NASA Technical Memorandum 102802

Velocity Filtering Applied to Optical Flow Calculations

Yair Barniv

August 1990

National Aeronautics and Space Administration

NASA Technical Memorandum 102802

Velocity Filtering Applied to Optical Flow Calculations

Yair Barniv, Ames Research Center, Moffett Field, California

August 1990

National Aeronautics and
Space Administration

Ames Research Center
Moffett Field, California 94035-1000

CONTENTS

ABBREVIATIONS

CFAR	constant false alarm rate
DF	delta function
FOE	focus of expansion
FOV	field of view
FP	floating point
FT	Fourier transform
HPF	high-pass filtering
LPF	low-pass filtering
MF	matched filter
PSF	point-spread function
SNR	signal-to-noise ratio
2-D	two-dimensional
3-D	three-dimensional

SUMMARY

Optical flow is a method by which a stream of two-dimensional images obtained from a passive sensor is used to map the three-dimensional surroundings of a moving vehicle. The primary application of the optical-flow method is in helicopter obstacle avoidance. Optical-flow calculations can be performed to determine the 3-D location of some chosen objects, or, in principle, of all points in the field of view of the moving sensor as long as its trajectory is known. Locating a few objects, rather than all three-dimensional points, seems to require fewer calculations but it requires that the objects between successive frames be identified. The application of velocity filtering to this problem eliminates the need for identification altogether because it automatically tracks *all* moving points based on the assumption of constant speed and gray level per point. Velocity filtering is a track-before-detect method and, as such, it not only tracks but also integrates the energy of each pixel during the whole observation time, thus enhancing the signal-to-noise ratio of each tracked pixel. This method is, in addition, very efficient computationally because it is naturally amenable to parallel processing. A substantial amount of engineering and experimental work has been done in this field, and the purpose here is to expand on the theoretical understanding and on the performance analysis of the velocity filtering method, as well as to present some experimental results.

1 INTRODUCTION

Optical flow is a method by which a stream of two-dimensional (2-D) images obtained from a passive sensor is used to map the three-dimensional (3-D) surroundings of a moving vehicle. The primary application of the optical-flow method is in helicopter obstacle avoidance. Optical-flow calculations can be performed to determine the 3-D location of some chosen objects (referred to as feature-based) or, in principle, of all points in the field of view (FOV) of the moving sensor (referred to as pixel-based). In both cases it is assumed that the flight trajectory is known. Locating a few objects rather than all 3-D points seems to require fewer calculations, but it does require that the objects be identified between successive frames. In pixel-based methods there is no need to identify objects since every pixel is defined to constitute an object, and it is recognizable by its (assumed) constant gray level and its approximate location.

Velocity filtering is a track-before-detect method; as such, it not only tracks but also integrates the energy of each pixel during the whole observation time, thus enhancing the signal-to-noise ratio (SNR) of each tracked pixel. This method is, in addition, very efficient computationally, because it is naturally amenable to parallel processing. In this paper we develop and apply the theory of velocity filtering to pixel-based passive ranging via optical flow. The basic theory of the velocity filter as realized in the frequency domain is given in reference 1. Application of this method—still in its frequency-domain interpretation—to the understanding of human visual motion sensing can be found in reference 2, and application to object tracking in references 3-5. The method of "velocity filtering" has been suggested as a pixel-based solution in reference 6. More detailed comments about these references will be presented in the next section.

The purpose of this paper is to provide some insight into the performance of the velocity-filtering method as applied to optical flow in order to be able to answer such questions as the following.

1. What is the smallest object that can be detected, or, equivalently, what is the minimum required contrast of a single pixel for reliable detection (velocity measurement) as a function of its image speed?

2. What is the required FOV to detect a given-size object at a given range and under given visibility conditions (fog, rain, or haze)?

3. How many velocity filters are required, and, as a result, what is the required computational throughput and memory?

Toward that end, the following tasks, which, so far as I know are areas of original work, are addressed.

1. Derive and interpret the space-domain shift-and-add algorithm from the general 3-D matched-filter formulation.

2. Develop a formulation for the velocity-passband of the filter and a method for covering a given 2-D velocity domain with complementing filters.

3. Develop the velocity-profile (hyperbolic) filter for optical-flow use and calculate its depth passband and accuracy.

4. Implement the optical-flow shift-and-add version of the velocity-profile filtering algorithm and develop appropriate preprocessing and interpolation methods.

In terms of an overall sensor/computer system, there is always a variety of trade-offs available to the system designer. For example, a very narrow FOV will enable the detection/tracking of small obstacles at a long distance in fog, but it will also provide a very limited view of the world. In this work, some tools are developed that can help the designer make these trade-offs.

Section 2 summarizes some of the basic concepts regarding the representation of moving objects in time-augmented space and the interpretation of the resulting Fourier transforms (FTs). The FT of a finite-time moving-point is given in section 3, its corresponding matched-filter (MF) is derived in section 4, and the MF is implemented in section 5. Section 6 deals with the passband and resolution of the velocity filter. In section 7 the general velocity filter formulation is applied to the optical-flow problem and is modified to also apply in the case of non-constant object velocities. In section 8 the new algorithm is implemented, and its performance on two real-imagery sets is discussed.

2 PRELIMINARIES

The basic concepts and mathematical tools that are applied in subsequent sections are presented here.

2.1 Motion as Time-Augmented Space

The motion of a physical point in one or two dimensions can be described as a line or a surface in a space that is augmented by the time dimension. Let us start with the simplest example of a point moving at some constant speed along the x-axis of a 1-D coordinate system. If the x-location and the corresponding times for each value of x are listed, this relationship can be plotted as a straight line in a 2-D coordinate system of (x, t). For a point that started from x_0 at t_0, and travels at a constant speed v, an equation of a

straight line is obtained: $x = x_0 + v(t - t_0)$ as shown in figure 1. I. If, instead of a constant speed, there is a constant acceleration, a parabola in the (x, t) space would be obtained.

Figure 1 also shows an example of a constant-velocity point in the (x, y) plane that describes a straight line in the 3-D space (x, y, t). This line can be given in two different forms; one is a parametric form in which t serves as a parameter, that is, $(x_0 + v_x t, y_0 + v_y t, t)$, and the other is as an intersection of two planes: the plane parallel to the y-axis, $x = x_0 + v_x t$, and the plane parallel to the x-axis, $y = y_0 + v_y t$. The parametric form has the advantage that it can be easily related to the vector $(v_x, v_y, 1)$ which is parallel to the line.

2.2 The Dirac Delta Function

The Dirac delta function $x(t) = \delta(t)$ can be defined as the limit when $T \to 0$ of a rectangle pulse function centered on $t = 0$ whose width is T and whose height is $1/T$, or more generally by the properties

$$\begin{cases} \delta(t) = 0 & t \neq 0 \\ \int_{-\infty}^{+\infty} \delta(t)\,dt = 1 \end{cases} \tag{1}$$

The most important property of the delta function is its sifting property, that is,

$$\int_{-\infty}^{+\infty} \delta(t - \lambda) f(\lambda)\,d\lambda = f(t) \tag{2}$$

One can similarly define a 2-D Dirac delta function such that its integral between $\pm\infty$ in 2-D equals 1, that is,

$$\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} \delta^2(x, y)\,dx dy = 1 \tag{3}$$

and it has the sifting property

$$\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} f(x, y)\,\delta^2(x - x_1, y - y_1)\,dx dy = f(x_1, y_1) \tag{4}$$

where the superscript 2 differentiates between the 1- and 2-D delta functions. From equation 2 we see that $\delta^2(x, y)$ can be replaced by $\delta(x)\delta(y)$ and still yield the same result, because the 2-D integral could then be separated into two 1-D integrals. This is why, for all practical purposes, the following equality (see ref. 7) can be used:

$$\delta^2(x, y) = \delta(x)\delta(y) \tag{5}$$

Another useful equality that can be proven by a simple change of variables is

$$\delta^2(ax, by) = \frac{1}{|ab|}\delta^2(x, y) \tag{6}$$

2.3 3-D Fourier Transform of a Moving Point

Earlier in this section, the trajectory or geometrical location of a point moving at a constant velocity on the plane was discussed. Thus, there is a functional relationship between location and time of the form

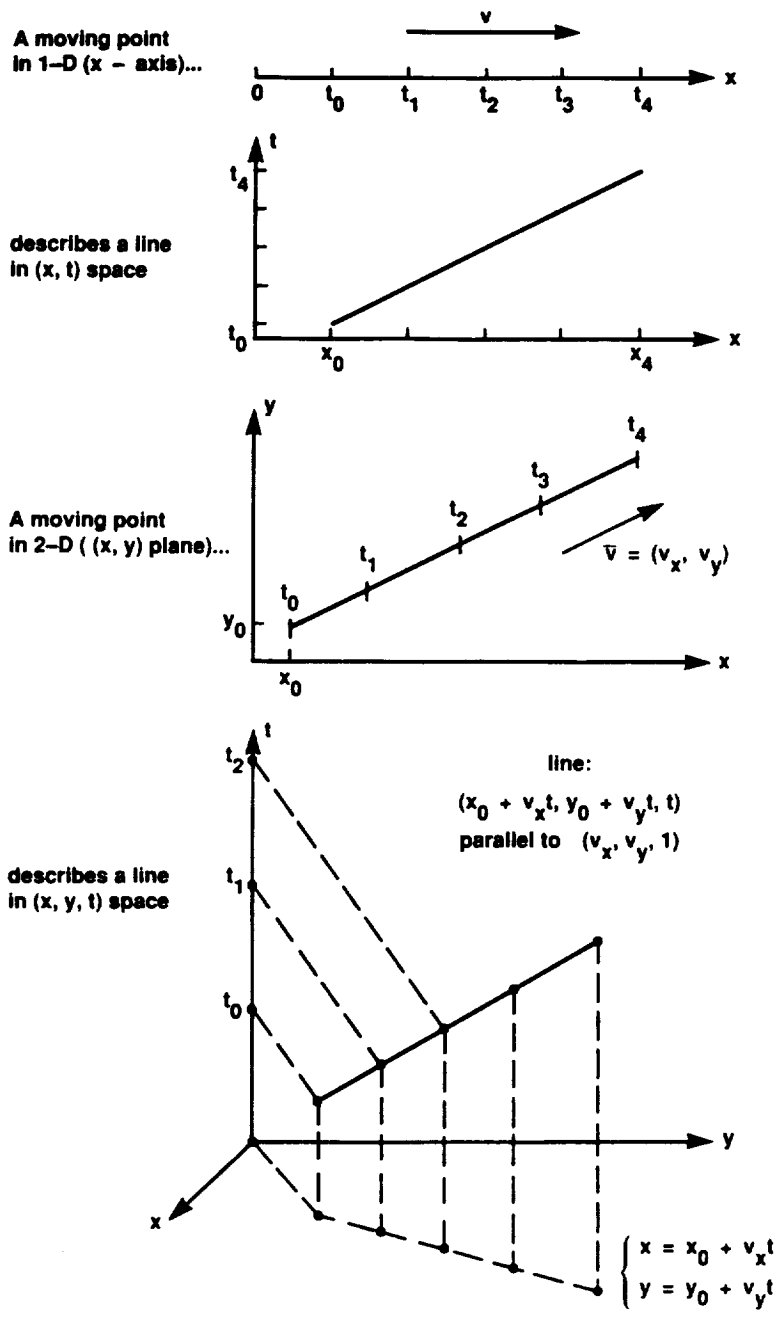

Figure 1. Motion translates to time-augmented space.

$x(t)$, $y(t)$. Here we want to refer to the amplitude of the point (in addition to its trajectory) which is a function of location *and* time.

A point that moves on the (x, y) plane with a fixed velocity vector $\bar{v} \equiv (v_x, v_y)$ can be described as a moving 2-D delta function (DF), that is,

$$\begin{aligned} s(x, y, t) &= \delta(x - x_0 - v_x t)\,\delta(y - y_0 - v_y t) \\ &= \delta^2(\bar{r} - \bar{r}_0 - \bar{v}t) \end{aligned} \tag{7}$$

Notice that s was chosen to be the amplitude function of the moving point so that at any given time, its 2-D integral over the (x, y) plane is unity. Also notice that, using $\bar{r} \equiv (x, y)$, a new vectorial form for the 2-D delta function, which is more convenient and concise to work with, has been defined through equation 7.

The 3-D Fourier transform (FT) of $s(x, y, t)$ is

$$\begin{aligned} S(\bar{k}, \omega) &= \int\int\int \delta^2(\bar{r} - \bar{r}_0 - \bar{v}t)\ \exp\{-j\omega t\}\ \exp\{-j\bar{k} \cdot \bar{r}\}\, dt\, d\bar{r} \\ &= 2\pi\ \exp\{-j\bar{k} \cdot \bar{r}_0\}\delta(\omega + \bar{k} \cdot \bar{v}) \end{aligned} \tag{8}$$

where the vector $\bar{k} \equiv (k_x, k_y)$ is used to denote the spatial frequencies in radians per meter and the dot product is used to denote scalar vector multiplication. The integrals in equation 8 and in all subsequent equations are between the limits of $\pm\infty$ unless specified otherwise. The result in equation 8 is composed of a phase term which depends only on the starting point $\bar{r}_0$ at $t = 0$, and on a 1-D delta function that can be written in a more revealing form as

$$\delta(v_x k_x + v_y k_y + \omega) \tag{9}$$

To understand the meaning of a DF we equate its argument to zero, because, by definition, the DF is active (goes to infinity) when its argument equals zero and it is zero elsewhere. Doing that results in

$$v_x k_x + v_y k_y + \omega = 0 \tag{10}$$

which is the equation of a plane in the 3-D FT domain that passes through the origin. This plane is known to be perpendicular to the vector $(v_x, v_y, 1)$.

This result can be summarized as follows. If the (x, y, t) and the (k_x, k_y, ω) coordinate systems are overlaid so that x and k_x, y and k_y , and t and ω are given by the same axes correspondingly, then the line describing the moving point in (x, y, t) (see fig. 1) is perpendicular to the plane in the 3-D FT domain that represents its FT. This interesting geometrical relationship between the spatial/temporal function (line) and its 3-D FT (a perpendicular plane) is shown in figure 2. This figure also shows the vectorial equation for a plane in 3-D which can be obtained from equation 10 by replacing the first two terms with the scalar vector product $\bar{k} \cdot \bar{v}$. It is seen that the "height," or ω, of any point on the plane is given by the above scalar vector product, that is, $\omega = -\bar{k} \cdot \bar{v}$.

2.4 3-D Fourier Transform of a Moving 2-D Line

It is known that an infinite line (in 2-D or 3-D) sliding along itself appears stationary. This is why only the component of motion of a moving infinite-length line perpendicular to itself is observable. Therefore,

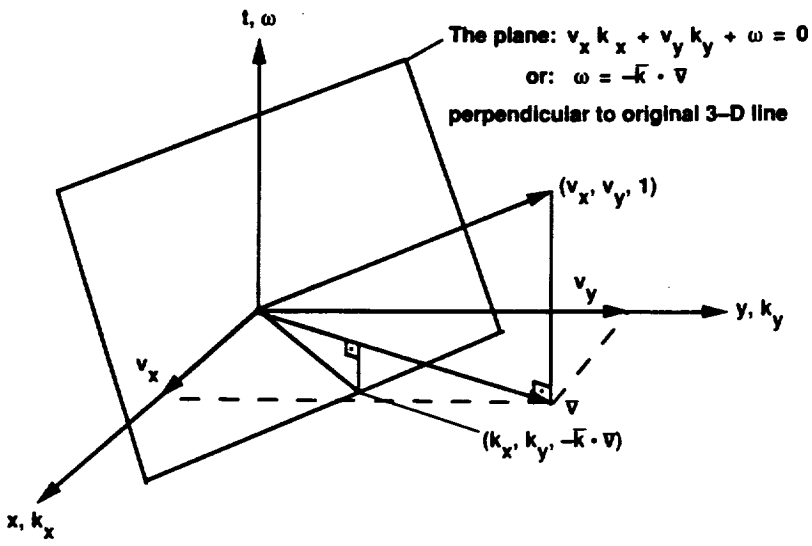

Figure 2. 3-D Fourier transform of a moving point is a plane.

all velocity vectors of a moving infinite line that have the same component perpendicular to the line will produce the same FT. As was shown for a moving point, the augmentation of the 2-D (x, y) plane to the (x, y, t) volume makes the moving line in 2-D appear as a stationary plane in 3-D.

