PASSIVE RANGING USING IMAGE EXPANSION

Yair Barniv, 22 April 1993

Abstract

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.

Keywords:
Divergence, Optical Flow, Passive Depth Estimation, Object Expansion, Triangulation.

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 (OF), 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 in the image plane. The trajectories of all imaged points constitute the 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. As with all other passive-ranging methods, it is assumed that the scene and its illumination sources are temporally constant (see [5]), and that all points belonging to the same object share the same range.

The OF at any given point in the image plane may belong to any 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, is defined by 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 [6, 7], make use of the lateral motion alone. Two basic limitations are implicit in these methods. First, they perform poorly in the image plane close to the focus of expansion (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 unaccounted-for parameters. "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 [8], the depth-error is inversely proportional to the triangulation baseline (see (13) ahead).

In this work we discuss methods of extracting depth information from the divergence of the OF as it is approximately obtained from the affine transformation matrix. We use the term "divergence" (or "local divergence") to refer to the mathematical definition of the derivative-vector operator denoted by ∇ which 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.

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 Figure 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. In this paper we assume textured objects, so our algorithm roughly derives the equivalent of local divergence.

Figure 1: Texture and size cues

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, 19, 20, 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, 23, 24] as well as *vorticity* detectors [24, 25, 26].

The organization of this report is as follows. Section 2 contains the theory of 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 and Section 6 discusses error analysis.

2 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] to [16].

Figure 2: The geometry of projection onto the image plane

Consider the projection, p, of some point P onto the surface of a spherical camera as shown in Figure 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 $\nabla z \triangleq [z_x \; z_y]^T$ exists (T denotes the matrix transpose operation). The distance of any point on that surface from the sphere's center can then be approximated in the neighborhood of P by

$$z \approx z_0 + [x \; y] \cdot \nabla z \; , \tag{1}$$

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 $\mathbf{V} \triangleq [V_x \; V_y \; V_z]^T$ and its rotational velocity $\omega \triangleq [\omega_x \; \omega_y \; \omega_z]^T$. It is convenient to normalize $\mathbf{V}$ by z_0 and denote $[v_x \; v_y \; v_z] \triangleq [V_x \; V_y \; V_z]/z_0$.

The motion of the camera causes the stationary point P and its surrounding to describe a 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 $\mathbf{v}(\mathrm{p}) \triangleq [\dot{u}\ \dot{v}]_p^T$. The spatial partial derivatives of $\dot{u}$ and $\dot{v}$ are denoted by $\dot{u}_x, \dot{u}_y, \dot{v}_x, \dot{v}_y$. From [9], the following equations hold at p,

$$\begin{aligned} \dot{u} &= -v_x - \omega_y \ , & \dot{v} &= -v_y + \omega_x \ , \\ \dot{u}_x &= v_z + v_x z_x \ , & \dot{v}_y &= v_z + v_y z_y \ , \\ \dot{u}_y &= \omega_z + v_x z_y \ , & \dot{v}_x &= -\omega_z + v_y z_x \end{aligned} \tag{2}$$

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

$$\mathrm{div(p)} \triangleq \dot{u}_x + \dot{v}_y = 2v_z + [v_x\ v_y] \cdot \nabla z \tag{3}$$

To interpret the above equation, suppose the camera only moves in the Z direction, that is, $v_x = v_y = 0$, so that div(p)$= 2v_z = 2V_z/z_0$. Thus, div(p) is twice the reciprocal of the time-to-collision of P with the camera's center. 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. 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 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

$$\chi(R) \triangleq \frac{1}{A(R)} \int_R \mathrm{div(p)}\ \mathrm{d}s = \frac{1}{A(R)} \frac{\mathrm{d}A(R)}{\mathrm{d}t} \ , \tag{4}$$

where $A(R)$ is the object area and $\mathrm{d}s$ is the elemental area. In words, the global divergence equals the temporal rate of change of the normalized object area.

