)^2}$) and maximum deviation ($max\,|\,\delta_i - \hat{\delta}_i\,|$) as a function of σ for $N = 10{,}000$ samples. Figures 4, 5 and 6 show the RMS error for $\beta = 0.0, 0.1, 0.25$ respectively. Figures 7, 8 and 9 show the same for the maximum error. Immediately, we see that the **CoM3** and **CoM5** estimators are problematic. What is surprising is the error of the **CoM7** estimator in the presence of noise at low stripe widths. However, this is understandable as, when the stripe

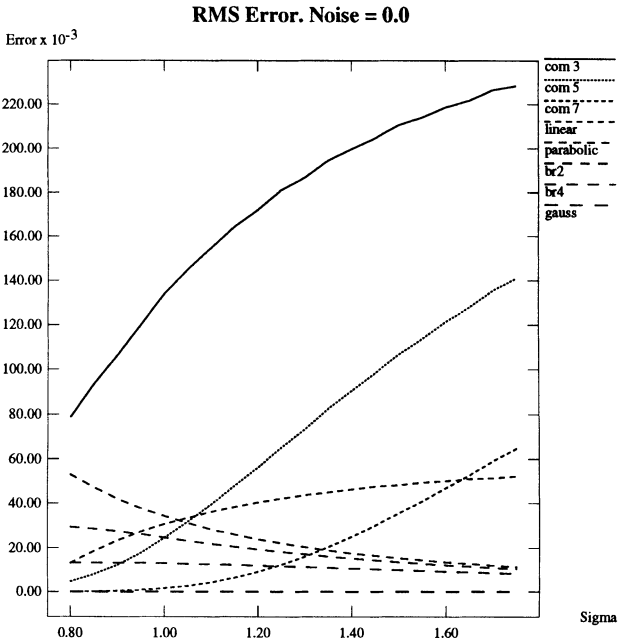

Fig. 4. RMS error versus σ for the estimators, noise = 0.0 (algorithms are listed large to small at maximum σ)

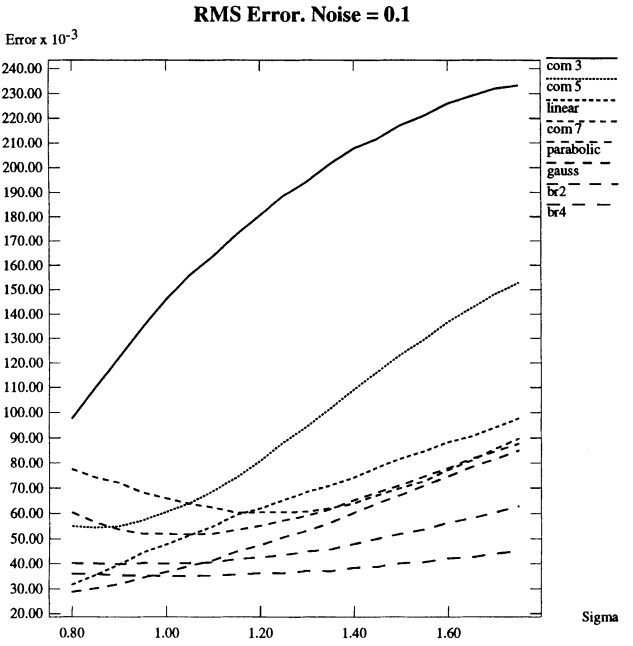

Fig. 5. RMS error versus σ for the estimators, noise = 0.1 (algorithms are listed large to small at maximum σ)