The moving line in 2-D and the corresponding 3-D plane are depicted in figure 3. The moving line can be expressed as a 1-D delta function in the (x, y) plane, that is,

$$\delta[a_x(x - v_x t) + a_y(y - v_y t)] = \delta[\bar{a} \cdot (\bar{r} - \bar{v}t)] \tag{11}$$

where $\bar{a} \equiv (a_x, a_y)$ is a unit-length vector perpendicular to the line. Spatially integrating δ of equation 11 results in the length of the line when that length is finite. The 3-D FT of equation 11 can be written as

$$L(k_x, k_y, \omega) = \iiint \delta[\bar{a} \cdot (\bar{r} - \bar{v}t)] \exp\{-jwt\} exp\{-j\bar{k} \cdot \bar{r}\} dt d\bar{r} \tag{12}$$

where the DF in the integrand is active for $t = \bar{a} \cdot \bar{r} / \bar{a} \cdot \bar{v}$. The sifting property in the t-dimension can now be used to integrate over t, so that the following double integral remains:

$$\begin{aligned} L(k_x, k_y, \omega) &= \iint \exp\{-j\omega \frac{\bar{a} \cdot \bar{r}}{\bar{a} \cdot \bar{v}}\} \exp\{-j\bar{k} \cdot \bar{r}\} \, d\bar{r} = 4\pi^2 \delta^2 (\frac{\omega \bar{a}}{\bar{a} \cdot \bar{v}} + \bar{k}) \\ &= 4\pi^2 \delta(\frac{\omega a_x}{\bar{a} \cdot \bar{v}} + k_x) \, \delta(\frac{\omega a_y}{\bar{a} \cdot \bar{v}} + k_y) \end{aligned} \tag{13}$$

Each DF above represents a plane in 3-D; the first is a plane parallel to the k_y-axis, and the second is a plane parallel to the k_x-axis. These planes intersect on the 3-D line given by

$$\omega = -\frac{k_x \bar{a} \cdot \bar{v}}{a_x} \qquad\qquad \omega = -\frac{k_y \bar{a} \cdot \bar{v}}{a_y} \tag{14}$$

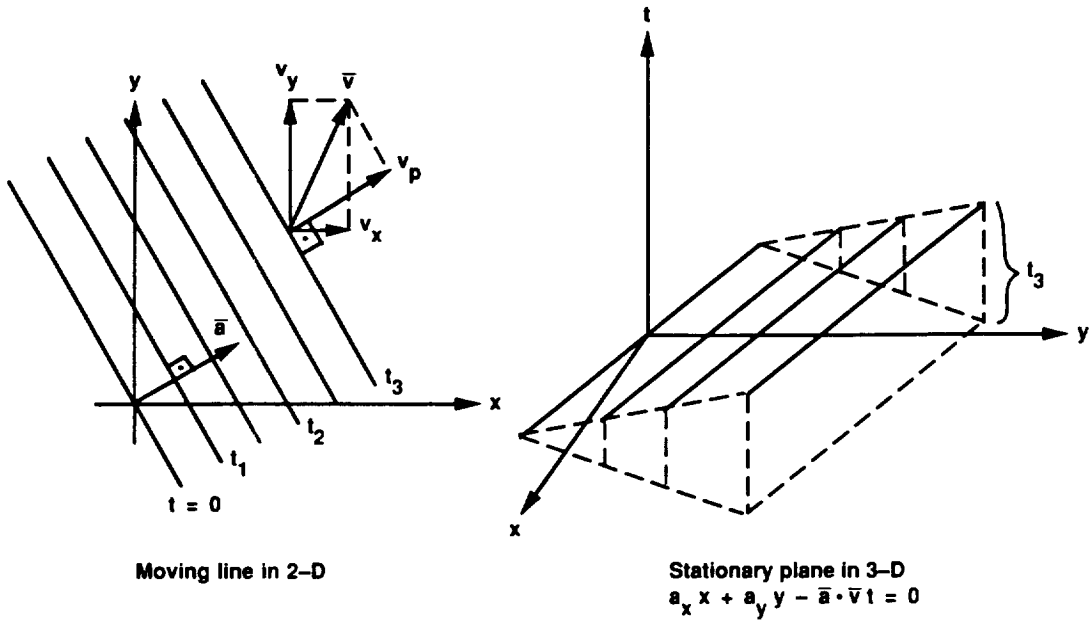

Figure 3. 3-D representation of a moving 2-D line.

and a possible vector parallel to the intersection line has the components $(a_x, a_y, -\bar{a} \cdot \bar{v})$. The $\bar{a}$ vector shown in figure 3 is perpendicular to the advancing line, so that $\bar{a} \cdot \bar{v} = \bar{a} \cdot \bar{v}_p = \mid \bar{a} \mid\mid \bar{v}_p \mid$ since $\bar{a}$ and $\bar{v}_p$ are parallel. Thus, it has been proven that the other component of $\bar{v}$, perpendicular to $\bar{v}_p$, has no effect on the resultant 3-D line in the FT domain. If the original moving-line equation is written as the plane in the 3-D (x, y, t) domain shown in figure 3, that is,

$$a_x x + a_y y - \mid \bar{a} \mid\mid \bar{v}_p \mid t = 0 \tag{15}$$

it is noted that this plane is perpendicular to the line in the FT domain, given by equation 14, which represents its transform. Recalling the result of the previous section, we notice the expected duality with the result here; that is, a *line* in (x, y, t) (which represents a moving point in 2-D) transforms into a perpendicular *plane* in the 3-D FT domain, and a *plane* in (x, y, t) (which represents a moving line in 2-D) transforms into a perpendicular *line* in the 3-D FT domain.

2.5 3-D Fourier Transform of a Moving Wave

A moving wave in 2-D can be described as $\cos(\cdot)$ using for its argument the same one used for the DF in the previous section, that is, $\bar{a} \cdot (\bar{r} - \bar{v}t)$. In this case (a_x, a_y) are the spatial frequencies of the wave along the x and y axes, respectively, and the vector $\bar{a}$ is still perpendicular to the constant-phase lines of the wave in the (x, y) plane as before. The moving wave is shown in figure 4, and its 3-D FT is given by

$$\begin{aligned} W(k_x, k_y, \omega) &= \iiint \cos[\bar{a} \cdot (\bar{r} - \bar{v}t)] \exp\{-j\omega t\} \exp\{-j\bar{k} \cdot \bar{r}\} dt\, d\bar{r} \\ &= 4\pi^3 [\delta(\omega + \bar{a} \cdot \bar{v})\, \delta^2(\bar{k} - \bar{a}) + \delta(\omega - \bar{a} \cdot \bar{v})\, \delta^2(\bar{k} + \bar{a})] \end{aligned} \tag{16}$$

The 3-D FT of the moving 2-D wave consists of only two points, as shown in figure 4. These points are antisymmetric with respect to the origin and, thus, reside on a line that passes through the origin. All waves having the same direction $\bar{a}$—irrespective of their frequency—yield two points on this line. The speed of the wave (perpendicular to its front) determines the height of the two points above or below the (k_x, k_y) plane. In other words, when the speed is zero, the resulting two points in the 3-D FT domain degenerate into the 2-D FT plane. It is the *motion* that raises the two points, or, for that matter, *the 2-D FT of any stationary 2-D pattern*, into the ω-dimension by *slanting* the zero-speed 2-D FT at an angle that is proportional to the speed.

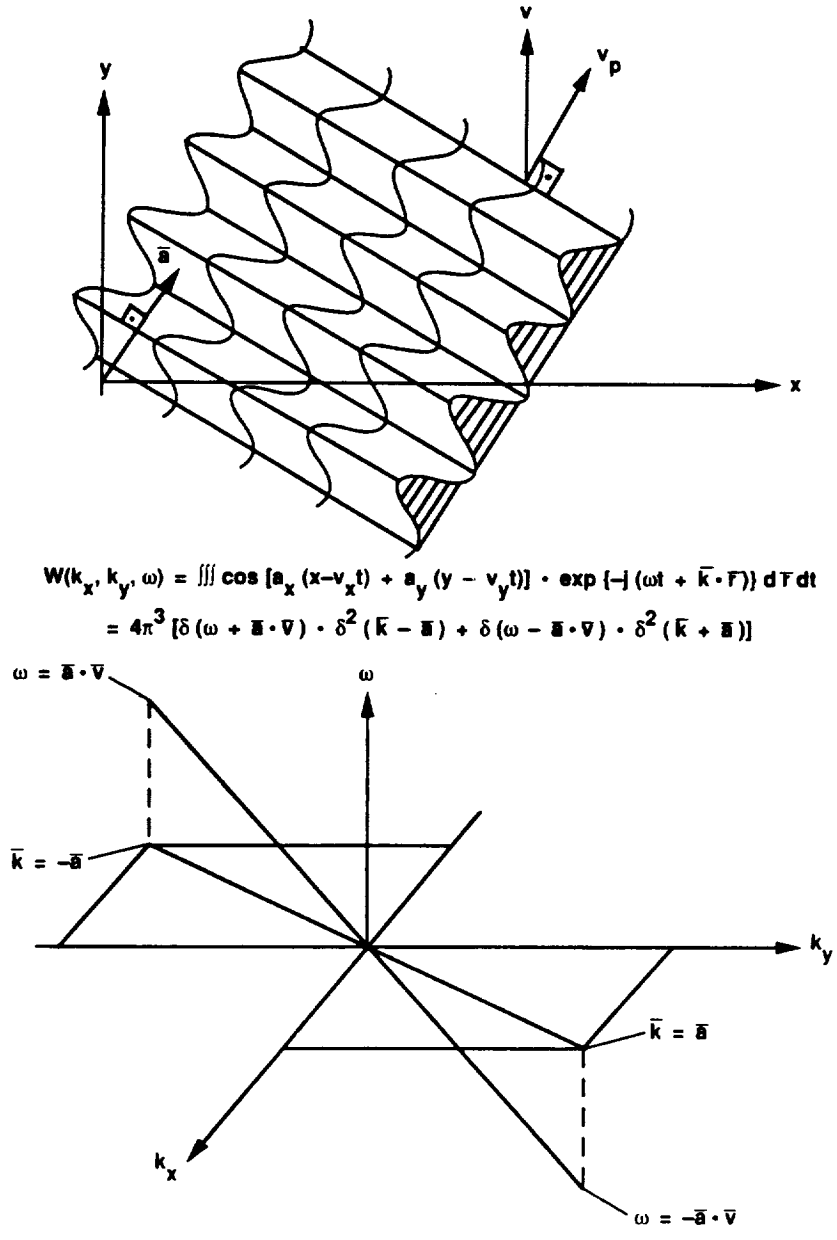

Figure 4. 3-D Fourier transform of a moving 2-D wave.

2.6 Discussion of References

Having developed the necessary mathematical tools, the references mentioned in the Introduction can be discussed in more detail.

Reference 1: Reed et al. The concept of one-, two-, and three-dimensional Fourier transform is well known (e.g., see ref. 8). Reed et al. seem to be the first to apply the 3-D matched filter to a moving constant-velocity object-detection problem. In this paper, they develop the MF equations and the signal-to-noise (SNR) expressions for the MF output. The complete development is done in the frequency domain, although it is mentioned that alternative approaches can be used, such as "space-domain transversal filtering schemes."

Reference 2: Watson and Ahumada. In their paper, Watson and Ahumada model the basic vision cell as a 3-D filter in the FT domain. Referring to figure 4, the filter is composed of two blobs, or ellipsoids, which are centered on the two points shown (denoted by "ω =" in the figure) and sit on the (k_x, k_y) plane. The authors refer to this filter as a "scalar filter" and they explain why such a filter is only sensitive to the velocity component of $\bar{v}$ perpendicular to the constant-phase lines, that is, to v_p in figure 4. They then suggest that vision cells work in "direction groups," where each direction group is composed of 10 scalar filters. The number 10 results from the single-blob angular size in the (k_x, k_y) plane, which is $\approx 36°$ at its –3 dB points, such that 10 filters fit along a circle in this plane.

The ω measurements of all scalar filters (just two suffice in principle) of a direction group determine a plane such as the one shown in figure 2. Watson and Ahumada derive the "vector motion sensor" which is interpreted herein as representing the plane described in section 2.3. This plane is simply the FT of a moving point, and it will be shown in section 4 that it is also the MF to be used in detecting a moving point. It is apparent from figure 2 that the plane (filter) through the origin can be determined by two additional points. Thus, what Watson and Ahumada do is fit a plane based on 10 (noisy) 3-D points which are determined by the (k_x, k_y) locations and ω measurements of the 10 scalar filters.

Watson and Ahumada give some geometrical interpretations of the FT filtering operation that are not provided in reference 1, but they do not use a plane in the 3-D FT domain to define a moving-point filter directly.

Reference 3: Porat and Friedlander. Porat and Friedlander apply filtering in the 3-D FT domain for the purpose of detecting moving objects in mosaic-sensor imagery. They use a bank of directional filters, where each filter is defined as a plane in the 3-D FT domain. The bank of filters is chosen to cover all possible velocity directions. This paper seems to be the first publication in which the use of planes in the 3-D FT domain for matched filtering is reported.

References 4 and 5: Reed et al.; Stotts et al. Both papers implement the MF suggested by Reed et al. in reference 1 and obtain experimental results for the detection of an aircraft-like object. The authors derive an expression for the filtering loss as a result of velocity mismatch and show that the experimental results are robust and agree with the theory.

Reference 6: Kendall and Jacobi. In this work, they use Reed's space-domain filtering which performs the equivalent of filtering by a plane in the 3-D FT domain. No derivation is given to show that the

two are equivalent and that they represent matched filtering for the case in which the imagery is corrupted by white additive noise. This method yielded some preliminary results when implemented in a single spatial dimension and applied to two imagery sets.

3 3-D FOURIER TRANSFORM OF A FINITE-TIME MOVING POINT

So far it has been assumed that the data of interest are continuous and exist between $\pm\infty$ in all three dimensions (x, y, t). The analysis will proceed with continuous data as long as it is meaningful; however, from this point on, things will be made more realistic by requiring causality and finite-time duration so that $0 \leq t \leq T < \infty$. This assumption is expressed by using a rectangular-pulse amplitude function of the form

$$a(t) = \begin{cases} 1 & \text{for} \quad 0 \leq t \leq T \\ 0 & \text{otherwise} \end{cases} \tag{17}$$

and by assuming that a point having this time-limited amplitude is moving with a velocity $\bar{v}$ from the initial location $\bar{r}_0 \equiv (x_0, y_0)$. This point can be described by using DF as before, that is,

$$s(\bar{r}, t) = a(t)\delta^2(\bar{r} - \bar{r}_0 - \bar{v}t) \tag{18}$$

which has a finite duration as opposed to the infinite-duration case considered in equation 7.

As before, the interest here is in the FT of this "signal," which is

$$S(\bar{k}, \omega) = \iiint a(t)\delta^2(\bar{r} - \bar{r}_0 - \bar{v}t) \exp\{-j\omega t\} \exp\{-j\bar{k} \cdot \bar{r}\} dt \, d\bar{r} \tag{19}$$

The spatial integration is performed first and then the temporal one, so that

$$\begin{aligned} S(\bar{k}, \omega) &= \int_{-\infty}^{+\infty} a(t) \exp\{-j\omega t\} \exp\{-j\bar{k} \cdot (\bar{r}_0 + \bar{v}t)\} \, dt \\ &= \exp\{-j\bar{k} \cdot \bar{r}_0\} \int_0^T \exp\{-j(\omega + \bar{k} \cdot \bar{v})t\} \, dt \\ &= T \exp\{-j\bar{k} \cdot \bar{r}_0\} \exp\{-j(\omega + \bar{k} \cdot \bar{v})T/2\} \operatorname{sinc}[(\omega + \bar{k} \cdot \bar{v})T/2\pi] \end{aligned} \tag{20}$$

where

$$\operatorname{sinc}(x) = \frac{\sin(\pi x)}{\pi x} \tag{21}$$

Equation 20 reverts to what is already known, because, when $T \to \infty$, it is the same as equation 8 except for the temporal phase term which would disappear for an $a(t)$ defined to be symmetrical around $t = 0$. Note the plane equation in the argument of the sinc function in equation 20 which is no longer an ideal flat, zero-width plane but rather is a plane that bulges with a thickness that is determined by the sinc function.

4 MATCHED FILTERING APPLIED TO A MOVING POINT

In this section, the matched filter (MF) for a moving point is derived. First, the MF in the FT domain is derived and then its effect in the spatial/temporal domain is interpreted.

4.1 Matched Filtering in the Fourier Transform Domain

An MF is the filter that optimally enhances the signal's amplitude when it is embedded in white or colored Gaussian noise. The well-known form of the MF for a signal embedded in white Gaussian noise is (from refs. 9 and 10)

$$H(\bar{k},\omega) = \frac{S^*(\bar{k},\omega)}{N_0}\exp\{-j\omega T\} \tag{22}$$

where N_0 is the constant-spectrum value of the white noise, asterisk denotes complex conjugation, and the output of the MF is read out when it peaks at $t = T$. The temporal phase term makes sure that the filter is realizable, so that its output is delayed until the end of the data stream. Passing the moving-point signal (which generally also includes noise) through the MF of equation 22 amounts to taking the 3-D inverse transform of the product,

$$G(\bar{k},\omega) = S(\bar{k},\omega)H(\bar{k},\omega) = T^2\,\mathrm{sinc}^2[(\omega + \bar{k}\cdot\bar{v})T/2\pi]\exp\{-j\omega T\}/N_0 \tag{23}$$

which yields, by inverse FT transformation,

$$g(\bar{r},t) = \frac{T^2}{(2\pi)^3 N_0}\iiint \mathrm{sinc}^2[\cdot]\exp\{j[\bar{k}\cdot\bar{r} + \omega(t-T)]\}\,d\bar{k}\,d\omega \tag{24}$$

Substituting $x = (\omega + \bar{k}\cdot\bar{v})/2\pi$,

$$\begin{aligned} g(\bar{r},t) &= \frac{T^2}{(2\pi)^2 N_0}\int \mathrm{sinc}^2(xT)\exp\{j2\pi x(t-T)\}\,dx \iint \exp\{j\bar{k}\cdot[\bar{r} - \bar{v}(t-T)]\}\,d\bar{k} \\ &= T/N_0\,\delta^2[\bar{r} - \bar{v}(t-T)]\cdot \mathrm{tra}(t/T) \end{aligned} \tag{25}$$

where $\mathrm{tra}(t/T)$ is the triangle function shown in figure 5. By definition, the output of an MF has to be sampled at $t = T$ to get the maximum output, or read-out of the last frame when the data are discrete in time. At that instant, there is a point detection at the origin, $\bar{r} = 0$, in the form of a $\delta^2(\bar{r})$ having an amplitude proportional to T. The peak of the MF output is at the origin because spatially non-causal images, which exist for both positive and negative values of $\bar{r}$ components, are being used. Note, that equation 25 includes the factor T, so that the result of MF operation is linearly proportional to the observation time.

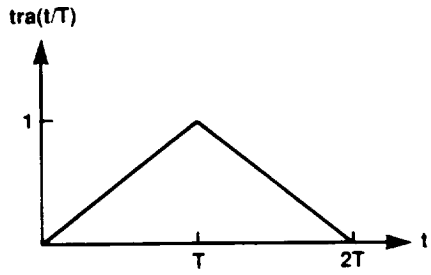

Figure 5. Triangle function.