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 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 Figure 3, we define our specific AFTR by

$$\begin{bmatrix} \tilde{u} \\ \tilde{v} \end{bmatrix} = s \begin{bmatrix} \mathrm{C}\theta & -\mathrm{S}\theta \\ \mathrm{S}\theta & \mathrm{C}\theta \end{bmatrix} \begin{bmatrix} u - u_0 \\ v - v_0 \end{bmatrix} + \begin{bmatrix} a + u_0 \\ b + v_0 \end{bmatrix}, \tag{5}$$

where s is a scaling (or expansion) factor, $\mathrm{C}\theta \triangleq \cos(\theta)$ and $\mathrm{S}\theta \triangleq \sin(\theta)$, and θ is the angle

Figure 3: Mapping of a point through the affine transformation

by which the object in I_1 is clock-wise 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 Figure 3 we notice that, first, the object expanded about 50%, second, it rotated about 25^0 counter-clock-wise, 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 $[a\ b]^T$.

3.2 Vehicle maneuvers and image plane motion

Here we calculate the transformation an object's projection 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]),

$$\dot{u} = -fv_x + uv_z + \omega_x \frac{uv}{f} - f\omega_y(1 + \frac{u^2}{f^2}) + v\omega_z$$

Figure 4: Window

$$\dot{v} = -fv_y + vv_z - \omega_y \frac{uv}{f} + f\omega_x(1 + \frac{v^2}{f^2}) - u\omega_z \ , \tag{6}$$

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

$$\frac{\Delta v_1 - \Delta v_0}{u_1 - u_0} = -\omega_z - \frac{v_0 \omega_y}{f} \tag{7}$$

The rotation of the left side of the square (where $u_0 = u_2$) is similarly found as

$$\frac{\Delta u_2 - \Delta u_0}{v_0 - v_2} = -\omega_z - \frac{u_0 \omega_x}{f} \tag{8}$$

When the point (u_0, v_0) coincides with the image center (p in Figure 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 (our camera's), 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

$$\frac{\Delta u_1 - \Delta u_0}{u_1 - u_0} = v_z + \frac{v_0 \omega_x}{f} - \frac{(u_0 + u_1)\omega_y}{f} \tag{9}$$

and for the left side of the square by

$$\frac{\Delta v_0 - \Delta v_2}{v_0 - v_2} = v_z - \frac{u_0 \omega_y}{f} + \frac{(v_0 + v_2)\omega_x}{f} \ , \tag{10}$$

Both expressions approach v_z at the same point p of Figure 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.

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 a unity matrix. This question has been investigated extensively by Mostafavi and Smith in [27, 28]. Their results are summarize below.

For images having a circularly symmetric Gaussian correlation function,

$$R(\tau_u, \tau_v) = \exp\left\{-\frac{1}{2\Delta^2}[\tau_u^2 + \tau_v^2]\right\} , \tag{11}$$

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, defined as

$$d \triangleq \sqrt{|\ 1 - 2s\cos\theta + s^2\ |} \approx \sqrt{(1-s)^2 + \theta^2} \ \text{ for small } \theta \text{ and } s \approx 1 \tag{12}$$

Figure 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 5^0), or for scaling alone ($s = 1.087$), or for any of their combinations such that (12) yields $d = 0.087$. Figure 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 $\omega_z = 20^0$/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,

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

as extrapolated from Figure 6, will increase from $\sigma_h = 0.025$ to $\sigma_h = 0.070$ pixels. In [8] we have found the depth error:

$$\sigma_z = \frac{\sqrt{2} z_0^2 \sigma_h}{bh} , \tag{13}$$

where b is the triangulation baseline and h 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 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-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.

3.4 Converging on the correct affine transformation

We now derive the equations and algorithm necessary to obtain the required affine transformation. 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,

$$\epsilon \triangleq I_1(\tilde{u}, \tilde{v}) - I_0(u, v) ; \qquad J \triangleq \frac{1}{2A} \iint_A \epsilon^2 \mathrm{d}A \tag{14}$$

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

$$\hat{X}(k+1) = \hat{X}(k) - \left\{ \nabla^2 J[\hat{X}(k)] \right\}^{-1} \nabla J[\hat{X}(k)] , \tag{15}$$

where

$$X(k) \triangleq [a \;\; b \;\; s \;\; \theta]^T , \tag{16}$$

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

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

$$\frac{\partial J}{\partial a} = \frac{1}{A} \iint_A \epsilon \frac{\partial \epsilon}{\partial a} \mathrm{d}A = \frac{1}{A} \iint_A \epsilon \frac{\partial I_1}{\partial a} \mathrm{d}A , \tag{17}$$

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

$$\frac{\partial I_1}{\partial a} = \frac{\partial I_1}{\partial \tilde{u}}\frac{\partial \tilde{u}}{\partial a} + \frac{\partial I_1}{\partial \tilde{v}}\frac{\partial \tilde{v}}{\partial a} \tag{18}$$