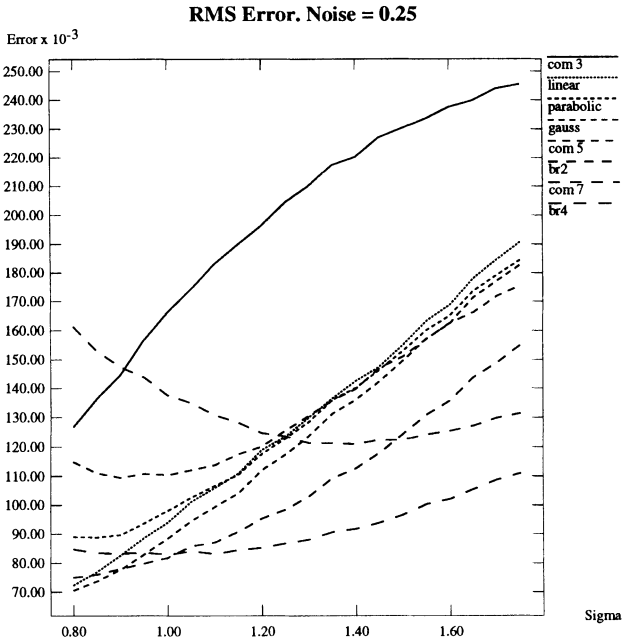

Fig. 6. RMS error versus σ for the estimators, noise = 0.25 (algorithms are listed large to small at maximum σ)

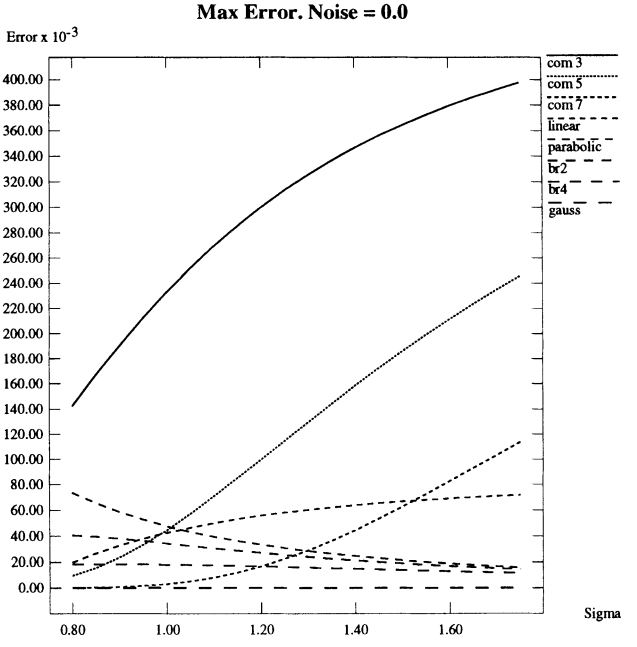

Fig. 7. Max error versus σ for the estimators, noise = 0.0 (algorithms are listed large to small at maximum σ)

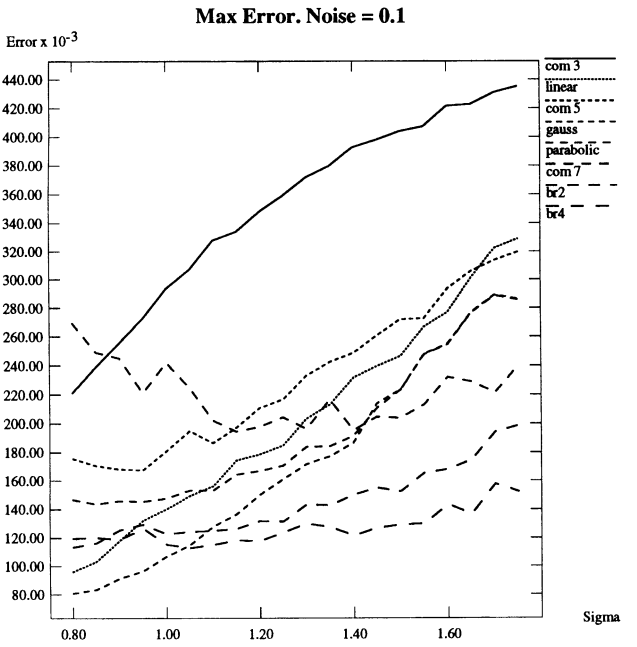

Fig. 8. Max error versus σ for the estimators, noise = 0.1 (algorithms are listed large to small at maximum σ)