4.2 Matched Filtering in the Spatial/Temporal Domain

It is useful to also consider the MF operation in the spatial/temporal domain because, in some cases, the implementation turns out to be simpler. The spatial/temporal form of the MF is obtained by finding the inverse transform of its 3-D Fourier transform, $H(\bar{k},\omega)$, as given by equation 22. Using equation 20, in equation 22, gives

$$h(\bar{r},t) = F^{-1}[H(\bar{k},\omega)] = \frac{T}{(2\pi)^3 N_0} \iiint \exp\{j[\bar{k}\cdot\bar{r}_0 - \omega T/2 + \bar{k}\cdot\bar{v}T/2]\} \times \mathrm{sinc}[(\omega + \bar{k}\cdot\bar{v})T/2\pi] \exp\{j(\bar{k}\cdot\bar{r} + \omega t)\}\, d\bar{k}d\omega \tag{26}$$

Using the substitution $x = (\omega + \bar{k}\cdot\bar{v})/2\pi$, gives

$$h(\bar{r},t) = \frac{T}{(2\pi)^2 N_0} \int \mathrm{sinc}(xT) \exp[j2\pi x(t - T/2)]\, dx \iint \exp\{j\bar{k}\cdot[\bar{r} + \bar{r}_0 - \bar{v}(t-T)]\}\, d\bar{k}$$
$$= 1/N_0\, \delta^2[\bar{r} + \bar{r}_0 - \bar{v}(t-T)]\, w(t/T) \tag{27}$$

where $w(t/T) = 1$ for $0 \le t \le T$ denotes a time window of width T, which is equal in this case to $a(t)$ of equation 17.

We now want to understand the operation of convolving *any* 3-D imagery set, $I(\bar{r},t)$, (not necessarily that of a point object) with the impulse response of equation 27. In general, a 3-D convolution can be written as

$$I_c(\bar{r},t) = I(\bar{r},t) * h(\bar{r},t) = \iiint I(\bar{\rho},\tau)\, h(\bar{r}-\bar{\rho}, t-\tau)\, d\tau\, d\bar{\rho} \tag{28}$$

which, in this case, is

$$I_c(\bar{r},t) = 1/N_0 \iiint I(\bar{\rho},\tau)\, \delta[\bar{r} - \bar{\rho} + \bar{r}_0 - \bar{v}(t - \tau - T)]\, w[(t-\tau)/T]\, d\tau d\bar{\rho} \tag{29}$$

The temporal limits of integration in equation 29 are determined by the period of existence of the imagery, which is assumed to be $(0,T)$. This time-window multiplied by another window that appears explicitly in the integrand, makes a combined window, denoted by $W(t,\tau,T)$, which behaves as follows:

$$W(t,\tau,T) = w(\tau/T)\, w[(t-\tau)/T] = \begin{cases} 1 & \text{for } 0 \le \tau \le t \quad \text{and } 0 \le t \le T \\ 1 & \text{for } t - T \le \tau \le T \quad \text{and } T \le t \le 2T \\ 0 & \text{otherwise} \end{cases} \tag{30}$$

The width, or integral, of this time-window over τ between the limits of 0 and t, has the same form as the $\mathrm{tra}(t/T)$ function except that its peak has a value of T instead of 1. Thus, after performing the spatial integration, equation 29 can be reduced to

$$I_c(\bar{r},t) = 1/N_0 \int_0^t I[\bar{r} + \bar{r}_0 - \bar{v}(t - \tau - T), \tau]\, d\tau \quad \text{for } 0 \le t \le T \tag{31}$$

A similar expression for the region $T \le t \le 2T$ is of no interest because all that is needed at the end is to sample equation 31 at $t = T$.

To check the validity of the above general result, it is applied to the specific moving-point imagery of equation 18, that is,

$$\begin{aligned} I_c(\bar{r},t) &= 1/N_0 \int_0^t a(\tau)\, \delta^2[\bar{r}+\bar{r}_0-\bar{v}(t-\tau-T)-\bar{r}_0-\bar{v}\tau]\, d\tau \\ &= 1/N_0 \int_0^t \delta^2[\bar{r}-\bar{v}(t-T)]\, d\tau \\ &= t/N_0\, \delta^2[\bar{r}-\bar{v}(t-T)] \end{aligned} \tag{32}$$

The interpretation of equation 32 is the following. When discrete images are considered, $I_c(\bar{r},t)$ represents the whole imagery set so that $I_c(\bar{r},1)$ is the first frame and, say, $I_c(\bar{r},20)$ is the $20th$ frame of the convolved imagery (after it has passed the MF). If we start with N original frames, we end up with $2N$ frames out of the MF. Since the MF is sampled at T, we are only interested in frame number N. It is seen that the results of equations 32 and 25 agree at time $t = T$ (frame number N), as expected.

5 IMPLEMENTATION OF THE MATCHED FILTER

To see what the operation of equation 31 means in terms of implementation, we rewrite it, sampled at $t = T$, that is, as

$$I_c(\bar{r},T) = 1/N_0 \int_0^T I(\bar{r}+\bar{r}_0+\bar{v}\tau,\tau)\, d\tau \tag{33}$$

It is seen that the shifted versions of the original images have to be added (integrated), starting with the first at $\tau = 0$, that is, $I(\bar{r}+\bar{r}_0,0)$, and ending with the last at $\tau = T$, that is, $I(\bar{r}+\bar{r}_0+\bar{v}T,T)$. This means that the last frame is shifted toward the origin by the vector $\bar{v}T$ relative to the first one. The additional shift toward the origin of size $\bar{r}_0$, which is common to all frames, is a result of the particular MF being used, which was matched to a point at $\bar{r}_0$. That shift would bring the point $\bar{r}_0$ to the origin. Since it is desired to *preserve* the original location of any general point $\bar{r}_0$ in the resultant image, this point has to be shifted back to where it came from. This can be accomplished by discarding the $\bar{r}_0$ term from equation 27. The final effect would thus be that *all* points (pixels) after MF stay in the locations they had at $t = 0$ or at the first frame of the original imagery.

We now return to the velocity-related spatial shift toward the origin, $\bar{v}t$. The direction of this shift is such that if a pixel has a velocity away from the origin, it is shifted back toward the origin as if to cancel out its velocity. Figure 6 shows the shifting operation performed on the last frame for a 1-D imagery. It is seen that, for a positive speed, the shifting is to the left. Since the magnitude of the shift is proportional to time, the effect of equation 33 is to align the objects having velocity $\bar{v}$ so that they appear *stationary* and collocated in their initial (general) locations in all frames. Once all frames are aligned, they are integrated over the total time T. Only those objects having velocity $\bar{v}$ will integrate coherently; others that have different velocities will appear smeared out in the resultant image. The degree of smearing will be proportional to the deviation of their velocities from the one for which the MF has been tuned, as explained in the next section. So far the 3-D space has been discussed as a continuous volume, with an occasional reference to frames and pixels. In reality we are dealing with finite-size images, say of $N \times N$ pixels, which appear at a certain rate, say every T_f seconds. In such a case the effective part of the image that can be worked with is limited. The limitation is determined by the tuning velocity $\bar{v}$ and the integration time T; it is the intersection of the frame with itself shifted by $-\bar{v}T$. As shown in figure 7, a typical point A, found in the first frame, will arrive at point B at the last frame—thus it will stay within the FOV.

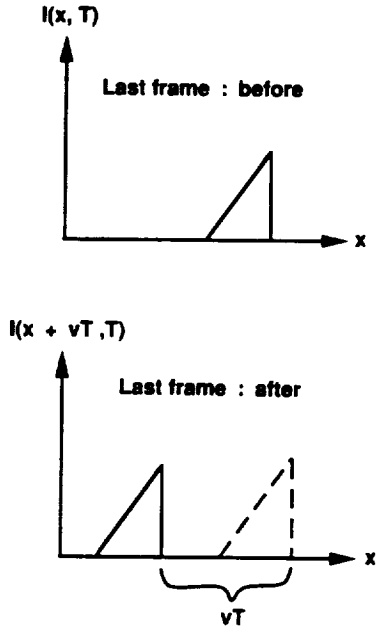

Figure 6. Last frame: before and after shift.

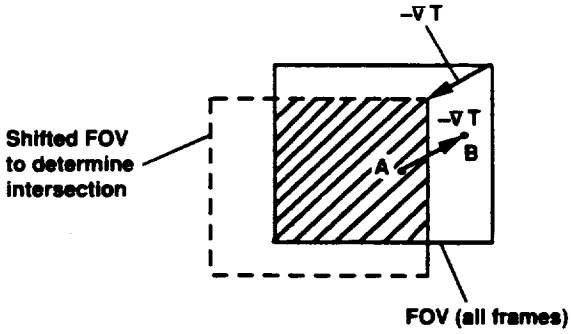

Figure 7. Active area of frame.

6 VELOCITY RESOLUTION OF A 3-D MATCHED FILTER

6.1 General Derivation

A general unknown imagery may contain all possible velocity vectors; thus, it will be passed through a bank of filters tuned to all the discrete velocity vectors that span the required range. In order to cover the range of velocities in an economical way, it is first neccessary to find out the resolutions of a single filter in both angle and length of the velocity vector. To do that, we start from the MF result of equation 22, where an MF tuned to $\bar{v}$ is used for an object moving at a different velocity $\bar{v}_0$, for which equation 20 is used with $\bar{v}_0$ replacing $\bar{v}$. Equation 23 can now be rewritten for such a case as

$$G(\bar{k},\omega) = T^2 \operatorname{sinc}[(\omega + \bar{k}\cdot\bar{v}_0)T/2\pi]\operatorname{sinc}[(w + \bar{k}\cdot\bar{v})T/2\pi]$$

$$\times \exp(j\bar{k}\cdot\Delta\bar{v}T/2)\exp(-j\omega T)/N_0 \tag{34}$$

where

$$\Delta\bar{v} \equiv \bar{v} - \bar{v}_0$$

The 3-D inverse FT, as in equation 24, sampled at $t = T$ will be

$$g(\bar{r},T) = \frac{T^2}{(2\pi)^3 N_0}\int \mathrm{sinc}[(\omega + \bar{k}\cdot\bar{v}_0)T/2\pi]\,\mathrm{sinc}[(\omega+\bar{k}\cdot\bar{v})T/2\pi]\,d\omega \iint \times \exp[j\bar{k}\cdot(\bar{r}+\Delta\bar{v}T/2)]\,d\bar{k} \tag{35}$$

Using Parseval's theorem, that is,

$$\int f^*(t)g(t)\,dt = \frac{1}{2\pi}\int F^*(\omega)G(\omega)\,d\omega \tag{36}$$

and the FT pair,

$$\omega[(t+T/2)/T]\exp(-jxt) \overset{FT}{\rightarrow} T\,\mathrm{sinc}[(\omega+x)T/2\pi] \tag{37}$$

the first integral over ω in equation 35 is replaced by

$$\begin{aligned}\frac{1}{2\pi}\int T\,\mathrm{sinc}[\cdot]\;T\,\mathrm{sinc}[\cdot]\,d\omega &= \int w[(t+T/2)/T]\exp\{j\bar{k}\cdot\bar{v}_0 t\} \\ &\quad\cdot w[(t+T/2)/T]\exp\{-j\bar{k}\cdot\bar{v}t\}\,dt \\ &= \int_{-T/2}^{T/2}\exp\{-j\bar{k}\cdot\Delta\bar{v}t\}\,dt = T\,\mathrm{sinc}\left(\frac{\bar{k}\cdot\Delta\bar{v}T}{2\pi}\right)\end{aligned} \tag{38}$$

Now equation 35 reads

$$g(\bar{r},T) = \frac{T}{4\pi^2 N_0}\iint \mathrm{sinc}\left(\frac{\bar{k}\cdot\Delta\bar{v}T}{2\pi}\right)\exp\{j\bar{k}\cdot(\bar{r}+\Delta\bar{v}T/2)\}\,d\bar{k} \tag{39}$$

We will skip some simple but unwieldy derivation, and write the final result as

$$g(\bar{r},T) = 1/N_0\;\delta(\Delta v_x y - \Delta v_y x)\;, \quad -T\Delta v_x \le x \le 0 \tag{40}$$

which is a 2-D line DF on the bounded segment shown in figure 8. The total area under this segment is

$$\int_{-T\Delta v_y}^{0}\int_{-T\Delta v_x}^{0}\delta(\Delta v_x y - \Delta v_y x)\,dxdy = T \tag{41}$$

and it is distributed evenly along it. It is noted that the dimensions of $\delta(\Delta v_x y - \Delta v_y x)$ are $\delta([m/sec][m])$= $[sec]\delta([m^2])$, which is the same as those of $t\delta^2(\bar{r})$ in equation 32, that is, $[sec]\delta([m])\delta([m])$ or $[sec][m^{-2}]$.

The line DF of equation 40 reduces to that of equation 32 evaluated at $t = T$ when $\Delta v_x = 0$ (which implicitly also makes $\Delta v_y = 0$). The meaning of this result is that, because of the velocity mismatch, the point detection of equation 32 smears over the length of the line segment of equation 40, which is

$$l = T\sqrt{\Delta v_x^2 + \Delta v_y^2} = T\,|\Delta\bar{v}| \tag{42}$$

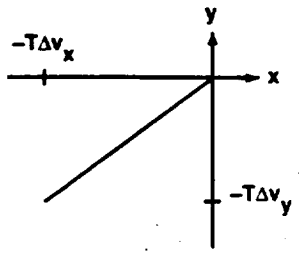

Figure 8. Line-segment delta function.

that is, the moving pixel is now detected in the form of a smeared line segment rather than a single point at the origin, as it was for $\Delta\bar{v} = 0$. This will cause location uncertainty along with a loss in gain.

Note that, because equation 42 depends on the absolute value of the speed mismatch, this equation reveals the effect of both mismatch in *speed* and mismatch in the velocity-vector *direction*. For mismatch in direction of size $\Delta\alpha$,

$$|\Delta\bar{v}| = 2\ |\bar{v}|\ \sin(\Delta\alpha/2) \tag{43}$$

Note too that it is the absolute value of the velocity vector mismatch that appears in the result of equation 42 and not the *relative* velocity vector mismatch.

The gain loss is simpler to understand when it is thought of in terms of discrete pixels rather than in terms of continuous spatial distances. For $l = p$ along, say, the x-axis (p denotes the side width of a pixel) the peak smears over 2 pixels so that the loss is by a factor of 2 or –6 dB. Equation 42 can be rewritten for the discrete case as

$$l_p \approx N\ |\Delta\bar{v}_p| \tag{44}$$

where N is the total number of frames (replacing the integration time T), and $\bar{v}_p$ is the velocity measured in pixels per frame. The expression of equation 44 is inaccurate for small $|\Delta\bar{v}_p|$ values (of the order of $1/N$ pixel/frame) because of the pixels' discretization. However, we are particularly interested in such small values because they correspond to losses of the order of –3 dB. We thus want to derive an accurate expression for that range of interest as explained below.

It is realized that the loss in gain results from smearing a 1-pixel area over more than a single pixel. Figure 9 shows a pixel that is smeared as a result of shifting it by $(\Delta x, \Delta y)$. For gain losses of the order of –3 dB, the peak of the MF output will still occur in the nominal pixel that is denoted by (0,0). Thus, we want to find the intersection areas between the shifted (smeared) pixel and the nominal pixel at each one of the N frames and sum them up. When $(\Delta x, \Delta y) = (0, 0)$ this sum equals N.

Counting frames from zero, that is, $0 \leq n \leq N-1$, the shift in pixel-width units at frame n is given by

$$\Delta x = n\,|\Delta\bar{v}_p|\cos\alpha\ ,\quad \Delta y = n\,|\Delta\bar{v}_p|\sin\alpha \tag{45}$$

The contribution that frame n makes to the nominal pixel is $(1 - n\Delta x)(1 - n\Delta y)$ so that the loss (to be denoted by L) can be written as the summation of the contributions of all N frames divided by the total

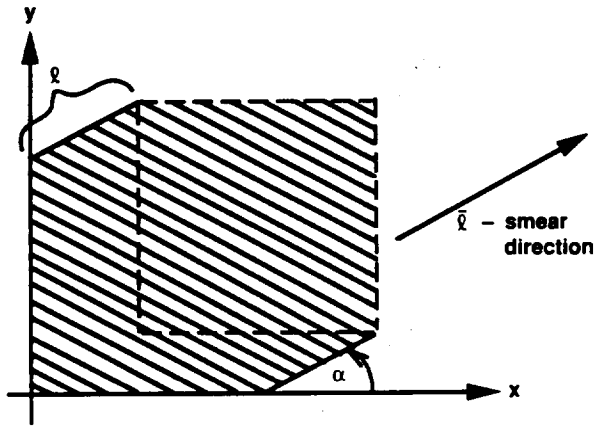

Figure 9. Smearing detected pixel in a general direction.

nominal contribution of N pixels to normalize it to unity; that is,

$$\begin{aligned} L &= 1/N \, \Sigma_{n=0}^{N-1} (1 - n\,|\Delta\bar{v}_p| \cos\alpha)(1 - n\,|\Delta\bar{v}_p| \sin\alpha) \\ &= 1/N \, \Sigma_{n=0}^{N-1} \left[1 - n\,|\Delta\bar{v}_p| \,(\cos\alpha + \sin\alpha) + n^2 \,|\Delta\bar{v}_p|^2 \sin(2\alpha)/2\right] \end{aligned} \tag{46}$$

For small $|\Delta\bar{v}_p|$ and large N the above sum will be approximated by an integral, that is,

$$L \approx 1/N \int_0^N (\cdot)\, dn = 1 - N/2 \,|\Delta\bar{v}_p| \,(\cos\alpha + \sin\alpha) + N^2/6 \,|\Delta\bar{v}_p|^2 \sin(2\alpha) \tag{47}$$

To find the conditions for which the loss is –3 dB, the loss, L, is equated to $1/\sqrt{2}$. For a general loss of $L \leq 1$, the following quadratic equation has to be solved:

$$l_p^2/6 \sin(2\alpha) - l_p/2(\cos\alpha + \sin\alpha) + (1 - L) = 0 \tag{48}$$

where $l_p = N\,|\Delta\bar{v}_p|$ is the length of smear in pixels as defined by equation 44. The solution is

$$l_p = \frac{1}{2\sin(2\alpha)} \left[3(\cos\alpha + \sin\alpha) - \sqrt{9 + \sin(2\alpha)(24L - 15)}\right] \tag{49}$$

When there is no loss, or $L = 1$, we find from equation 49 that there is no smear, or $l_p = 0$ as expected.