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,

$$\begin{aligned} \tilde{u} = &\quad s[\mathrm{C}\theta(u-u_0) - \mathrm{S}\theta(v-v_0)] + u_0 + a \\ \tilde{v} = &\quad s[\mathrm{S}\theta(u-u_0) + \mathrm{C}\theta(v-v_0)] + v_0 + b\,, \end{aligned} \tag{19}$$

so that

$$\begin{aligned} \frac{\partial \tilde{u}}{\partial a} &= 1\ ; & \frac{\partial \tilde{v}}{\partial a} &= 0 \\ \frac{\partial \tilde{u}}{\partial b} &= 0\ ; & \frac{\partial \tilde{v}}{\partial b} &= 1 \\ \frac{\partial \tilde{u}}{\partial s} &= \mathrm{C}\theta(u-u_0) - \mathrm{S}\theta(v-v_0)\ ; & \frac{\partial \tilde{v}}{\partial s} &= \mathrm{S}\theta(u-u_0) + \mathrm{C}\theta(v-v_0) \\ \frac{\partial \tilde{u}}{\partial \theta} &= -s[\mathrm{S}\theta(u-u_0) + \mathrm{C}\theta(v-v_0)]\ ; & \frac{\partial \tilde{v}}{\partial \theta} &= s[\mathrm{C}\theta(u-u_0) - \mathrm{S}\theta(v-v_0)] \end{aligned} \tag{20}$$

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 derivative 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 θ, we thus have

$$\frac{\partial^2 J}{\partial a \partial \theta} = \frac{\partial}{\partial a}\left(\frac{\partial J}{\partial \theta}\right) = \frac{1}{A}\iint_A \frac{\partial \epsilon}{\partial a}\frac{\partial \epsilon}{\partial \theta} + \epsilon\frac{\partial}{\partial a}\left[\frac{\partial I}{\partial u}\frac{\partial u}{\partial \theta} + \frac{\partial I}{\partial v}\frac{\partial v}{\partial \theta}\right] \tag{21}$$

After some algebra, we get

$$\frac{\partial^2 J}{\partial a \partial \theta} = \frac{1}{A}\iint_A U\frac{\partial u}{\partial a}\frac{\partial u}{\partial \theta} + V\frac{\partial v}{\partial a}\frac{\partial v}{\partial \theta} + W\left[\frac{\partial u}{\partial a}\frac{\partial v}{\partial \theta} + \frac{\partial u}{\partial \theta}\frac{\partial v}{\partial a}\right] + \epsilon\left[\frac{\partial I}{\partial u}\frac{\partial^2 u}{\partial a \partial \theta} + \frac{\partial I}{\partial v}\frac{\partial^2 v}{\partial a \partial \theta}\right]\ , \tag{22}$$

where

$$U \triangleq \left(\frac{\partial I}{\partial u}\right)^2 + \epsilon\frac{\partial^2 I}{\partial u^2}\ ; \quad V \triangleq \left(\frac{\partial I}{\partial v}\right)^2 + \epsilon\frac{\partial^2 I}{\partial v^2}\ ; \quad W \triangleq \epsilon\frac{\partial^2 I}{\partial u \partial v} + \frac{\partial I}{\partial u}\frac{\partial I}{\partial v} \tag{23}$$

The other mixed second derivatives of J are obtained similarly. The second non-mixed 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 $\tilde{u}$ and $\tilde{v}$ 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. Differentiating equations (20) yields 10 second derivatives for $\tilde{u}$ and 10 for $\tilde{v}$; all are zero except:

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

$$\frac{\partial^2 u}{\partial \theta^2} = s[-\mathrm{C}\theta(u - u_0) + \mathrm{S}\theta(v - v_0)] = -\frac{\partial v}{\partial \theta}; \quad \frac{\partial^2 v}{\partial \theta^2} = -s[\mathrm{S}\theta(u - u_0) + \mathrm{C}\theta(v - v_0)] = \frac{\partial u}{\partial \theta} \quad (24)$$

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 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 equations (14) to (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 pixel's area during its integration (or exposure) time.

