N94-25504

Expansion-Based Passive Ranging

Yair Barniv
NASA Ames ResearchCenter

94-24744

SUMMARY

This paper describes a new technique of passive ranging which is based on utilizing the image-plane expansion experienced by every object as its distance from the sensor decreases. This technique belongs in the feature/object-based family.

The motion and shape of a small window, assumed to be fully contained inside the boundaries of some object, is approximated by an affine transformation. The parameters of the transformation matrix are derived by initially comparing successive images, and progressively increasing the image time separation so as to achieve much larger triangulation baseline than currently possible. Depth is directly derived from the expansion part of the transformation.

To a first approximation, image-plane expansion is independent of image-plane location with respect to the focus of expansion (FOE) and of platform maneuvers. Thus, an expansion-based method has the potential of providing a reliable range in the difficult image area around the FOE. In areas far from the FOE the shift parameters of the affine transformation can provide more accurate depth information than the expansion alone, and can thus be used similarly to the way they have been used in conjunction with the Inertial Navigation Unit (INU) and Kalman filtering. However, the performance of a shift-based algorithm, when the shifts are derived from the affine transformation, would be much improved compared to current algorithms because the shifts—as well as the other parameters—can be obtained between widely separated images.

Thus, the main advantage of this new approach is that, allowing the tracked window to expand and rotate, in addition to moving laterally, enables one to correlate images over a very long time span which, in turn, translates into a large spatial baseline—resulting in a proportionately higher depth accuracy.

PRECEDING PAGE BLANK NOT FILMED

234

ACRONYMS USED IN TEXT

FOE	- Focus of Expansion
FOV	- Field of View
INU	- Inertial Navigation Unit
LOS	- Line of Sight
OF	- Optical Flow
PSF	- Point-Spread-Function
SNR	- Signal-to-Noise Ratio
PSR	- Peak-to-Sidelobes Ratio
TBD	- Track Before Detect
TTC	- Time To Collision
3-D	- 3-Dimensional
AFTR	- Affine Transformation
CW, CCW	- Clockwise, counterclockwise

1 INTRODUCTION

Passive ranging is an area of considerable interest for applications such as obstacle avoidance for rotorcraft nap-of-the-earth navigation and spacecraft landing. Two main passive-ranging methods can potentially be employed for this purpose; one based on motion and the resulting image-plane optical flow, and the other based on stationary stereo. Both methods can be thought of as special cases of a more general triangulation method known as "bearing-only" or "direction-of-arrival" (*e.g.*, [1, 2, 3, 4]). In this paper we chose to concentrate on monocular OF-based ranging.

The motion of an imaging sensor causes each imaged point of the scene to correspondingly describe a time trajectory on the image plane. The trajectories of all imaged points constitute what's called the "optical flow" (OF). A forward-looking imaging sensor, such as a TV camera or a Forward-Looking Infra-Red (FLIR), is typically used as the source of optical flow data. The various methods of extracting depth information from the OF can be classified as belonging into three main classed as we did in [5]. The method described in this paper can be considered object-based or feature-based depending on the definition of features. If the features are chosen based on some local image property, such as texture or edge, then we are dealing with a feature-based method. If, the feature is chosen through some pre-segmentation to be wholly contained inside a physical object, then our new technique can be considered to be object-based; that is how we chose to regard it in this paper.

Like all other passive-ranging methods, we assume that the scene and its illumination sources are temporally constant (see [6]). We also assume that all points belonging to the same object share the same range. In [5] we differentiated between detect-then-track and track-before-detect (TBD) algorithms (akin to filtering and smoothing respectively) and pointed out the advantages of the TBDs in terms of SNR-performance and robustness (see [7, 8, 9]). We will return to this subject after presenting our new algorithm, and show that it is a TBD one.

The OF at any given point in the image plane consists of three kinds of motion: lateral translation, expansion (or divergence), and rotation (curl). When considering a window of some finite size, one can approximately describe its time evolution by an affine transformation which, in the most general case, has six parameters: four belonging to the 2×2 multiplying matrix, and two belonging to the vector of lateral translation. Most depth-estimation methods, such as described in [10, 11], make use of the lateral motion alone. Two basic limitations are implicit in these methods. First, they cannot perform in the image plane close to the FOE, and second, they can only use a relatively short triangulation baseline because far-apart images would not correlate due to the misadjustment in the other components of the affine transformation (besides the shifts) which are not accounted for. "Triangulation baseline" is the term we use for the distance the platform travels between the frames to be correlated. As we have shown in our earlier work [12], the depth-error is inversely proportional to the triangulation baseline (see (18)

ahead).

In this work I will discuss methods of extracting depth information from the divergence of the OF as it is approximately obtained from the affine transformation matrix. I use the term "divergence" (or "local divergence") to refer to the mathematical definition of the derivative-vector operator denoted by ∇ which, in this case, scalar-multiplies the velocity vector at a point. Divergence is thus defined for an infinitesimal area and time. We use "expansion", or "global divergence", as a short-hand for the "rate of area expansion" to denote the *average* divergence over the area of some finite-size window or of an object. We will soon see why the divergence of the velocity vector, $\nabla \cdot \mathbf{v}(p)$ at some point p actually measures the rate of area expansion (which explains the above proliferation of terms).

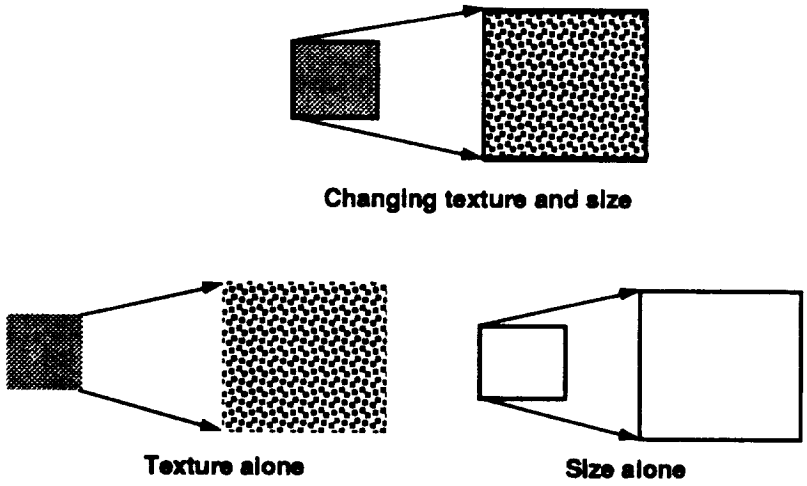

Figure 1: Texture and size cues.

The idea of using divergence as a source of depth information is not new. The works of Longuet-Higgins and Prazdny [13], Prazdny [14, 15], Koenderink [16], Koenderink and van Doorn [17, 18], and Nelson and Aloimonos [19] elaborate quite extensively on this subject. Recently, an interesting extension to these works was reported by Ringach and Baram in [20]; although it is field-based, it explicitly assumes that the scene is composed of objects (defined by their borders) and derives the global divergence for all objects without the need to actually delineate or identify them. The local- and global-divergence methods are intended for different kinds of objects as exemplified in figure 1. The local-divergence method is intended for textured objects with no well-defined edges, whereas the global-divergence method is intended for objects with little or no texture but having well-defined edges. In this paper we rely upon the objects being textured, so our algorithm roughly derives the equivalent of local divergence.

If we examine a window centered on the FOE, its translational motion is zero by definition, but it still expands as the depth decreases, and this expansion is left as the only source of depth information. Thus, there are two new aspects to our work; one is the direct derivation of depth from expansion, and the other is enabling the use of a long triangulation baseline for

even using just the conventional translation-based methods (as well as for the expansion-based ones). The later feature is the one that transforms this algorithm from a track-then-detect to a TBD, because the accuracy (or SNR) of the final result is based on the largest available baseline, as opposed to (Kalman) filtering of results that were individually obtained based on a single-interframe baseline.

This is why one can consider this work to represent an extension of the existing translation-based algorithms such as the one developed by Sridhar, Phatak, and Cheng in [10, 11] and Sridhar, Suorsa and Hussien in [21] which derive the image-plan translations of "points of interest" (small windows) through spatial cross-correlation between consecutive images and subsequent Kalman filtering of their image-plane trajectories.

In order to round off the picture, we also need to refer to another closely-related area of research represented by the work of Merhav and Bresler (see [22, 23, 24, 25]). The first three papers primarily address image-plane motion estimation, which is, of course, equivalent to depth. Also, they rely upon the assumption (that we do not need to make) that the image statistics in the X and Y directions are separable. The fourth paper suggests a stochastic-gradient approach to image-plane motion estimation which can be thought of as a precursor of the work reported here.

As a last comment, it is noteworthy that utilizing divergence (or expansion) for depth derivation has been largely motivated by advances in the understanding of visual processing in humans and primates. For example, experiments with humans suggest the existence of divergence (looming) detectors in the human visual system [26, 27, 28] as well as *vorticity* detectors [28, 29, 30].

The organization of this report is as follows. Section 2 contains the theory relating Divergence, Expansion, and Depth. Section 3 presents the idea of using the affine transformation to relate objects in different frames. Section 4 presents simulation results. Section 5 presents the practical algorithm that iterates over increased frame separation. Section 6 discusses the error analysis.

2 OPTICAL FLOW, DIVERGENCE AND EXPANSION

The basic equations for the divergence in the image plane are derived in this section. This derivation is based on prior work described in [13] to [20].

It is convenient to think for a moment of imaging the outside scene onto a spherical surface because such projections are identical irrespective of the camera-axis direction. In fact, with

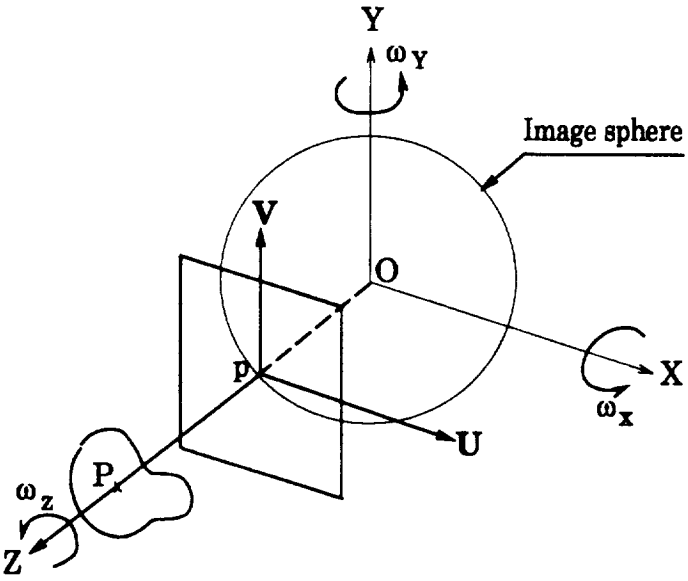

Figure 2: The geometry of projection onto the image plane.

such geometry, the camera axis is *defined* to coincide with the line-of-sight (LOS) from the center of the sphere to any imaged point as seen in figure 2. Another motivation for regarding the image plane as a sphere is that this geometry is similar to that of imaging the world by a lens onto a spherical retina, *e.g.*, in the human eye. Let us define the coordinate system of the spherical camera to have its origin at the sphere's center and its Z axis to pass through the imaged point P of some object. The sphere is defined to be of unity radius. Consider the projection of P onto the sphere at point p. At that point define the origin of an (U, V) plane tangent to the sphere which is called *local projective image plane* (image plane, for short); this image plane approximates the sphere at the point of tangency. Let us assume that P is found on a smooth surface described by some function $z = f(x, y)$ so that its gradient $\nabla z = (z_X, z_Y)$ exists. The distance of any point on that surface from the sphere's center can then be approximated in the neighborhood of P by

$$z \approx z_0 + \nabla z \cdot (x, y) \,, \tag{1}$$

where z_0 is the distance between points O and P. The relative motion of the camera with respect to the scene is defined by its translational velocity $\mathbf{V} \triangleq (V_X, V_Y, V_Z)$ and rotational velocity $\omega \triangleq (\omega_X, \omega_Y, \omega_Z)$. It is convenient to normalize $\mathbf{V}$ by z_0 and define $(v_X, v_Y, v_Z) \triangleq (V_X, V_Y, V_Z)/z_0$.

The motion of the camera causes the stationary point P and its surrounding to describe a retinal velocity field (or optical flow) around p on the image plane. We denote image-plane projections by (u, v), to correspond with (U, V), and their temporal derivatives by (u_t, v_t). Thus, the image-plane velocity vector at p is defined as $\mathbf{v}(p) \triangleq (u_t, v_t) \mid_p$. The spatial partial derivatives of (u_t, v_t) are denoted by $u_{tx}, u_{ty}, v_{tx}, v_{ty}$. From [13] we know that the following equations hold at p,

$$u_t = -v_X - \omega_Y \,, \qquad v_t = -v_Y + \omega_X \,,$$

$$u_{tx} = v_Z + v_X z_X \,, \qquad v_{ty} = v_Z + v_Y z_Y \,,$$
$$u_{ty} = \omega_Z + v_X z_Y \,, \qquad v_{tx} = -\omega_Z + v_Y z_X \tag{2}$$

Using the above equations, the divergence at p (denoted $\mathrm{div}(p)$) can be expressed as

$$\mathrm{div}(p) = \nabla \cdot \mathbf{v}(p) \triangleq u_{tx} + v_{ty} = 2v_Z + \nabla z \cdot (v_X, v_Y) \tag{3}$$

To interpret the above equation, suppose that the camera only moves in the Z direction. In that case $v_X = v_Y = 0$ and $\nabla \cdot \mathbf{v}(p) = 2v_Z = 2V_Z/z_0$, that is, $\mathrm{div}(p)$ is twice the reciprocal of the time-to-collision (TTC) of P with the camera's center. Because of this interpretation, $\mathrm{div}(p)$ is termed "immediacy" in [16] and other papers, that is, it measures the immediacy of an imminent collision. In the opposite case, when $(v_X, v_Y) \neq (0,0)$ and $v_Z = 0$, there can still be a relative depth change between the camera and the patch because it is generally slanted. In other words, $\mathrm{div}(p)$ will still have the same interpretation as before, except that the imminent collision is going to be with some point on the plane tangent to the patch at P and not with the point P itself. Thus both terms of the immediacy have a valid physical interpretation. Notice that the rotational velocities do not appear in $\mathrm{div}(p)$. This is a very important (and well-known) observation because it says that *the TTC information is wholly contained in the imagery; no additional information is needed* (such as from the INU).

Nelson and Aloimonos describe in [19] a straight-forward mechanism for evaluating the divergence from a sequence of images. In practice, this algorithm can only provide a rough estimate of the local divergence.

The global divergence is defined (see [20]) as the *average* divergence over the area of each object, and denoted by $\chi(R)$ for an object whose projection onto the image plane is R (assuming, for the moment, that its boundary ∂R is well defined). Thus,

$$\chi(R) \triangleq \frac{1}{A(R)} \int_R \mathrm{div}(p)\, \mathrm{d}s = \frac{1}{A(R)} \int_{\partial R} \nabla \cdot \mathbf{v}(p)\, \mathrm{d}s = \frac{1}{A(R)} \int_{\partial R} \mathbf{v}(p) \cdot \mathbf{n}\, \mathrm{d}l \,, \tag{4}$$

where $A(R)$ is the object area, $\mathrm{d}s$ is the elemental area, $\mathrm{d}l$ is the elemental length along ∂R, $\mathbf{n}$ is a unit vector normal to ∂R, and the equality is based on the divergence theorem. In words, the average divergence equals the line integral of the normal component of the velocity vector at the edge along the edge of the object. This line integral can easily be shown (see [20]) to have an intuitive interpretation, that is,

$$\chi(R) = \frac{1}{A(R)} \frac{\mathrm{d}A(R)}{\mathrm{d}t} \,, \tag{5}$$

i.e., the global divergence equals the temporal rate of change of the normalized object area.

To find the relationship between global divergence, expansion, and time-to-collision, consider the similar-triangles equation relating the image-plane projection at $(u, 0)$ of some point

similar to P but located at $(x = l, y = 0, z = z_0)$ in figure 2,

$$u = \frac{l}{z_0} \tag{6}$$

Taking the derivative of u with respect to z_0, we find that

$$\mathrm{d}u = -\frac{u}{z_0}\mathrm{d}z_0 = \frac{u}{z_0}V_Z\mathrm{d}t = uv_Z\mathrm{d}t\,, \tag{7}$$

Thus,

$$\frac{1}{u}\frac{\mathrm{d}u}{\mathrm{d}t} = v_Z \tag{8}$$

If we repeat this derivation for an *area* change, $\mathrm{d}A$, rather than for a length change, $\mathrm{d}u$, we would find, using $\mathrm{d}A/A = 2\mathrm{d}u/u$, that

$$\frac{1}{A}\frac{\mathrm{d}A}{\mathrm{d}t} = 2v_Z \tag{9}$$

Comparing (9) to (5), it is seen that χ has the interpretation of twice the TTC. Thus, the normalized (by the area) temporal rate of change of the projected area A of some object, that is, its *rate of area expansion*, equals twice the TTC.

3 ESTIMATING THE RATE OF EXPANSION

In this section we introduce the affine transformation, and develop the algorithm necessary to estimate the object's rate of expansion.

3.1 The affine transformation

The affine transformation (AFTR) can be used to relate object's projections at different frames (or times); its most general form is defined by six parameter. However, we intuitively judged that four parameters should suffice because they directly convey the physically-interpretable changes one would expect to occur. We thus define our specific AFTR by

$$\begin{bmatrix} \hat{u} \\ \hat{v} \end{bmatrix} = s\begin{bmatrix} C\theta & -S\theta \\ S\theta & C\theta \end{bmatrix}\begin{bmatrix} u - u_0 \\ v - v_0 \end{bmatrix} + \begin{bmatrix} a + u_0 \\ b + v_0 \end{bmatrix}, \tag{10}$$

where s is a scaling (or expansion) factor, $C\theta \triangleq \cos(\theta)$ and $S\theta \triangleq \sin(\theta)$, and θ is the angle by which the object in I_1 is CW rotated with respect to its original orientation in I_0. Thus, this AFTR maps points (u, v) from one frame (I_0) onto the corresponding points $(\hat{u}, \hat{v})$ in another frame (I_1). In figure 3 we notice that, first, the object expanded about 50%, second, it rotated about 25^0 CCW, and third, it moved up and right. This is indeed the order of mappings conveyed

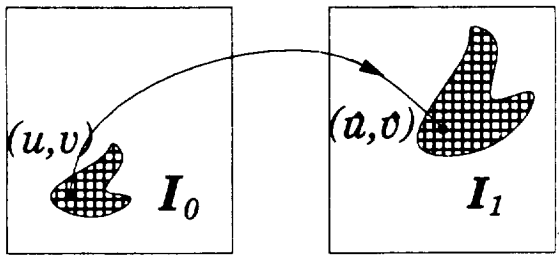

Figure 3: Mapping of a point through the affine transformation.

by the above definition although the order of scaling and rotation is immaterial. Notice that scaling and rotation is performed around the arbitrarily-defined center point of the object located at (u_0, v_0), and shifting is performed later—back to the original center point plus an incremental shift by the vector (a, b).