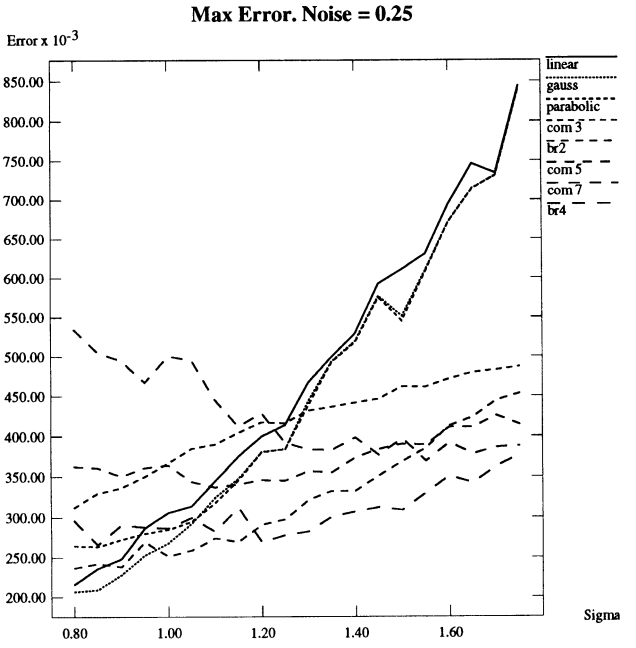

Fig. 9. Max error versus σ for the estimators, noise = 0.25 (algorithms are listed large to small at maximum σ)

width is low, the stripe intensities fall quickly at non-central pixels, causing the noise to more quickly dominate the signal and have a greater effect.

To compare the algorithms, we also summed the RMS error for $\sigma = 0.8 - 1.8$ (by 0.05) for the three values of β.

β	Gaussian	CoM3	CoM5	CoM7	Linear	Parabolic	BR2	BR4
0.00	0.00	3.71	1.36	0.31	0.87	0.49	0.39	0.24
0.10	1.07	3.90	1.86	1.32	1.36	1.23	0.93	0.77
0.25	2.49	4.25	2.67	2.63	2.62	2.61	2.12	1.86

From this, we can see good performance over a range of σ and β for the **BR2** and **BR4** estimators. This is also clear in Figs. 5 and 6, however, the **Gaussian** estimator has obvious benefits as the noise level or stripe width decreases. It is also interesting that the figures show to what extent the choice of estimator is linked to the specific stripe width and noise level. For our stripe system, we have observed:

Target color	Mean stripe peak intensity	Stripe σ	Background range
white	201	1.69	13–15
grey	165	1.31	11–12
black	60	1.22	10–12

Hence, for our striper, the noise seems to be about 2–3 quanta, or about 1–5% of the peak intensity. We think that the increase in σ as the intensity increases is explained by the gamma compression of the camera flattening the stripe peak.

7 Algorithm Behavior on Sensor Saturation

No algorithm should produce wildly unreasonable estimates of the peak position when the sensor is saturated. When saturation occurs, the measured intensity values at the peak and nearby pixels are usually at some limiting value (e.g., 255). Moreover, because of the effects of saturation on the physical sensor, adjacent pixels whose true signal is below saturation may also be affected or become saturated. Hence, use of these adjacent pixels may also not be possible.

The **Gaussian, Linear** and **Parabolic** algorithms given in Sect. 3 have a definite problem in this situation, resulting in a division by zero. We propose that these algorithms can be modified to use the midpoint of the saturated region:

```
if overflow-occurred
{
    peak_position = last_overflowed_pixel - overflow_length/2 + 0.5
}
else use_normal_algorithm
```

The other algorithms do not actually perform too badly, provided the region of saturation is only 1–3 pixels. We tested the behavior of the algorithms by an experiment where the pixel values were generated using the formula given in Sect. 4. Then, whenever the intensity value was greater than 0.5, it was set to 0.5 (i.e., the saturation limit). This limit allowed a maximum of three consecutive saturated pixels. The algorithms were applied at all subpixel offsets from –0.5 to 0.5, and the deviation of the estimated subpixel offset away from the true offset were recorded. Figure 10 shows the deviations for the **Gauss** and the **BR4** algorithms.
The maximum deviations for the algorithms are:

Algorithm	Gaussian	CoM3	CoM5	CoM7	Linear	Parabolic	BR2	BR4
Max Deviation	0.320	0.255	0.059	0.049	0.320	0.320	0.175	0.071

Some improvement might be possible by a function of the non-saturated pixels surrounding the saturated region, but the utility of these algorithms depends on how the sensor responds when saturated.