Equation 49 can now be used to calculate $|\Delta\bar{v}_p|$ for a –3 dB loss by evaluating it for $L = 1/\sqrt{2}$ and dividing the resulting l_p by N. From equation 44, a large N yields a small $|\Delta\bar{v}_p|$ for any given fixed loss. Figure 10 shows the relationship between the total smear length during N frames, l_p, and the direction of the velocity mismatch, $\Delta\bar{v}$, for the cases of –3 dB and –6 dB losses. It is seen that if a –6-dB loss is allowed, the smear length is 1 pixel when its direction is along either the x or the y axes, that is, $\alpha = 0$, or 90° (more generally for $\alpha = n \cdot 90°, n = 0, 1, 2, \ldots$). This result is already known because it corresponds to the simple case in which the detection pixel (or the peak) smears evenly over the area of two pixels. For the –6-dB loss, if the smear is in any diagonal direction (45°), its length can only be 0.9 pixels. If only a loss of –3 dB is allowed, then the allowed smear length along the axes goes down to 0.58 pixel and, in a diagonal direction, to 0.47 pixel.

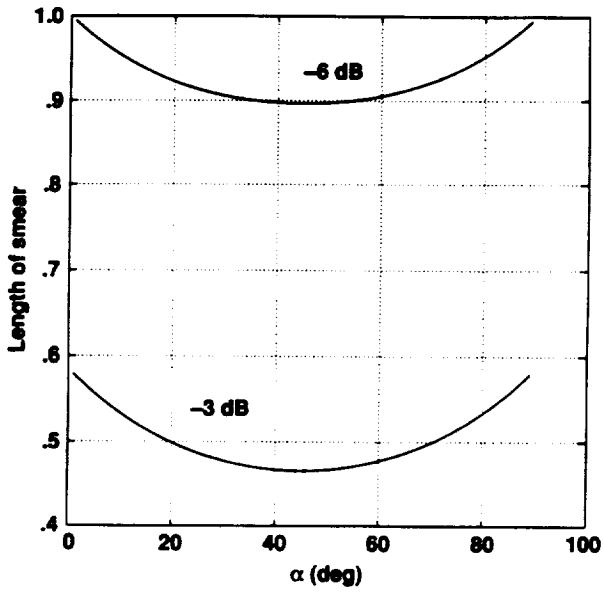

Figure 10. Smear length for –3 dB and –6 dB loss as function of smear direction.

When the smear length is larger than 1 pixel, the loss is always larger than 6 dB. This is why the high-loss region of l_p is regarded as a *rejection* rather than a *loss* region. An approximate relationship for the rejection capability of the velocity filter in this region is given by

$$\begin{aligned} \mathrm{L} &\approx \tfrac{1}{1+N\lvert\Delta\bar{v}_p\rvert}, \text{ for } \lvert\Delta\bar{v}_p\rvert \leq 1 \text{ pixel/frame} \\ &\approx 1/N \text{ otherwise} \end{aligned} \tag{50}$$

For example, for $N = 32$ frames, the nominal gain for a perfect velocity match is 32. If the speed changes by 0.1 pixel/frame (from whatever it was), $|\Delta\bar{v}_p| = 0.1$, and $l_p = 3.2$ pixels. This means that the peak will smear over 4.2 pixels, that is, a loss of 12.5 dB. Looking at it the other way around, the gain separation between pixels that have a speed difference of 0.1 pixels/frame is 12.5 dB.

6.2 Required Number of Velocity Filters

In this section the results regarding the allowed smear length are used to arrive at an estimate for the number of velocity filters required to cover any given range of speeds, say, 0 to some V_{max}, and in all velocity directions.

Figure 11 shows the results of allowed smear for the –3 dB and –6 dB cases in a polar coordinate system. For a given number of integrated frames N, dividing the pixel values of the figure by N allows reading the axes in terms of the Δv_x and Δv_y components of the velocity-vector mismatch. For example, for $N = 10$, a value of 1 pixel in the figure is equivalent to a speed mismatch of 0.1 pixel/frame. With that interpretation of figure 11, the vector $\Delta\bar{v}_p$ can be drawn from the center of the figure to any point along the closed curve (for either the –3 dB or –6 dB curve). Thus, the closed curve can be seen as enclosing a

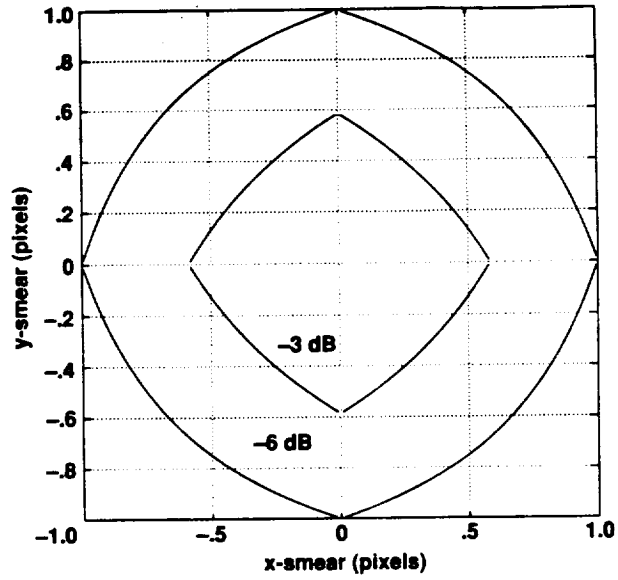

Figure 11. Smear lengths shown in polar coordinates.

two-dimensional area in the *velocity domain* (v_x, v_y). If the velocity mismatch vector falls inside this area, then the mismatch loss is equal to or less than the one described by the curve.

The above regions, approximated (and referred to) by circles, can now be used to construct a bank of filters so as to cover any required region in the velocity plane in a complementary way. Figure 12 shows an example in which it is desired to cover the speed range between 0 and some V_{max}. The number of such circles—each standing for a velocity filter tuned to its center velocity vector—is

$$N_f \approx V_{max}^2 / |\Delta \bar{v}_p|^2 \tag{51}$$

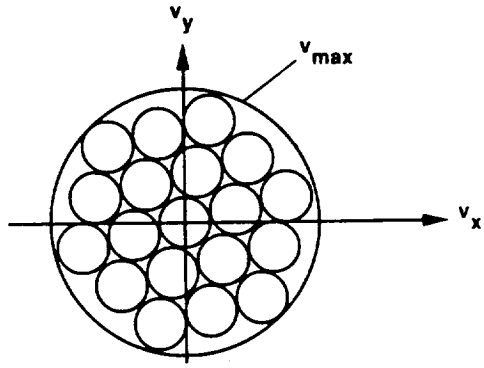

Figure 12. Covering velocity plane with complementing filters.

For example, if V_{max} = 1 pixel/frame, and $|\Delta \bar{v}_p|$= 0 .05 pixel/frame (for a 3 dB loss), then 400 filters are needed to cover the required range of velocity vectors. On the other hand, if it is required to cover only a one-dimensional strip, as is the case for optical-flow calculations with a given focus of expansion, then only $V_{max}/(2l_p)$ filters are needed, where $2l_p$ is the diameter of the approximated "circle" filter of figure 11. Continuing the above example, only 10 filters will be needed in that case.

6.3 Velocity Filter Algorithm

It has been shown that, for a straightforward velocity filtering in the spatial-temporal domain that is performed in batch, it is necessary to shift each image in the velocity direction backward in time to align all frames with the first one. This process is shown in figure 13 for a point that moves over three frames, where, in general, frame n is shifted by $-|\Delta \bar{v}_p|\ (n-1)$. The shifting operation makes it necessary to expand each original frame by adding a strip of zeros of width $|\Delta \bar{v}_p|\ (N-1)$ in the $-\bar{v}_p$ direction so that the shifted pixels will have a place to be written on. After shifting, the frames are summed up. It is seen in the figure that the three points are aligned after shifting so that the summation of the frames will result in a value of 3.

The interest here, however, is in performing the same process recursively, that is, as each frame is collected, the "first" frame is redefined to be the first one of the current integration-time window. This means to discard the first frame of the current window and then shift the window one frame forward as each new frame is received. Let us now see what this operation takes in terms of calculations by writing down the algorithmic steps. When a new frame arrives:

1. Store the new frame as is (before shifting) for later use in step 3.

2. Shift the new frame by $-\bar{v}_p(N-1)$ pixels (backward).

3. Subtract the first frame of the current window; this is the *original (unshifted)* version of that frame which was stored in step 1.

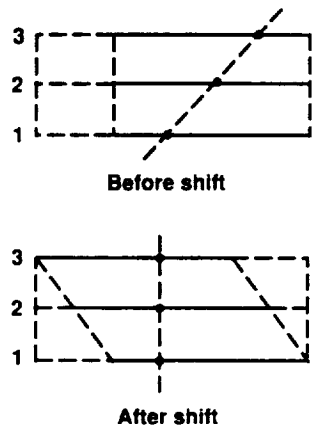

Figure 13. Shift-and-sum processing.

4. Shift the window frame by $+\bar{v}_p$ (forward).

5. Sum up the window from step 4 to the result of step 2.

These steps can be written as

$$\text{sum}_{i+1} = \mathcal{sh}_{+1}(\text{sum}_i - \mathit{fr}_1) + \mathcal{sh}_{-N+1}(\mathit{fr}_c) \tag{52}$$

where sum_i and sum_{i+1} are the previous and updated sum frames, respectively, and $\mathcal{sh}$ denotes the shifting operation in a direction indicated by the sign of its subscript (positive in the velocity-vector direction) and by an amount given by the subscript size in units of $\bar{v}_p$. The frame fr_1 above is the first frame of the old window, and fr_c is the current new frame. The recursion can start from $\text{sum}_0 = 0$, so that it builds up to a steady state after the first N frames have been collected.

6.4 Computational Load and Memory Requirement

The computational requirements of the recursive algorithm described above will be determined in this section. Starting with the computations rate, note that there are three basic types of operations: shift, interpolation, and summation. The need for interpolation arises because the amount of shifting is generally given by a non-integer number of pixels. Only one possible interpolation scheme is used here, and it should be kept in mind that many others are also useful (discussed in section 8.4).

To shift a single pixel into some non-integer location requires the following algorithmic steps:

1. Get the pixel's location and value from memory.

2. Calculate the shifted-pixel non-integer (x, y) location (two floating-point (FP) adds) and get the remainders $(\Delta x, \Delta y)$ (two roundoffs).

3. Calculate intersection areas of the shifted pixel with the underlying 4 pixels, that is,

$$\begin{aligned} A &= 1 + \Delta x \Delta y - \Delta x - \Delta y \\ B &= \Delta y - \Delta x \Delta y \\ C &= \Delta x \Delta y \\ D &= \Delta x - \Delta x \Delta y \end{aligned} \tag{53}$$

which takes one FP multiplication and five FP adds.

4. Multiply the pixel's value by the four intersection areas (four FP multiplications).

5. Add these four values to the four underlying pixels (four FP adds).

Counting operations without distinguishing their types, a total of 18 operations are required to shift, interpolate, and add 1 pixel. Since in equation 53 there is one add, one subtract, and two shifts, the total number of operations that the recursive algorithm takes is 38 operations/pixel.

The rate of computations depends, of course, on the frame rate. If the frames come at R_f frame/sec, and if M-pixel frames are used, then the computations rate is $38 R_f M$ operations/sec per velocity filter,

or, for K filters,

$$Q = 38 R_f M K \qquad \text{operations/sec} \tag{54}$$

To get an idea about the order-of-magnitude of the required rate, we will use an example in which $R_f = 30$ frames/sec, $M = 512 \times 512$, and $K = 400$, which results in $1.19 \cdot 10^{11}$ operations/sec. Note, that a computer with that capability is beyond the state of the art, and that these kinds of numbers create the incentive to trade off performance for some possible reduction in the computation rate requirement.

The required memory can be written as

$$\text{MEM} = M(K + N) \tag{55}$$

where the term MK represents K filters for which a single sum-frame of M pixels has to be stored, and the term MN represents the storage of N original frames so they can be subtracted as required by equation 53). For the same example, and for $N = 32$ (N is the number of frames in the integration window), MEM = $1.13 \cdot 10^8$ words.

7 APPLICATION TO THE OPTICAL FLOW PROBLEM

7.1 General

In this section, the general theory developed above is applied to the optical-flow problem. The optical-flow case—under the assumption of a known focus-of-expansion (FOE)—offers various ways of reducing the computational requirement to a manageable level because it is basically a one- and not a two-dimensional problem, at least in terms of the image coordinates. This means that only a one-dimensional strip has to be covered in the velocity-vector plane that corresponds to a strip in the image plane. In other words, the frames can be divided into some fixed number of angular sectors—all having the FOE as their vertex—and then a bank of velocity filters applied so as to cover a linear strip in the velocity-vector plane which is oriented the same as the sector in the image plane. This method of coverage is shown in figure 14 which is an overlay of the velocity and image planes. The –3-dB contours of the passband of the velocity filters' are shown as circles, where the lowest-speed circle (besides zero speed) touches the origin so that it corresponds to some minimum pixel distance from the FOE.

Each velocity filter in figure 14 is applied to all pixels underneath it. The product of the number of filters and the number of pixels, which determines the computational rate, stays constant if the filtering process described by the figure is replaced by another one in which every pixel has its own dedicated filter centered on it.

The optical-flow case is one-dimensional in the sense that the velocity direction is known for each pixel; however, it has one additional dimension that has been ignored so far: the sensor/object range (or depth, or distance). The simple filter-per-pixel processing could suffice if each pixel corresponded to a unique velocity, but, because a pixel may represent objects at any depth, the pixel actually corresponds to a *range* of velocities in the image plane. Thus, a set of filters is needed for each pixel to cover the range of depths corresponding to that range of image-plane velocities.

The question now is how to cover the depth dimension for each pixel in the most efficient way. It will be shown that, for any given depth, the pixel's speed in the image plane (or distance from the FOE) has

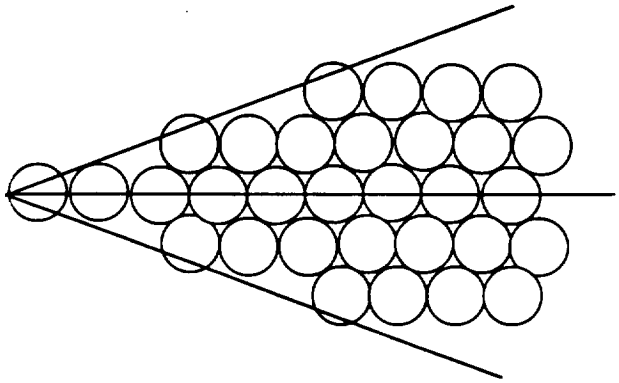

Figure 14. Overlay of velocity and image planes for velocity-filters coverage.

some *speed profile* which, in general, cannot be approximated by a constant. The term "profile" is used because, although the distance from the FOE has non-zero time-derivatives of all orders, it is a known function of time with only one unknown parameter, that is, the depth. In other words, it is being suggested that "depth filters" be used rather than constant-velocity filters.

7.2 Why Constant-Speed Filters Cannot Work

In the forward-looking optical-flow problem each pixel—representing the angular squint from the sensor's axis—moves in the image plane radially away from the FOE, and its distance from the FOE is a hyperbolic function of time. Assuming, for simplicity, that the flight-velocity vector coincides with the sensor's axis, let the (x, y, z) coordinate system at the sensor be set up so that the image plane lies in the (x, y) plane, the x-axis is parallel to the ground, and the z-axis points in the flight direction. The object's location in the image plane is denoted by (θ_x, θ_y) to correspond with the (x, y) axes.

With that, the projection of a stationary object at (x, y, z) onto the image plane can be expressed by its image-plane coordinates (to be measured in pixels)

$$\theta_x = dfx/z \quad \theta_y = dfy/z \tag{56}$$

where f is the imaging-device focal length and d is number of pixels per unit length. The time-derivatives of the projection coordinates are

$$\dot{\theta}_x = v\theta_x/z \quad \dot{\theta}_y = v\theta_y/z \tag{57}$$

where v is the sensor's speed, z denotes the sensor-to-object depth, and $\dot{z} = -v$. We can now write, for any θ irrespective of direction in the image plane,

$$\begin{aligned} \theta &= dfr/z \\ \dot{\theta} &= v\theta/z \end{aligned} \tag{58}$$

where $\theta = \sqrt{\theta_x^2 + \theta_y^2}$ and $r = \sqrt{x^2 + y^2}$. Also, the initial values for z and θ can be used, that is, z_0 and θ_0 at $t = 0$, to write

$$\theta = \frac{\theta_0 z_0}{z_0 - vt} \quad , \quad \dot{\theta} = \frac{v\theta}{z_0 - vt} \tag{59}$$

There is some minimum θ for which these equations are meaningful, which corresponds to the velocity filter whose circular –3-dB contour touches the origin $\theta = 0$ (the leftmost in figure 14. To find this minimum, recall from equation 44 that $|\Delta \bar{v}_p| = l_p/N$ pixels/frame. Since, for the minimum-speed pixel, $\dot{\theta}$ plays the role of $|\Delta \bar{v}_p|$, it follows that

$$\dot{\theta}_0 = \frac{l_p}{N} \quad \text{pixels/frame} \tag{60}$$

or, for a frame rate of R_f frames/sec,

$$\dot{\theta}_0 = \frac{l_p R_f}{N} \quad \text{pixels/sec} \tag{61}$$

Replacing $\dot{\theta}$ by $\dot{\theta}_0$ in equation 59 and setting $t = 0$, gives the minimum processible pixel distance from the FOE as

$$\theta_{0_{min}} = \frac{l_p z_0 R_f}{vN} \quad \text{pixels} \tag{62}$$

With typical values of $z_0 = 50$ m, $R_f = 32$ frames/sec, $v = 15$ m/sec, $N = 16$ frames, and $l_p \approx 0.5$ pixel (as read from fig. 10 for a –3-dB loss), $\theta_{0_{min}} = 3.33$ pixels. This means that for the given depth of $z_0 = 50$ m, all θ's between zero and 6.66 pixels along any radius centered on the FOE will be processed by the same lowest-speed velocity filter.