3.2 Vehicle's maneuvers and image-plane motion

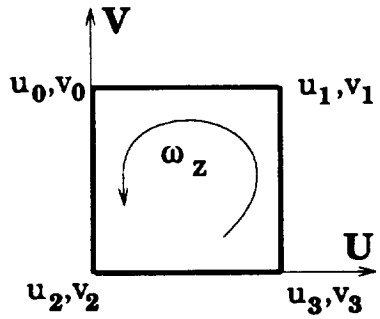

Figure 4: Window.

In this subsection we calculate the transformation that an object's projection undergoes as a result of platform maneuvers so as to relate it with the AFTR as defined above. To do that, we start from the well-known equations for the temporal derivatives of the image-plan projections (u, v). Repeated as in [21], and adapted to our earlier notation, we have

$$u_t = -fv_X + uv_Z + \omega_X \frac{uv}{f} - f\omega_Y(1 + \frac{u^2}{f^2}) + v\omega_Z$$

$$v_t = -fv_Y + vv_Z - \omega_Y \frac{uv}{f} + f\omega_X(1 + \frac{v^2}{f^2}) - u\omega_Z\,, \tag{11}$$

where f is the focal length. Now consider the shifts experienced by the corners of the window shown in figure 4. The differences between their shifts can be used to yield rotation and expan-

sion. The rotation of the upper side of the square (where $v_1 = v_0$) during some interframe time can be approximated by

$$\frac{\Delta v_1 - \Delta v_0}{u_1 - u_0} = -\omega_Z - \frac{v_0 \omega_Y}{f} \tag{12}$$

When the point (u_0, v_0) coincides with the FOE, this reduces to $-\omega_Z$. The rotation of the left side of the square (where $u_0 = u_2$) is similarly found as

$$\frac{\Delta u_2 - \Delta u_0}{v_0 - v_2} = -\omega_Z - \frac{u_0 \omega_X}{f}\,, \tag{13}$$

which also reduces to $-\omega_Z$ at the FOE. We have used the rotations of vertical and horizontal lines to show that, first, they rotate slightly differently, that is, in principle, the square distorts, and, second, this rotation approximately equals the platform roll. Comparing the two terms on the right of (12) for equal platform roll and yaw, the yaw term is smaller by a factor of f/v_0. At a distance of, say, 50 pixels from the FOE, and with f=622 (using our camera as an example), this factor is 12.4. Since the expansion-based algorithm suggested here is intended to mainly enhance depth derivation around the FOE, we conclude that image-plane rotation is reasonably approximated by platform roll.

Next, let us analyze the expansion factor. For the upper side of the square it is approximated by

$$\frac{\Delta u_1 - \Delta u_0}{u_1 - u_0} = v_Z + \frac{v_0 \omega_X}{f} - \frac{(u_0 + u_1)\omega_Y}{f} \tag{14}$$

and for the left side of the square by

$$\frac{\Delta v_0 - \Delta v_2}{v_0 - v_2} = v_Z - \frac{u_0 \omega_Y}{f} + \frac{(v_0 + v_2)\omega_X}{f}\,, \tag{15}$$

At the FOE, both expressions approach v_Z as the square size goes to zero. Again, horizontal and vertical lines expand slightly differently, but, to a good approximation, this expansion equals v_Z (the TTC). The superfluous terms are an order of magnitude smaller than v_Z for areas up to 50 pixels from the FOE and small angular speeds.

Our conclusion is that, over the expected range of flight scenarios, the affine transformation represents a good approximation to the actual mapping that is taking place between different frames. If this approximation is not adequate, one can always use the full 6 degrees of freedom available in the general affine transformation.

3.3 What happens when scaling and rotation are ignored

In this subsection we elaborate on the importance of using the AFTR even for an algorithm which calculates depth based on the shifts alone. Ignoring the AFTR amounts to taking it to be

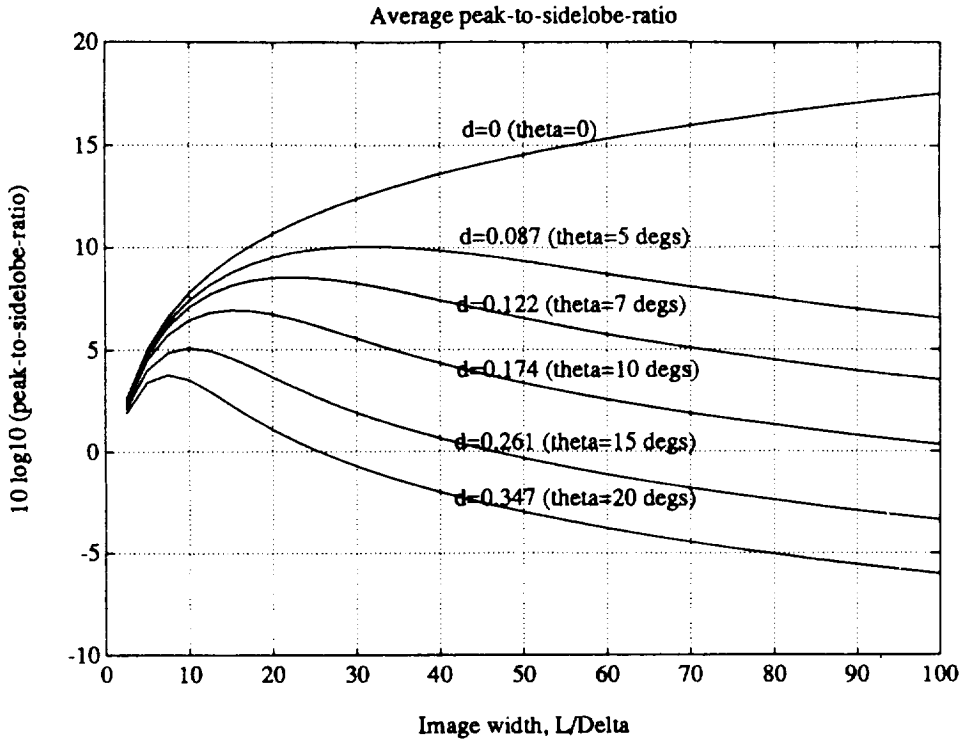

Figure 5: Average peak-to-sidelobes ratio as a function of image size for different distortions.

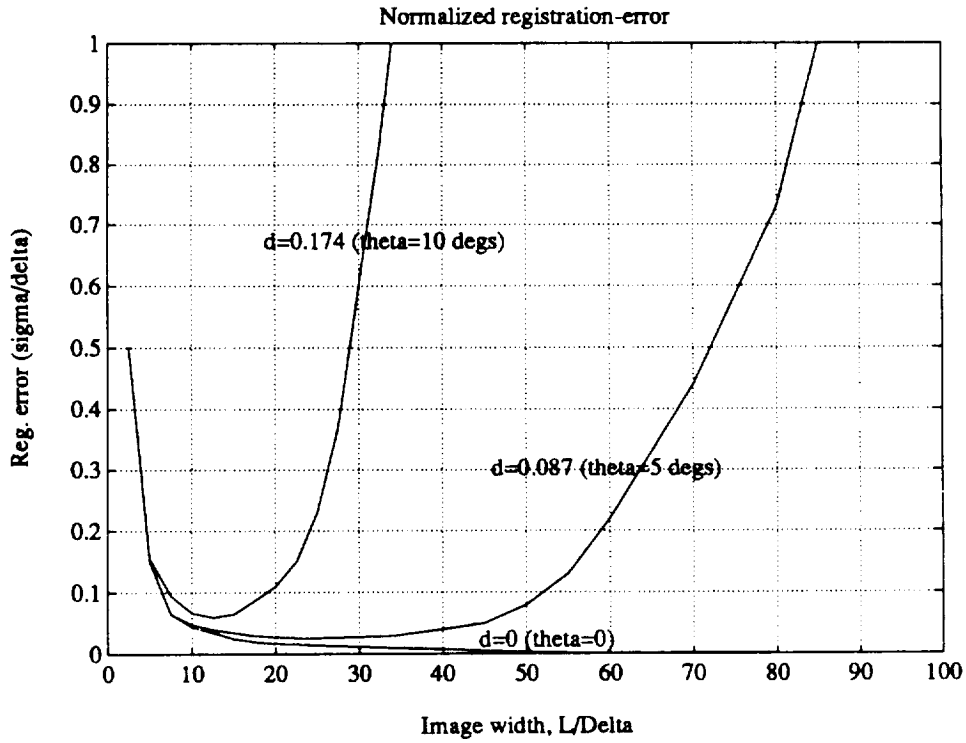

Figure 6: Registration-error standard deviation as a function of image size for different distortions.

a unity matrix. This question has been investigated quite extensively by Mostafavi and Smith in [31, 32]. For completeness, we summarize their results here.

For images having a circularly symmetric Gaussian correlation function,

$$R(\tau_u, \tau_v) = \exp\left\{-\frac{1}{2\Delta^2}[\tau_u^2 + \tau_v^2]\right\} , \tag{16}$$

where τ_u, τ_v are the spatial shifts, and Δ the "correlation width", the effects of non-compensated rotation (by θ) and/or scaling (by s) are determined by the combined geometric-distortion parameter d,

$$d \triangleq \sqrt{|\, 1 - 2s\cos\theta + s^2 \,|} \approx \sqrt{(1-s)^2 + \theta^2} \text{ for small } \theta \text{ and } s \approx 1 \tag{17}$$

Figure 5 shows the effect of d on the peak-to-sidelobes ratio (PSR). Peak stands for the maximum value of the cross-correlation function between two frames, and "sidelobes" stands for the standard deviation of the cross-correlation function far from the peak. The reference image is taken as a square of size $L \times L$. The other (sensed) image is much larger than the reference image. In the figure, L appears normalized by the correlation width because what counts is the effective number of "independent" image objects. Six graphs are shown for different d values. The graph for $d = 0.087$, for example, can be used for rotation alone (of 5^0), or for scaling alone ($s = 1.087$), or for any of their combinations such that (17) yields $d = 0.087$. Figure 6 similarly shows the behavior of the registration error.

Let us use the following example to demonstrate the effect of uncompensated rotation or scaling errors. Take speed $V_X = 25$ m/s, depth $z_0 = 120$ m, image-plane location 10 pixels from the FOE, a rolling maneuver of $\omega_Z = 20^0$/s, $L = 21$, $\Delta = 1.5$ pixels, and frame rate of 2 fr/s. This low frame rate is used to achieve a large triangulation baseline as will be explained later. Only two consecutive frames are used in this example.

In a single interframe time the platform rotates 10^0 and there is an expansion by a factor of $s = 120/(120 - 25 \cdot 0.5) = 1.1163$, so that $d = 0.21$. The PSR will incur a loss of ≈ 3 (6 dB in PSR power)—as read from figure 5. This is why, without using the AFTR, one needs to use a higher frame rate, say, 10 fr/s. The registration error, as extrapolated from figure 6, will increase from 0.025 to 0.070 pixels. In [12] we have found the depth error:

$$\sigma_z = \frac{\sqrt{2} z_0^2 \sigma_u}{bu} , \tag{18}$$

where b is the triangulation baseline. Thus, the depth error incurred by a geometrically-compensated algorithm ($b = 12.5$ m) is 4.1 m while that incurred by a non-compensated algorithm ($b = 2.5$ m) is 57 m (out of 120 m !).

This example shows that, even in the conventional shift-based algorithm, neglecting to compensate for the AFTR in the process of cross-correlation is costly in two ways. First, it

either degrades the PSR which may hinder locking onto the correct peak (false alarm) or impose a short b, and second, even when correct peak detection is achieved, the depth error would increase around tenfold.

3.4 Converging on the correct affine transformation

In this subsection we derive the equations and algorithm necessary to obtain the correct affine transformation. The basic idea is to use Newton's equation (see [33]) iteratively to converge from the initially-assumed transformation matrix into the correct one by minimizing an appropriate error measure, or cost-function.

We thus start by defining the cost-function, J, as the integral over the window area, A, of the squared difference of image gray levels, that is,

$$\epsilon \triangleq I_1(\hat{u},\hat{v}) - I_0(u,v)\,; \qquad J \triangleq \frac{1}{2A}\iint_A \epsilon^2 \mathrm{d}A \tag{19}$$

If all points (u,v) inside the window (defined in I_0) are correctly mapped into $(\hat{u},\hat{v})$ of I_1, then the above cost should equal zero. In practice, however, we can only expect to *minimize* this cost albeit not to drive it to zero. Our plan is to find the gradient and second derivatives of J so that we can use Newton's method to *solve* for the minimum assuming that the cost-function is quadratic in the four parameters to be estimated. Since this assumption only holds approximately, it is necessary, in practice, to iterate a few times until the solution converges. The iterative update equation for the estimated parameter vector $\hat{X}(k)$ becomes

$$\hat{X}(k+1) = \hat{X}(k) - \left\{\nabla^2 J[\hat{X}(k)]\right\}^{-1} \nabla J[\hat{X}(k)]\,, \tag{20}$$

where

$$X(k) \triangleq \begin{bmatrix} a \\ b \\ s \\ \theta \end{bmatrix} \tag{21}$$

The four components of the cost-function gradient are calculated next. Starting with the first shift-parameter, a,

$$\frac{\partial J}{\partial a} = \frac{1}{A}\iint_A \epsilon \frac{\partial \epsilon}{\partial a}\mathrm{d}A = \frac{1}{A}\iint_A \epsilon \frac{\partial I_1}{\partial a}\mathrm{d}A\,, \tag{22}$$

because only the $I_1(\hat{u},\hat{v})$ part of ϵ depends on a through $\hat{u},\hat{v}$. Developing that relationship,

$$\frac{\partial I_1}{\partial a} = \frac{\partial I_1}{\partial \hat{u}}\frac{\partial \hat{u}}{\partial a} + \frac{\partial I_1}{\partial \hat{v}}\frac{\partial \hat{v}}{\partial a} \tag{23}$$

Similar equations are obtained for the other three parameters by substituting them in place of a in (23). The above four equations require the partials of $\hat{u}, \hat{v}$ with respect to all four parameters. These are obtained by differentiating the two scalar equations obtained from (10), that is,

$$\begin{aligned} \hat{u} &= s[C\theta(u-u_0) - S\theta(v-v_0)] + u_0 + a \\ \hat{v} &= s[S\theta(u-u_0) + C\theta(v-v_0)] + v_0 + b\,, \end{aligned} \tag{24}$$

so that

$$\begin{aligned} \frac{\partial \hat{u}}{\partial a} &= 1\,; & \frac{\partial \hat{v}}{\partial a} &= 0 \\ \frac{\partial \hat{u}}{\partial b} &= 0\,; & \frac{\partial \hat{v}}{\partial b} &= 1 \\ \frac{\partial \hat{u}}{\partial s} &= C\theta(u-u_0) - S\theta(v-v_0)\,; & \frac{\partial \hat{v}}{\partial s} &= S\theta(u-u_0) + C\theta(v-v_0) \\ \frac{\partial \hat{u}}{\partial \theta} &= -s[S\theta(u-u_0) + C\theta(v-v_0)]\,; & \frac{\partial \hat{v}}{\partial \theta} &= s[C\theta(u-u_0) - S\theta(v-v_0)] \end{aligned} \tag{25}$$

We now need the ten second derivatives of the symmetrical matrix $\nabla^2 J[\hat{X}(k)]$. In order to simplify notation, we will drop the "dA" from the integrals, the subscript 1 from I, and the hats from u, v; these will now be understood whenever not specified. Let us start with one of the mixed second derivatives, say that of a and θ. We thus have

$$\frac{\partial^2 J}{\partial a \partial \theta} = \frac{\partial}{\partial a}\left(\frac{\partial J}{\partial \theta}\right) = \frac{1}{A}\iint_A \frac{\partial \epsilon}{\partial a}\frac{\partial \epsilon}{\partial \theta} + \epsilon \frac{\partial}{\partial a}\left[\frac{\partial I}{\partial u}\frac{\partial u}{\partial \theta} + \frac{\partial I}{\partial v}\frac{\partial v}{\partial \theta}\right] \tag{26}$$

After some more algebra, we get

$$\frac{\partial^2 J}{\partial a \partial \theta} = \frac{1}{A}\iint_A U\frac{\partial u}{\partial a}\frac{\partial u}{\partial \theta} + V\frac{\partial v}{\partial a}\frac{\partial v}{\partial \theta} + W\left[\frac{\partial u}{\partial a}\frac{\partial v}{\partial \theta} + \frac{\partial u}{\partial \theta}\frac{\partial v}{\partial a}\right] + \epsilon\left[\frac{\partial I}{\partial u}\frac{\partial^2 u}{\partial a \partial \theta} + \frac{\partial I}{\partial v}\frac{\partial^2 v}{\partial a \partial \theta}\right], \tag{27}$$

where

$$U \triangleq \left(\frac{\partial I}{\partial u}\right)^2 + \epsilon\frac{\partial^2 I}{\partial u^2}\,; \quad V \triangleq \left(\frac{\partial I}{\partial v}\right)^2 + \epsilon\frac{\partial^2 I}{\partial v^2}\,; \quad W \triangleq \epsilon\frac{\partial^2 I}{\partial u \partial v} + \frac{\partial I}{\partial u}\frac{\partial I}{\partial v} \tag{28}$$

The other mixed second derivatives of J are similar and can be obtained by substituting the other parameters in place of a and θ in (28). The second (non-mixed) derivatives can, of course, be obtained by substituting the same parameter twice. For example, the second derivative of J with respect to a is

$$\frac{\partial^2 J}{\partial a^2} = \frac{1}{A}\iint_A U\left(\frac{\partial u}{\partial a}\right)^2 + V\left(\frac{\partial v}{\partial a}\right)^2 + 2W\frac{\partial u}{\partial a}\frac{\partial v}{\partial a} + \epsilon\left[\frac{\partial I}{\partial u}\frac{\partial^2 u}{\partial a^2} + \frac{\partial I}{\partial v}\frac{\partial^2 v}{\partial a^2}\right] \tag{29}$$

Notice that the above equations require two kinds of building blocks; these are the first and second (also mixed) spatial derivatives of the I_1 image as well as the first and second (also mixed)

derivatives of $\hat{u}$ and $\hat{v}$ with respect to the four transformation parameters. The image spatial derivatives are calculated by convolving it with a simple Sobel-operator-type 3×3 window. The first derivatives of $\hat{u}$ and $\hat{v}$ where already calculated for the gradient as in (25). Differentiating (25) once again yields 10 second derivatives for $\hat{u}$ and 10 for $\hat{v}$. Out of these 20, all turn out to be zero except the following four:

$$\frac{\partial^2 u}{\partial s \partial \theta} = -S\theta(u-u_0) - C\theta(v-v_0) = -\frac{\partial v}{\partial s}; \quad \frac{\partial^2 v}{\partial s \partial \theta} = C\theta(u-u_0) - S\theta(v-v_0) = \frac{\partial u}{\partial s}$$

$$\frac{\partial^2 u}{\partial \theta^2} = s[-C\theta(u-u_0) + S\theta(v-v_0)] = -\frac{\partial v}{\partial \theta}; \quad \frac{\partial^2 v}{\partial \theta^2} = -s[S\theta(u-u_0) + C\theta(v-v_0)] = \frac{\partial u}{\partial \theta} \quad (30)$$

At this point all the components necessary for a single iteration on the Newton's solution have been derived.

