This paper describes a new technique for passive ranging which is of special interest in areas such as covert nap-of-the-Earth helicopter flight and spacecraft landing. This technique is based on the expansion experienced by the image-plane projection of an object as its distance from the sensor decreases. The motion and shape of a small window, assumed to fall inside the boundaries of some object, is approximated by an affine transformation. The parameters of the transformation matrix (expansion, rotation, and translation) are derived by initially comparing successive images, and progressively increasing the image time separation. This yields a more favorable geometry for triangulation (larger baseline) than is currently possible. Depth is directly derived from the expansion part of the transformation, and its accuracy is proportional to the baseline length.
The idea of using divergence as a source of depth information is not new. The works of Longuet-Higgins and Prazdny [9], Prazdny [10, 11], Koenderink [12], Koenderink and van Doorn [13, 14], and Nelson and Aloimonos [15] elaborate extensively on this subject. Recently, an interesting extension to these works was reported by Ringach and Baram [16]; although it is field based, it explicitly assumes that the scene is composed of objects 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 Fig. 1. The local-divergence method is intended for textured objects lacking well-defined edges, whereas the global-divergence method is intended for objects with little or no texture but having well-defined edges. We assume textured objects, 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, yet it expands as the depth decreases. This expansion serves 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 even for just the conventional translation-based methods. 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 [6, 7] and Sridhar, Suorsa and Hussien in [17] which derive the image-plane translations of “points of interest” (small windows) through spatial cross correlation between consecutive images and subsequent Kalman filtering of their image-plane trajectories.

Reference to another closely related area of research represented by the work of Merhav and Bresler (see [18—21]) is called for. The first three papers primarily address image-plane motion estimation, which is, of course, equivalent to depth. Also, they rely upon the assumption (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 [22—24] as well as vorticity detectors [24—26].

The organization of this report is as follows. Section II contains the theory of divergence, expansion, and depth. Section III presents the idea of using the AFTR to relate objects in different frames. Section IV presents simulation results. Section V presents the practical algorithm that iterates over increased frame separation, and Section VI discusses error analysis.

II. OPTICAL FLOW, DIVERGENCE AND EXPANSION

The basic equations for the divergence in the image plane are summarized in this section; these are based on prior work described in [9—16].

Consider the projection $p$ of some point $P$ onto the surface of a spherical camera as shown in Fig. 2. At that point define the origin of an image plane $(U, V)$ tangent to the sphere. This image plane approximates the sphere at the point of tangency. Assume that $P$ is located on a smooth surface described by some function $z = f(x, y)$ so that its gradient $\nabla z \triangleq [z_x, z_y]$ exists (T denotes the matrix transpose operation). The motion of the camera causes the stationary point $P$ and its surrounding to describe a velocity field estimation which can be thought of as a precursor of the work reported here.

The basic equations for the divergence in the image plane are summarized in this section; these are based on prior work described in [9—16].

Consider the projection $p$ of some point $P$ onto the surface of a spherical camera as shown in Fig. 2. At that point define the origin of an image plane $(U, V)$ tangent to the sphere. This image plane approximates the sphere at the point of tangency. Assume that $P$ is located on a smooth surface described by some function $z = f(x, y)$ so that its gradient $\nabla z \triangleq [z_x, z_y]$ exists (T denotes the matrix transpose operation). The distance of any point on that surface from the center of the sphere can then be approximated in the neighborhood of $P$ by

$$z \approx z_0 + [xy] \cdot \nabla z$$

where $z_0$ is the depth of $P$. The relative motion of the camera with respect to the scene is defined by its translational velocity $V \triangleq [V_x, V_y, V_z]^T$ and its rotational velocity $\omega \triangleq [\omega_x, \omega_y, \omega_z]^T$. It is convenient to normalize $V$ by $z_0$ and denote $\hat{V} \triangleq [V_x, V_y]^T / z_0$.

The motion of the camera causes the stationary point $P$ and its surrounding to describe a velocity field.
(or optical flow) around \( p \) on the image plane. We denote image-plane projections by \((u, v)\) and their temporal derivatives by \((\dot{u}, \dot{v})\). Thus, the image-plane velocity vector at \( p \) is given by \( \mathbf{v}(p) = [\dot{u} \dot{v}]^T \). The spatial partial derivatives of \( u \) and \( v \) are denoted by \( u_x, u_y, v_x, v_y \). From [9], the following equations hold at \( p \),

\[
\begin{align*}
\dot{u} &= -v_x - \omega_y, \\
\dot{v} &= -v_y + \omega_x \\
u_x &= v_z + v_y z_y, \\
u_y &= v_z - v_x z_x, \\
v_x &= v_z + v_y z_y, \\
v_y &= v_z - v_x z_x.
\end{align*}
\]