Differentiating between a pixel's gray-level and the actual gray-level value of the scene at any continuous location on the image plane is important in estimating the scene values $I_1(\tilde{u}, \tilde{v})$ as required in (14) because $(\tilde{u}, \tilde{v})$ are generally non-integers. 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 $I_1(\tilde{u}, \tilde{v})$, we use interpolation as shown in Figure 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 $(\tilde{u}, \tilde{v})$ in image I_1, and we want to estimate $I_1(\tilde{u}, \tilde{v})$. 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

Figure 7: The interpolation method

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:

$$\begin{aligned} I_1(\tilde{u}, \tilde{v}) &\triangleq (1-\delta u)(1-\delta v) I_1(\tilde{u}_0, \tilde{v}_0) + \delta v(1-\delta u) I_1(\tilde{u}_0, \tilde{v}_0 + 1) \\ &\quad + \delta u(1-\delta v) I_1(\tilde{u}_0 + 1, \tilde{v}_0) + \delta u \delta v I_1(\tilde{u}_0 + 1, \tilde{v}_0 + 1) \end{aligned} \tag{25}$$

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.

4.1 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 $(\tilde{u}, \tilde{v})$, $I_1(\tilde{u}, \tilde{v})$, ϵ, 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 non-maneuvering 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 Figure 8 to demonstrate the effect of expansion as the depth decreases from 150 to 138 to 126 m. Figure 9 (top) shows the case of a 21×21 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

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, s; $L = 21$ (top), $L = 41$ (bot)

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 $\left\{\nabla^2 J[\hat{X}(k)]\right\}^{-1} \nabla J[\hat{X}(k)]$. The four graphs in each figure are scaled as necessary for convenient presentation. At the bottom, Figure 9 shows the same for a 41×41 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 Δs give the calculated correction for the case where the error occurs in s alone. In such a case, the correction part of equation (15) simplifies to

$$\Delta s = \frac{\mathrm{d}J/\mathrm{d}s}{\mathrm{d}^2 J/\mathrm{d}s^2} \tag{26}$$

   It can be seen from the figures that Δs 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 Δs graphs do not show any discontinuity.

In the next set of error measurements we investigated the error sensitivity to rotation angle, θ, 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

Figure 10: Frames 0, 4, and 8 of simulated textured wall as seen while rolling with no translational motion

Figure 11: Sensitivity of the cost-function and its derivatives to rotation, θ; $L = 21$ (top), $L = 41$ (bot)

of 150 m. The set of 3 images (number 0, 4, and 8) are shown in Figure 10 to demonstrate the effect of rotation. Figure 11 shows the cases of a 21×21 window size (top) and 41×41 (bottom) which are 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.

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

$$\Delta\theta = \frac{\mathrm{d}J/\mathrm{d}\theta}{\mathrm{d}^2J/\mathrm{d}\theta^2} \tag{27}$$

   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. Figure 12 shows the cases of a 21×21 and 41×41 windows which are centered on the FOE. We observe that:

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.

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 Δa give the calculated correction for the case where the error occurs in a (or b) alone.

4. Figure 13 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

Figure 12: Sensitivity of the cost-function and its derivatives to shift; $L = 21$ (top), $L = 41$ (bot)

Figure 13: Sensitivity of the cost-function and its derivatives to shift over a wide range (21×21 window)

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. We later refer to this region as "the capture zone".

4.2 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 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 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). Figure 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 that:

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. This is especially impressive because these shifts are small—only (0.548,0.548) pixels.

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 Figure 15. The transformation parameters are calculated at frame number 2 by comparison with frame 0. As before, the

-bb-error = =

-bb-error = =

Figure 14: Convergence for forward flying, no maneuvers, $L = 21$: at the FOE (top), at (20,20) from the FOE (bot)

-bb-error = =

-bb-error = =

Figure 15: Convergence for general maneuvers at the FOE (top) and at (20,20) from the FOE (bot)

.

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.

5 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.