4 SIMULATIONS OF THE COST-FUNCTION AND ITS DERIVATIVES

We now want to examine the behavior of the cost-function and its derivatives as a function of the four parameters in open loop, that is, without trying to correct the errors yet. For the following experimental results we used simulated imagery where the scene is composed of a wall normal to the initial flight trajectory. This wall is painted with a random Gaussian colored noise having spatial correlation width of 2 pixels in each of the two spatial dimensions. In this section we describe the main features of our Flight/Vision simulation and the open-loop error measurements.

4.1 Flight/Vision simulation

We have developed a simple simulation that enables us to generate a sequence of images (imagery) as obtained from an optical sensor that travels and maneuvers as prescribed. This simulation is described here.

The scenery is composed of a flat wall oriented normal to the initial LOS. The gray levels of the wall are derived by passing a white Gaussian noise through a two-dimensional Gaussian-shaped low-pass filter of some desired width. The wall is densely sampled by "wall-pixels" which, when imaged onto the camera's focal plane, are much finer than the "chip-pixels" of the camera. Typically 25 wall-pixels fall inside a single chip-pixel at the beginning of the run; this is chosen so that the wall can approximately be considered continuous. The correlation width of the low-pass filter above is chosen in terms of equivalent chip-pixels. In all simulations described later we chose correlation width of 2 chip-pixels because that is a typical width for the lens'

point-spread-function. The number of wall-pixels impinging on each chip-pixel is proportional to the depth squared because the camera uses a fixed angular field of view. Thus, to maintain a constant wall brightness on the image plane, we have to factor the wall brightness (or gray-level) by the depths-squared inverse. This compensation is nothing more than simulating the dependence of light radiation (power-per-area) on the inverse of the range squared.

The camera is initially located across from (and pointing to) the wall center at a distance of z_0 m. It is generally flying towards the wall center and can perform any desired maneuvers on its way. Each ray from the center of a wall-pixel to the camera's focal point (in world coordinates) gets transformed into the camera's coordinates through the 3×3 rotation matrices corresponding to yaw, pitch, and roll (*e.g.*, see [34]). The camera coordinates of the ray are used in the projection equations to yield the image coordinates of the ray's piercing point, that is,

$$u = f\frac{x}{z}\,; \qquad v = f\frac{y}{z}\,, \tag{31}$$

We now assume a point-spread-function (PSF), having the shape of a chip-pixel and centered on the (non-integer) (u, v) point, to impinge upon the grid of chip pixels. This is where interpolation becomes necessary.

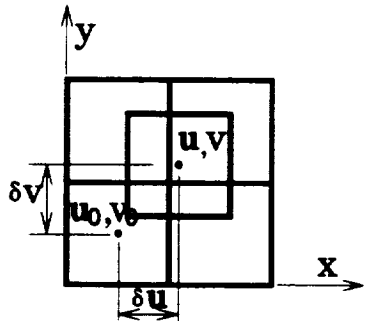

Figure 7: The interpolation method.

The method of interpolation is explained with the help of figure 7. The (u, v) point falls at a distance of $(\delta u, \delta v)$ from some integer point (u_0, v_0). We thus assign the PSF areas intersected by each of the 4 chip-pixels to these pixels. The corresponding areas are thus assigned as follows:

$$\begin{aligned}
(1-\delta u)(1-\delta v) &\qquad \text{to pixel } (u_0, v_0)\,; \\
(1-\delta u)\delta v &\qquad \text{to pixel } (u_0, v_0+1)\,; \\
\delta u(1-\delta v) &\qquad \text{to pixel } (u_0+1, v_0)\,; \\
\delta u \delta v &\qquad \text{to pixel } (u_0+1, v_0+1)\,,
\end{aligned} \tag{32}$$

where $(\delta u, \delta v)$ are derived as

$$\delta u = u - \text{int}(u)\,; \qquad \delta v = v - \text{int}(v)\,, \tag{33}$$

and int(·) is a function that rounds off its argument to the nearest lower integer.

As we have said above, there are, around 25 such partial contributions into every chip pixel—each contribution resulting from the center of a wall-pixel being projected onto some different (u, v) point. We have found that choosing the ratio between the sides of a chip-pixel and a wall-pixel to equal 5, using a chip-pixel-size PSF, and interpolating as prescribed above, results in a realistically-appearing textural behavior of the wall as it gets closer to the camera during the simulated flight. Examples showing the time evolution of the imaged wall will be shown in the sequel.

4.2 Simulation of the error equations

The error equations are, in principle, simulated as prescribed by equations (19) to (30). However, since we are dealing with a spatially-discretized images, it is necessary to implement these equations in a discrete form as well. There are no conceptual problems associated with replacing integrals by summations. However, all we know about the real physical image values comes from the pixels' gray-level data. It is important to understand that the gray-level of a pixel represents the value of a double integral over its area (average), where the spatially-continuous radiation emanating from the scene serves as the integrand. Another way to put it is that each pixel collects all the photons impinging anywhere within its boundaries during its integration time (interframe time).

Differentiating between a pixel's gray-level and the actual value of the scene at any (continuous) location on the image plan is important in estimating the scene values $I_1(\hat{u}, \hat{v})$ as required in (19) because $(\hat{u}, \hat{v})$ are generally non-integers. There is no such problem in estimating $I_0(u, v)$ because, by definition, we start from the pixel's center (integer) and thus take its gray-level as the best estimate of the scene value at this pixel's center. For the estimation of $I_1(\hat{u}, \hat{v})$, we use an interpolation method that looks identical to the one used for the imagery generation, although it is conceptually completely different.

Referring once more to figure 7, here is the problem. Say we have an estimate for the value of the scene at the center point of some initial pixel, that is, we have $I_0(u_0, v_0)$. This point has been mapped into location $(\hat{u}, \hat{v})$ in image I_1, and we want to estimate $I_1(\hat{u}, \hat{v})$. The relevant information available from image I_1 is its pixel values for the four pixels shown in the figure because these are directly affected by the original scenery patch (of pixel size). We can think of the value of each such pixel as a random variable crosscorrelated with $I_0(u_0, v_0)$ in proportion with the intersected areas as defined by (32). This led us to use the rather ad hoc interpolation method:

$$\begin{aligned} I_1(\hat{u}, \hat{v}) \triangleq\ & (1 - \delta u)(1 - \delta v) I_1(\hat{u}_0, \hat{v}_0) + \delta v (1 - \delta u) I_1(\hat{u}_0, \hat{v}_0 + 1) \\ & + \delta u (1 - \delta v) I_1(\hat{u}_0 + 1, \hat{v}_0) + \delta u \delta v I_1(\hat{u}_0 + 1, \hat{v}_0 + 1) \end{aligned} \tag{34}$$

This method has the advantage that it yields the expected results when $(\hat{u}, \hat{v})$ take on integer values, and it provides a continuous estimate inside the convex hull defined by the values of the four nearest pixels. The same interpolation method is used for estimating the image values as well as their first and second derivatives.

4.3 Open-loop error measurements

In the first set of open-loop error simulations we investigated the error sensitivity to the scaling factor s in isolation as a function of window size . The flight trajectory used for this set is non-maneuvering and constant-velocity towards the center of the wall starting from a depth of 150 m at a speed of 1 m/fr. The set of 3 images (number 0, 12, and 24) are shown in figure 8 to demonstrate the effect of expansion as the depth decreases from 150 to 138 to 126 m. Figure 9 shows the case of a 11×11 window size which is centered on the FOE. The first and fifth frames are used for I_0 and I_1 respectively so that the baseline is $b = 4$ m. The figure shows four curves. Three curves belong to the cost-function and its first and second derivatives as derived in the previous section. The fourth curve shows the correction for s as calculated by the Newton's algorithm of (20), that is, the third component of $\left\{\nabla^2 J[\hat{X}(k)]\right\}^{-1} \nabla J[\hat{X}(k)]$. The four graphs in each figure are scaled as necessary for convenient presentation. Figure 10 and figure 11 only differ from figure 9 by the window size as indicated in their titles. Figures 12 and 13 represent contraction—as opposed to expansion—and they serve to verify symmetry in comparison with figures 10 and 11 respectively.

The following observations are noteworthy.

1. The absolute values of all four variables increase monotonically with the window size. The reason is that, since the free variable is an expansion factor, it causes each pixel of the window to shift in linear proportional to its distance from the center of the window. Thus, the larger the window, the larger are the shift errors experienced by its pixels.

2. The values of the cost-function and its first and second derivatives roughly agree; this is not obvious because each derivative is obtained directly from the corresponding image derivatives. Low-pass-filtering of the image derivatives and the fact that we deal with discrete pixel values and have to resort to interpolation, can account for the numerical disparities.

3. The actual value of s, to be denoted s_a, is shown by the vertical bars in all figures. It is noticed that, in all 5 cases it falls closer to the minima of the cost-functions than to the zero crossings of the first derivatives. We do not have a satisfactory explanation for this behavior except to assume that these are noise-like inaccuracies resulting from the quantization and interpolation operations; they clearly diminish as the window becomes larger. It warrants commenting here that it is the zero crossing of the derivative which

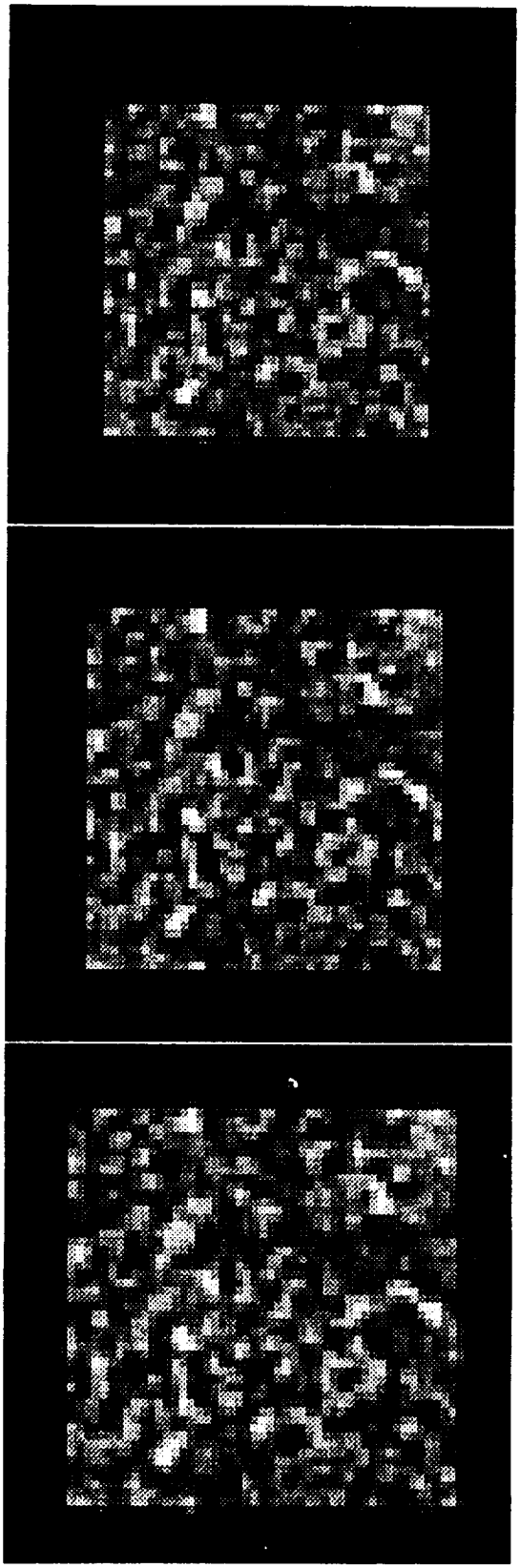

Figure 8: Frames 0, 12, and 24 of simulated textured wall seen while flying forward.

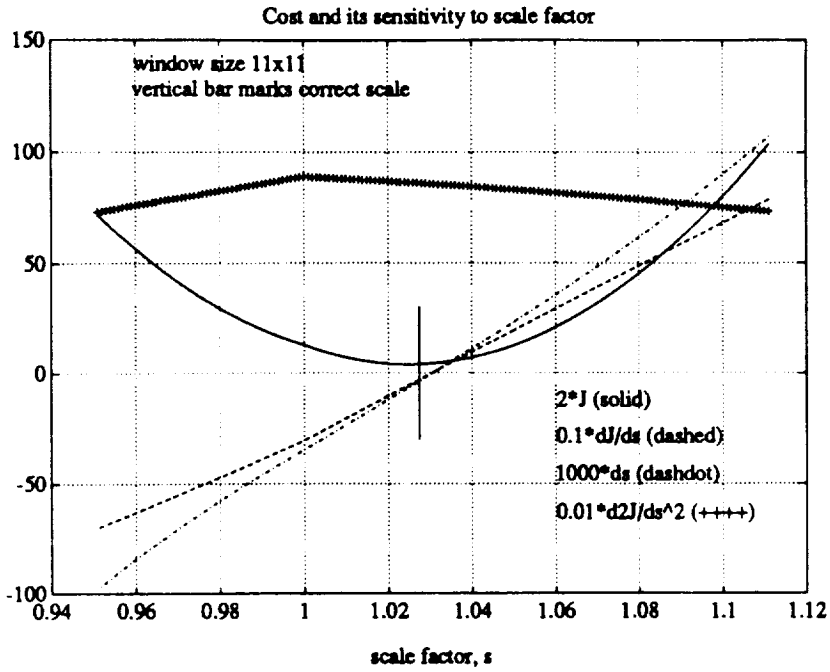

Figure 9: Sensitivity of the cost-function and its derivatives to the scale factor (11×11 window).

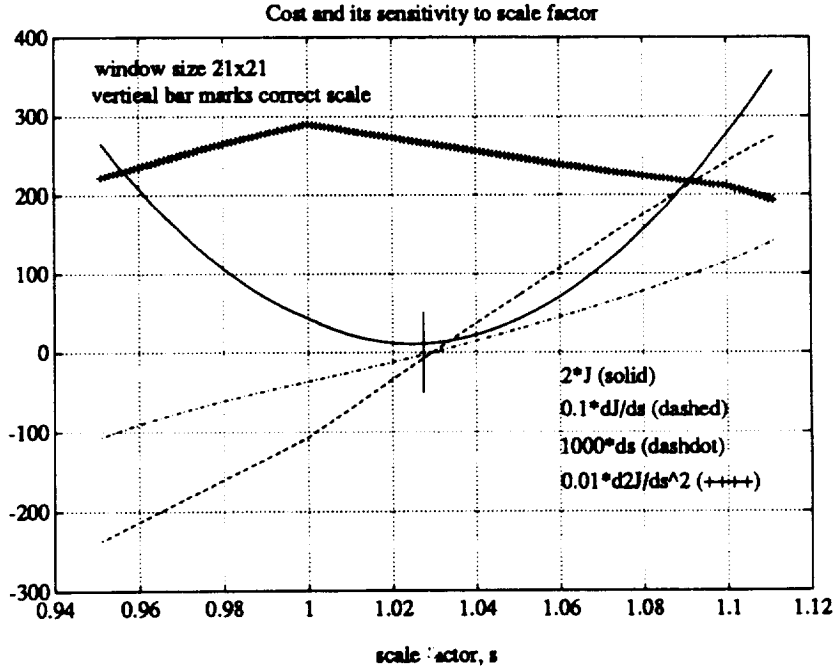

Figure 10: Sensitivity of the cost-function and its derivatives to the scale factor (21×21 window).

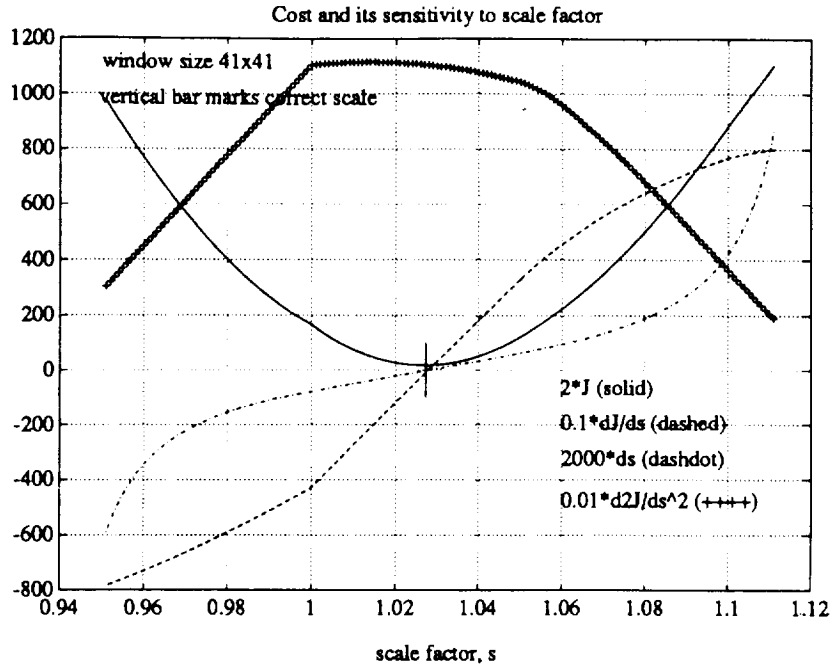

Figure 11: Sensitivity of the cost-function and its derivatives to the scale factor (41×41 window).