Using that, the divergence at \( p \) (denoted by \( \text{div}(p) \)) can be expressed as

\[
\text{div}(p) = \frac{\partial}{\partial z} (2v_z + [v_x, v_y, v_z] \cdot \nabla z) = 2v_z + [v_x, v_y] \cdot \nabla z.
\]

To interpret the above equation, suppose the camera only moves in the \( Z \) direction, that is, \( v_z = v_x = 0 \), so that \( \text{div}(p) = 2v_z = \partial v_z / \partial z \). Thus, \( \text{div}(p) \) is twice the reciprocal of the time-to-collision of \( P \) with the center of the camera. \( \text{div}(p) \) is termed “immediacy” in some papers because it measures the imminence of an impending collision. On the other hand, if \([v_x, v_y] \neq [0 0] \) but \( v_z = 0 \), there can still be a relative depth change between the camera and the patch at \( P \) because the patch may be generally slanted. \( \text{div}(p) \) will still have the same interpretation as before, except that the impending 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 valid physical interpretations. Note that the rotational velocities do not appear in \( \text{div}(p) \); this means that the time-to-collision information is wholly contained in the imagery and no additional information is needed.

The global divergence is defined (see [16]) as the average divergence over the area of an object, and denoted by \( \chi(R) \) for an object whose projection onto the image plane is \( R \). It is shown that

\[
\chi(R) = \frac{1}{A(R)} \int_R \text{div}(p) ds = \frac{1}{A(R)} \frac{dA(R)}{dt}
\]

where \( A(R) \) is the object area and \( ds \) is the elemental area. In words, the global divergence equals the temporal rate of change of the normalized object area.

III. ESTIMATING THE RATE OF EXPANSION

In this section we introduce the AFTR and develop the algorithm necessary to estimate the rate of expansion of the object.

A. Affine Transformation

The AFTR can be used to relate projections of the object at different frames (or instances); its most general form is defined by six parameters. However, we judged that four parameters should suffice because they directly convey the physically interpretable changes one would expect to occur. In accordance with Fig. 3, we define our specific AFTR by

\[
\begin{bmatrix}
\tilde{u} \\
\tilde{v}
\end{bmatrix} = \begin{bmatrix}
C\theta & -S\theta \\
S\theta & C\theta
\end{bmatrix} \begin{bmatrix}
u - u_0 \\
v - v_0
\end{bmatrix} + \begin{bmatrix}a + u_0 \\
b + v_0
\end{bmatrix}
\]

where \( s \) is a scaling (or expansion) factor, \( C\theta \triangleq \cos(\theta) \) and \( S\theta \triangleq \sin(\theta) \), and \( \theta \) is the angle by which the object in \( I_1 \) is clockwise rotated with respect to its original orientation in \( I_0 \). Thus, this AFTR maps points \((u, v)\) from one frame \((I_0)\) into the corresponding points \((\tilde{u}, \tilde{v})\) in another frame \((I_1)\). In Fig. 3 we notice that, first, the object expanded about 50%, second, it rotated about 25° counterclockwise, and third, it moved up and right. This is indeed the order of mappings conveyed by the above definition although the order of scaling and rotation is immaterial. Notice that scaling and rotation are 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 \( [a \ b]^T \).

B. Vehicle Maneuvers and Image Plane Motion

Here we calculate the transformation that the projection of an object undergoes as a result of platform maneuvers so it can be related to the AFTR. Starting from the well-known equations for the temporal derivatives of the image-plane projections \((u, v)\) (see [17]),

\[
\begin{align*}
\dot{u} &= -fv_x + uv_z + \omega_z \frac{uv}{f} \\
&- f\omega_y \left(1 + \frac{u^2}{f^2}\right) + v\omega_z \\
\dot{v} &= -fv_y + vv_z - \omega_z \frac{uv}{f} \\
&+ f\omega_x \left(1 + \frac{v^2}{f^2}\right) - u\omega_z
\end{align*}
\]

where \( f \) is the focal length. Now consider the shifts experienced by the corners of the window shown in Fig. 4. The differences between their shifts can be used to yield rotation and expansion. The rotation of the top side of the square (where \( v_1 = v_0 \) during some
interframe time can be approximated by
\[
\frac{\Delta v_1 - \Delta v_0}{u_1 - u_0} = -\omega_z - \frac{v_0 u \omega_y}{f}. \tag{7}
\]

The rotation of the left side of the square (where \( u_0 = u_2 \)) is similarly found as
\[
\frac{\Delta u_2 - \Delta u_0}{v_0 - v_2} = -\omega_z - \frac{u_0 v \omega_x}{f}. \tag{8}
\]

When the point \((u_0, v_0)\) coincides with the image center (in Fig. 2), i.e., when \( (u_0, v_0) = (0,0) \), both expressions above reduce to \(-\omega_z\). Comparing the two terms on the right hand side of (7) (or (8)) for equal platform roll and yaw, the yaw (or pitch) term is smaller by a factor of \( f/v_0 \). At, say, 50 pixels from the FOE, and with \( f = 622 \) pixels (that of our camera), this factor is 12.4. As we see, the top and left sides rotate slightly differently, i.e., the square distorts, and this factor is 12.4. As we see, the top and left sides rotate slightly differently, i.e., the square distorts, and this rotation approximately equals the platform roll.