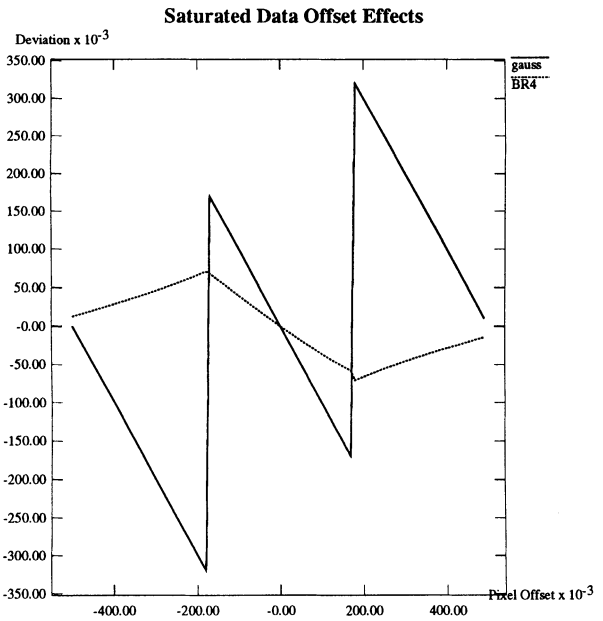

Fig. 10. Deviations of pixel offsets using saturated data

8 Empirical Testing

We tested all algorithms in the laser stripe range sensor described in Sect. 2. The experiments used three different test objects. These were a cube, a trapezoid with its top surface at an angle of 10° to the horizontal, and an equilateral prism (see Fig. 11). In all cases, the object was oriented so that all range values along the stripe were equal. The test objects were aligned so that their surface normals lay in the $X - Z$ plane. The coordinate system, the relative positioning of the cameras and the direction of motion of the micro-stepper are clarified in Fig. 2.

The experiments obtained a series of range images comprising 100 range stripes each. A single data point was chosen from each stripe, such that all the data points chosen lay along a line parallel to the x-axis. Secondly, the stripes were taken with a very small micro-stepper movement (0.2 mm for the prism and 0.3 mm for trapezoid and the cube), thereby leading to high data density. Also, the depth quantization was kept small (0.03 mm), so that the estimation errors would be of larger magnitude than the quantization errors.

For each surface, each algorithm was used to detect the stripe peak, and then the depth calculated from each camera was noted and their average computed. Having collected the data, a linear least-squares fit ($z = \hat{a} + \hat{b}x$) was computed for each set of data points. The slope of this fitted line was computed, and the minimum and maximum values of the errors were recorded. The variance of the errors was also computed. The comparative statistics are shown below.

Surface	Algorithm	$\hat{a}$	$\hat{b}$	Error(mm)		
				Min.	Max.	Variance
Flat Surface	Gaussian	49.7680	0.000112	−0.12762	0.09815	0.00249
	CoM – 3 pt	49.6466	0.000006	−0.06152	0.08812	0.00122
	CoM – 5 pt	49.7025	0.000130	−0.10121	0.09972	0.00156
	CoM – 7 pt	49.6239	0.000036	−0.09555	0.09658	0.00156
	Linear	49.7878	0.000090	−0.07676	0.08499	0.00120
	Parabolic	49.7397	0.000297	−0.14026	0.14496	0.00271
	BR – 2nd	49.6804	−0.000171	−0.21498	0.11769	0.00323
	BR – 4th	49.6808	−0.000411	−0.13912	0.16080	0.00343