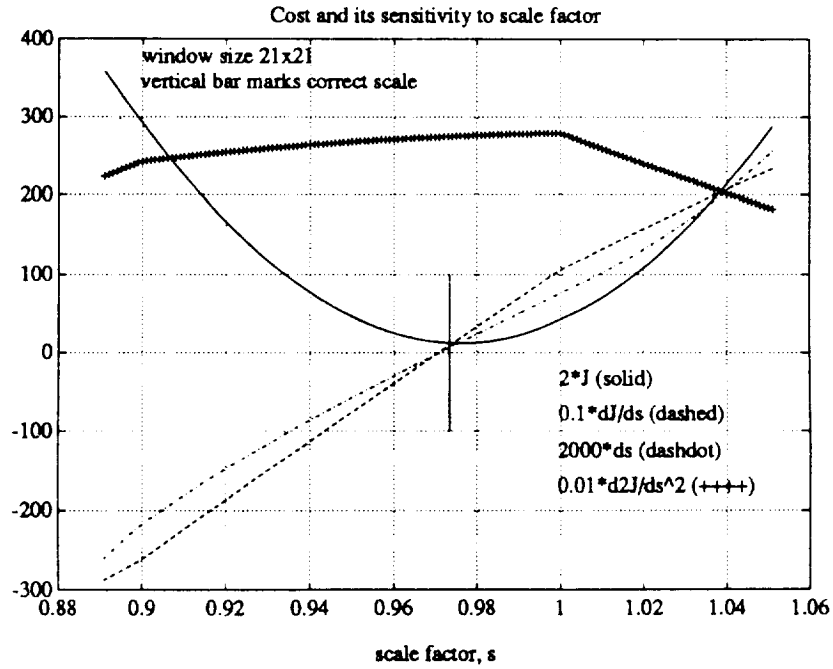

Figure 12: Sensitivity of the cost-function and its derivatives to the scale factor (21×21 window).

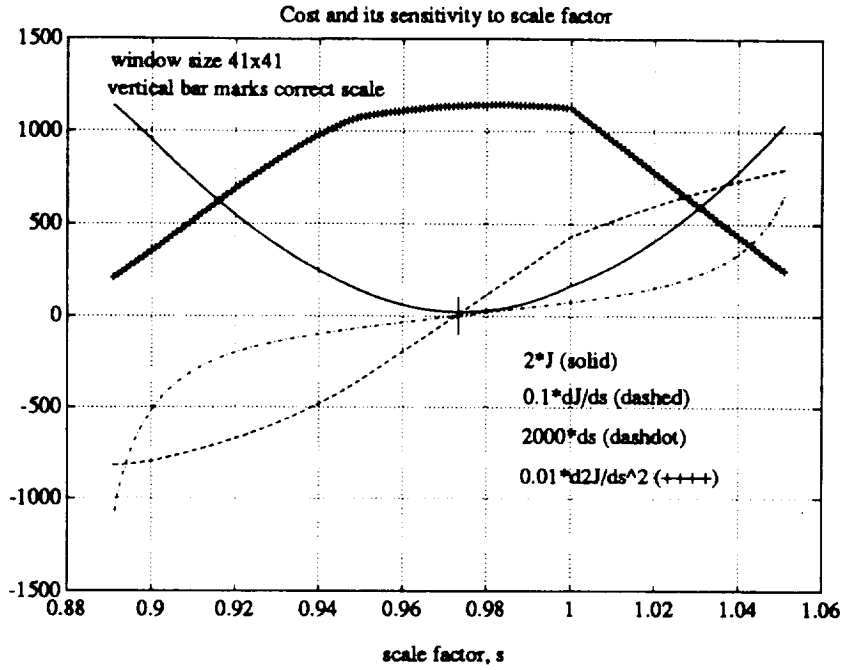

Figure 13: Sensitivity of the cost-function and its derivatives to the scale factor (41×41 window).

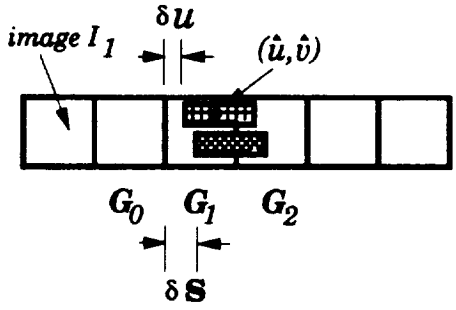

Figure 14: Interpolation example.

matters and not the minimum of the cost-function because that is where the closed-loop system would converge to.

4. The second derivative shows a sharp slope change at $s = 1$; the first derivative and the cost-function itself show corresponding behavior. The reason for that is explained by analyzing our interpolation method as shown in figure 14 for a simple one-dimensional case. The black dot represents the center point, $(\hat{u}, \hat{v})$, of one of the I_0 pixels that got shifted—as a result of expansion by some factor $s > 1$—to its new location in image I_1. The rectangle centered on the dot represents the original I_0 pixel. This new location is shifted by δu with respect to where it would fall if s equaled unity. Let us take the gray-level of this particular I_0 pixel as unity with all its neighbors being zeroes. This pixel will cause the gray-levels of image I_1 to become

$$G_0 = 0\ ; \qquad G_1 = (1 - \delta u)\ ; \qquad G_2 = \delta u \tag{35}$$

In order to generate the error curves, we sweep the value of s over some range around $s = 1$. The lower rectangle in the figure represents the location of the corresponding swept pixel for some $s > 1$ (denoted by s_s) which is different from the actual s_a. This swept pixel is shown shifted by δs. Interpolating for the current value of $s = s_s$, we have

$$I_1(\hat{u}, \hat{v}) = G_1(1 - \delta s) + G_2 \delta s = (1 - \delta u)(1 - \delta s) + \delta u \delta s = 1 - \delta s - \delta u + 2\delta u \delta s \tag{36}$$

When s sweeps through values less than unity, *i.e.*, $s_s < 1$, we have

$$I_1(\hat{u}, \hat{v}) = G_1(1 - \mid \delta s \mid) + G_0 \mid \delta s \mid = G_1(1 - \mid \delta s \mid) = (1 - \delta u)(1 - \mid \delta s \mid)\ , \tag{37}$$

which is always less than the corresponding result for a positive δs.

We thus conclude that, for an expansion, when the actual s is larger than 1, sweeping s_s over values of $s_s > 1$ always results in $I_1(\hat{u}, \hat{v})$ larger than those resulting from symmetrical (around $s = 1$) values of $s_s < 1$. This effect becomes more pronounced as the window size increases because the window pixels are, on the average, farther from its center and they experience larger δu shifts. When the actual s is smaller than 1, we see the opposite behavior as exemplified by figures 12 and 13. In these, the actual s is $s_a = 146/150 = 0.9733$. It is important to realize that, since the closed-loop algorithm performs around s_a and not around $s = 1$, it is not affected by the above phenomenon.

5. The curves of ds give the calculated correction for the case where the error occurs (through sweeping) in s alone. In such a case, the correction part of equation (20) simplifies to

$$\mathrm{d}s = \frac{\mathrm{d}J/\mathrm{d}s}{\mathrm{d}^2J/\mathrm{d}s^2} \tag{38}$$

It can be seen from the figures that ds approximately agrees with this equation. Also, the discontinuities in the first and second derivatives at $s = 1$ cancel each other in (38) so that the ds graphs do not show any discontinuity.

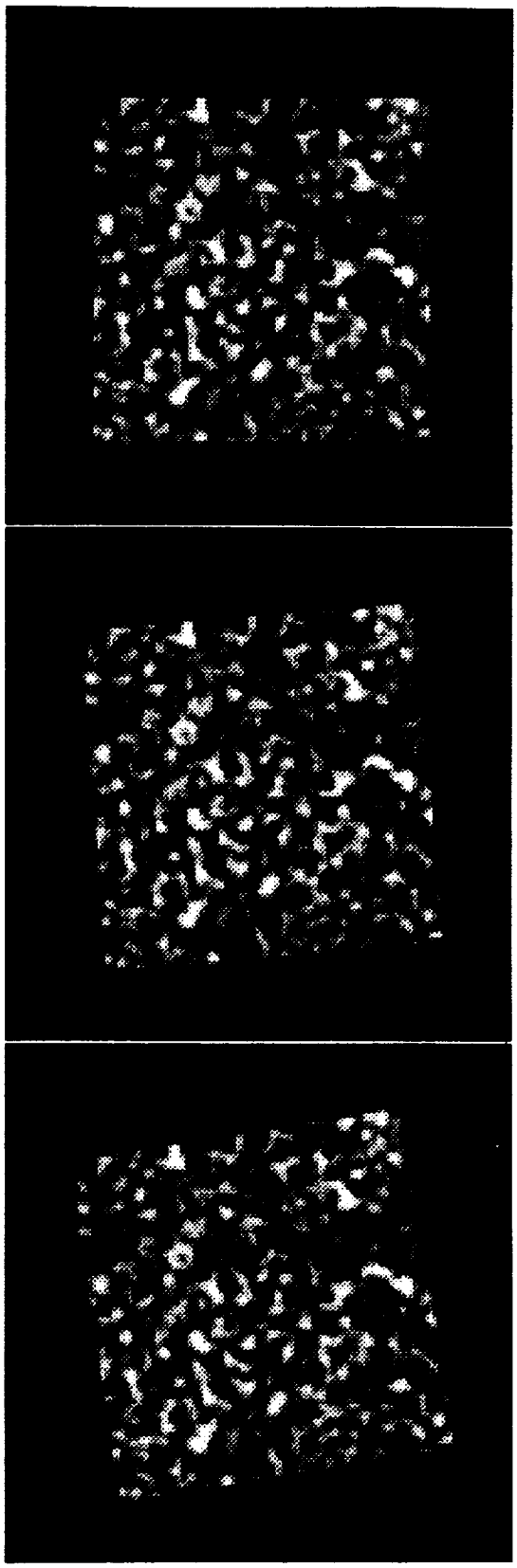

Figure 15: Frames 0, 4, and 8 of simulated textured wall as seen while rolling with no lateral motion.

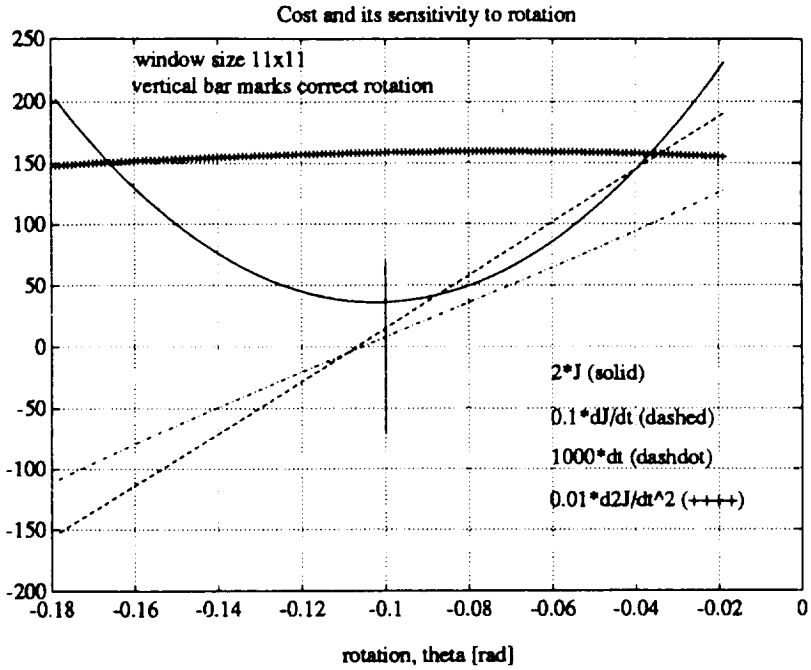

Figure 16: Sensitivity of the cost-function and its derivatives to rotation (11×11 window).

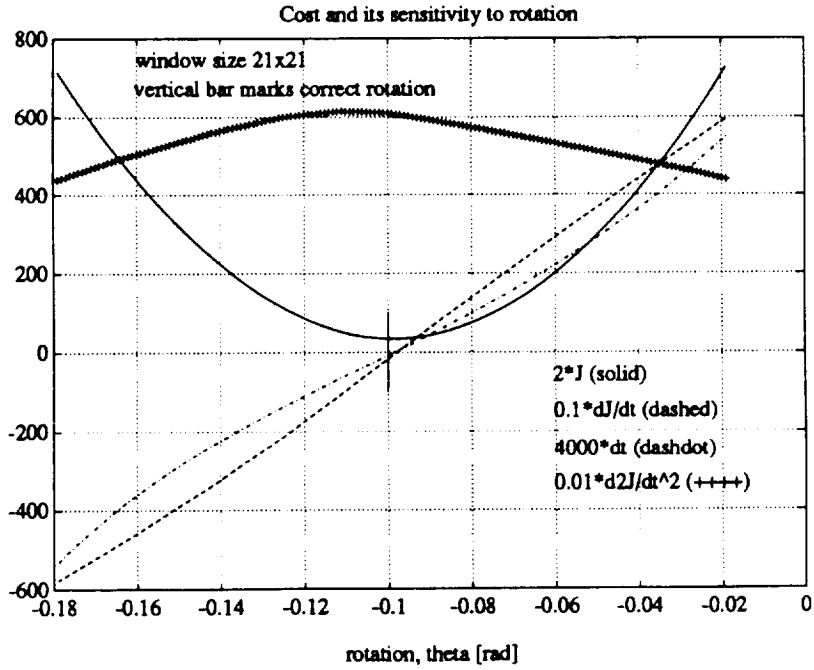

Figure 17: Sensitivity of the cost-function and its derivatives to rotation (21×21 window).

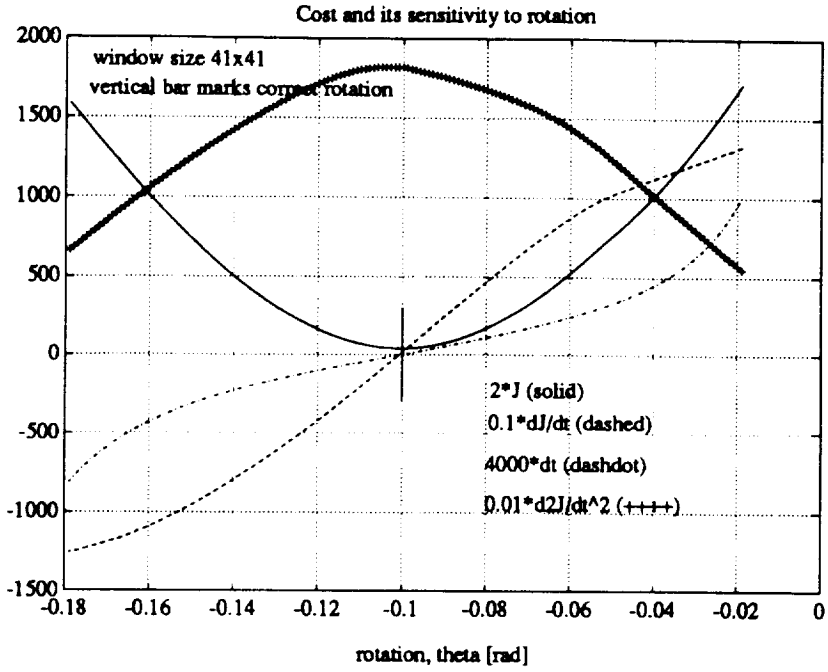

Figure 18: Sensitivity of the cost-function and its derivatives to rotation (41 × 41 window).

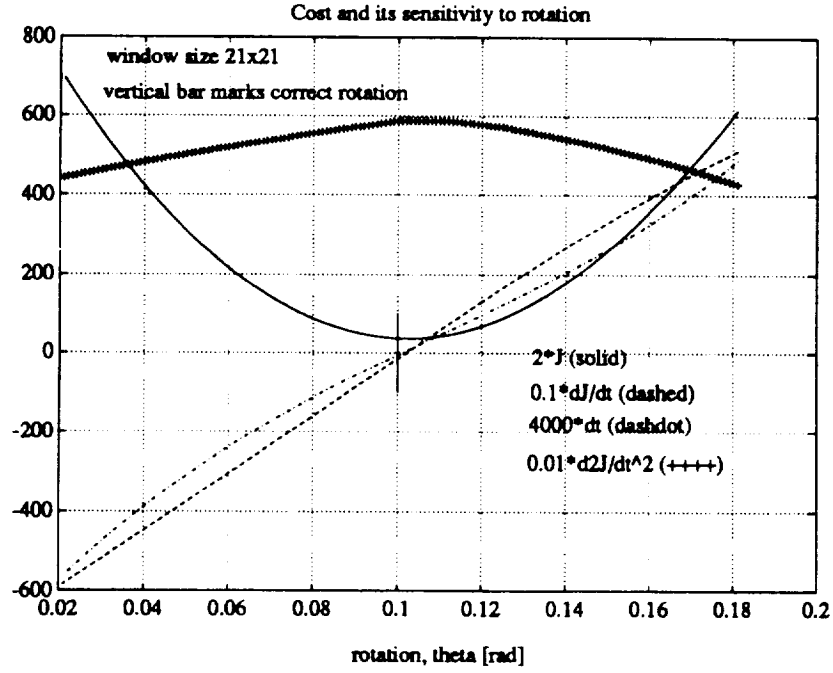

Figure 19: Sensitivity of the cost-function and its derivatives to rotation, 21 × 21 window, positive rotation.

In the next set of error measurements we investigated the error sensitivity to rotation angle θ in isolation as a function of window size. For this set the camera does not travel laterally; it only rolls at -0.02 rad/fr while pointing towards the center of the wall from a constant depth of 150 m. The set of 3 images (number 0, 4, and 8) are shown in figure 15 to demonstrate the effect of rotation. Figure 16 shows the case of a 11×11 window size which is centered on the FOE. Figures 17 and 18 correspond to windows of size 21 and 41 respectively. The first and sixth frames are used for I_0 and I_1 respectively so that the total roll used in generating the first 3 figures is of -0.1 rad. Figure 19 shows a roll in the opposite direction for a symmetry check. The same four curves as before are shown in all figures.

The following observations can be made.

1. The absolute values of all four variables increase monotonically with the window size. The reason here is the same that applied to the scaling-only cases. The larger the window the larger the shifts experienced by pixels which are farther from the window center.

2. The values of the cost-function and its first and second derivatives roughly agree as for the s curves.

3. The actual value of θ is shown by the vertical bars in the figures. It is noticed that the bars fall close to the minima of the cost-functions and also to the zero crossings of the first derivatives. The larger the window, the more accurate these results are.

4. There are no marked discontinuities as found in the s curves because the reason that caused it there does not apply here.