Next, let us analyze the expansion factor. For the top side of the square it is approximated by
\[
\frac{\Delta u_1 - \Delta u_0}{u_1 - u_0} = v_z + \frac{v_0 u \omega_y}{f} - \frac{(u_0 + u_1) \omega_y}{f}. \tag{9}
\]
and for the left side of the square by
\[
\frac{\Delta v_0 - \Delta v_2}{v_0 - v_2} = v_z - \frac{u_0 v \omega_x}{f} + \frac{(v_0 + v_2) \omega_x}{f}. \tag{10}
\]
Both expressions approach \( v_z \) at the same point \( p \) of Fig. 2 as the square size goes to zero. Again, horizontal and vertical lines expand slightly differently, but both converge onto \( v_z \), i.e., the time-to-collision inverse.

The above derivation shows that the image-plane rotation is well approximated by platform roll, and the expansion by \( v_z \). These approximations become equalities at the image center. In practical flight situations, the FOE is not too far from the image center. Since this algorithm is mainly intended to enhance depth derivation around the FOE point of the image plane, we conclude that the affine transformation represents a good approximation to the actual mapping that takes place, between different frames.

### C. 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 a unity matrix. This question has been investigated extensively by Mostafavi and Smithin \([27, 28]\). Their results are summarized below.

For images having a circularly symmetric Gaussian correlation function,
\[
R(r_u, r_v) = \exp\left\{-\frac{1}{2\Delta} \left( r_u^2 + r_v^2 \right) \right\} \tag{11}
\]
where \( r_u, r_v \) are the spatial shifts, and \( \Delta \) the “correlation width,” the effects of non-compensated rotation (by \( \theta \)) and/or scaling (by \( s \)) are determined by the combined geometric-distortion parameter \( d \), defined as
\[
d \equiv \sqrt{|1 - 2s \cos \theta + s^2|}
\approx \sqrt{(1-s)^2 + \theta^2} \quad \text{for small } \theta \text{ and } s \approx 1. \tag{12}
\]

Fig. 5 shows the effect of \( d \) on the peak-to-sidelobes ratio (PSR). Peak is the maximum value of the cross-correlation function, and “sidelobes” is its standard deviation far from the peak. The reference image is taken as a square of size \( L \times L \) and the sensed image is much larger. In the figure, \( L \) appears normalized by the correlation width because what counts is the effective number of “independent” objects. The graph for \( d = 0.087 \), for example, can be used for rotation alone (of \( 5^\circ \)), or for scaling alone (\( s = 1.087 \)), or for any of their combinations such that (12) yields \( d = 0.087 \). Fig. 6 similarly shows the behavior of the registration error.

Let us use an example to demonstrate the effect of uncompensated rotation or scaling errors. Take speed \( V_x = 25 \) m/s, depth \( z_0 = 120 \) m, a rolling maneuver of \( \omega_z = 20^\circ/s \), \( L = 21 \), \( \Delta = 1.5 \) pixels, and frame rate of \( 2 \) fis/s. This low frame rate is used to achieve a large triangulation baseline as is explained later. Only two consecutive frames are used in this example. In a single interframe time the platform rotates \( 10^\circ \), and there is an expansion by a factor of \( s = 120/(120 - 25 - 0.5) = 1.1163 \), so that \( d = 0.21 \). The PSR will incur a loss.
of \( \approx 3 \) (6 dB in PSR power), as read from Fig. 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 Fig. 6, will increase from \( \sigma_h = 0.025 \) to \( \sigma_h = 0.070 \) pixels. In [8] we have found the depth error:

\[
\sigma_z = \frac{\sqrt{2} h z}{bh}
\]

(13)

where \( h \) is the triangulation baseline and \( b \) the image-plane distance from the FOE (say, 10 pixels). Thus, the depth error incurred by a geometrically compensated algorithm (\( b = 12.5 \) m) is 4.1 m while that incurred by a noncompensated 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-correlating any two frames 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.

D. Converging on Correct Affine Transformation

We now derive the equations and algorithm necessary to obtain the required AFTR. Initially guessing this matrix, we use Newton’s equation (see [29]) iteratively to minimize an appropriate cost function and thereby solve for the correct matrix parameters.