Next, we want to check how well an accelerating pixel having a hyperbolic time trajectory, as given by equation 59, can be approximated by a constant-speed pixel. In other words, the length of time for which the pixel's speed remains inside a single constant-speed velocity filter is to be determined.

For a pixel found initially at θ_0, the initial speed is $\dot{\theta}_0 = v\theta_0/z_0$. If the speed were constant, θ would increase during N frames by $\dot{\theta}_0 N/R_f$ pixels. On the other hand, the actual increase in θ during N frames is hyperbolic as given by equation 59, that is,

$$\frac{z_0\theta_0}{z_0 - vN/R_f} - \theta_0 = \frac{vN\theta_0}{z_0 R_f - vN} \tag{63}$$

We now want to limit the difference between the actual increase and its constant-speed approximation to be less than l_p, that is,

$$\frac{vN\theta_0}{z_0 R_f - vN} - \frac{v\theta_0 N}{z_0 R_f} < l_p$$

$$v\theta_0 N\left[\frac{1}{z_0 R_f - vN} - \frac{1}{z_0 R_f}\right] = \frac{\theta_0 v^2 N^2}{z_0 R_f(z_0 R_f - vN)} < l_p \tag{64}$$

This leads to a quadratic equation for N with the result that

$$N \le \frac{z_0 R_f l_p}{2v\theta_0}\left[\sqrt{1 + 4\theta_0/l_p} - 1\right] \tag{65}$$

or, approximately,

$$N \le \frac{z_0 R_f}{v}\sqrt{\frac{l_p}{\theta_0}} \tag{66}$$

The meaning of this bound is that with any larger N, the integrated energy over N frames will spill outside of the constant-velocity-filter passband. Alternatively, for any given N there is an upper bound on θ. These statements are clarified by the following example.

Example

For $v = 10$ m/sec, $z_0 = 50$ m, $R_f = 32$ frames/sec, $\theta_0 = 45$ pixels, and $l_p = 0.5$ pixel, it is found from equation (65) that $N \leq 16$, or that the integration time is less than 0.5 sec. Alternatively, for $N = 16$, θ_0 must be less then 45 pixels.

7.3 Using Velocity Profile Filters

Because we have seen that a constant-velocity filter can only be useful in limited cases (where either N or θ_0 is small), it is suggested that *variable-speed* filters be used such that each filter is tuned to the velocity *profile* generated by a hyperbolic time-function. First, it is realized that each hyperbola is determined (for a given speed v) by the pixel's initial distance from the FOE, θ_0, and the range z_0 of the object that this pixel represents. Now the relevant problem is to find the passband of such a filter in terms of the *range of depths* that will still fall inside it although it was tuned to some fixed depth z_0. To answer this question, the difference between two θ's calculated from equation 59 will be derived for a fixed θ_0 and v, but with two different values for z_0. One value is the tuned-for z_0 itself and the other is some z_b that denotes any general value of z_0 inside the filter's passband.

Thus, the passband of the velocity-profile filter is defined by requiring that the absolute value of the difference between the above two θ's after $T = N/R_f$ seconds is less than l_p (l_p defined as positive), or

$$-l_p \leq \theta_b(T) - \theta(T) = \theta_0 \left[\frac{z_b}{z_b - vT} - \frac{z_0}{z_0 - vT} \right] \leq l_p \tag{67}$$

When solved for the width of the passband in terms of sensor/object ranges, this inequality translates to

$$z_{bl} \leq z_b \leq z_{bh} \tag{68}$$

where the low end of the passband is

$$z_{bl} = \frac{z_0(1 + \theta_0/l_p) - vN/R_f}{\frac{z_0 R_f}{vN} + \theta_0/l_p - 1} \tag{69}$$

and the high end is

$$z_{bh} = \frac{z_0(1 - \theta_0/l_p) - vN/R_f}{\frac{z_0 R_f}{vN} - \theta_0/l_p - 1} \tag{70}$$

Example

With $v = 10$ m/sec, $N = 16$ frames, $R_f = 32$ frames/sec, $\theta_0 = 10$ pixels, $z_0 = 50$ m, and $l_p = 0.5$ pixel, $z_{bl} = 36.0$ m and $z_{bh} = 86.8$ m. In other words, if a velocity-profile filter is tuned for a pixel at distance of 10 pixels from the FOE and for a nominal object range of 50 m, the passband (or bandwidth) of this filter

extends between 36 and 86.8 m. If a larger range of depths is to be covered for a pixel at the same radius, several such filters have to be used and they have to be tuned to overlap, say, at their –3-dB points.

Figure 15(a) shows the time-evolution of the pixel used in the example above for the nominal, high, and low object/sensor ranges. It is seen that after 16 frames, the pixel separation between the nominal (center) graph and the graphs for the high and low depths are $\pm l_p = \pm 0.5$ pixels. Figure 15(b) shows a similar case except that $\theta_0 = 100$ instead of 10 pixels and the depth bandwidth of the filter is only between 48.0 m and 52.1 m. It is thus seen that many more profile filters are needed to cover any given range of depths for pixels that are farther from the FOE.

Figure 16 shows the dependence of the number of profile filters needed to cover a given range of depths on the pixel's distance from the FOE. The upper part of the figure shows horizontal lines with tic marks. Consider, for example, the second such line from the bottom (corresponding to a pixel at a distance of 20 pixels from the FOE) which is divided into four unequal segments by the tic marks. Starting from the maximum depth of $z_{bh} = 120$ m in this example (the high –3-dB point of the highest-depth filter), equation 70 is solved for the midpoint of the filter, z_0. Notice that the solution of z_0 in terms of z_{bh} is the same as that of z_{bl} in terms of z_0 in equation 69. Now, using the $z_0 = 77$-m depth just found, z_{bl} is solved for from equation 69 to find the low –3-dB point of the same (highest-depth) filter which is at $z_{bl} = 57$ m. This same point will also serve as the high –3 dB of the next filter below (in terms of depth) because all filters are to overlap at their –3 dB points. Thus, equation 70 is used repeatedly to go down in depth in the order of z_{bh}, z_0, z_{bl} per filter, and the process continued for lower-depth filters until the required range of sensor/object distances is covered. In this case, only two filters are needed: i.e., the highest-depth filter that covers the depths from 57 m to 120 m (with midpoint at 77 m), and the filter below that covers the depths from 38.5 m to 57 m (with midpoint at 46 m). Thus, every three tic marks define one filter, and the total number of filters required is (approximately) half the number of tic marks.

The lower part of figure 16 shows the total number of filters that can be counted from the upper part. It is seen that the total number of filters happens to be almost linear with the pixel's distance from the FOE, and that the densities of the filters increase with the pixel's distance from the FOE and decreases with the depth.

At this point it becomes apparent that the filter passbands, as seen in the upper part of figure 16, also represent the *accuracy* with which the three-dimensional space in front of the sensor can be mapped. Each filter can be thought of as a bin of the 3-D volume included in the FOV (two projection distances and depth). As expected, the depth accuracy improves with increasing projection values (pixel's distance from the FOE) and decreasing depth. There is always some minimum pixel distance from the FOE for which only one filter (or actually half a filter) can cover the whole range of depths. There is no point in processing pixels that are closer to the FOE because this minimum-distance pixel already corresponds to the coarsest accuracy, or to an uncertainty the size of the whole range of depths of interest.

The next four figures (figs. 17 through 20) are similar to figure 16 but with different parameters. The following general behavior should be noted.

1. The number of filters increases with the speed, v.

2. The density of the filters increases as the depth decreases.

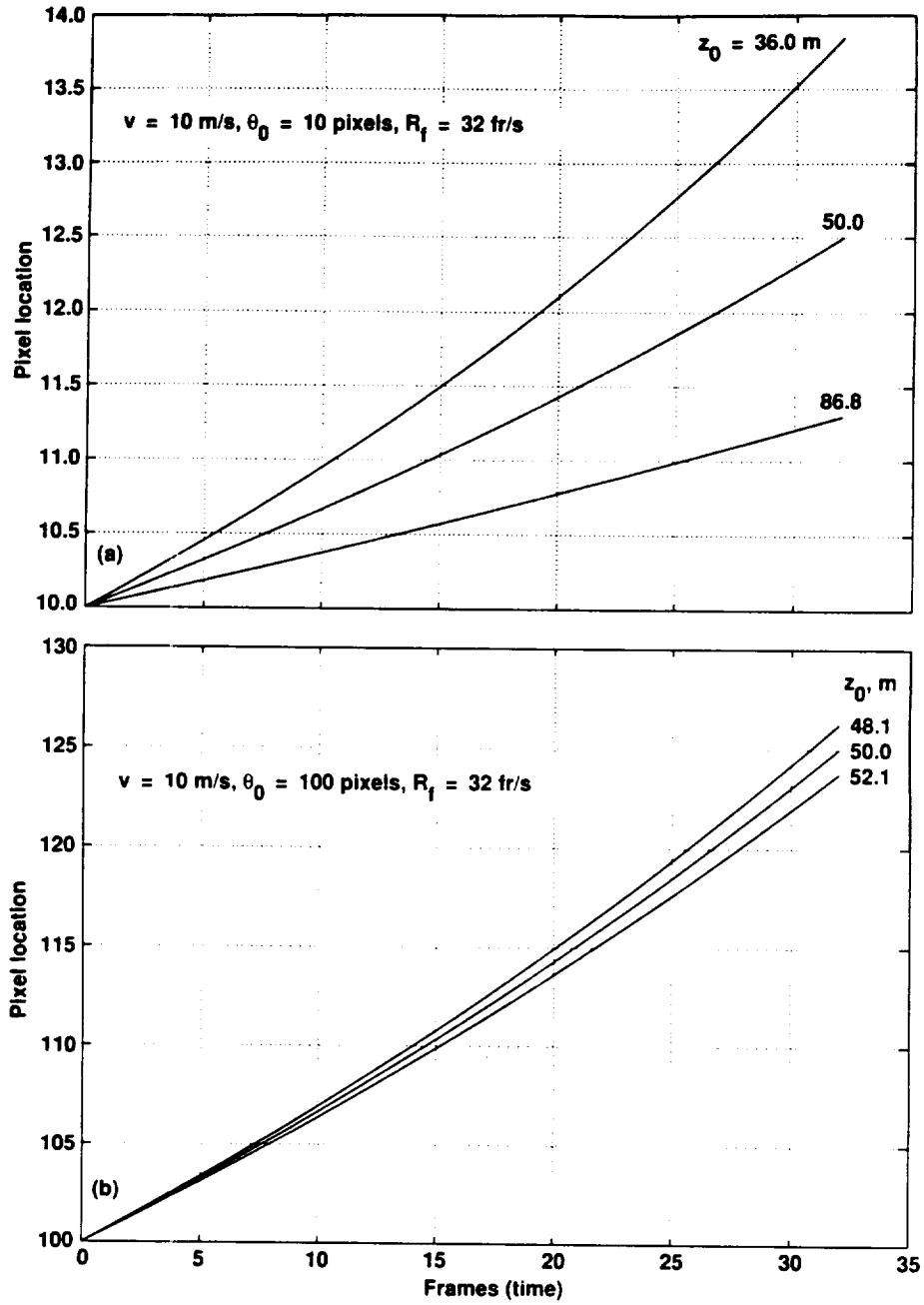

Figure 15. Time-trajectories of a pixel receding from FOE for nominal and extreme object/sensor ranges: v = 10 m/sec, R_f = 32 frames/sec. (a) θ_0 = 10 pixels, (b) θ_0 = 100 pixels.

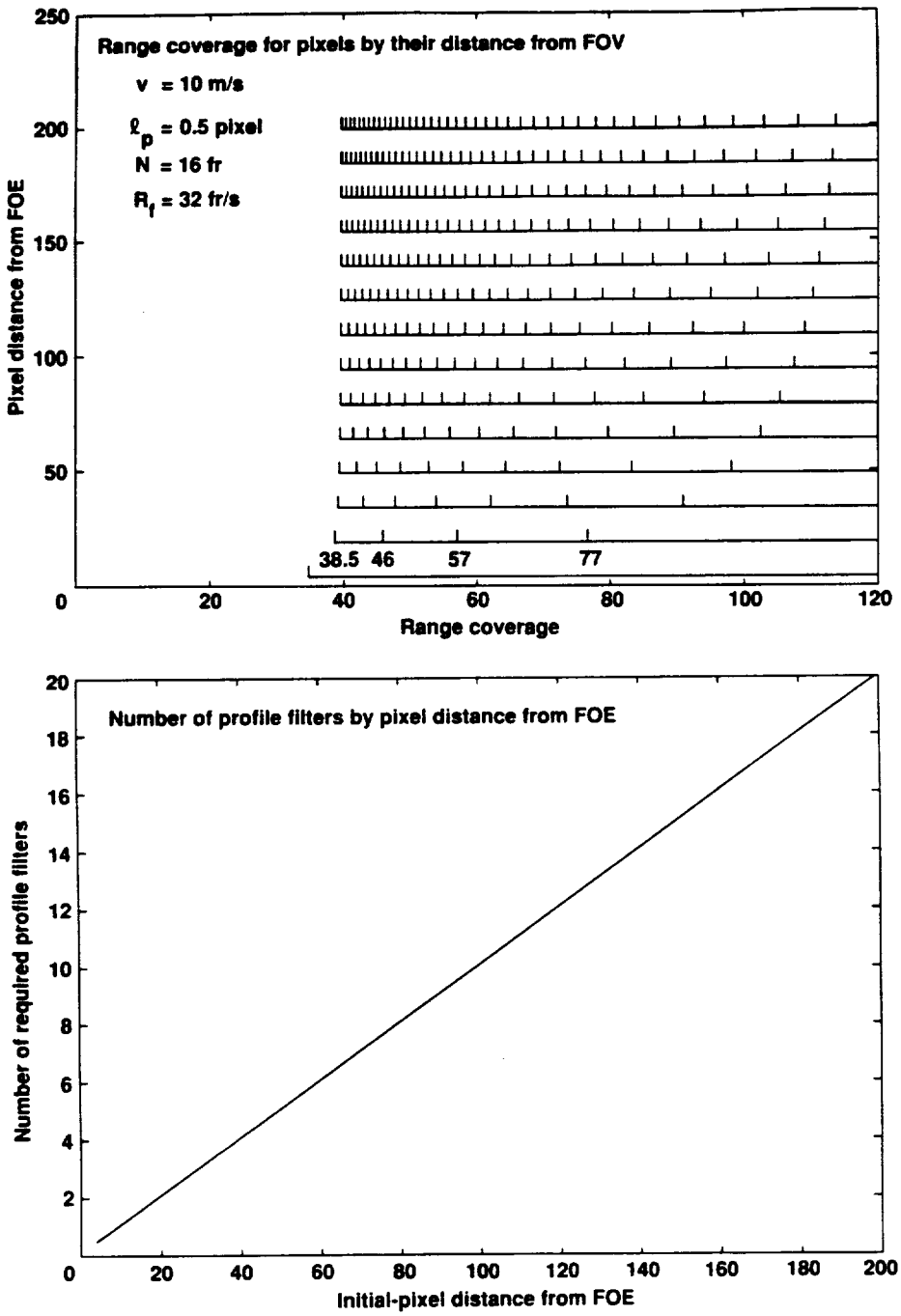

Figure 16. Number of filters required to cover a given range of sensor/object distances: v = 10 m/sec, l_p = 0.5 pixel, N = 16 frames, R_f = 32 frames/sec.