5. The curves of $\mathrm{d}\theta$ give the calculated correction for the case where the error occurs (through sweeping) in θ alone. In such a case equation (20) simplifies to

$$\mathrm{d}\theta = \frac{\mathrm{d}J/\mathrm{d}\theta}{\mathrm{d}^2J/\mathrm{d}\theta^2} \tag{39}$$

   In the figures $\mathrm{d}\theta$ approximately agrees with this equation.

In the next set of error measurements we investigated the error sensitivity to image-plane shifts, a, in isolation as a function of window size. For this set the camera is stationary except that it is panning at 0.0005 rad/f while pointing towards the center of the wall from a constant depth of 150 m. Images numbers 0 and 4 are used for I_0 and I_1 respectively. The panned images are not shown because they look quite indistinguishable—being shifted only by about a pixel. Figure 20 shows the case of a 21×21-size window (top) and 41×41-size window (bottom) when both are centered on the FOE. The following observations can be made.

1. As opposed to the previous cases, where s or θ served to generate the errors, here there is very little sensitivity to the window size because the shifts are equal for all pixels within the window of any size.

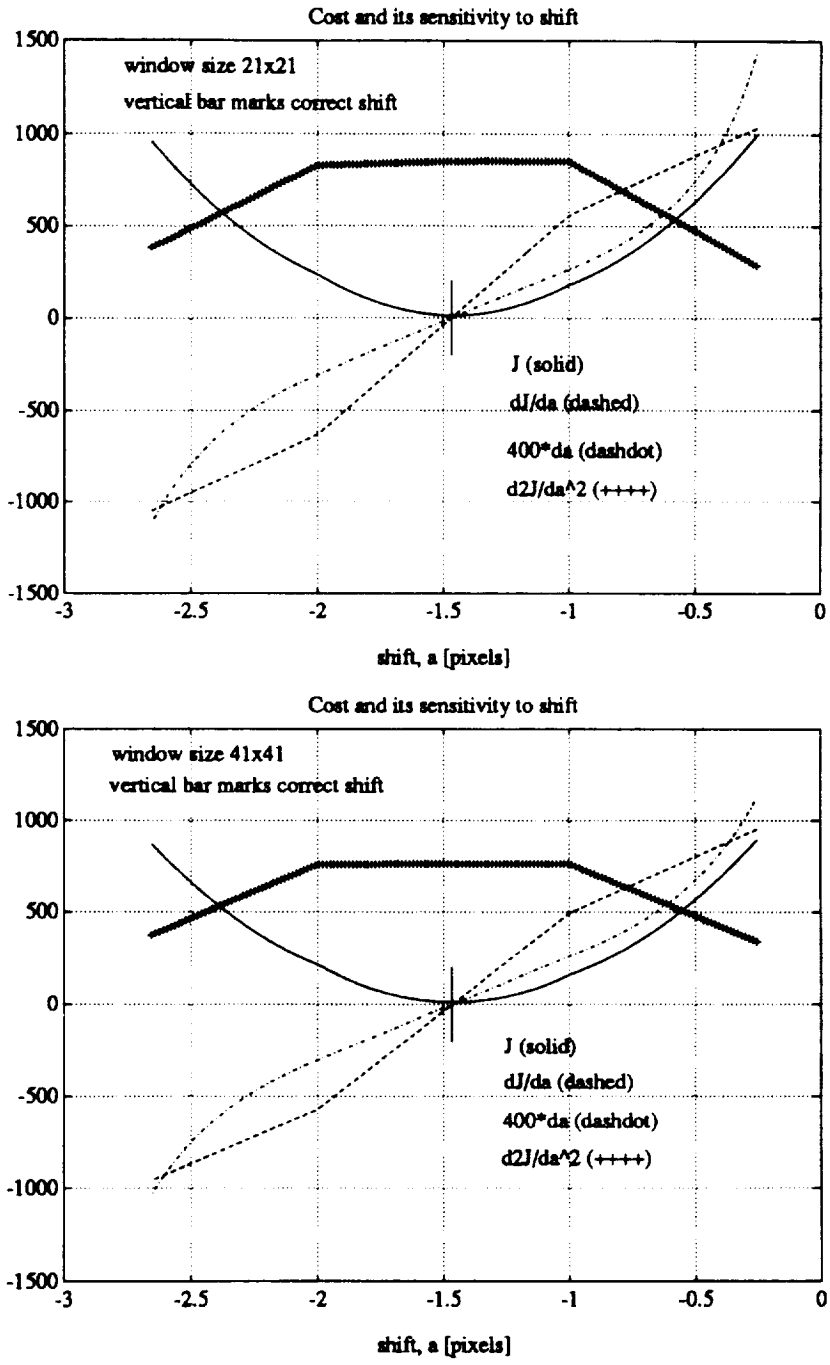

Figure 20: Sensitivity of the cost-function and its derivatives to shift; $L = 21$ (top), $L = 41$ (bot).

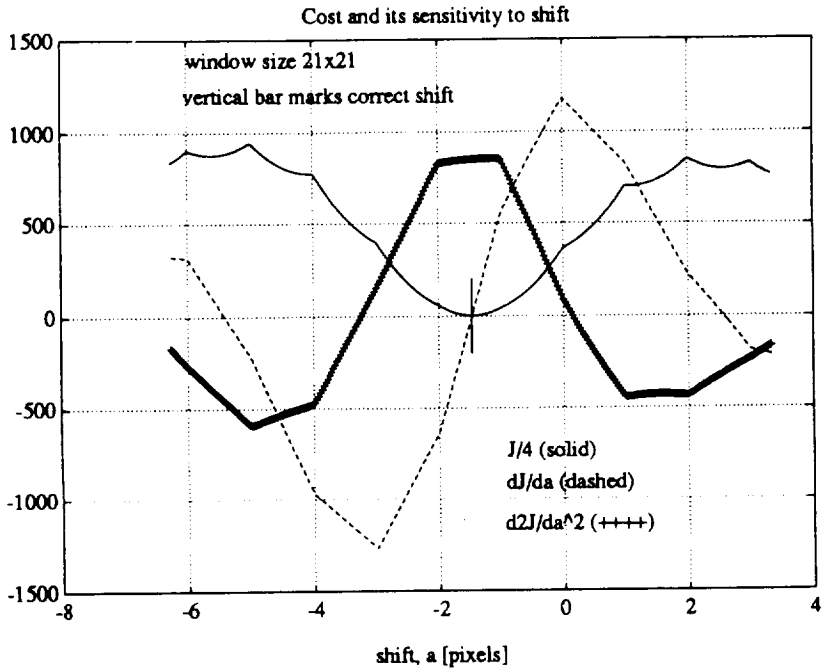

Figure 21: Sensitivity of the cost-function and its derivatives to shift over a wide range (21×21 window).

2. The actual value of a is marked by the vertical bars in all figures. These bars fall close to the minima of the cost-functions and also to the zero crossings of the first derivatives. As before, the larger the window, the more accurate these results are.

3. The second-derivative discontinuities at the integer pixel shifts can be explained by arguments similar to those used in the case of the s curves.

4. The curves of da give the calculated correction for the case where the error occurs (through sweeping) in a (or b) alone. In such a case, the correction part of equation (20) simplifies to

$$\mathrm{d}a = \frac{\mathrm{d}J/\mathrm{d}a}{\mathrm{d}^2J/\mathrm{d}a^2} \tag{40}$$

   In the figures, da approximately agrees with this equation.

5. Figure 21 shows the behavior of the cost-function curve for large shifts—where it becomes highly non-linear. The Newton's solution loses much of its value at such large errors. However, convergence is still possible inside the error region defined by the nearest zero-crossing of the first derivative on either side of the zero-error point (± 4 pixels here). Inside this region the correction still shows the right sign.

4.4 Closed-loop performance

In this subsection we summarize the results of closed-loop runs. These runs are divided into four groups. The first three groups parallel the open-loop cases of forward-flying, rolling, and panning (yaw). In the fourth run there are maneuvers in all variables so we could test the most general case. Within each group there are two kinds of parameters. One parameter is the window size, and the other is the location of the window with respect to the FOE.

In each run the errors are corrected using the Newton's method for six iterations. Theoretically, Newton's method should "converge" in one shot for any ideal parabolic cost-function. We allow for discrepancies from the ideal by (1) iterating on the solution more than once, (2) factoring the corrections by an experimental factor of 0.75 to prevent overshoots, and then, (3) bounding δs by ± 0.03, $\delta\theta$ by ± 0.03 rad, and δa, δb by ± 0.75 pixels.

Each of the graphical results for all runs include five curves to show the convergence of the cost-function, J, and the four parameters: s, θ, a, and b. In addition, there are four bars (arbitrarily located between iteration number 4 and 5) whose ordinates show the ground-truth values of the four parameters for ready visual comparison. The bars are marked by the parameter symbols.

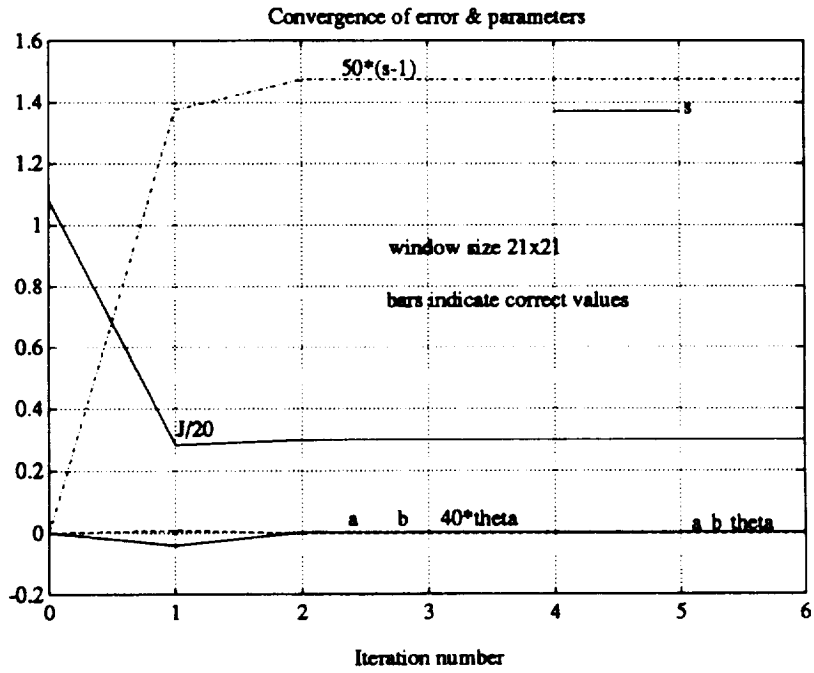

Figure 22: Convergence for forward flying and no maneuvers at the FOE (21 × 21 window).

Let us start with the results for forward-flying with no maneuvers. The initial depth is 150 m and the velocity is 1 m/fr towards the center of the wall. The transformation parameters are calculated at the time of frame number 4 by comparing it to frame number 0 (skipping the

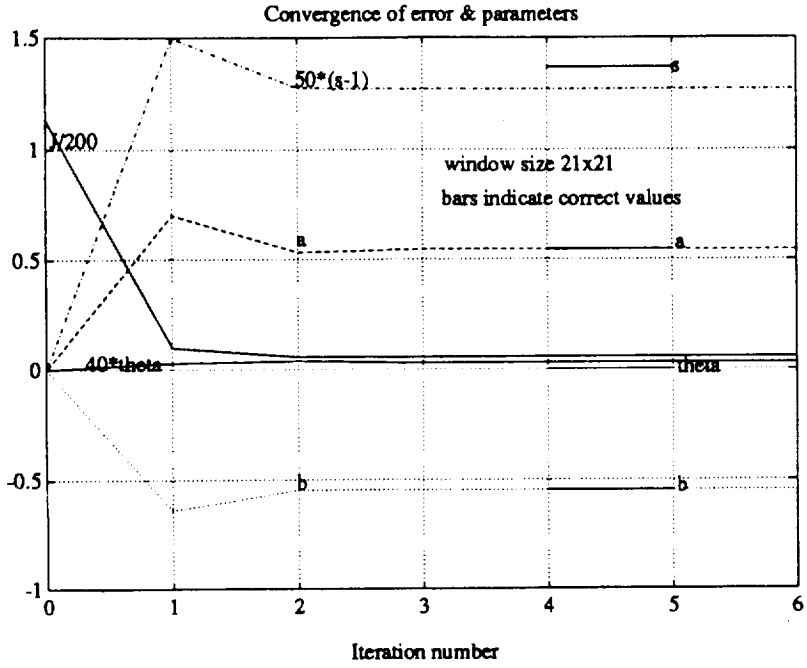

Figure 23: Convergence for forward flying and no maneuvers at (20,20) from the FOE (21 × 21 window).

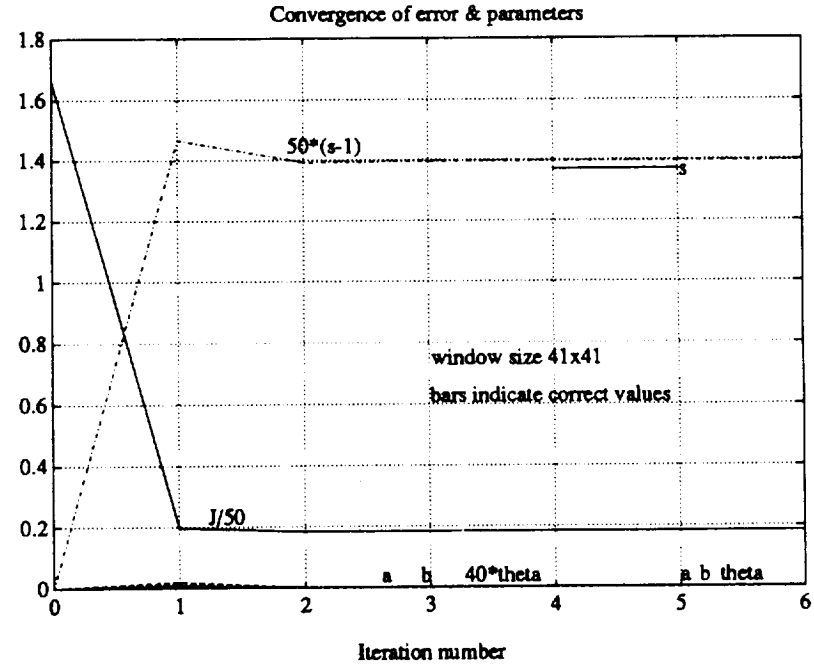

Figure 24: Convergence for forward flying and no maneuvers at the FOE (41 × 41 window).

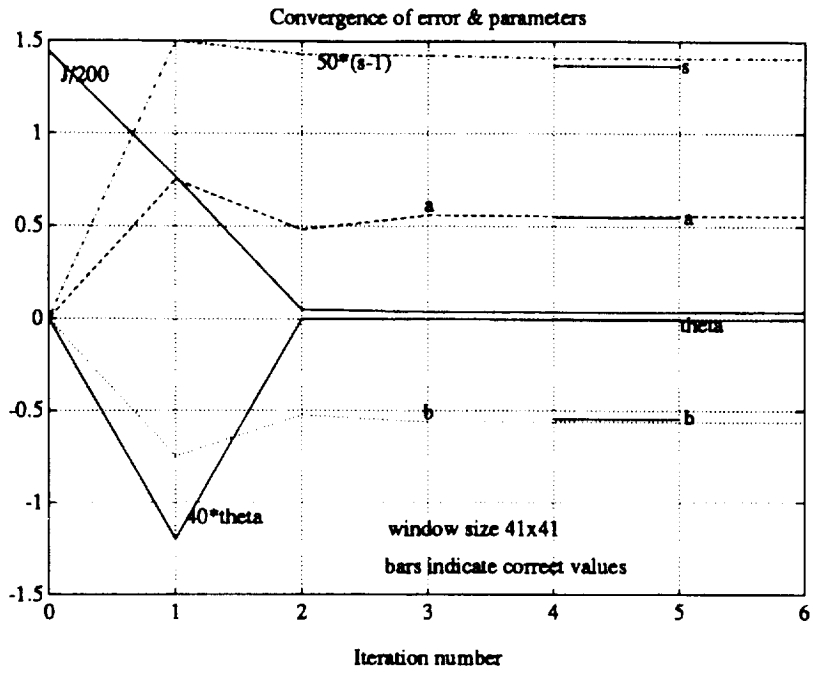

Figure 25: Convergence for forward flying and no maneuvers at (20,20) from the FOE (41×41 window).

intermediate frames). These runs are intended to demonstrate expansion alone for a window centered on the FOE, and expansion-plus-shift for a window centered on the point (20,20) with respect to the FOE. The following observations can be made:

1. The cost-function and all parameters practically converge in two iterations. When no parameter correction hits its bounds, convergence is achieved in a single iteration.

2. The accuracies—especially for s—improve noticeably as the window size doubles (4 times the window area), but they are still very good for the 21×21-size window. For example, from figure 22, the correct expansion (indicated by the s bar) is 150/146=1.0274, which corresponds to 146 frames-to-collision, whereas the converged value is $s = 1.0296$ which corresponds to 135 frames-to-collision.

3. The converged shifts for the (20,20) point practically show no error. This is especially impressive because these shifts are small—only (0.548,0.548) pixels.

Next, we present the results for roll-only flying without any forward or lateral motion. The depth is constant at 150 m. The transformation parameters are calculated at the time of frame number 2 by comparing it with frame number 0. the roll-angle difference is 0.04 rad between these two frames. In these runs we demonstrate rotation alone for a window centered on the FOE, and rotation-plus-shift for a window centered on the point (20,20) with respect to the FOE. The following observations can be made:

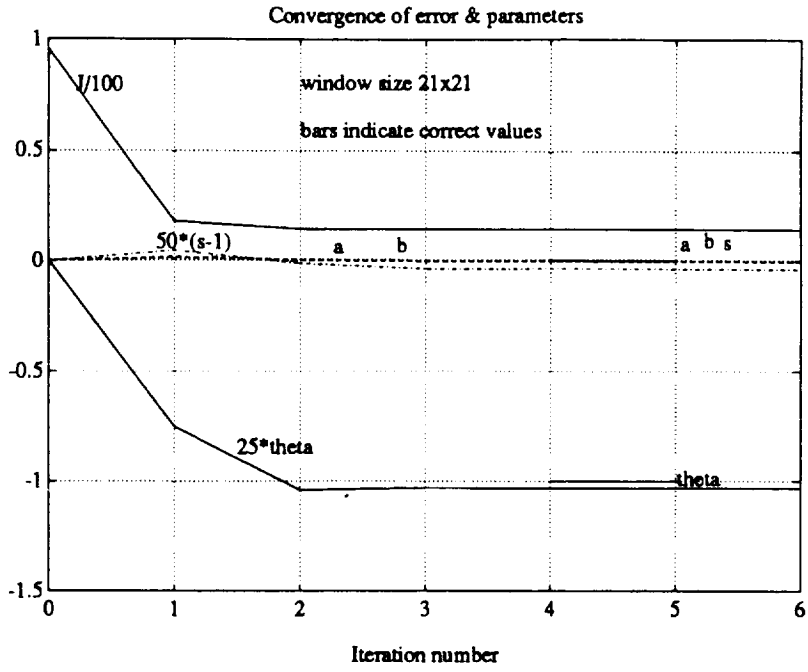

Figure 26: Convergence for roll-only maneuver at the FOE (21 × 21 window).