The cost function \( J \) is defined as the integral over the window area \( A \), of the squared difference of image gray levels, that is,

\[
\epsilon \triangleq I_1(\tilde{u}, \tilde{v}) - I_0(u, v)
\]

\[
J \triangleq \frac{1}{2A} \int_{A} \epsilon^2 dA.
\]

(14)

If the mapping between all \((u, v)\) points inside the window (in \( I_0 \)) and the corresponding \((\tilde{u}, \tilde{v})\) (in \( I_1 \)) is correct, then the above cost should equal zero. In practice, however, we can only expect to minimize it. Newton’s method assumes that the cost function is quadratic and uses the Gradient and Hessian to solve for its minimum. Since this assumption only holds approximately, it is necessary, in practice, to iterate a few times on the Newton’s solution until it converges. The iterative update equation for the estimated parameter vector \( \hat{X}(k) \) becomes

\[
\hat{X}(k + 1) = \hat{X}(k) - \{\nabla^2 J[\hat{X}(k)]\}^{-1}\nabla J[\hat{X}(k)]
\]

(15)

where

\[
X(k) \triangleq [a \ b \ s \ \theta]^T
\]

(16)

and \((a, b)\) are the image-plane shifts, \( s \) the scaling factor, and \( \theta \) the rotation angle.

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

\[
\frac{\partial J}{\partial a} = \frac{1}{A} \int_{A} \frac{\partial \epsilon}{\partial a} dA = \frac{1}{A} \int_{A} \epsilon \frac{\partial I_1}{\partial a} dA
\]

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

\[
\frac{\partial I_1}{\partial a} = \frac{\partial I_1}{\partial \tilde{u}} \frac{\partial \tilde{u}}{\partial a} + \frac{\partial I_1}{\partial \tilde{v}} \frac{\partial \tilde{v}}{\partial a}
\]

(17)

Similar equations are obtained for the other three parameters by substituting them in place of \( a \) in (18). The above four equations require the partials of \( \tilde{u}, \tilde{v} \) with respect to all four parameters. These are obtained by differentiating the two scalar equations obtained from (5), that is,

\[
\tilde{u} = s[C \theta (u - u_0) - S \theta (v - v_0)] + u_0 + a
\]

\[
\tilde{v} = s[S \theta (u - u_0) + C \theta (v - v_0)] + v_0 + b
\]

(19)

so that

\[
\frac{\partial \tilde{u}}{\partial a} = 1
\]

\[
\frac{\partial \tilde{v}}{\partial a} = 0
\]

\[
\frac{\partial \tilde{u}}{\partial b} = 0
\]

\[
\frac{\partial \tilde{v}}{\partial b} = 1
\]

\[
\frac{\partial \tilde{u}}{\partial s} = C \theta (u - u_0) - S \theta (v - v_0)
\]

\[
\frac{\partial \tilde{v}}{\partial s} = S \theta (u - u_0) + C \theta (v - v_0)
\]

\[
\frac{\partial \tilde{u}}{\partial \theta} = -s[S \theta (u - u_0) + C \theta (v - v_0)]
\]

\[
\frac{\partial \tilde{v}}{\partial \theta} = s[C \theta (u - u_0) - S \theta (v - v_0)].
\]

(20)
We now need the ten second derivatives of the symmetrical matrix $\nabla^2 J[\hat{X}(k)]$. Let us start with one of the mixed second derivatives which can then serve as a template to find all the others. To simplify notation, we drop the “$dA$” from the integrals, the subscript 1 from $I$, and the tilde from $u,v$ whenever understood from the context. For the mixed derivative of $a$ and $\theta$, we thus have
\[
\frac{\partial^2 J}{\partial a \partial \theta} = \frac{\partial}{\partial a} \left( \frac{\partial J}{\partial \theta} \right) = \frac{1}{A} \int_A \frac{\partial \epsilon}{\partial a} \frac{\partial \epsilon}{\partial \theta}
\]
\[
+ \epsilon \frac{\partial}{\partial a} \left[ \frac{\partial I}{\partial u} \frac{\partial u}{\partial \theta} + \frac{\partial I}{\partial v} \frac{\partial v}{\partial \theta} \right].
\]
After some algebra, we get
\[
\frac{\partial^2 J}{\partial a \partial \theta} = \frac{1}{A} \int_A U \frac{\partial u}{\partial a} \frac{\partial u}{\partial \theta} + V \frac{\partial v}{\partial a} \frac{\partial v}{\partial \theta}
\]
\[
+ W \left[ \frac{\partial u}{\partial a} \frac{\partial u}{\partial \theta} + \frac{\partial u}{\partial \theta} \frac{\partial u}{\partial a} \right] + \epsilon \left[ \frac{\partial I}{\partial a} \frac{\partial^2 u}{\partial a^2} + \frac{\partial I}{\partial v} \frac{\partial^2 v}{\partial a^2} \right].
\]
where
\[
U \triangleq \left( \frac{\partial I}{\partial u} \right)^2 + \epsilon \frac{\partial^2 I}{\partial u^2}
\]
\[
V \triangleq \left( \frac{\partial I}{\partial v} \right)^2 + \epsilon \frac{\partial^2 I}{\partial v^2}
\]
\[
W \triangleq \epsilon \frac{\partial^2 I}{\partial u \partial v} + \frac{\partial I}{\partial u} \frac{\partial I}{\partial v}.
\]
The other mixed second derivatives of $J$ are obtained similarly. The nonmixed second derivatives are obtained by substituting the same parameter twice.