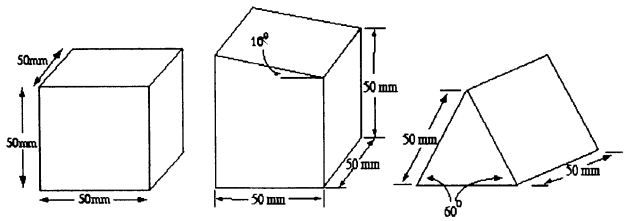

Fig. 11. Detail of experimental objects

Surface	Algorithm	$\hat{a}$	$\hat{b}$	Error(mm)		
				Min.	Max.	Variance
10° slope, left to right	Gaussian	57.2598	−0.05306	−0.24558	0.38329	0.02463
	CoM – 3 pt	57.7386	−0.05443	−0.45631	0.56070	0.05423
	CoM – 5 pt	57.6771	−0.05466	−0.37467	0.31389	0.03110
	CoM – 7 pt	57.6287	−0.05478	−0.29864	0.34434	0.02416
	Linear	57.6665	−0.05391	−0.30376	0.31999	0.01737
	Parabolic	57.7211	−0.05436	−0.23738	0.37868	0.01899
	BR – 2nd	57.8916	−0.05389	−0.27106	0.28737	0.01836
	BR – 4th	57.8261	−0.05382	−0.28686	0.33492	0.02110
10° slope, right to left	Gaussian	51.3929	0.05424	−0.31441	0.51424	0.03381
	CoM – 3 pt	51.8246	0.05257	−0.43193	0.47276	0.04573
	CoM – 5 pt	51.7356	0.05321	−0.34414	0.41777	0.03457
	CoM – 7 pt	51.6648	0.05277	−0.32581	0.42064	0.03128
	Linear	51.7305	0.05263	−0.35943	0.40299	0.03645
	Parabolic	51.7699	0.05281	−0.35278	0.41456	0.03496
	BR – 2nd	50.9346	0.05460	−0.33865	0.48187	0.03299
	BR – 4th	50.9251	0.05439	−0.30189	0.47461	0.03184

Surface	Algorithm	$\hat{a}$	$\hat{b}$	Error(mm)		
				Min.	Max.	Variance
60° slope, left to right	Gaussian	70.2352	−0.34806	−0.43505	0.45969	0.04737
	CoM – 3 pt	70.3217	−0.34726	−0.42918	0.51835	0.05024
	CoM – 5 pt	70.3418	−0.34816	−0.37187	0.55610	0.04146
	CoM – 7 pt	70.1715	−0.34771	−0.39819	0.51866	0.04106
	Linear	70.2851	−0.34810	−0.43443	0.42978	0.04897
	Parabolic	70.3154	−0.34841	−0.39460	0.50010	0.04764
	BR – 2nd	70.1493	−0.34781	−0.46431	0.51234	0.05624
	BR – 4th	70.1202	−0.34813	−0.47256	0.47195	0.04401
60° slope, right to left	Gaussian	23.5725	0.34834	−0.34344	0.44277	0.03918
	CoM – 3 pt	23.5071	0.34925	−0.33353	0.41938	0.03397
	CoM – 5 pt	23.5518	0.34852	−0.31148	0.35788	0.03213
	CoM – 7 pt	23.3997	0.34865	−0.40910	0.48459	0.03624
	Linear	23.5029	0.34894	−0.38680	0.47643	0.03994
	Parabolic	23.5249	0.34873	−0.39773	0.44770	0.03675
	BR – 2nd	23.5494	0.34841	−0.31710	0.46174	0.03628
	BR – 4th	23.5244	0.34870	−0.34116	0.52795	0.02985

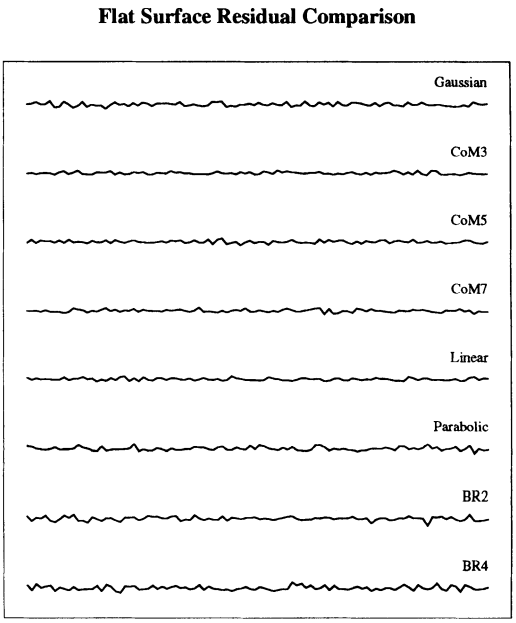

Fig. 12. Comparative depth residuals across a flat surface. The data is separated for clarity

The measured values for $\hat{a}$ are not particularly relevant. The values of $\hat{b}$ are of interest, because they specify the slope of the surface as measured by the algorithms used. The true value of $\hat{b}$ is 0 in the case of the cube, ± 0.0538 in the case of the trapezoid, and ± 0.3464 for the prism. These values were derived by careful physical measurements of the objects, but are still subject to the usual measurement errors. However, they provide us with a basis for comparing the accuracies of the different peak-picking methods.