5.1 The capture zone

We have touched on the question of convergence in regard to Figure 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 Figures 5, 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 a wrong peak; this is equivalent to 15 dB in power ratio. Taking a window of size 21×21, Δ of 1.5 pixels is ≈ 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: $\theta = \pm 8.5^0$. We have thus shown that the capture zone is quite wide. Images from real scenes are highly non-stationary so 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. 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 non-maneuvering, 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,

$$u = f\frac{x}{z}\ ; \qquad v = f\frac{y}{z} \tag{28}$$

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

$$1 = \frac{fl}{z_0}\ ; \qquad s_1 = \frac{fl}{z_1}\ ; \qquad z_1 = z_0 - V_z t_1\ , \tag{29}$$

for a frame separation of t_1, from which

$$s_1 = \frac{z_0}{z_0 - V_z t_1}\ ; \qquad z_0 = \frac{s_1 V_z t_1}{s_1 - 1} \tag{30}$$

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

$$s_2 = \frac{s_1 t_1}{t_2 - s_1(t_2 - t_1)} \tag{31}$$

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

$$a_2 = a_1 t_2/t_1\ ;\ \ b_2 = b_1 t_2/t_1\ ;\ \ \theta_2 = \theta_1 t_2/t_1 \tag{32}$$

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 $s_i,\ t_i$ which corresponds to the largest triangulation baseline that yielded convergence.

5.3 Performance of the iterative algorithm

Here we present results obtained by running the iterative algorithm on our simulated imagery. It is a non-maneuvering, forward-flying example with initial depth $z_0 = 150$ m and velocity

of 2 m/s. The window of size 21×21 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 θ 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 the converged values of the previous frame separation.

Figure 16: Depth convergence with iterations (increased triangulation baseline).

Figure 16 shows how the initial-depth estimate improves with the iteration number (or frame separation). The final result was obtained from frame pair (0,34). We have thus effectively used a triangulation baseline of 68 m which constitutes a substantial fraction of the initial depth. The final result here is 0.178% accurate. We have run the algorithm on various other simulated and real-data cases—at and around the FOE. Generally, the depth accuracies are very good (around 5%), and they improve as we get closer to the FOE.

6 Error analysis

Here 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 (13) where σ_h can be read from Figure 6.

The accuracy of the depth derived from expansion is determined by the accuracy of the expansion factor s. When all 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 scaling factor alone. The s accuracy is determined by the additive noise at the peak (denoted by $C_N(0,0)$). This noise has been neglected so far 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 [27] as

$$\mathrm{var}\{C_N(0,0)\} = L^{-2}\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} g(u,v)R(u,v)R_N(u,v)\mathrm{d}u\mathrm{d}v \ , \tag{33}$$

where $R_N(u,v)$ is the additive-noise correlation function.

Figure 17: Loss in correlation peak value due to residual errors in scaling factor

For simplicity we use equal $R(\tau_u,\tau_u)$ and $R_N(\tau_u,\tau_u)$ —both given by (11). We want to find the change in s 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 [27], is plotted in Figure 17. 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 $\sqrt{\mathrm{var}\{C_N(0,0)\}} = 0.000177$. 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 (30), where we had

$$z_0 = \frac{sV_z\Delta t}{s-1} \ , \tag{34}$$

so that

$$\frac{\mathrm{d}z_0}{z_0} = -\frac{\mathrm{d}s}{s(s-1)} \approx -\frac{\mathrm{d}s}{s-1} \tag{35}$$

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

$$\sigma_{zs} = \frac{\sigma_s z_0}{s-1} \tag{36}$$

For $s = 1.0274$, as was used to create Figure 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])

$$\hat{z}_0 = kz_s + (1-k)z_t \ , \tag{37}$$

where z_s is the expansion-based depth measurement and z_t the 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

$$k = \frac{\sigma_{zt}^2 - \rho\sigma_{zt}\sigma_{zs}}{\sigma_{zs}^2 + \sigma_{zt}^2 - 2\rho\sigma_{zt}\sigma_{zs}} \ , \tag{38}$$

and the minimum error is

$$\mathrm{E}\{e^2\} \triangleq \mathrm{E}\{(z_0 - \hat{z}_0)^2\} = \frac{\sigma_{zt}^2\sigma_{zs}^2(1-\rho^2)}{\sigma_{zs}^2 + \sigma_{zt}^2 - 2\rho\sigma_{zt}\sigma_{zs}} \tag{39}$$

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 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 crosscorrelate 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 crosscorrelation.

Derivation of depth from expansion is more robust in many ways compared to 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

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

[6] 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.

[7] 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.

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

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

[10] 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.

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

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

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

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

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

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

[23] 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.

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

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

[26] 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.

[27] 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.

[28] 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.

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

[30] Y. Barniv. Expansion-based passive ranging. Technical Memorandum 104025, NASA, Ames Research Center, Moffett Field, CA, June 1993.