The above equations require two kinds of building blocks; the first and second spatial derivatives of the $I_1$ image and the first and second 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 $\times$ 3 window. Differentiating equations (20) yields 10 second derivatives for $\hat{u}$ and 10 for $\hat{v}$; all are zero except:
\[
\frac{\partial^2 u}{\partial s \partial \theta} = -S\theta(u - u_0) - C\theta(v - v_0) = -\frac{\partial v}{\partial s}
\]
\[
\frac{\partial^2 v}{\partial s \partial \theta} = C\theta(u - u_0) - S\theta(v - v_0) = \frac{\partial u}{\partial s}
\]
\[
\frac{\partial^2 u}{\partial \theta^2} = s[-C\theta(u - u_0) + S\theta(v - v_0)] = -\frac{\partial v}{\partial \theta}
\]
\[
\frac{\partial^2 v}{\partial \theta^2} = -s[C\theta(u - u_0) + S\theta(v - v_0)] = \frac{\partial u}{\partial \theta}.
\]
At this point all the components necessary for a single iteration on the Newton’s solution have been derived.

IV. SIMULATIONS OF COST FUNCTION AND ITS DERIVATIVES

We now examine the behavior of the cost function and its derivatives as a function of the four parameters in open loop, that is, without yet trying to correct the errors. For the following experimental results we used simulated imagery, where the scene consists of a wall normal to the initial flight trajectory. The wall is textured by a random Gaussian colored noise having spatial correlation width of 2 pixels in each dimension.

The error equations are, in principle, simulated as prescribed by (14)–(24), but, since we are dealing with spatially discretized images, it is necessary to implement these equations in a discrete form. 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. Note that a pixel’s gray-level is obtained by integrating the impinging radiation, emanating from the scene, over the area of the pixel during its integration (or exposure) time.

Differentiating between the gray-level of a pixel and the actual gray-level value of the scene at any continuous location on the image plane is important in estimating the scene values $I_1(\hat{u}, \hat{v})$ as required in (14) because $I_1(\hat{u}, \hat{v})$ are generally nonintegers. There is no such problem in estimating $I_0(u,v)$ because, by definition, we start from the center of a pixel (integer) and, thus, take its gray-level as the best estimate. For the estimation of $I_1(\hat{u}, \hat{v})$, we use interpolation as shown in Fig. 7.

Say we have an estimate for the value of the scene at the center point of some initial pixel $I_0(u_0,v_0)$. 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 4 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. This led us to use the rather ad hoc interpolation method:
\[
I_1(\hat{u}, \hat{v}) \triangleq (1 - \delta u)(1 - \delta v)I_1(u_0, v_0) + \delta v(1 - \delta u)I_1(u_0, v_0 + 1)
\]
\[
+ \delta u(1 - \delta v)I_1(u_0 + 1, \hat{v}) + \delta u \delta v I_1(u_0 + 1, \hat{v} + 1).
\]
This method has the advantage that it yields the expected results when \((\tilde{u}, \tilde{v})\) 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.

A. Open-Loop Error Measurements

An open-loop error measurement consists of the following procedure. Choose a single parameter of the AFTR matrix of (5), say, the scaling factor \(s\), as a scanned variable, and keep the other three parameters constant. When \(s\) is scanned, so are \((\tilde{u}, \tilde{v}), I_1(\tilde{u}, \tilde{v}), \) and \(J\) of (14). Assume that \(I_1\) is a scaled version of \(I_0\) by some \(s_0\). Then, when \(s\) passes the value \(s = s_0\) during scanning, the cost function \(J\) dips to its minimum and its derivatives behave correspondingly.

In the first set of open-loop error simulations we investigated the error sensitivity to the scaling factor \(s\) as a function of window size. The flight trajectory used for this set is nonmaneuvering and of 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 Fig. 8 to demonstrate the effect of expansion as the depth decreases from 150 to 138 to 126 m. Fig. 9(a) shows the case of a 21 \(\times\) 21 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 belong to the cost function and its first and second derivatives as derived earlier; the fourth shows the correction for \(s\) as calculated by the Newton’s algorithm of (15), i.e., the third component of \(\{\nabla^2 J(\tilde{X}(k))\}^{-1}\nabla J(\tilde{X}(k))\). The four graphs in each figure are scaled as necessary for convenient presentation. Fig. 9(b) shows the same for a 41 \(\times\) 41 window. 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 in the window to shift proportionately to its distance from the window center. 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 with each other; this is not so 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 use interpolation, can account for numerical discrepancies.