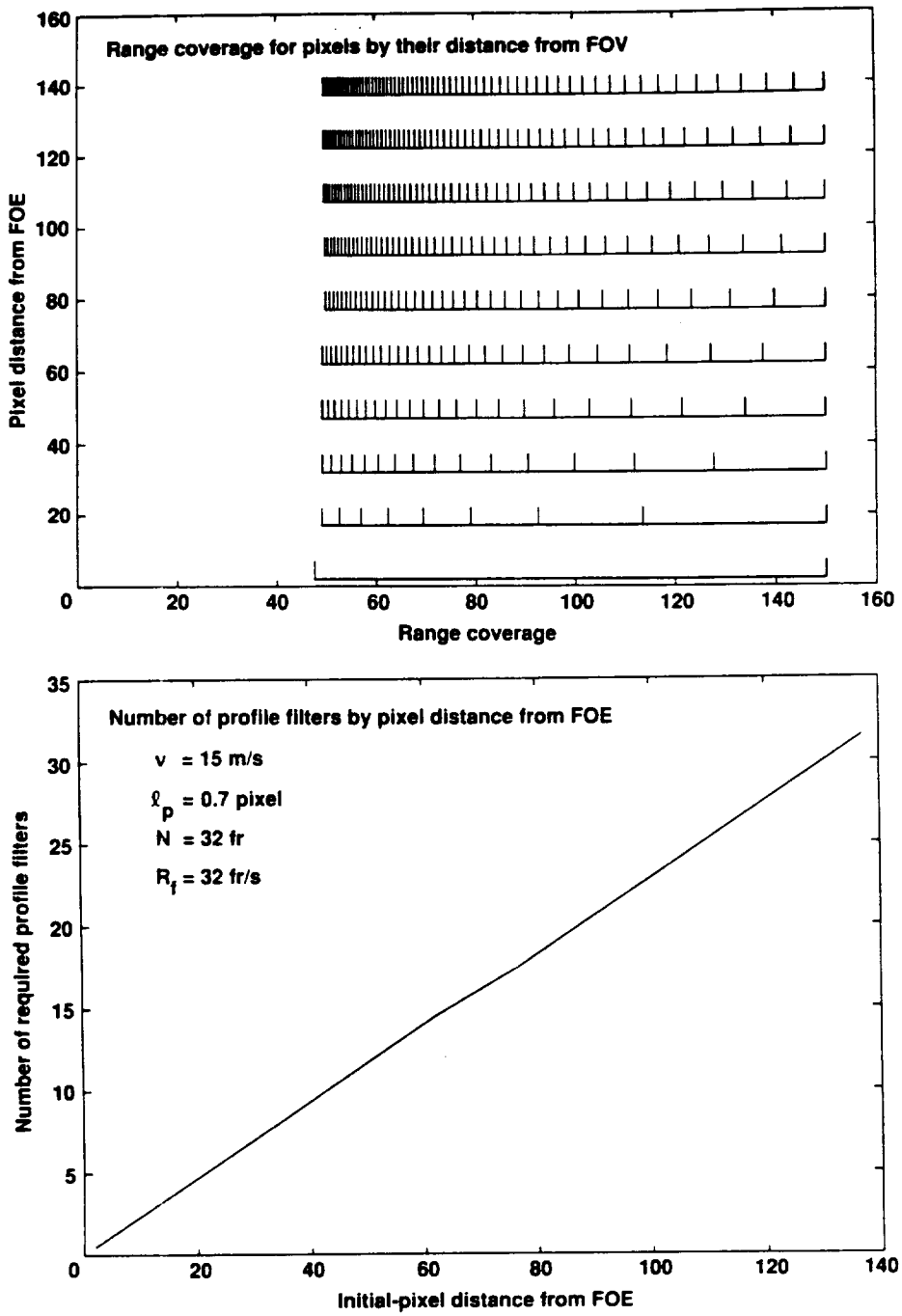

Figure 17. Number of filters required to cover a given range of depths: v = 15 m/sec, l_p = 0.7 pixel, N = 32 frames, R_f = 32 frames/sec.

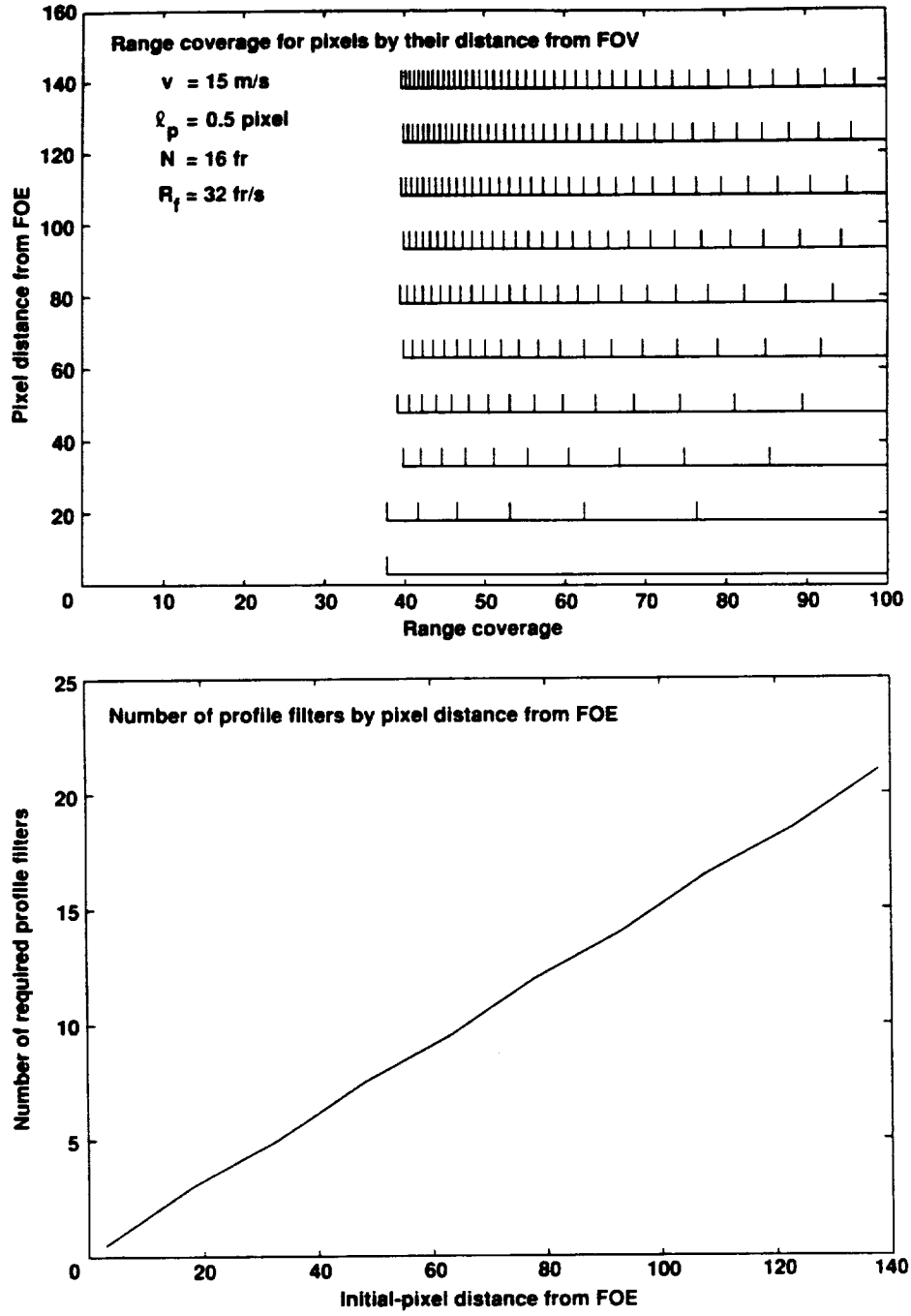

Figure 18. Number of filters required to cover a given range of depths: v = 15 m/sec, l_p = 0.5 pixel, N = 16 frames, R_f = 32 frames/sec.

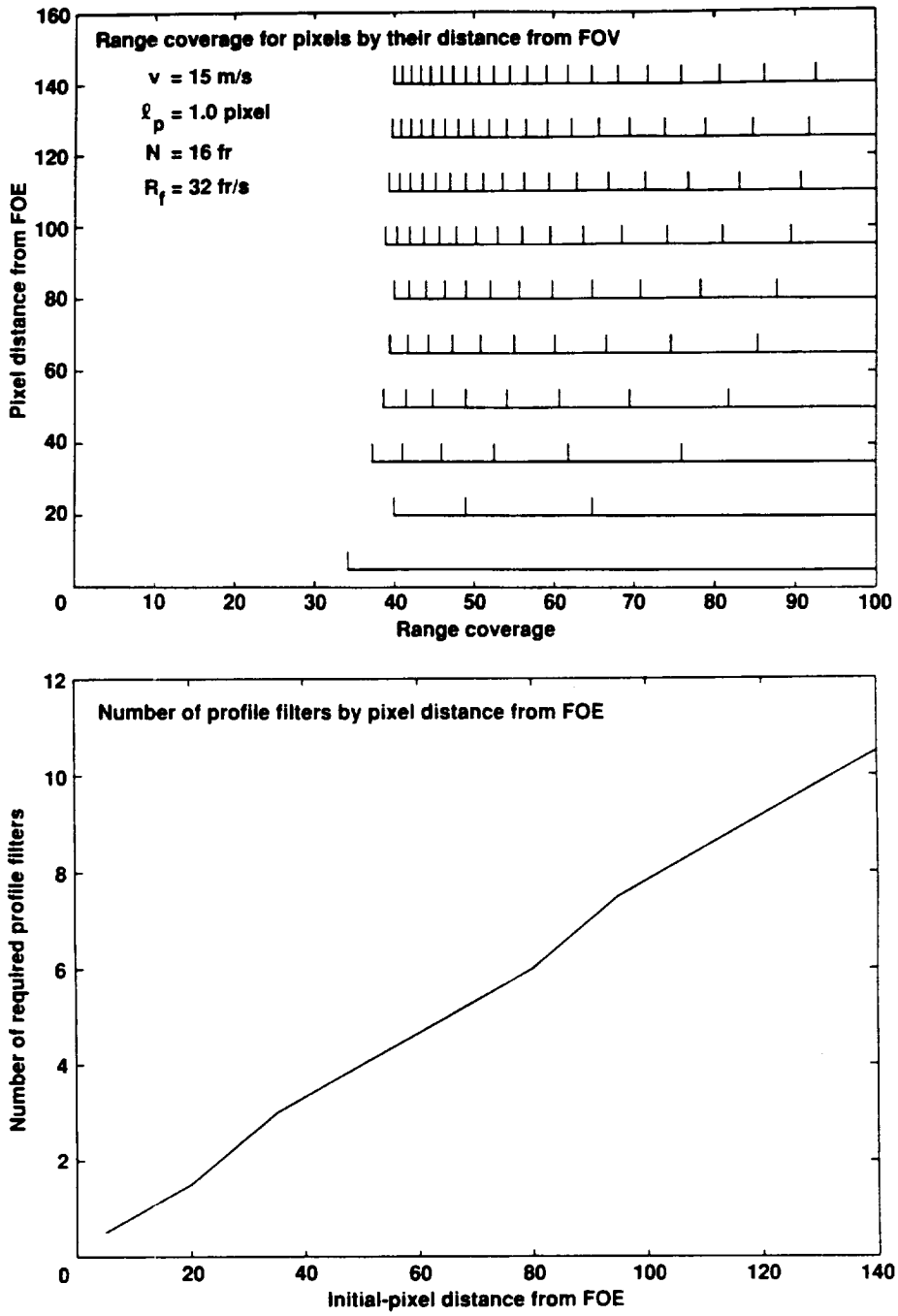

Figure 19. Number of filters required to cover a given range of depths: v = 15 m/sec, l_p = 1.0 pixel, N = 16 frames, R_f = 32 frames/sec.

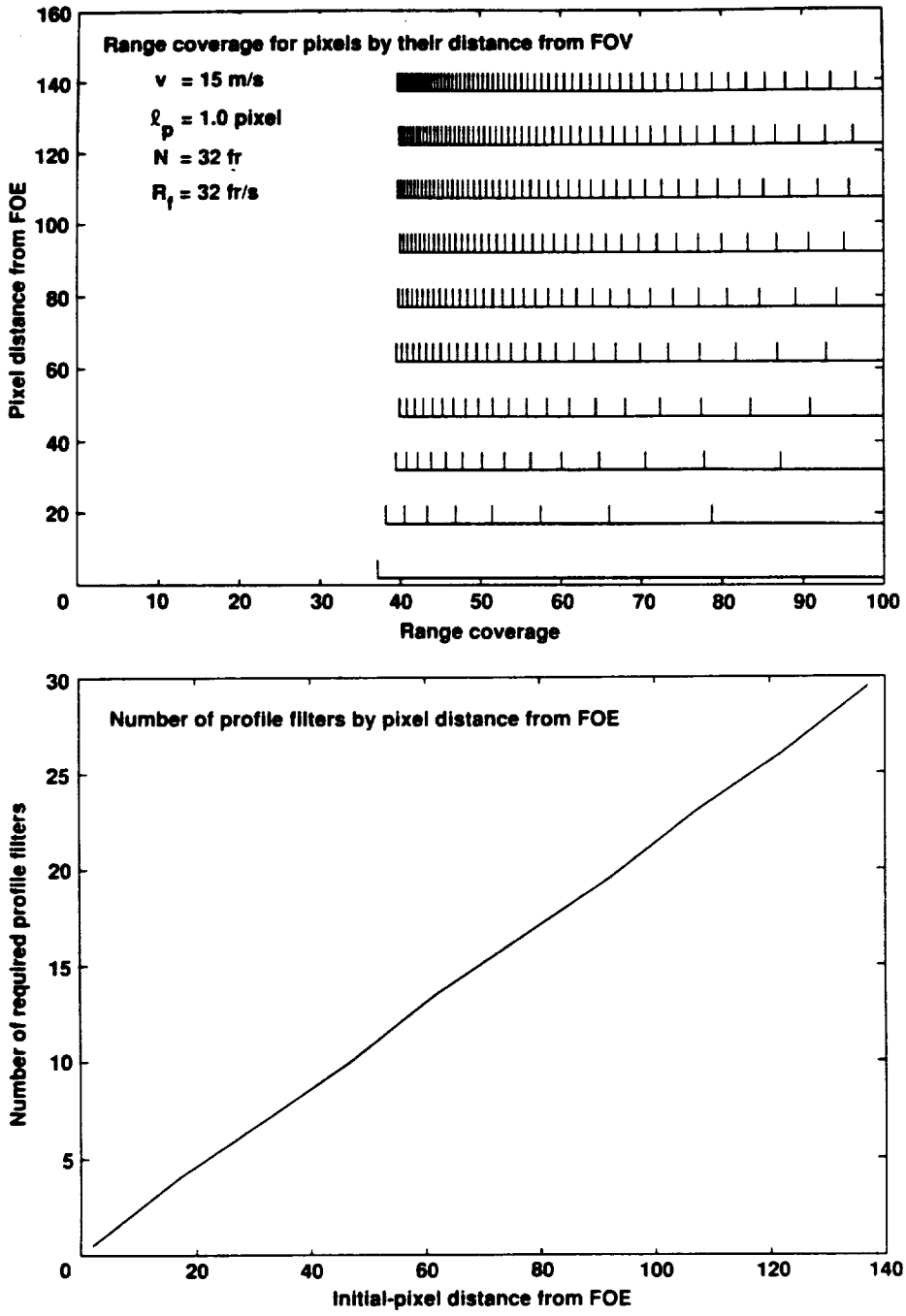

Figure 20. Number of filters required to cover a given range of depths: v = 15 m/sec, l_p = 1.0 pixel, N = 32 frames, R_f = 32 frames/sec.

3. The number of filters decreases as l_p increases because each filter is allowed to have a wider passband; thus there is decreased gain at the ends of the passband.

4. The number of filters increases with the number of integrated frames N, because increasing N makes a higher-gain and, thus, a narrower filter (for the same l_p).

5. It is seen from equation 69 that increasing the frames' rate, R_f, has the opposite effect of increasing N.

8 ALGORITHM IMPLEMENTATION

8.1 Main Routines

The algorithm is implemented as follows. First, equation 69 is used repeatedly for all $1 \leq \theta_0 \leq \theta_{max}$ to get the depth centers of all filters required to cover a given range of sensor-object depths. A $\theta_{max} = 200$ was chosen because pixels at that radius in the first frame will normally leave the FOV by the last frame, and they are not processed anyhow. This can be described by the simple-minded "program" below.

input N, R_f, v, l_p

do for $\theta_0 = 1, \theta_{max}$

$z = z_{max}$

until $z \leq z_{min}$

$z =$ FUNCTION (z)

enddo

The result of this part (SUBROUTINE ZFILTER) is stored in

1. nf(200) which gives the total number of depth filters needed for each θ in the range of 1 to 200 pixels distance from the FOE.

2. $z(200, 50)$ are the center depths for the same range of θ and for up to 50 filters per θ.

The second part of the algorithm has the general form shown below.

```
DO FOR ALL PIXELS

   convert cartesian coordinates of pixel with respect to the FOE
   into polar coordinates; thus get distance and angle from the
   horizontal axis in the image plane.

DO FOR ALL DEPTH FILTERS APPLICABLE TO THIS PIXEL
```

```
DO FOR ALL FRAMES

calculate pixel distance from FOE at this frame. Use same
angle to convert from polar to cartesian coordinates.

read value of this pixel at this frame (interpolate).

store value to calculate statistics

END FRAMES' LOOP

calculate variance of pixels\rq\ values which were picked up
from the frames according to the depth filter in use.

record the minimum variance among all depth filters up to
current one and the corresponding depth for this pixel.

END OF DEPTH FILTERS LOOP

END OF PIXELS' LOOP
```

The criterion for choosing the depth for a pixel is that its gray level must be approximately constant as it shows up in different locations in different frames. Thus, the depth-filer for which the ratio of variance/average over the frames is the minimum is chosen.

8.2 Preprocessing

The original imagery cannot be processed directly by the velocity-filtering method. The reason is that most practical imagery contains large objects (composed of many pixels), where each object is defined by a nearly-constant gray level. If we try to track any particular pixel found in the inside of the object, we may associate it with other pixels belonging to the same object in the other frames instead of with its own shifted versions, because the pixels are indistinguishable due to their similar gray level. This problem is encountered not only for pixels that are completely inside the object but even for pixels found on the edges of the object when their velocity vectors (away from the FOE) point toward the inside of the object. Since there is nothing to track in a featureless region (inside of an object), this behavior of the algorithm is not surprising. In fact, no algorithm can do better in such cases.

The natural thing to do in order to isolate features of an image is to employ some form of high-pass-filtering (HPF). The signed-gradient operation, which is simply the spatial derivative along the radius from the FOE, was chosen. This direction is chosen because each velocity filter picks up pixels from different frames along such lines. Thus, a "feature" is a variation in gray level in the radius direction. Two different methods of gradient-based preprocessing are considered here; they are discussed in the following.