The tabulated results are better illustrated by the comparative graphs shown in Figs. 12–14. The graphs are the plots of the variation of the residuals from the line of best fit applied to the measured data. The plots have each been offset by a different amount so that they are appear together on one graph. The scale on the Y-axis is the same for all the data sets. The maximum absolute variation of the plots from the line of best fit is about 0.5 mm.

Figure 12 shows the variation across the surface of the cube. The **CoM3** and **BR2** algorithms appear to show the least amount of perturbation, followed closely by the plots derived from the **Linear** and **BR4** methods.

The results with the trapezoid (see Fig. 13) clearly show systematic errors in the imaging system, particularly with all the **CoM** algorithms. These are caused by the aliasing of the image of the peak creeping from one pixel to the next, crossing the inter-pixel gap. We estimate this gap is itself almost as wide as a pixel. The non-uniform response across a pixel response is also a source of the observed periodic effect (see e.g., [1]).

The performance of all the algorithms deteriorates dramatically in the case of the prism (see Fig. 14). The variations are more pronounced and more frequent than in the previous two examples. This is because the acute angle of the object,

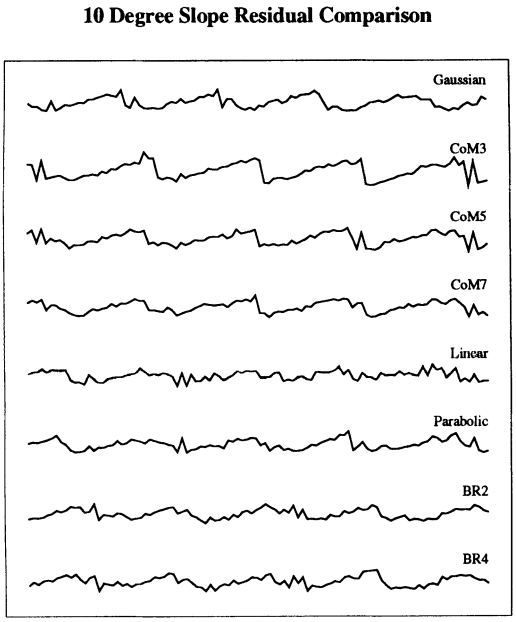

Fig. 13. Comparative depth residuals across a 10°, left-to-right slope. The data is separated for clarity

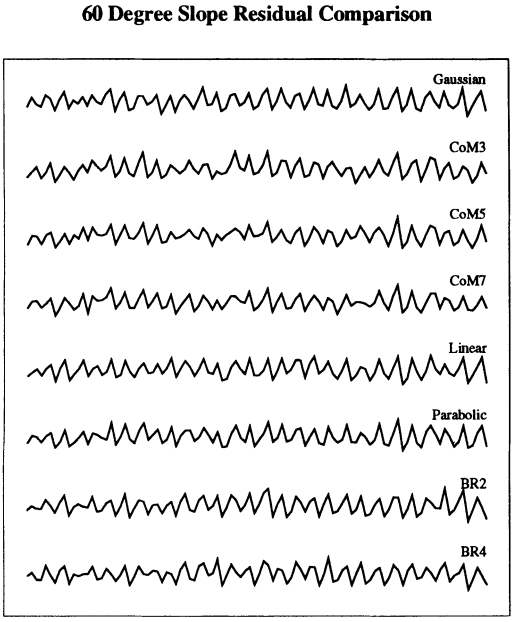

Fig. 14. Comparative depth residuals across a 60°, left-to-right slope. The data is separated for clarity

which is very close to the angle of the camera axis to the horizontal, causes the stripe to move across the imaging sensors pixels more quickly.