3) The correct values of \(s\) are shown by the vertical bars in all figures. It is noticed that they fall closer to the minima of the cost functions than to the zero
crossings of the first derivatives. We assume that these are noise-like inaccuracies resulting from the quantization and interpolation operations; they clearly diminish as the window size increases. Notice that it is the zero crossing of the derivative which 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 change in slope 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 detailed in [30]. Since the closed-loop algorithm performs around the actual \( s \) and not around \( s = 1 \), it is not affected by this phenomenon.

5) The curves of \( \Delta s \) give the calculated correction for the case where the error occurs in \( s \) alone. In such a case, the correction part of (15) simplifies to

\[
\Delta s = \frac{dJ}{ds} \frac{ds}{d^2J/ds^2}. \tag{26}
\]

It can be seen from the figures that \( \Delta s \) approximately agrees with this equation. Also, the discontinuities in the first and second derivatives at \( s = 1 \) cancel each other in (26) so that the \( \Delta s \) graphs do not show any discontinuity.

In the next set of error measurements we investigated the error sensitivity to rotation angle \( \theta \) 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 Fig. 10 to demonstrate the effect of rotation. Fig. 11(a) shows the case of a \( 21 \times 21 \) window size and Fig. 11(b) shows a \( 41 \times 41 \) window size which is centered on the FOE. The first and sixth frames are used for \( I_0 \) and \( I_1 \), respectively, so the total roll used in generating the figures is of \(-0.1\) rad. The following observations can be made.
Fig. 10. Frames 0, 4, and 8 of simulated textured wall as seen while rolling with no translational motion.

1) The absolute values of all four variables increase monotonically with the window size for the same reason they did in the $s$ curves.

2) The values of the cost-function and its first and second derivatives roughly agree.

3) The actual value of $\theta$ 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) The curves of $\Delta \theta$ give the calculated correction for the case where the error occurs in $\theta$ alone. In such a case (15) simplifies to

$$\Delta \theta = \frac{dJ}{d\theta}$$

In the figures $\Delta \theta$ approximately agrees with this equation.

In the last set we repeated the same for image-plane shifts $a$. Here the camera is stationary except that it is panning at 0.0005 rad/frame; it is initially pointing to the wall center. Images numbers 0 and 4 are used for $I_0$ and $I_1$. The time-sequence of panned images are not shown because they look indistinguishable—being shifted by only about one pixel. Fig. 12 shows the cases of a $21 \times 21$ and $41 \times 41$ windows which are centered on the FOE. We observe the following.

1) As opposed to the previous cases where $s$ or $\theta$ 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.

2) The correct $a$ values, as marked by the vertical bars, fall close to the minima of the cost functions and also to the zero crossings of the first derivatives.

3) The curves of $\Delta a$ give the calculated correction for the case where the error occurs in $a$ (or $b$) alone.

4) Fig. 13 shows the behavior of the cost function curve for large shifts—where it becomes highly nonlinear. 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 ($\pm 4$ pixels here). Inside this region the correction still shows the right sign. We later refer to this region as “the capture zone.”

B. Closed-Loop Performance

Here we summarize the results of closed-loop runs which are divided into two groups. The first group parallels the open-loop case of forward-flying. For brevity we skip the closed-loop parallels of the rotation-only and shift-only cases. The second run corresponds to maneuvers in all variables. In each run the errors are corrected using Newton’s method for six iterations; this is what we mean by “closing
the loop”. 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 with an experimental factor of 0.75 to prevent overshoots, and then, (3) bounding $\delta s$ by $\pm 0.03$, $\delta \theta$ by $\pm 0.03$ rad, and $\delta a, \delta b$ by $\pm 0.75$ 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: $s$, $\theta$, $a$, and $b$. In addition, there are four horizontal bars (arbitrarily located between iteration number 4 and 5) whose ordinates show the correct calculated values of the four parameters for ready visual comparison. The bars are marked by the parameter symbols.

For forward-flying with no maneuvers, $z_0 = 150$ m and $V_x = 1$ m/fr. The transformation parameters are calculated at frame number 4 by comparing it with frame 0 (skipping the intermediate frames). Fig. 14 demonstrates expansion alone for a window centered on the FOE, and expansion-plus-shift for a window centered at (20,20) pixels from the FOE. Based on these results and others (not shown here) we conclude the following.

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

2) The accuracies (especially for $s$) improve with the window size. For the case shown, the correct expansion is $150/146 = 1.0274$, corresponding to 146 frames-to-collision whereas the converged value is $s = 1.0296$, corresponding to 135 frames-to-collision.

3) The converged shifts for the (20,20) pixel practically show no error.

For a general maneuver, $V_x = 1$ m/s, $z_0 = 150$ m, pitch and yaw rates are 0.0005 rad/s each, and the roll-rate is 0.02 rad/s. The results are shown in Fig. 15.
The transformation parameters are calculated at frame number 2 by comparison with frame 0. As before, the system converges within two iterations, and the accuracies improve with the window size. The accuracy of $s$ is around 6% for the FOE point and 16% for the (20,20) point.