Gradient operation. Straightforward gradient processing as described above brings up the features needed for velocity filtering but it has a crucial drawback when the preprocessed images are later processed with the main velocity-filtering algorithm. The problem is that close objects are always observed on the background of farther objects found behind them. The background objects, since they are farther from the sensor, move at lower speeds away from the FOE than the close objects. As a result, the gray-level differences at the edges of the close objects change with time as *different* background objects pass by behind the close object. In other words, the edges of objects, as obtained from the gradient operation, have a varying gray level over different frames, thus, they cannot be used with any algorithm that relies on constant gray level for every object over all frames. Only in unrealistic cases, in which the scenery is composed of nonoverlapping objects situated against a uniform background, will the simple gradient work. For this reason, the alternative discussed below is considered.

Gradient used only as a test. Another way of obtaining the feature-rich parts of the imagery is to use the gradient as a test in choosing the edges out of the *original* nongradient imagery. This is done by zeroing out all parts of the original imagery that do not have a gradient (its absolute value) above some threshold. It is important to emphasize that the featureless parts of the images have to be actually zeroed out; processing them cannot be simply skipped, for the reason explained earlier regarding edge pixels that move into the inside of the object.

8.3 Object's Expansion Between Frames

The linear dimensions of an object grow inversely proportional to the depth. Some comments are in order in this regard.

First, there is the question of compliance with the sampling theorem. Since the objects in all practical cases can be of any size, there will always be some objects that are smaller than a pixel; thus, there is the potential of aliasing. A simple way to avoid aliasing is to blur the optical point-spread function (PSF) in front of the sensor so that it is larger than two pixels.

Assume, for example, an object with a perfect (step-function) edge. Using a PSF of the above size will make this edge show up in roughly two pixels—*irrespective* of the size of the object itself, which increases with time. Thus, in such an ideal case, the algorithm does not have to accommodate for the object' s growth between frames. However, if an object is assumed that has a gradual gray-level transition at its edges which span a few pixels, the size (or width) of the edges will grow with time. In such a case the algorithm does have to accommodate for the object' s growth between frames. Moreover, in this case the different amplitudes of the gradient images at different depths must be compensated for. The reason is that an edge changes its width in terms of pixels as the depth changes, but it always describes an overall fixed gray-level difference between the two objects of the nongradient image which are found at both sides of the edge.

The problem is that there will always be a mix of object edges that fall into the above two categories. Moreover, every non-ideal object goes from the category of sharp-edges to that of the wide-edges as it approaches the sensor. Thought has been given to an edge-width test for determining the category, but it has not been tried out as yet. For the purposes of this report each category was simply used separately for the whole run.

8.4 Methods of Interpolation

The need for interpolation arises because spatial samples (pixels) of the physical scene are being dealt with. Since any discernible change of gray level in the image is *defined* to be an isolated object, it must be possible to deal with small objects of the order of 1- to 3-pixels. When considering edges, this size refers to the width of the edge. Thus, there is no point in trying to estimate the gray levels of the underlying objects; instead, the pixel readings themselves must be used to describe the gray levels of the physical object.

First method. If a pixel at frame No. 1 is considered to be the object itself or a part of the object, then this "object" (pixel) will normally intersect 4 pixels of any later frame as it moves with the optical flow. For the add-and-shift algorithm, the gray level of the object must be read at all frames. The question is what to do when the original pixel-object is expected to fall in between 4 pixels—that is, there are four observations that are affected by the original pixel's gray level, but they are also affected by other unaccounted-for pixels. The gray-levels of these other pixels that contribute into the four observed pixels can be thought of as random variables having a zero mean. The point is that by interpolating the values of the four observed pixels, a random variable is obtained having an *average* equal to the tracked pixel gray level and some variance.

Tracking a pixel-object through the frames means summing up its interpolated gray levels over all frames, that is, summing up N random variables. The variance of the sum variate (normalized by the average) decreases proportionate to N. Since the test for assigning depth to a pixel makes use of the sample variance (which is itself a random variable), there is some probability of wrong depth assignment associated with the randomness of the interpolation scheme described above.

As explained above, the origin of depth-assignment errors is in the interpolation scheme which is based on 4 pixels. Note that the tracked object-pixel occupies only one fourth of the area of these 4 pixels, whereas three fourths of the area is random. One obvious way to increase the ratio between nonrandom and random pixel areas involved in the interpolation is to *define* the object to be larger than one pixel, say 2×2 pixels. In such a case the above ratio increases from 1/3 to 4/5 because 9 pixels must now be observed instead of 4.

The definition of an object as a square of 2×2 pixels has a side effect in the form of blurring the image or low-pass filtering (LPF) it. This is why it is not suggested to go on and define 3×3 pixels, for example, as objects. It was found that a 2×2 size is a good compromise because with smaller objects the sampling theorem limits are encountered, and with larger ones the image is too blurred.

So far the effect of object expansion (or growth) on the interpolation process has been ignored. If this factor, is to be included, more than 4 pixels must be used. What is done in the algorithm is to observe *all* pixels in the current frame that have any common area with the expanded pixel tracked from frame 1, and to then interpolate on these pixels.

Second method. Another method was tried that avoids, at least in principle, the randomness associated with the first method above. Here it is, assumed that the minimum-size object at frame 1 is 2×2 pixels. At any later frame this object occludes at least 1 pixel of the later frame completely. This means that, given the gray-level values of the 4 pixels from frame-1, their contribution to the later-frame pixel can be calculated proportionately to their intersected area with that pixel.

The performance of this method was poor compared with that of the first method. Its sensitivity to the assumption that the underlying object can be described by the values of 4 pixels appears to be the reason. Here, the actual gray levels of the 4 pixels from frame 1 are used to predict the gray level of a pixel in a later frame, whereas in the first method we essentially work with the *sums* of the 4 pixel gray levels.

8.5 The Algorithm Parameters

This section lists all the algorithm parameters that have to be chosen and explains the considerations associated with their choice. When working with either the gradient images themselves or with the gradient-as-a-test method, it is necessary to specify the FOE and the threshold, Gth, under which the gradient value will be set to zero.

For the main algorithm it is necessary to choose the following parameters (in addition to specifying the FOE):

1. Beginning and end rows and columns of the images that are to be processed.

2. Gth: Threshold on the absolute value of the gradient imagery under which pixel values are set to zero.

3. Rth: The threshold on the ratio of rms/average (of object gray levels over all frames) under which the corresponding depth-filter is not allowed to be considered for a pixel. If this parameter is chosen too low (say, 0.1), more first-frame pixels are never assigned a depth; thus, they will appear blank in the final depth image. However, those pixels for which this ratio (for any depth filter) goes under the threshold will have good depth estimates. If this parameter is chosen high (say, 0.6), it allows almost all pixels to get a depth estimate but these estimates are, in general, less reliable.

4. Side: The side size in pixels of the minimum-size object (2 means 2×2 pixel objects). Choosing this parameter to be 1 causes the depth estimate for a large percentage (10% to 20%) of the pixels to be in error, meaning completely wrong depths. Choosing it to be 2 yields much better results because it does spatial LPF (or pixel averaging) along with decreasing the error variance associated with the interpolation. Choosing this parameter to be 3 or higher effectively blurs the image excessively and averages out objects with their (time-varying) background at the edges; thus, it undermines the essential premise of constant gray level over all frames.

5. z_{max} and z_{min}, that is, maximum and minimum depths: These parameters determine the range of depths over which centers of depth filters will be precalculated. Since a fixed number of depth filters is allowed and since calculating of their centers start from z_{max} going down, it usually happens that for large θ's, all depth filters are consumed before arriving at z_{min}. This is why it is necessary to "focus" the choice of depths range so that a limited range of interest is covered.

6. l_p: This is the width of the depth filter which is used here to define the separation between two adjacent depth filters, as explained in section 6. Choosing this parameter to be 0.5 or 1 pixel for example, causes the depth filters to overlap at their –3 dB or –6 dB points, respectively (see fig. 10). Obviously, when a higher l_p is chosen, coverage of the depths' range is allowed with fewer filters. This makes the algorithm more efficient but less accurate in terms of pixels' depth assignment.

7. R_f, or frames' rate in frames per second: This parameter is the actual frames' rate, so there is no choice to be made.

8. v, or the vehicle's speed in meters per second: Again, this is the actual vehicle's speed.

9. N or number of frames: The algorithm is working in batch for a fixed number of frames, so that N is given by the choice of the data set itself.

10. Expand: Equals 1 when taking into account object expansion with decreasing depth; equals 0 otherwise.

8.6 Data Sets Used

Two data sets were used. One is the 15-frame imagery taken at the NASA/FSN image processing laboratory (pencils on a table) and the second is the SRI data set (ladder and thrown shirt) that originally contained 128 images. To limit the number of processed frames in the SRI set, only every other frame of the last 64 frames was used, thus reducing this set to 32 frames. This section summarizes the results of a few hundred runs by presenting the final product of some representative cases in the form of a depth image based on frame No. 1 of each set.

Data set scaling. In order to use a data set that was taken in a laboratory environment, it was necessary to scale it in terms of vehicle speed, frame rate, and distances. The general formula for the depth scale factor is

$$S = \frac{v}{R_f v_l} \tag{71}$$

where v is the vehicle's actual speed in meters per second, v_l is the laboratory speed of the camera in meters per frame, and R_f is the laboratory frames rate in frames per second, which is assumed equal to the actual frame rate.

In the case of the FSN data set, the images were taken every 1.25 cm. If $R_f = 1$ frame/sec is chosen arbitrarily, the lab speed is $v_l = 0.0125$ m/frame. Similarly, if $v = 10$ m/sec is arbitrarily chosen, then $S = 800$. The above arbitrary choices cause z_{max} and z_{min} to be chosen accordingly, and affect the depth scaling of the output but not the essential results. Given that the depths of relevant objects in this set are between 0.33 and 0.9 m (at the first frame), this translates to 260 and 720 m, respectively.

The original 512×512 imagery was reduced to 256×256 by averaging every 4 pixels into a single pixel. The FOE for the reduced imagery is at row = 105 and col = 120.

In the case of the SRI imagery, $v_l = 10.16 \cdot 10^{-3}$ m/frame, and $R_f = 32$ frames/sec (using every other frame). If $v = 10$ m/sec, the scale factor is $S = 30.76$. The FOE for the reduced 256×256 imagery of this set is at row = 107 and col = 115.

Results for the FSN imagery. The original images (first and last) are shown in figures 21 and 22, and their gradient versions thresholded above 4 (the gray-level rms is 35) are shown in figures 23 and 24. Figures 25 and 26 show the original first and last frames in the parts that have an absolute gradient values above 5.

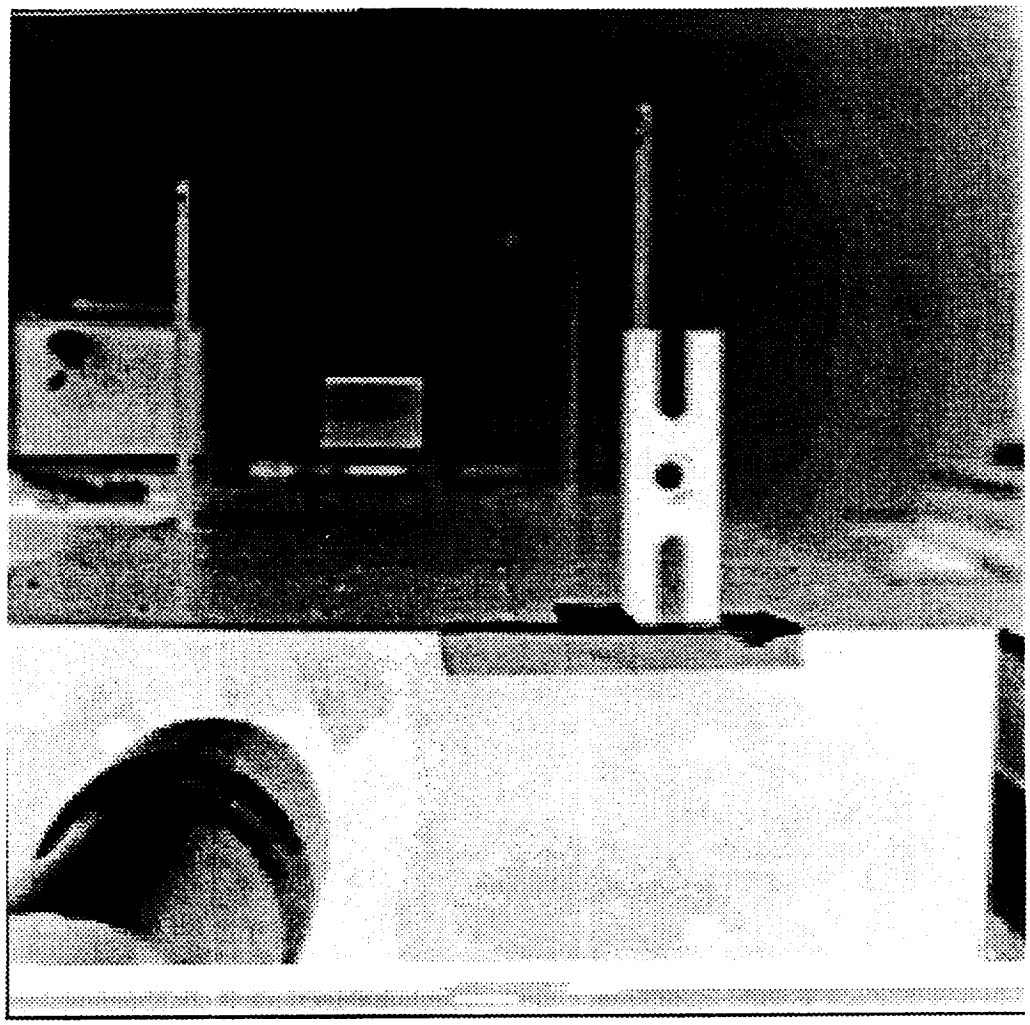

Figure 21. FSN: frame 1.

One important thing to notice about the gradient images is that, although the background is uniform, the edges of the same object do not have the same gray level at the beginning and end of the sequence. For example, the right-side edge of the closest pencil is much lighter in the last frame than it is in the first. Also, the left pencil seems to be out of focus at the first frame because its edges have much better contrast at the last frame.

Figure 27 shows the depth image obtained from processing the gradient imagery with gradient threshold $Gth = 2.5$, rms/average threshold $Rth = 0.5$, $z_{max} = 800$m, $z_{min} = 100$m, $l_p = 1$, side = 2, $v = 10$ m/sec, and allowing object expansion (expand = 1). The ground-truth (in parenthesis) and the min/max of measured depths for the three pencils and the bracket are shown in the figure. It is seen that the depth errors are of the order of 5% to 10%. The peripheral parts of the depth image are blank because all pixels that exit the FOV before the last frame are excluded.

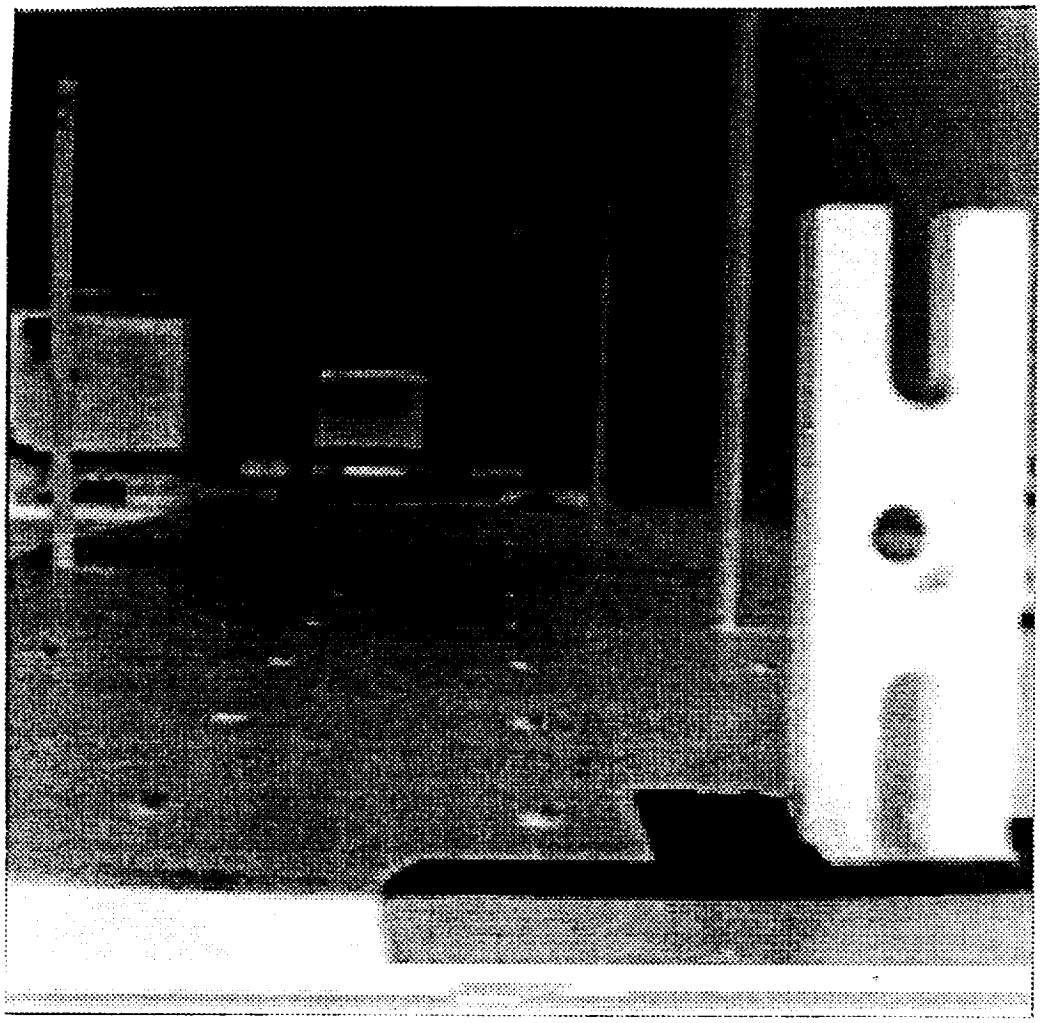

Figure 22. FSN: frame 15.