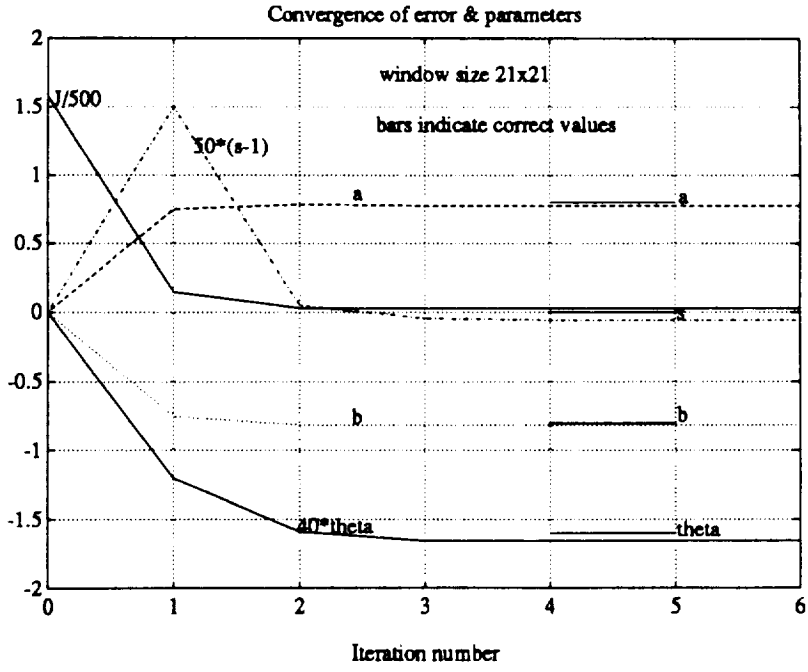

Figure 27: Convergence for roll-only maneuver at (20,20) from the FOE (21 × 21 window).

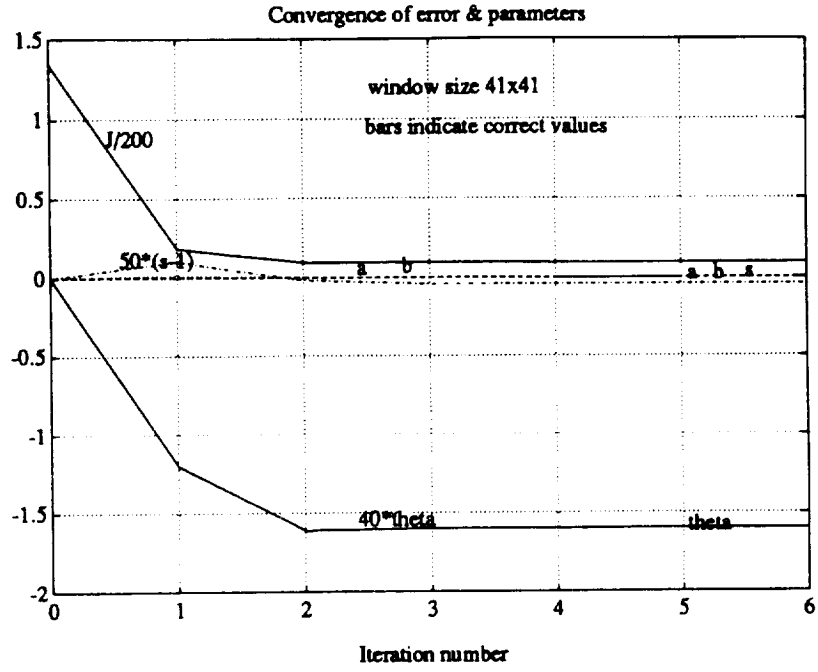

Figure 28: Convergence for roll-only maneuver at the FOE (41 × 41 window).

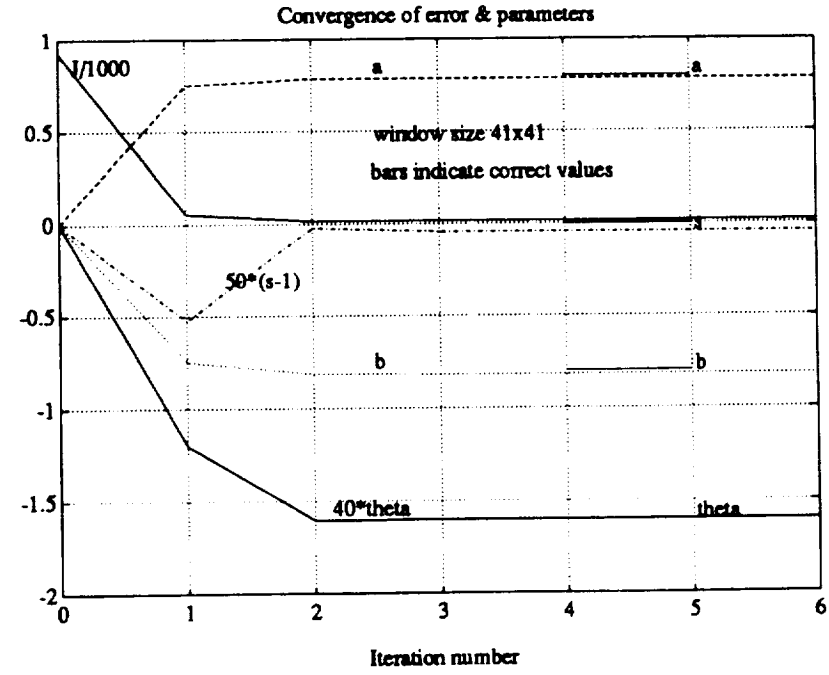

Figure 29: Convergence for roll-only maneuver at (20,20) from the FOE (41 × 41 window).

1. As before, the system practically converge within two iterations.

2. Although the cost-function—especially in figure 26—does not converge as close to zero as in all other case, the parameters still converge accurately to their respective values.

3. The accuracies improve noticeably as the window size doubles. From figure 24 and figure 25, θ virtually has zero error, while its error increases to 3.6% for the 21×21 window.

4. The expansion shows a transient for the (20,20) point, but it settles to zero after 2 iterations.

5. The converged shifts at the (20,20) point are remarkably close to the correct ones of (0.8,0.8) pixels.

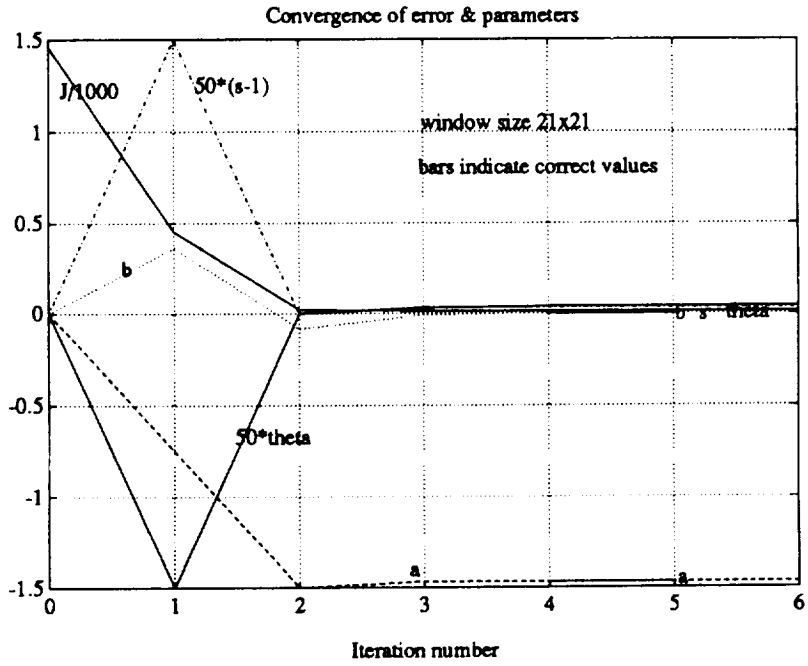

Figure 30: Convergence for yaw-only maneuver at the FOE (21×21 window).

Next, we present the results for yaw-only flying with no forward or lateral motion. The depth is constant at 150 m. The transformation parameters are calculated at the time of frame number 4 by comparing it with frame number 0; the yaw-angle difference is 0.002 rad. We translate this yaw angle by using the fact that, in our Flight/Vision simulation, the camera's FOV is taken as 10 degrees, and it corresponds with an image of size 128×128. This means that the expected shift is $\delta u = 1.467$ pixels. Thus, in these runs, we demonstrate δu-shift alone for a window centered on the FOE or on the point (26,26) with respect to the FOE. The following observations can be made:

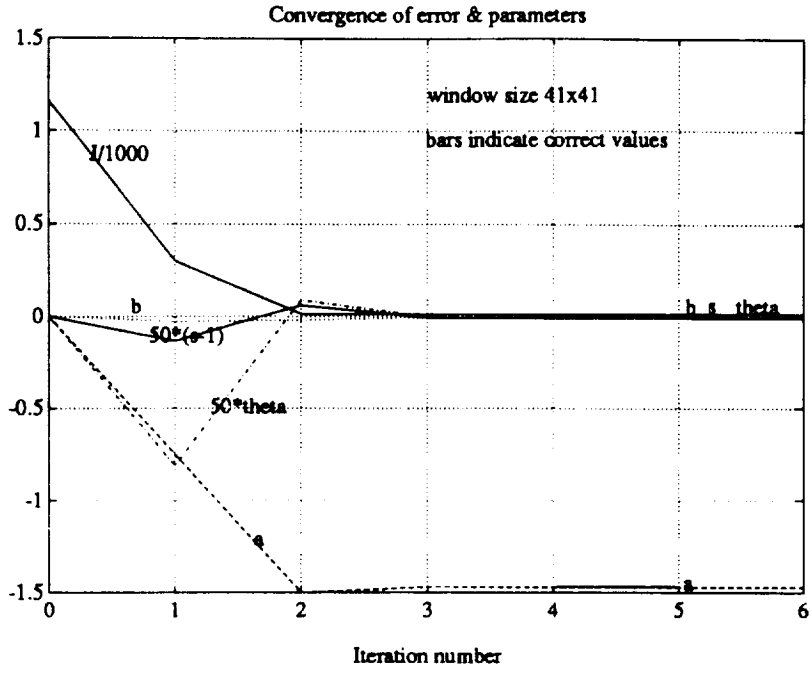

Figure 31: Convergence for yaw-only maneuver at (26,26) from the FOE (41 $\times$ 41 window).

1. Irrespective of the window size, or the location of the image-point with respect to the FOE, all converged parameters are close to being error free.

2. The expansion and rotation show transients which decay to zero after two iterations.

Lastly, we present the results for a general maneuver where the velocity is 1 m/s (starting from 150 m depth), pitch and yaw rates are 0.0005 rad/s each, and the roll-rate is 0.02 rad/s. The transformation parameters are calculated at the time of frame number 2 in figures 32, 33, and 35, and at frame number 4 in figure 34 by comparison with frame number 0. The following observations can be made:

1. The system converges within two iterations.

2. Generally, the accuracies improve with the window size.

3. The accuracy of s is around 6% for the FOE point—irrespective of the window size (21 to 61)—and it drops to 16% for the (20,20) point.

Summarizing the simulation results, we can conclude that the basic idea and algorithm are solid and perform very well. Although these simulations were done in apparently noise-free situation, they do get affected by the noise inherent in the pixel quantization.

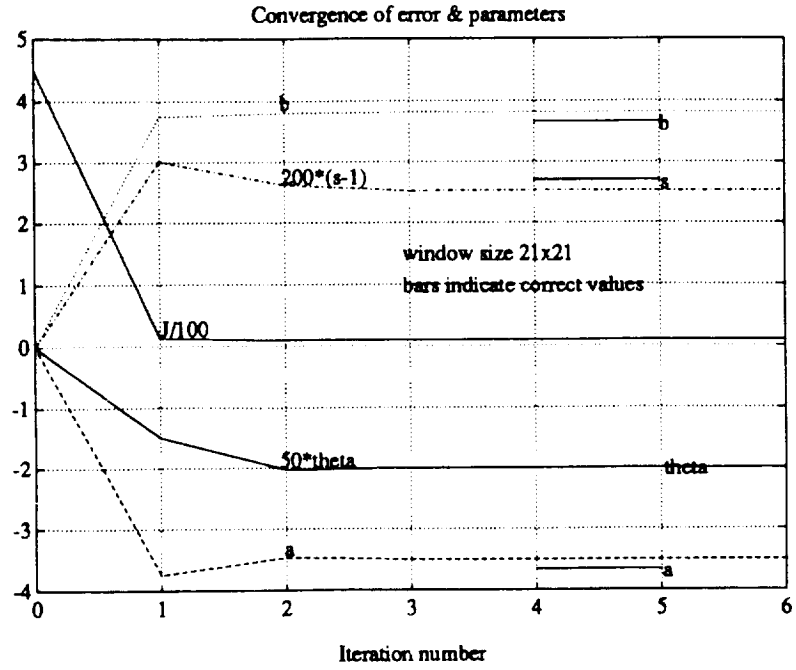

Figure 32: Convergence for general maneuvers at the FOE (21×21 window, 2-frames difference).

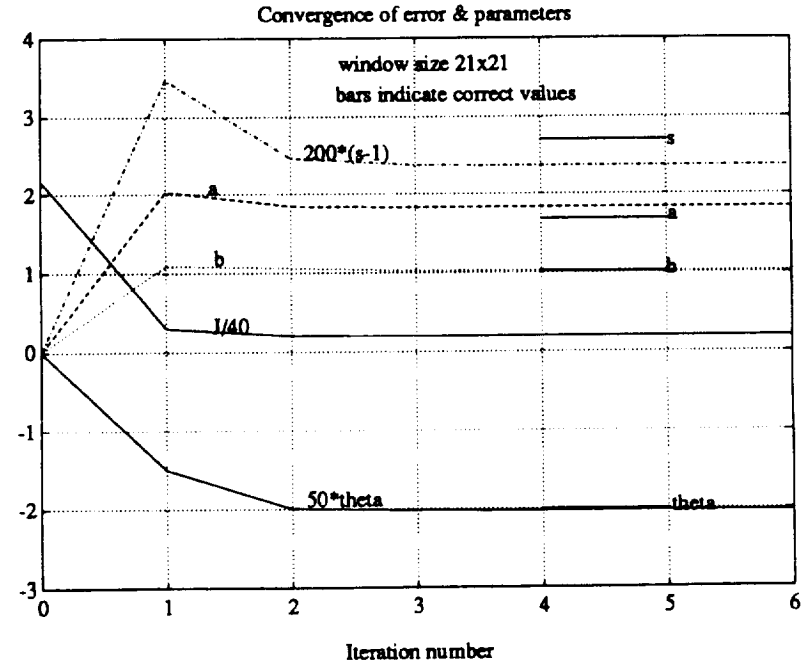

Figure 33: Convergence for general maneuvers at (20,20) from the FOE (21×21 window, 2-frames difference).

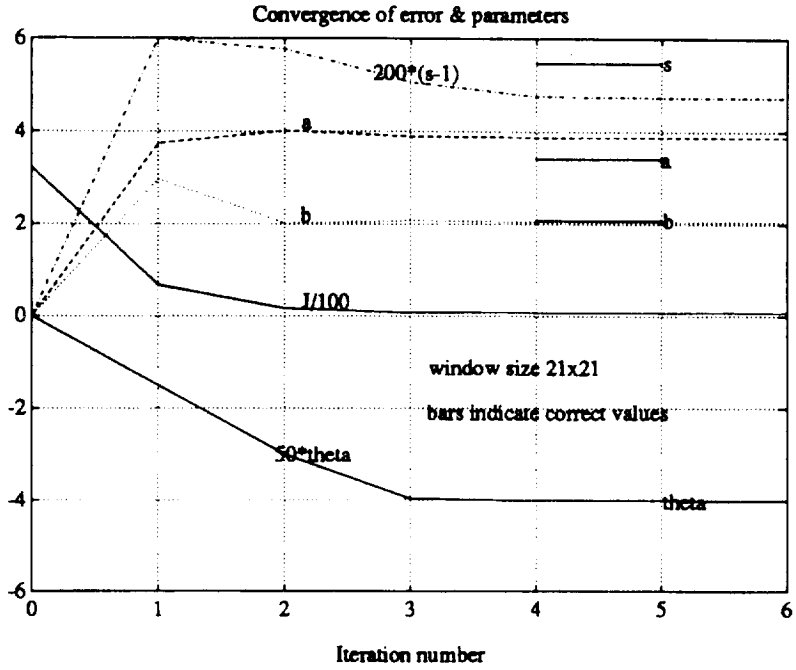

Figure 34: Convergence for general maneuvers at (20,20) from the FOE (21 × 21 window, 4-frames difference).

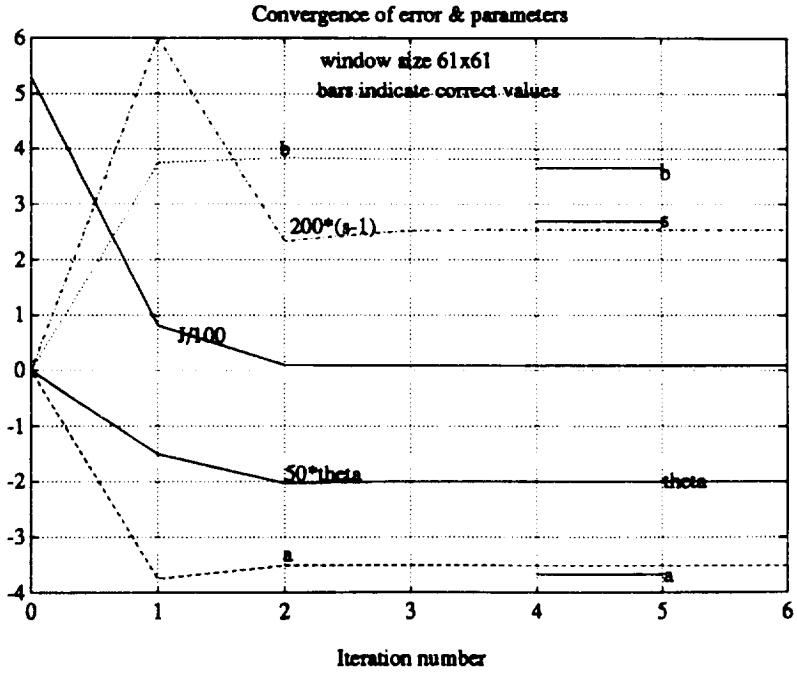

Figure 35: Convergence for general maneuvers at the FOE (61 × 61 window, 2-frames difference).