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

V. INCREASING THE TRIANGULATION BASELINE

In this section we use the above algorithm as the core on which a farther layer is built with the intention of increasing the accuracy and robustness of the practical algorithm.

A. Capture Zone

We have touched on the question of convergence in regard to Fig. 13 where 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 separately. Estimating the capture zone width is based on the bandwidth or correlation width of the images. For that, we used $\Delta = 1.5$ pixels (see (11)) in conjunction with Figs. 5 and 6.

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 locking onto
Fig. 13. Sensitivity of cost-function and its derivatives to shift over a wide range (21 × 21 window).

Fig. 14. Convergence for forward flying, no maneuvers, \( L = 21 \). (a) At FOE. (b) At (20,20) from FOE.
a wrong peak; this is equivalent to 15 dB in power ratio. Taking a window of size $21 \times 21$, $\Delta$ of 1.5 pixels is $\approx 0.07$ of the window size. Using the equations in [27]), we get the capture zone for $a$ (or $b$) as ±1.32 pixels. We similarly calculated the capture zone for the expansion factor as: $s = 0.871$ to 1.148, and for the rotation as: $\theta = \pm 8.5^\circ$. We have thus shown that the capture zone is quite wide. Images from real scenes are highly nonstationary so that $\Delta$ 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.

B. 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. Say we initially use frame 0 and frame 1, so the frame separation is one. The Newton’s equations are iterated on 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 (note that the first frame of the pair is kept fixed). 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 for some fixed frame separation; the outer loop iterates through increased frame separation. The implicit assumption here is that the flight trajectory is basically nonmaneuvering, or, in other words, it is the maneuvers which will determine the maximum usable triangulation baseline.

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,

$$u = f \frac{x}{z}, \quad v = f \frac{y}{z}.$$  \hspace{1cm} (28)
Let us project an object of length \( l \) onto the image plane and define its projection as unity. After decreasing the depth from \( z_0 \) to \( z_1 \), the projection changes to \( s_1 \). That can be written as

\[
1 = \frac{fl}{z_0}; \quad s_1 = \frac{fl}{z_1}; \quad z_1 = z_0 - Vz_1
\]

for a frame separation of \( t_1 \), from which

\[
s_1 = \frac{z_0}{z_0 - Vz_1}; \quad z_0 = \frac{s_1Vz_1}{s_1 - 1}
\]

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

\[
s_2 = \frac{s_1t_1}{t_2 - s_1(t_2 - t_1)}.
\]

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

\[
a_2 = a_1t_2/t_1; \quad b_2 = b_1t_2/t_1; \quad \theta_2 = \theta_1t_2/t_1.
\]

After the algorithm stops, (30) is used to calculate the current best estimate for the initial depth \( z_0 \), based on the last pair of \( s_1, t_1 \), which corresponds to the largest triangulation baseline that yielded convergence.

### C. Performance of Iterative Algorithm

Here we present results obtained by running the iterative algorithm on our simulated imagery. It is a nonmaneuvering forward-flying example with initial depth \( z_0 = 150 \) m and velocity of 2 m/s. The window of size \( 21 \times 21 \) is initially centered on pixel \( (74,74) \) (the FOE is at \( (64,64) \)). There are 40 frames in the set.

Frame 0 is always used as the basis for comparison, initially with frame 2, and then with frames 5, 10, 16, 22, 28, and 34. The initial conditions for \( a, b, s, \) and \( \theta \) for the first frame separation are 0.0, 0.0, 1.0, 0.0 because we do not know any better. The initial conditions for every iteration thereafter are predicted based on linear extrapolation, so that

\[
a_2 = a_1t_2/t_1; \quad b_2 = b_1t_2/t_1; \quad \theta_2 = \theta_1t_2/t_1.
\]

For the same example used earlier, where \( L/\Delta = 14 \), and assuming an image signal-to-noise ratio of 100, it is found from (33) that \( \sqrt{\text{var}[C_N(0,0)]} = 0.000177 \). 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 (30), where we had

$$z_0 = sV \frac{\Delta t}{s-1}$$

(34)

so that

$$\frac{dz_0}{z_0} = -\frac{ds}{s(s-1)} \approx -\frac{ds}{s-1}.$$  

(35)

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

$$\sigma_{zs} = \frac{\sigma_{z_0} s}{s-1}.$$  

(36)

For $s = 1.0274$, as was used to create Fig. 9 ($z_0 = 150$ m), and with $\sigma_s = 0.00325$, (36) yields $\sigma_{zs} = 17.8$ m which is close to the simulation results.