The aliasing gives a periodic structure to the estimators. Some portion of this periodic, and systematic error could probably be reduced by modeling the effect as

a function of local surface slope and uncorrected estimated subpixel position. In the case of our sensor, the errors observed here are symptomatic of the sensor structure and dominate most of the theoretical results discussed above. We have investigated using subpixel algorithms that fit an observed stripe profile to an empirically derived real stripe profile; however, the stripe profiles that we observed varied from pixel to pixel with little systematic character, and we concluded that modeling each pixel's response individually was unprofitable for our application.

9 Conclusions

The empirical results show that the **CoM3** algorithm has poor performance. The other methods display performance within the same range probably because of factors such as sensor structure, inter-pixel gaps, cross-talk, and integration of the sensor response over the width of the pixel. The **Linear** and **BR2** methods have been shown to possess high bias (Sect. 5). When we consider the errors produced, the sum of the RMS errors are highest for the **CoM3** and **CoM5** algorithms. They are joined by the **Parabolic** algorithm when we consider the maximum errors. This leaves us with only the **Gaussian, CoM7** and **BR4** algorithms as suitable candidates.

In the case of algorithms like the **CoM7** and **BR4**, which rely on a large number of points around the peak, we observe that specular reflections and transparency may cause problems since the outlying pixels have a substantial effect on the computation of the location of the peak. Also, in the case where the object has holes in it, causing internal reflections and mutual illumination, the weighted average method of the **CoM** algorithms will deliver a skewed estimate of the peak position. That is, when random noise levels are low, estimators using a small number of ponts will have advantages in avoiding effects arising from the structure of the sensed object.

We can see good performance over a range of σ and β for the **BR4** estimator. This is also clear in Fig. 5; however, the **Gaussian** estimator has obvious benefits as the noise level or stripe width decreases. It is also interesting that the figures show to what extent the choice of estimator is linked to the specific stripe width and noise level. For our striper, the noise seems to be about 2–3 quanta, or about 1–5% of the peak intensity. Note that in all cases we assumed that the intensity levels were below the saturation level of the sensor.

When comparing the speeds of the algorithms, the **Gaussian** is the slowest by about a factor of 2 over the **Linear** algorithm, which is the fastest in our implementation. However, the peak detection sub-process takes up only a small percentage of the total range image acquisition and peak detection time, so the speed of the algorithms is not a factor in their comparison.

In addition to these results, we have not seen published before a bias analysis (cf. Sect. 5) and a bias factor on the estimator (cf. Sect. 4). Finally, the aliasing effect observed in Sect. 8 does not seem to have been reported before in conjunction with subpixel range sensors.

These results apply to sampled and digitized signals, whereas some algorithms, namely the center-of-mass algorithm, can be applied directly to the video signal [3, 6]. Real-time digitized stripe detection and subpixel location can be achieved by scanning for the peak as the digitized signal is acquired. Both approaches have been implemented (elsewhere) in hardware and thus remove computational expense as a consideration, because the subpixel calculations are fast enough to be completed in the time before the next pixel is acquired, and one has to wait for the complete video scan anyway, when using standard video equipment.

Acknowledgements. The authors gratefully acknowledge the assistance of the University of Edinburgh and the European Institute of Technology (Grant EIT-061) for funding the research described in this paper.

References

1. B. G. Batchelor, D. A. Hill and D. C. Hodgson, "Automated Visual Inspection", IFS (Publications) Ltd, UK, North Holland, 1985.
2. F. Blais & M. Rioux, "Real-Time Numerical Peak Detector", Signal Processing 11, pp 145–155, 1986.
3. D. Braggins, "Achieving subpixel precision", Sensor Review, Volume 10, No. 4, pp 174–177, 1990.
4. P. J. MacVicar-Whelan & T. O. Binford, "Intensity Discontinuity Location to Subpixel Precision", Proceedings of the 7th International Joint Conference on Artificial Intelligence, pp 752–754, 1981.
5. D. K. Naidu, R. B. Fisher, "A Comparitive Analysis of Algorithms for Determining the Peak Position of a Stripe to Sub-Pixel Accuracy", Proc. 1991 British Machine Vision Association Conf., pp 217–225, Glasgow, 1991.
6. S. J. White, "Method and Apparatus for Locating Center of Reference Pulse in a Measurement System", U.S. Patent No. 4,628,469, December 1986.