5 INCREASING THE TRIANGULATION BASELINE

In this section we use the above algorithm as the core on which a farther layer is to be built with the intention of increasing the accuracy and robustness of the practical algorithm. The implicit assumption here is that the flight trajectory is basically non-maneuvering, or, in other words, it is the maneuvers which will determine the maximum usable triangulation baseline.

5.1 The capture zone

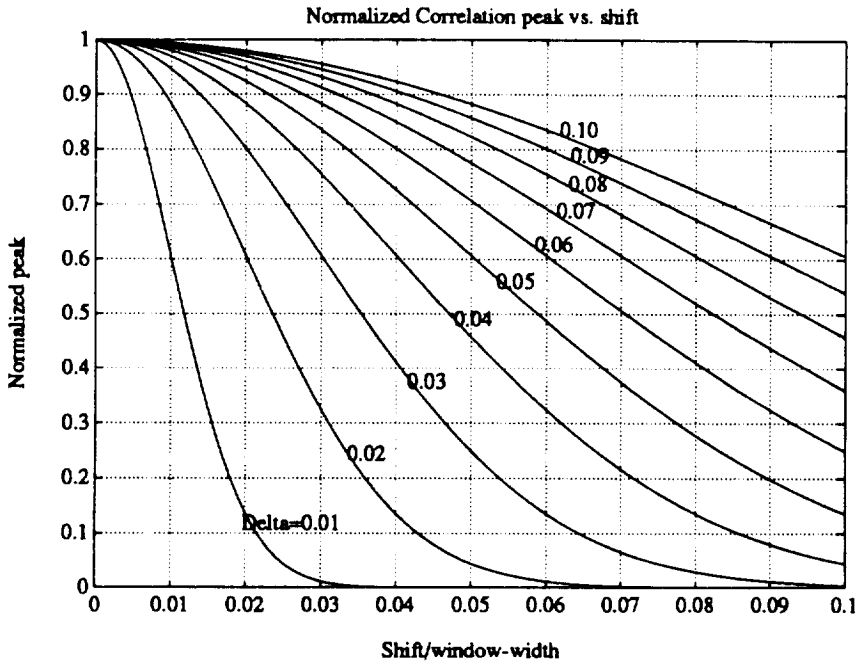

Figure 36: Average normalized correlation peak vs. shift, Delta in image-width fraction.

We have touched on the question of convergence in regard to figure 21. In that figure the "capture zone" is of ± 4 pixels—meaning that, as long as the error is within this zone, it always has the correct sign to drive it towards the stable solution. Thus, convergence is assured inside this zone, although its width is not usually known—especially when more than a single parameter is involved. It is possible, however, to estimate some lower bounds on the capture zone for each one of the four parameters. Estimating the width of the capture zone is based on the bandwidth or correlation width of the images. For that, we used $\Delta = 1.5$ pixels in conjunction with figures 5, 6, 36, and 37. What it means is that image-plane locations 1.5 pixels apart have gray-levels correlated with a correlation coefficient of $\exp\{-0.50\} = 0.606$ (see (16).

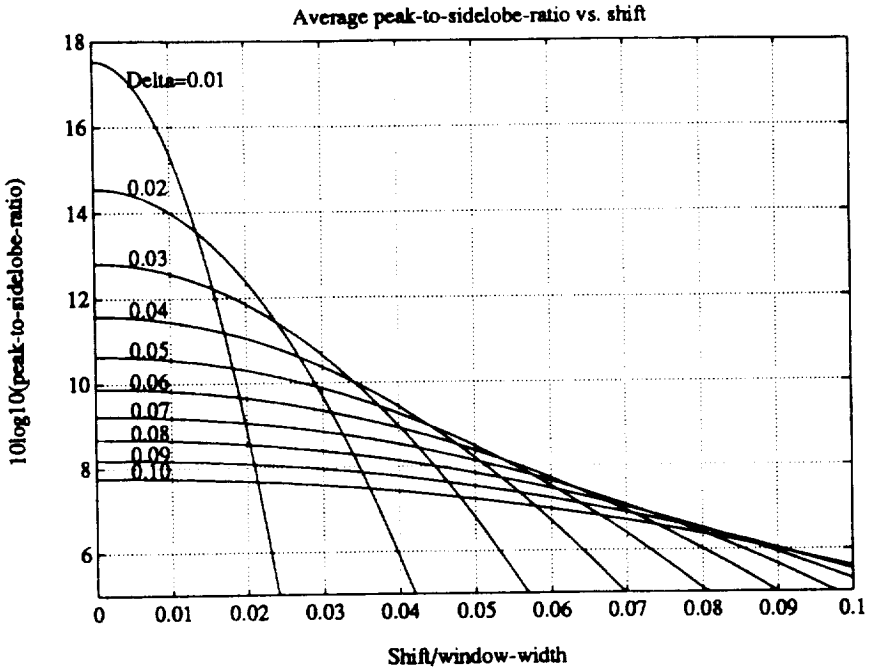

Figure 37: Average peak-to-sidelobe-ratio vs. shift, Delta in image-width fraction.

To estimate the capture zone, we arbitrarily assume that a PSR=7.5 is acceptable to provide a high enough probability of detecting the correct correlation peak and a low enough probability of false alarm (locking onto a wrong peak). This figure is equivalent to 15 dB in power ratios. Let us assume that the window size is 21×21; then Δ of 1.5 pixels is ≈ 0.07 of the window-size. From the corresponding graph in figure 37 we read that a PSR=7.5 is achieved for shifts less than 0.063 of the window size, *i.e.*, ± 1.32 pixels. Repeating this exercise for $\Delta = 2$ would result in a smaller capture zone of only 0.97 pixels.

A word about figures 36 and 37 is now in place. Figure 36 shows that the correlation peak drops slowly with the shift when Δ is large—as expected from (16). However figure 37 shows that the higher the Δ, the higher the PSR's initial value is, and the sharper its drop. This result is attributed to the fact that, when Δ is large, the effective number of independent image areas (objects) decreases. That has no effect on the mean correlation peak but it increases the sidelobes variance. The sidelobes variance of the cross-correlation, $C(\tau_u, \tau_v)$, is given by equation (A19) of [31],

$$\begin{aligned} \mathrm{var}\{C(\tau_u,\tau_v)\} = \quad & L^{-2} \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} g(u,v)R(u,v)R(\hat{u},\hat{v})\mathrm{d}u\mathrm{d}v + \\ & L^{-2} \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} g(u,v)R(u-\tau_u,v-\tau_v)R(u+\tau_u,v+\tau_v)\mathrm{d}u\mathrm{d}v \ , \end{aligned} \tag{41}$$

where

$$g(u,v) = \begin{cases} (1-|u/L|)(1-|v/L|), & (u,v) \in [-L,L]\times[-L,L] \\ 0 & \text{otherwise} \ , \end{cases} \tag{42}$$

is a triangular window that weighs the integrand, L^2 is the window area used for normalization, and $(\hat{u}, \hat{v})$ are the transformed (u, v) of (10). Far from the crosscorrelation peak, only the first term of (41) prevails, and that is what we used for constructing the above figures.

To estimate the capture zone for the expansion factor, s, and the rotation, θ, we refer back to figure 5. For the same case $L/\Delta = 14$, and we see from the figure that this is achieved with $d = 0.148$ which is equivalent to 8.5^0 of rotation or $s = 1.148$ of expansion. Overall, we have shown that the capture zone is quite wide, and there is some optimal window size that can be chosen for any given correlation width. Images from real scenes are highly non-stationary in the sense that Δ might be small for one part of the image and large for another. However it can never be smaller than the PSF which is why we used $\Delta = 1.5$ as a PSF-width estimate.

5.2 The iterative algorithm

In the iterative algorithm we start with frames which are close enough in time to ensure that the errors in the four parameters fall inside the worst-case capture zone. Let us say that we initially use frame-0 and frame-1, so the frame separation is one. The Newton's equations are iterated upon until the error converges. The converged parameters are then used to predict the initial values for a larger frame separation, say, between frame-0 and frame-4 (notice that the first frame of the pair is fixed here). The same is now repeated for this new frame separation. Thus, there are two nested iteration loops; the inner one iterates on the Newton's equations until convergence is achieved for some fixed frame separation; the outer loop iterates through increased frame separation. The algorithm can be summarized by the following pseudo code.

```
frame_separation = 1;
frame_0  = 0;
frame_1 = frame_0 + frame_separation;

while(frame < last_frame) {
  while(error has not converged) {
    solve Newton's eqs. to update
    a, b, s, theta;
  }
  if(final error is low) {
    increase frame separation;
    frame_1 = frame_0 + frame_separation;
    save last parameter values;
    predict initial parameter values for new frame separation;
  }
  else { declare previous-iteration results as final; }
```

}

When running in batch for some fixed number of frames, the outer iteration loop must stop when the frame separation cannot be increased any more. Another condition to stop is that the converged results of the last frame-separation are not satisfactory, as judged by some criteria.

The prediction of initial parameter values for the next (larger) frame separation is calculated from the converged parameters of the previous frame separation using the projection equations (31). Let us project an object of length l onto the image plane so that its projection is defined as unity. After decreasing the depth from z_0 to z_1, the projection changes to s_1. For a frame separation of t_1, that can be written as

$$1 = \frac{fl}{z_0} \; ; \qquad s_1 = \frac{fl}{z_1} \; ; \qquad z_1 = z_0 - V_Z t_1 \; , \tag{43}$$

from which

$$s_1 = \frac{z_0}{z_0 - V_Z t_1} \; ; \qquad z_0 = \frac{s_1 V_Z t_1}{s_1 - 1} \tag{44}$$

Rewriting the last equation for some s_2, t_2 instead of for s_1, t_1, and solving for s_2, we get

$$s_2 = \frac{s_1 t_1}{t_2 - s_1(t_2 - t_1)} \tag{45}$$

This is how the current expansion estimate (for the current frame separation) is used to predict the expansion estimate for a larger frame separation, t_2. The other three parameters are predicted based on linear extrapolation, so that

$$a_2 = a_1 t_2/t_1 \; ; \;\; b_2 = b_1 t_2/t_1 \; ; \;\; \theta_2 = \theta_1 t_2/t_1 \tag{46}$$

After the algorithm stops, (44) is used to calculate the current best estimate of the *initial* depth z_0 based on the last pair of s_i, t_i which corresponds to the largest triangulation baseline that yielded convergence.

5.3 Performance of the iterative algorithm

First we ran the iterative algorithm on our simulated imagery, and then on some real imagery.

Let us start with a typical run on the simulated imagery. It is a non-maneuvering, forward-flying case with velocity of 2 m/s. The first frame pair is made up of frame-0 and frame-2. The window of size 21 $\times$ 21 is initially centered on pixel (74,74) which is 10 pixels away from the FOE (which is at (64,64)) in u and v. There are 40 frames in the set. The following screen output reports progress in the estimation of the initial depth of 150 m. Each table-like block of numbers reports the convergence of the inner loop for the current frame separation. The inner-loop iteration number is k and the error is denoted by err.

```
Opened forward_flying frame 0
Opened forward_flying frame 2

k,a,b,s,theta err = 0  0.000000 0.000000 1.000000  0.000000 267.672333
k,a,b,s,theta err = 1  0.440236 0.406381 1.030000 -0.005372 141.133240
k,a,b,s,theta err = 2  0.236247 0.248784 1.021997  0.000912  79.217903
k,a,b,s,theta err = 3  0.293832 0.275524 1.020823 -0.000823  83.259949
k,a,b,s,theta err = 4  0.278676 0.270192 1.020453 -0.000340  82.172791
k,a,b,s,theta err = 5  0.283285 0.271448 1.020556 -0.000496  82.425667
k,a,b,s,theta err = 6  0.281893 0.271135 1.020525 -0.000446  82.349678
k,a,b,s,theta err = 7  0.282312 0.271216 1.020533 -0.000462  82.372147
k,a,b,s,theta err = 8  0.282187 0.271194 1.020531 -0.000457  82.365257
k,a,b,s,theta err = 9  0.282224 0.271201 1.020531 -0.000458  82.367386
k,a,b,s,theta err = 10 0.282213 0.271199 1.020531 -0.000458  82.366638

Current estimate of initial depth = 198.825650

Opened forward_flying frame 5

k,a,b,s,theta err = 0 0.705533 0.677997 1.052959 -0.001145 74.528183
k,a,b,s,theta err = 1 0.708271 0.690493 1.069563 -0.000779 55.280674
k,a,b,s,theta err = 2 0.712015 0.685594 1.065833 -0.000990 55.973583
k,a,b,s,theta err = 3 0.710209 0.685661 1.066386 -0.000896 55.718666
k,a,b,s,theta err = 4 0.710757 0.685674 1.066294 -0.000920 55.761593
k,a,b,s,theta err = 5 0.710619 0.685669 1.066310 -0.000915 55.753265
k,a,b,s,theta err = 6 0.710652 0.685670 1.066307 -0.000916 55.754974
k,a,b,s,theta err = 7 0.710644 0.685671 1.066308 -0.000916 55.754597

Current estimate of initial depth = 160.812401

Opened forward_flying frame 10

k,a,b,s,theta err = 0 1.421288 1.371342 1.142033 -0.001831 73.417000
k,a,b,s,theta err = 1 1.565485 1.555250 1.153044  0.000626 28.408524
k,a,b,s,theta err = 2 1.531177 1.529034 1.151938 -0.000007 27.785374
k,a,b,s,theta err = 3 1.540096 1.533858 1.152508  0.000302 27.518284
k,a,b,s,theta err = 4 1.537325 1.532594 1.152262  0.000157 27.572006
k,a,b,s,theta err = 5 1.538222 1.532950 1.152353  0.000221 27.549091
k,a,b,s,theta err = 6 1.537919 1.532843 1.152320  0.000194 27.556232
k,a,b,s,theta err = 7 1.538024 1.532876 1.152332  0.000205 27.553633
k,a,b,s,theta err = 8 1.537986 1.532865 1.152328  0.000200 27.554605
```

```
k,a,b,s,theta err = 9 1.538000 1.532869 1.152329  0.000202 27.554232

Current estimate of initial depth = 151.294483

Opened forward_flying frame 16

k,a,b,s,theta err = 0 2.460800 2.452590 1.268244  0.000323 122.735245
k,a,b,s,theta err = 1 2.784964 2.769740 1.269186 -0.001672  24.976557
k,a,b,s,theta err = 2 2.683191 2.691992 1.270512  0.000779  17.046618
k,a,b,s,theta err = 3 2.703243 2.702418 1.268434  0.000409  16.436787
k,a,b,s,theta err = 4 2.697401 2.699811 1.268962  0.000566  16.535572
k,a,b,s,theta err = 5 2.698910 2.700440 1.268833  0.000553  16.499908
k,a,b,s,theta err = 6 2.698529 2.700282 1.268864  0.000553  16.508102
k,a,b,s,theta err = 7 2.698624 2.700322 1.268857  0.000553  16.506060
k,a,b,s,theta err = 8 2.698600 2.700312 1.268859  0.000553  16.506615
k,a,b,s,theta err = 9 2.698606 2.700315 1.268858  0.000553  16.506516

Current estimate of initial depth = 151.021904

Opened forward_flying frame 22

k,a,b,s,theta err = 0 3.710583 3.712933 1.411131  0.000760 268.700623
k,a,b,s,theta err = 1 4.293034 4.237146 1.417225 -0.000956  30.175304
k,a,b,s,theta err = 2 4.125841 4.143232 1.415531 -0.001404   8.244106
k,a,b,s,theta err = 3 4.139481 4.153663 1.415352 -0.000450   8.060862
k,a,b,s,theta err = 4 4.137228 4.152442 1.415281 -0.000596   8.049483
k,a,b,s,theta err = 5 4.137534 4.152607 1.415305 -0.000568   8.049872
k,a,b,s,theta err = 6 4.137496 4.152581 1.415299 -0.000574   8.049752

Current estimate of initial depth = 149.947839

Opened forward_flying frame 28

k,a,b,s,theta err = 0 5.265904 5.285103 1.596075 -0.000731 522.403687
k,a,b,s,theta err = 1 6.015904 6.035103 1.601842 -0.001724  25.510233
k,a,b,s,theta err = 2 5.956887 5.961528 1.600377  0.000404  18.381941
k,a,b,s,theta err = 3 5.961518 5.965533 1.599840  0.000178  18.452717
k,a,b,s,theta err = 4 5.960965 5.965257 1.599872  0.000228  18.443174
k,a,b,s,theta err = 5 5.961043 5.965267 1.599865  0.000224  18.443617
k,a,b,s,theta err = 6 5.961033 5.965264 1.599866  0.000224  18.443457
```

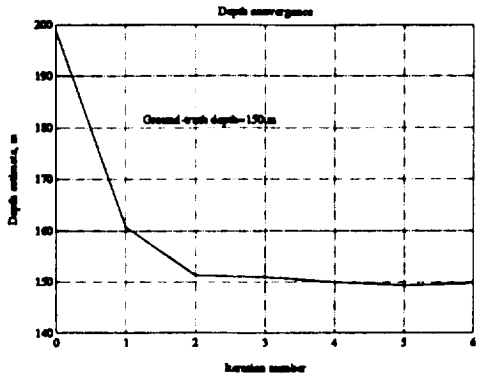

Figure 38: Depth convergence with iterations (increased triangulation baseline).

```
Current estimate of initial depth = 149.354233

Opened forward_flying frame 34

k,a,b,s,theta err = 0 7.238398 7.243535 1.835851  0.000272 938.172974
k,a,b,s,theta err = 1 7.988398 7.993535 1.834242 -0.009328 124.417595
k,a,b,s,theta err = 2 8.284004 8.308021 1.832178 -0.000755  30.430639
k,a,b,s,theta err = 3 8.285615 8.306216 1.832047 -0.000582  30.408218
k,a,b,s,theta err = 4 8.285925 8.305839 1.831991 -0.000556  30.394806
k,a,b,s,theta err = 5 8.285982 8.305765 1.831979 -0.000549  30.392294
k,a,b,s,theta err = 6 8.285994 8.305748 1.831976 -0.000548  30.391562
k,a,b,s,theta err = 7 8.285996 8.305745 1.831976 -0.000548  30.391680

Current estimate of initial depth = 149.733168

Final estimate of initial depth = 149.733168
```

There are a few interesting observations to make:

1. Frame-0 is always used as the basis for comparison —initially with frame-2, then with frames 5, 10, 16, 22, 28, and 34. The depth estimate improves with the frame separation as shown in figure 38

2. Notice that the first line of each block represents the initial conditions for a, b, s, and θ. In the first block, these are 0.0, 0.0, 1.0, 0.0 because we do not know any better. The last line of each block represents the converged values which are used to predict the initial conditions for the next block.