We assume that the depth information contained in $s$ and that contained in the lateral shifts $(a, b)$ are 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 this job here. However, we can still write down the combining algorithm for the initial depth unbiased estimate $\hat{z}_0$, as (see [29])

$$\hat{z}_0 = k z_s + (1-k) z_t$$  

(37)

where $z_s$ is the expansion-based depth measurement and $z_t$ the shifts-based one. $k$ is determined by the variances, $\sigma_{z_s}^2$ of $z_s$ and $\sigma_{z_t}^2$ of $z_t$, and by their correlation coefficient $\rho$, as

$$k = \frac{\sigma_{z_s}^2 - \rho \sigma_{z_s} \sigma_{z_t}}{\sigma_{z_s}^2 + \sigma_{z_t}^2 - 2 \rho \sigma_{z_s} \sigma_{z_t}}.$$  

(38)

and the minimum error is

$$E\{e^2\} = E\{(z_0 - \hat{z}_0)^2\}$$

$$= \frac{\sigma_{z_s}^2 \sigma_{s}^2 (1 - \rho^2)}{\sigma_{z_s}^2 + \sigma_{z_t}^2 - 2 \rho \sigma_{z_s} \sigma_{z_t}}.$$  

(39)

Close to the FOE, $\sigma_{zs} \ll \sigma_{zt}$ so that, irrespective of $\rho$, $k \ll 1$, and vice versa. This means that, even if we use some guessed $\rho$ 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 $\rho$, the proportions will not be optimal.

VII. SUMMARY

In this paper we developed a new expansion-based passive-ranging algorithm that can complement the existing shift-based algorithm in the image areas close to the FOE. The new algorithm estimates four parameters of geometrical distortion between images, which enables it to cross correlate far-apart frames, thus, to produce accurate results. This stands in contrast with respect to a shift-based algorithm which assumes zero geometrical distortion, and, thus, is limited in the frame time separation for cross correlation.

Derivation of depth from expansion is more robust in many ways compared with derivation from shifts. First, it is insensitive to the image-plane location, and, in particular, it performs best at the FOE, where shift-based algorithms are completely helpless. Second, it is insensitive to aircraft maneuvers, which do not even enter the solution, as they do in shift-based algorithms.

The significance of this work is that, using the new algorithm in conjunction with a shift-based one, can result in a robust and reliable monocular-vision depth estimation capability. 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

On the accuracy analysis of airborne techniques for passively locating electromagnetic emitters.

Application of linear statistical models to radar location techniques.

Statistical theory of passive location systems.

Effect of bias on bearing-only target location.

Determination of optical flow.

Simulation and analysis of image-based navigation system for rotorcraft low-altitude flight.

Kalman filter based range estimation for autonomous navigation using imaging sensors.

Error analysis of combined optical-flow and stereo passive ranging.

The interpretation of a moving retinal image.

Determining the instantaneous direction of motion from optical flow generated by a curvilinear moving observer.

Egomotion and relative depth map from optical flow.

Optic flow.

Invariant properties of the motion parallax field due to the movement of rigid bodies relative to an observer.

Local structure of movement parallax of the plane.

Obstacle avoidance using flow field divergence.

A diffusion mechanism for obstacle detection from size-change information.

Passive range estimation for rotocraft low altitude flight.

On-line vehicle motion estimation from visual terrain information. Part i: Recursive image registration.

On-line vehicle motion estimation for visual terrain information. Part ii: Ground velocity and position estimation.

Recursive image registration with application to motion estimation.

Air-to-surface range estimation by a stochastic gradient approach.

Looming detectors in the human visual pathway.

Visual responses to changing size and sideways motion for different directions of motion in depth: Linearization of visual responses.

Visual system responds to rotational and size-change components of complex proximal motion patterns.

Motion aftereffect: A global mechanism for the perception of rotation.

Visual responses to vorticity and the neural analysis of optic flow.

Image correlation with geometric distortion. Part i: Acquisition performance.

Image correlation with geometric distortion. Part ii: Effect on local accuracy.

Applied Optimal Estimation.

Expansion-based passive ranging.
Yair Barniv earned his first degree in electrical engineering from the Technion—Israel Institute of Technology. He earned his M.Sc. and Ph.D. degrees at Carnegie–Mellon University, Pittsburgh, PA, in digital/optical signal processing.

Early in his career he worked on the development of airborne and ground radars for fire-control systems and air-traffic control. Later, in the aircraft industry, he worked on missile guidance design, implementation, integration, and field testing. During that nine year period he also did performance analysis and feasibility/cost-effectiveness studies in a broad spectrum of projects related to electro-acoustics, electro-optics, electronic countermeasures (ECM), radar, and navigation. After graduation from Carnegie–Mellon University he joined Systems Control Technology. During this eight year period he did research on cruise-missile path optimization, closed-loop ECM, synthetic aperture radar (SAR) application to homing, ballistic missiles interception, fleet antimissile defense, and others. His main contribution during that period was an innovative approach to streak detection in IR imagery using dynamic programming. Recently Dr. Barniv has developed new techniques in the areas of nonacoustic ASW, velocity-discriminating SAR, and passive ranging for helicopter nap-of-the-Earth obstacle avoidance based on optical flow and stereo vision. Currently he is working on biological/neural-nets implementation of passive ranging algorithms.