3. The error in each block starts from some value and usually drops and stabilizes. If the initial guess falls far from the minimum but inside the capture zone, then the error starts

from a large value and drops sharply. If the initial guess happened to be good, then the errors are already "converged"; this is exemplified by the second block belonging to the frame pair (0,5).

4. The final result was obtained from the image pair (0,34)—which does not necessarily represent the maximum frame separation possible. We have thus effectively used a triangulation baseline of 68 m which constitutes a substantial fraction of the initial depth of 150 m. This is the reason why we regard this algorithm as a track-before-detect one. In this example, the accuracy of the final result is 0.178 percent.

We have run the algorithm on various other simulated cases—at and around the FOE. Generally, the depth accuracies are better than 2%, and they improve as we get closer to the FOE.

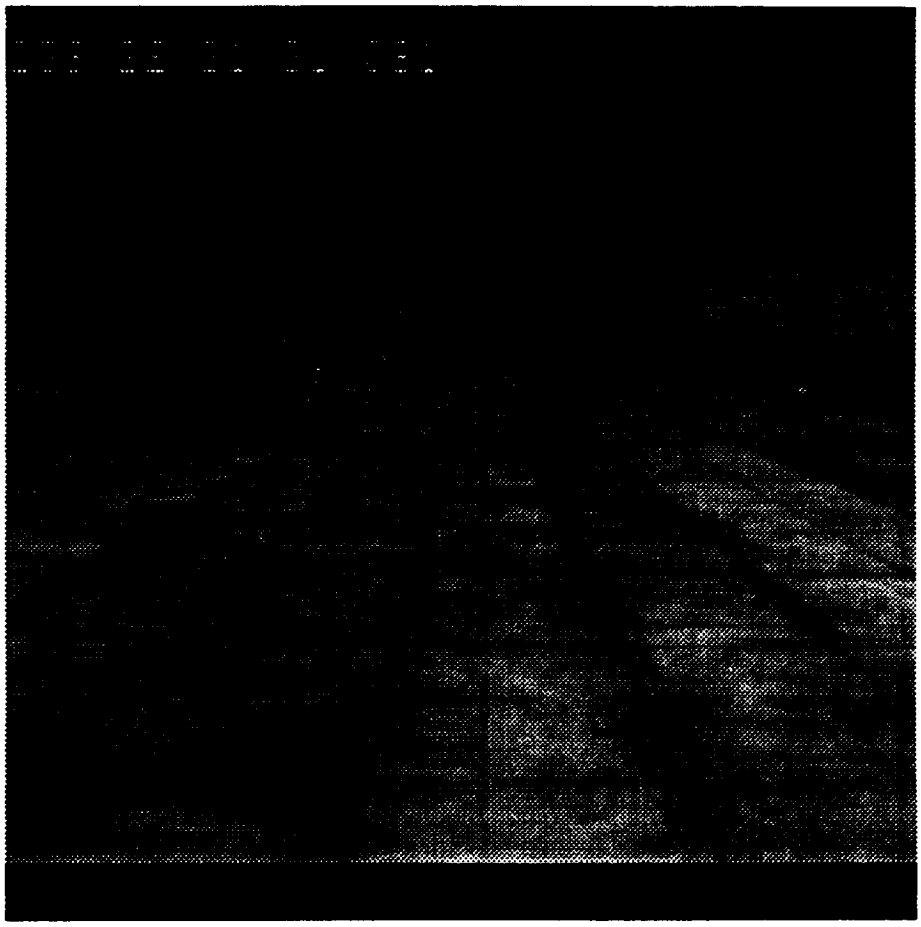

Figure 39: The first "newline" image.

We now present real-data cases from our imagery set "newline"; the first image of this sequence is shown in figure 39. The scene is that of a runway with a few surveyed trucks. The images are of size 512 × 512, the speed is 30.17 ft/s, and the frame rate 30 per second. There are only minor maneuvers in this flight. The convergence curve is shown in figure 40 for the

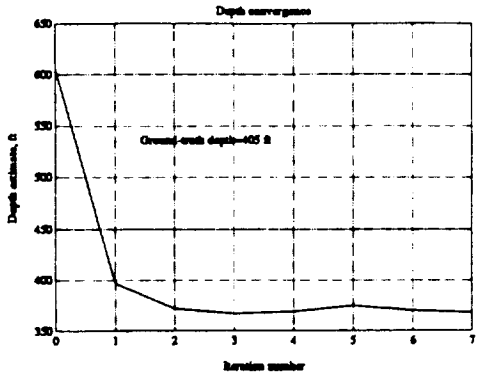

Figure 40: Depth convergence with iterations for "newline" leftmost truck.

leftmost truck which is at depth of 405 ft. The frame pairs used are: frame-0 with 2, 5, 10, 16, 22, 28, 34, and 40. Each iteration uses the next-larger frame separation. The converged depth resulting from the algorithm is 368, so that the accuracy here is of 9%. For the farther truck on

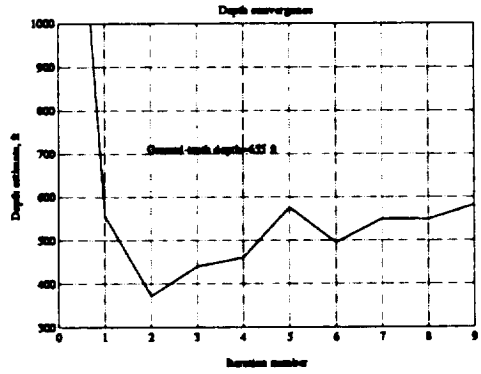

Figure 41: Depth convergence with iterations for "newline" leftmost truck.

the left, the algorithm ran ten iterations (last frame pair was (0,52)) and converged on a depth of 583 ft, where the ground-truth depth is 655 ft—accuracy of 11%. The convergence curve is shown in figure 41. The objects in these two examples show very little texture, and they are also small and far (TTC $\approx$ 10 s) which may explain why the algorithm does not perform that well. Still the results can be considered satisfactory.

6 ERROR ANALYSIS

In this section we analyze the depth error as achieved by combining the depth results from lateral translation and those from expansion. We have already discussed the accuracy of the depth derived from lateral translation which is given by (18) where σ_u is given by figure 6.

The accuracy of the depth derived from expansion is determined by that of the expansion factor. When all the (four) parameters have converged, and thus compensated for, the case becomes that of nominally zero distortion and shifts. Therefore we have to examine the sensitivity of the correlation peak value to residual errors in the expansion factor alone. This accuracy is determined by the additive noise at the peak (denoted by $C_N(0,0)$). Notice that, so far, we have neglected this noise because it is practically much smaller than the *sidelobe* noise which results from the randomness of the image itself. The additive noise at the peak is given by equation (19) of [31] which is similar to (41) but with $\tau_u = \tau_v = 0$ and one of the correlation functions replaced by that of the noise, $R_N(u,v)$, that is,

$$\mathrm{var}\{C_N(0,0)\} = L^{-2} \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} g(u,v) R(u,v) R_N(u,v) \mathrm{d}u\mathrm{d}v \tag{47}$$

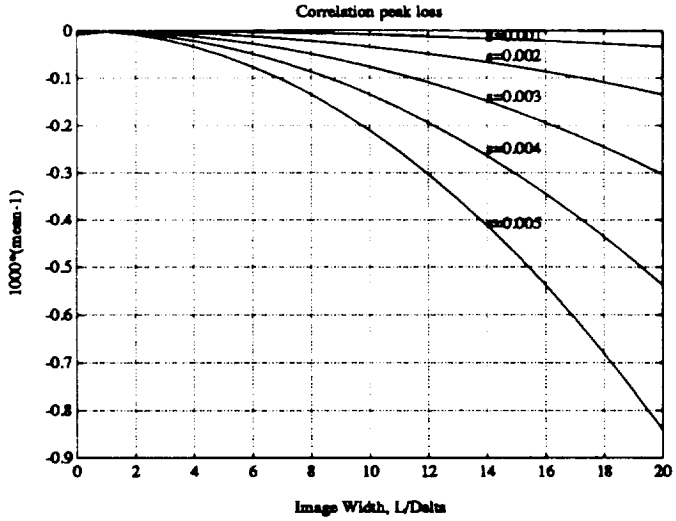

Figure 42: Loss in correlation peak value due to residual errors in scaling factor.

For simplicity we use equal $R(\tau_u, \tau_u)$ and $R_N(\tau_u, \tau_u)$ as given by (16). The question is now: what is the change in the expansion factor which causes a change in the correlation peak equal to the standard deviation, $\sqrt{\mathrm{var}\{C_N(0,0)\}}$. The correlation peak, as given by (5) of [31], is plotted in figure 42. For the same example used earlier, where $L/\Delta = 14$, and assuming an image signal-to-noise ratio of a 100, it is found from (47) that $\sqrt{\mathrm{var}\{C_N(0,0)\}} = 0.000177$. In the figure, the point having $L/\Delta = 14$ and an ordinate of -0.177 falls between the graphs of $s = 0.003$ and $s = 0.004$. Interpolating between these, results in $s = 0.00325$.

The relationship between the s error and the depth error is derived from (44), where we had

$$z_0 = \frac{sV_Z\Delta t}{s-1}\,, \tag{48}$$

so that

$$\frac{\mathrm{d}z_0}{z_0} = -\frac{\mathrm{d}s}{s(s-1)} \approx -\frac{\mathrm{d}s}{s-1} \tag{49}$$

We can thus express the expansion-based depth standard deviation as

$$\sigma_{zs} = \frac{\sigma_s z_0}{s-1} \tag{50}$$

For $s = 1.0274$, as was used to create figure 10 ($z_0 = 150$ m), and with $\sigma_s = 0.00325$, (50) yields $\sigma_{zs} = 17.8$ m which is close to the simulation results.

The depth information contained in the expansion factor, s, and that contained in the shifts ,(a, b), is likely to be correlated because it is the same additive noise that causes inaccuracies in both measurements. Developing the necessary covariance matrix that relates their errors is not an easy task, and we thus forego that job here. However, we can still write down the combining algorithm for the initial-depth unbiased estimate, $\hat{z_0}$, as (see [33])

$$\hat{z_0} = kz_s + (1-k)z_t \,, \tag{51}$$

where z_s is the expansion-based depth measurement and z_t the translation- (or shifts-) based one. k is determined by the variances, σ_{zs}^2 of z_s and σ_{zt}^2 of z_t, and by their correlation coefficient ρ, as

$$k = \frac{\sigma_{zt}^2 - \rho\sigma_{zt}\sigma_{zs}}{\sigma_{zs}^2 + \sigma_{zt}^2 - 2\rho\sigma_{zt}\sigma_{zs}} \,, \tag{52}$$

and the minimum error—using this k—is then

$$\mathrm{E}\{e^2\} \triangleq \mathrm{E}\{(z_0 - \hat{z}_0)^2\} = \frac{\sigma_{zt}^2\sigma_{zs}^2(1-\rho^2)}{\sigma_{zs}^2 + \sigma_{zt}^2 - 2\rho\sigma_{zt}\sigma_{zs}} \tag{53}$$

We know that, close to the FOE, $\sigma_{zs} \ll \sigma_{zt}$ so that, irrespective of ρ, $k \Rightarrow 1$, and vice versa. This means that, even if we use some guessed ρ of, say, 0.5 at this point, we will still be combining the measurements in a consistent way; that is the accurate measurement will contribute more than the inaccurate one—although, without knowing ρ, the proportions will not be optimal.

7 CONCLUDING REMARKS

In this paper we developed a new expansion-based passive-ranging algorithm that can complement the existing shift-based algorithm in the image areas near the FOE. We presented simulation and real-data results and compared them with the analysis results.

In the future we intend to develop this algorithm in two directions. One is to make it process an image sequence in real time and produce range maps. The other is to use it to segment an image into objects.

REFERENCES

[1] L.H. Wegner. On the accuracy analysis of airborne techniques for passively locating electromagnetic emitters. Report R-722-PR AD 729 767, NTIS ASTIA D.C., Rand Corp, 1971.

[2] J.L. Poirot and G.V. McWilliams. Application of linear statistical models to radar location techniques. *IEEE Trans. on Aerospace and Electronic Systems*, 10(6):830–834, November 1974.

[3] D.J. Torrieri. Statistical theory of passive location systems. *IEEE Trans. on Aerospace and Electronic Systems*, 20(2):183–198, March 1984.

[4] M. Gavish and E. Fogel. Effect of bias on bearing-only target location. *IEEE Trans. on Aerospace and Electronic Systems*, 26(1):22–25, January 1990.

[5] Y. Barniv. Neural networks application to divergence-based passive ranging. Technical Memorandum 103981, NASA, Ames Research Center, Moffett Field, CA, August 1992.

[6] B.K.P. Horn and B.G. Schunck. Determining optical flow. *Artificial Intelligence*, 17(3):185–203, August 1981.

[7] Y. Barniv. Dynamic programming solution for detecting dim moving targets. *IEEE Trans. on Aerospace and Electronic Systems*, 21(3):144–156, March 1985.

[8] Y. Barniv and O. Kella. Dynamic programming solution for detecting dim moving targets part II: Analysis. *IEEE Trans. on Aerospace and Electronic Systems*, 23(6):776–788, November 1987.

[9] Barniv, Y. (Ch. 4) (Yaakov Bar-Shalom, Editor). *Multitarget-Multisensor Tracking: Advanced Applications.* Artech House, 1990.

[10] B. Sridhar and A.V. Phatak. Simulation and analysis of image-based navigation system for rotorcraft low-altitude flight. In *Proceedings of the AHS Meeting on Automation Application for Rotorcraft*, Atlanta, GA, April 1988.

[11] B. Sridhar, V.H.L. Cheng, and A.V. Phatak. Kalman filter based range estimation for autonomous navigation using imaging sensors. In *Proceedings of the 11th Symposium on Automatic Control in Aerospace*, Tsukuba, Japan, July 1989.

[12] Y. Barniv. Error analysis of combined optical-flow and stereo passive ranging. *IEEE Trans. on Aerospace and Electronic Systems*, to be published, October 1992.

[13] H. C. Longuet-Higgins and K. Prazdny. The interpretation of a moving retinal image. *Proc. R. Soc., London B*, 208:385–397, 1980.

[14] K. Prazdny. Determining the instantaneous direction of motion from optical flow generated by a curvilinear moving observer. *Computer Vision, Graphics, and Image Processing*, 17:238–248, 1981.

[15] K. Prazdny. Egomotion and relative depth map from optical flow. *Biological Cybernetics*, 36:87–102, 1980.

[16] J. Koenderink. Optic flow. *Vision Research*, 26(1):161–180, 1986.

[17] J.J. Koenderink and A.J. van Doorn. Invariant properties of the motion parallax field due to the movement of rigid bodies relative to an observer. *Optica Acta*, 22(9):773–791, 1975.

[18] J.J. Koenderink and A.J. van Doorn. Local structure of movement parallax of the plane. *Journal of Optical Society of America*, 66(7):717–723, July 1976.

[19] R. C. Nelson and J. Aloimonos. Obstacle avoidance using flow field divergence. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 11(10):1102–1106, 1989.

[20] D. L. Ringach and Y. Baram. A diffusion mechanism for obstacle detection from size-change information. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 5(5):6–7, 1992.

[21] B. Sridhar, R.E. Suorsa, and B. Hussien. Passive range estimation for rotocraft low altitude flight. *Journal of Machine Vision and Applications*, 1993. Technical Memorandum 103899, NASA Ames Research Center. October 1991.

[22] S. Merhav and Y. Bresler. On-line vehicle motion estimation from visual terrain information, part i: Recursive image registration. *IEEE Trans. on Aerospace and Electronic Systems*, pages 588–605, September 1986.

[23] Y. Bresler and S.J. Merhav. On-line vehicle motion estimation for visual terrain information. part ii: Ground velocity and position estimation. *IEEE Trans. on Aerospace and Electronic Systems*, AES-22(5):588–603, September 1986.

[24] Y. Bresler and S.J. Merhav. Recursive image registration with application to motion estimation. *IEEE Trans. on ASSP*, pages 70–86, January 1987.

[25] S. Merhav. Air-to-surface range estimation by a stochastic gradient approach. Technical Memorandum MEMO, NASA, Ames Research Center, Moffett Field, CA, October 1992.

[26] D. Reagan and K.I. Beverley. Looming detectors in the human visual pathway. *Vision Research*, 18:415–421, 1978.

[27] D. Reagan and K.I. Beverley. Visual responses to changing size and sideways motion for different directions of motion in depth: Linearization of visual responses. *Journal of Optical Society of America*, 70(11):1289–1297, November 1980.

[28] M. Hershenson. Visual system responds to rotational and size-change components of complex proximal motion patterns. *Perception and Psychophysics*, 42(1):60–64, 1987.

[29] P. Cavanagh and O.E. Favreau. Motion aftereffect: A global mechanism for the perception of rotation. *Perception and Psychophysics*, 9:175–182, 1980.

[30] D. Reagan and K.I. Beverley. Visual responses to vorticity and the neural analysis of optic flow. *Journal of Optical Society of America*, 2(2), February 1985.

[31] Mostafavi H. and F.W. Smith. Image correlation with geometric distortion, part i: Acquisition performance. *IEEE Trans. on Aerospace and Electronic Systems*, 14(3):487–493, May 1978.

[32] Mostafavi H. and F.W. Smith. Image correlation with geometric distortion, ii: Effect on local accuracy. *IEEE Trans. on Aerospace and Electronic Systems*, 14(3):494–500, May 1978.

[33] A. Gelb. *Applied Optimal Estimation.* The MIT Press, 1974.

[34] P.N. Smith. Flight data acquisition for validation of passive ranging algorithms for obstacle avoidance. In *Proceedings of the American Helicopter Society Forum*, Washington, D.C., May 1990.