Figures 28 and 29 show the depth images obtained from processing the original images with gradient-as-a-test and expand = 0, 1, respectively. The other parameters common to these two cases are $Gth = 5$, $Rth = 0.4$, $z_{max} = 800\ m$, $z_{min} = 100\ m$, $l_p = 1$, and side = 2. The basic depths obtained are similar to those of the direct-gradient method, but there are more obvious and irregular errors, such as the short vertical black line on the right edge of the closest pencil. In comparing the last two figures, it is seen that compensating for object expansion made little difference with this imagery set. The reason may be that the set was obtained with a relatively wide optical PSF to begin with, so that further smoothing had little effect.

The performance of the algorithm on this imagery set was not up to expectation, a result of the unsatisfactory depth of the optical focusing (the PSF should be fixed irrespective of depth). This caused the images (especially the gradient ones) to show a non-fixed gray level which contradicts the uniform-gray-level assumption on which the algorithm is based.

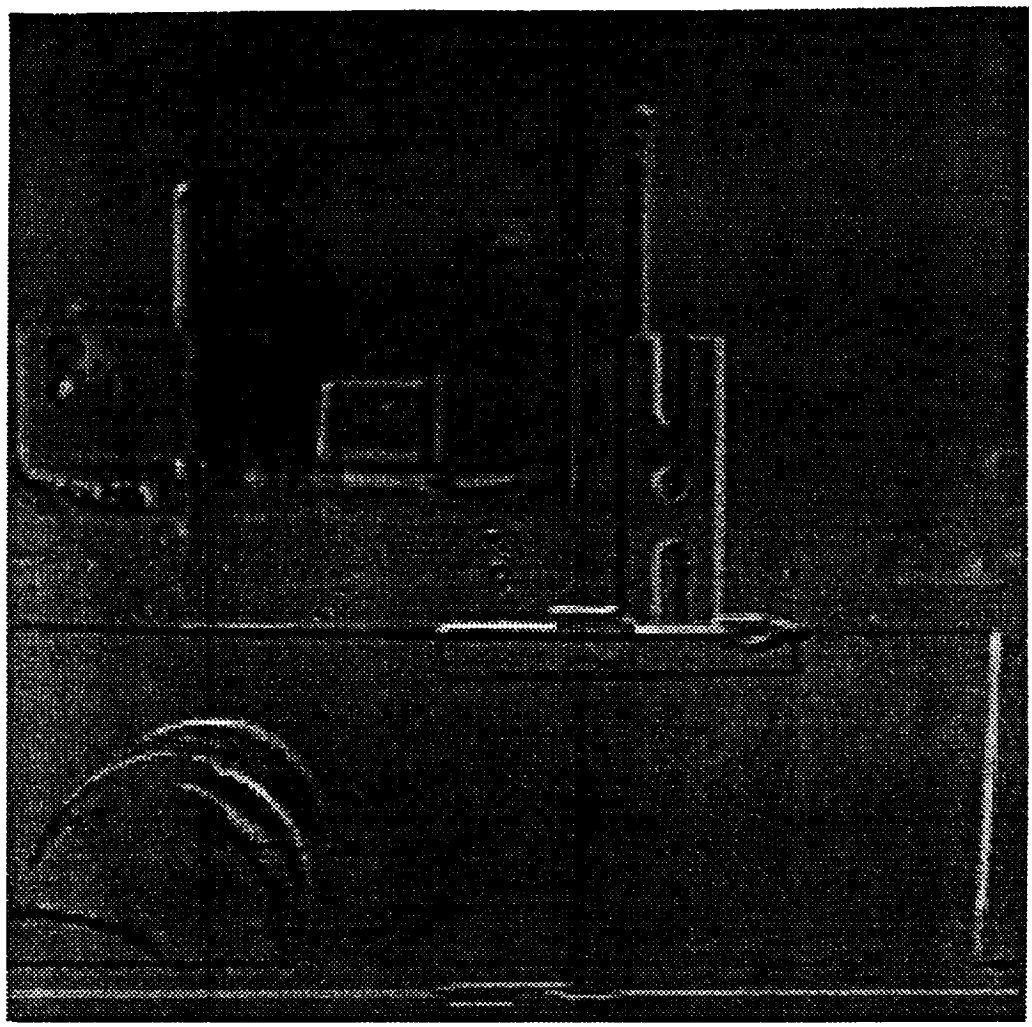

Figure 23. Gradient of FSN: frame 1.

Results for the SRI imagery. The original first image is shown in figure 30. The depth images obtained with the gradient and gradient-as-a-test preprocessing are shown in figures 31 and 32, respectively. All parameters are common for both cases: $Gth = 3$, $Rth = 0.4$, $l_p = 1$, side = 2 pixels, $v = 10\ m/sec$, $R_f = 32$ frames/sec, $N = 32$, expand = 0, $z_{max} = 120$ m, $z_{min} = 10$ m. In both methods there is a large percentage (10%-20%) of erroneous pixels, but it still looks as if the gradient method behaves better because it excludes most of the central region of the image which is found at a depth larger than z_{max}. Figure 33 shows a case similar to that in figure 31 except that $z_{max} = 250$ m, $z_{min} = 30$ m and expand = 1. The percentage of depth errors in this result is smaller (about 8%), and, in fact, this is the best result obtained so far through experimenting with the various parameters.

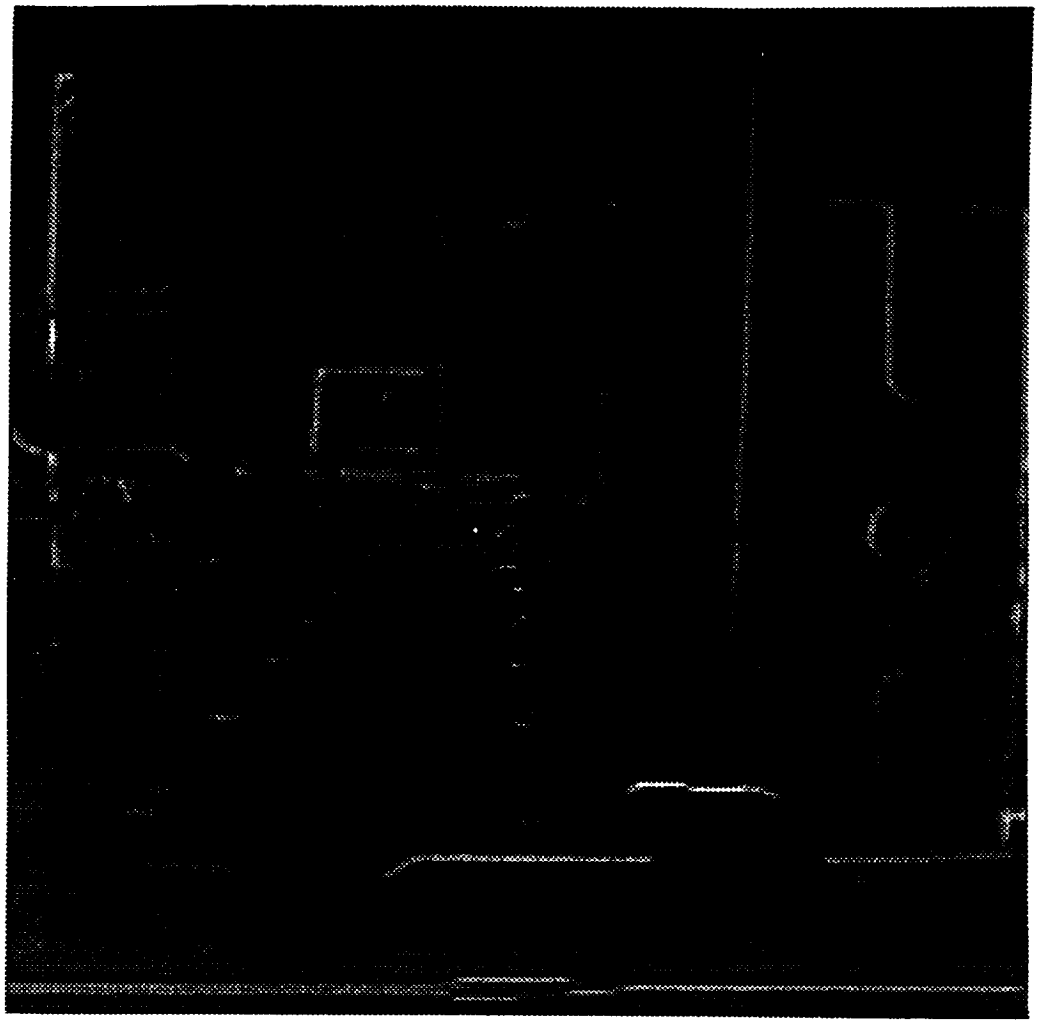

Figure 24. Gradient of FSN: frame 15.

8.7 General Discussion and Summary

Results obtained by applying the velocity filtering algorithm to two imagery sets, the FSN set and SRI set, have been presented. The results for the FSN set are clearly much better than those for the SRI set. The main reason is that the FSN imagery is much simpler, having distinct objects that move against a distant uniform background. Conclusions about the general performance potential of this algorithm are set forth below.

Given a camera that is *equally* focused at all depths and has a PSF of about 2 pixels, the performance is very much dependent on the scenery. If the scenery is composed of distinct objects that do not overlap during the entire observation time and are laid out against a uniform background, the algorithm is expected to perform well. Another necessary condition is that the objects must be separated by a few pixels at all times.

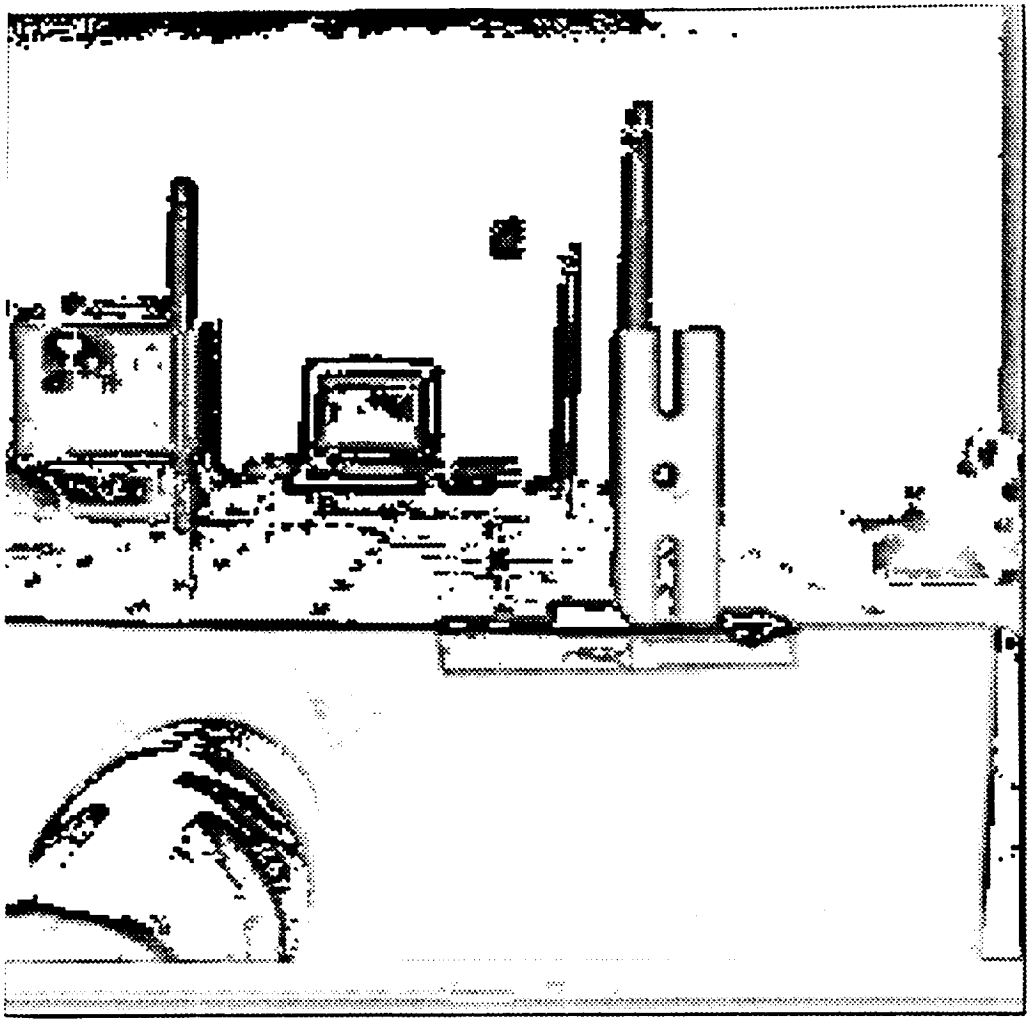

Figure 25. Original FSN frame 1 with gradient ≥ 5.

Fortunately, it is expected that real scenes will not suffer that much from the occlusion problem but will still contain objects that are of all sizes and that are at close proximity. One possible recourse is to use the gradient method with an automatic threshold setting so as to maintain a fixed number of above-threshold pixels (reminiscent of a constant-false-alarm-rate detection method, or CFAR). The same threshold should of course be used for all the frames. Doing that has the potential of creating more distance between distinct objects but it does not guarantee that behavior.

Although the algorithm requires further development and refinement and may not be applicable to all situations, it is considered to be practical and useful.

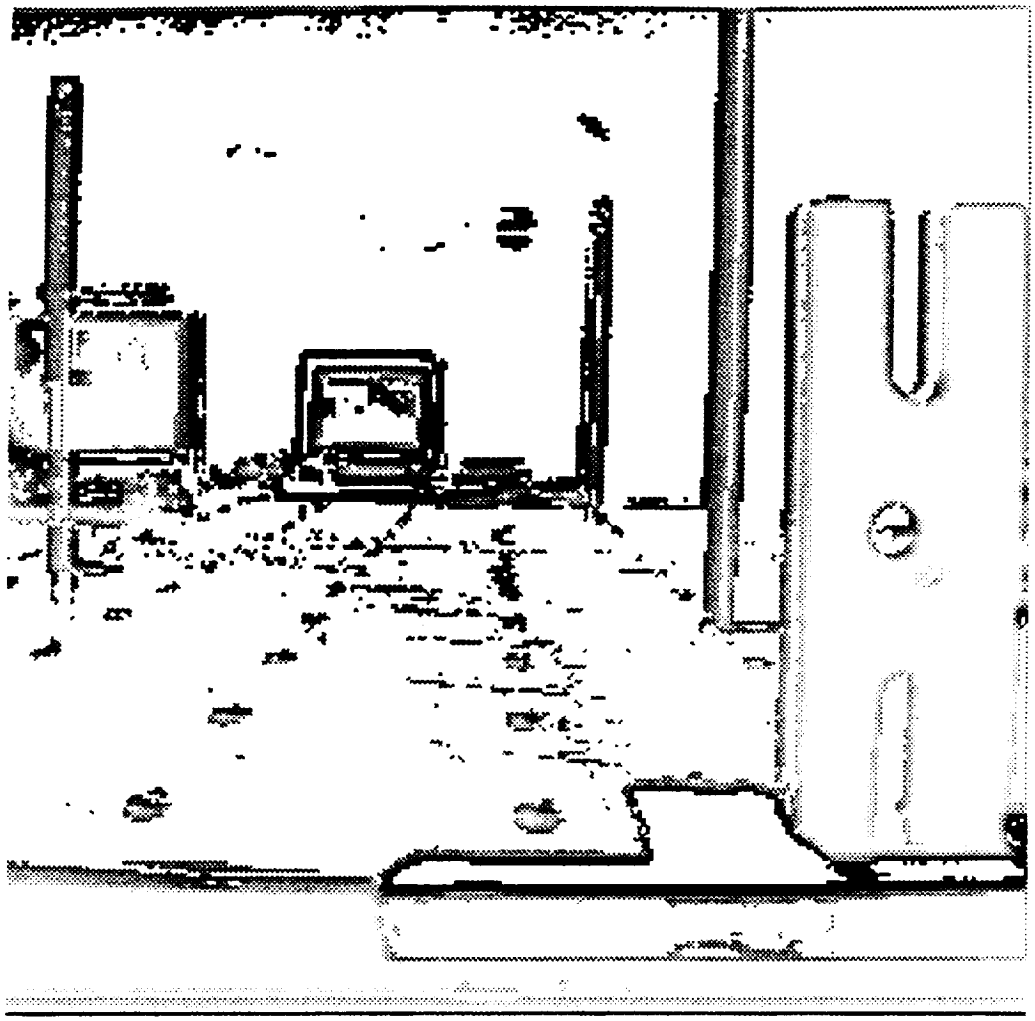

Figure 26. Original FSN frame 15 with gradient ≥ 5.

8.8 Future Work

All the practical problems encountered in applying the algorithm to real imagery do not appear in the earlier analysis which dealt with performance evaluation for cases that agree with the scene conditions specified above. Further analysis, algorithm development, and experimental work is clearly necessary. The following constitute the principal tasks that remain to be done.

1. Develop methods to deal with the problem of object occlusions. The main idea that can be utilized here is that the image-plane speed of an occluding object is known to always be larger than that of the background object, simply because it is closer.

2. Develop methods to dilute scene areas where many small objects interact by excluding such areas altogether, requiring some minimum pixel distance between objects and deleting the in-between ones, or picking up objects having gray levels within some given band.

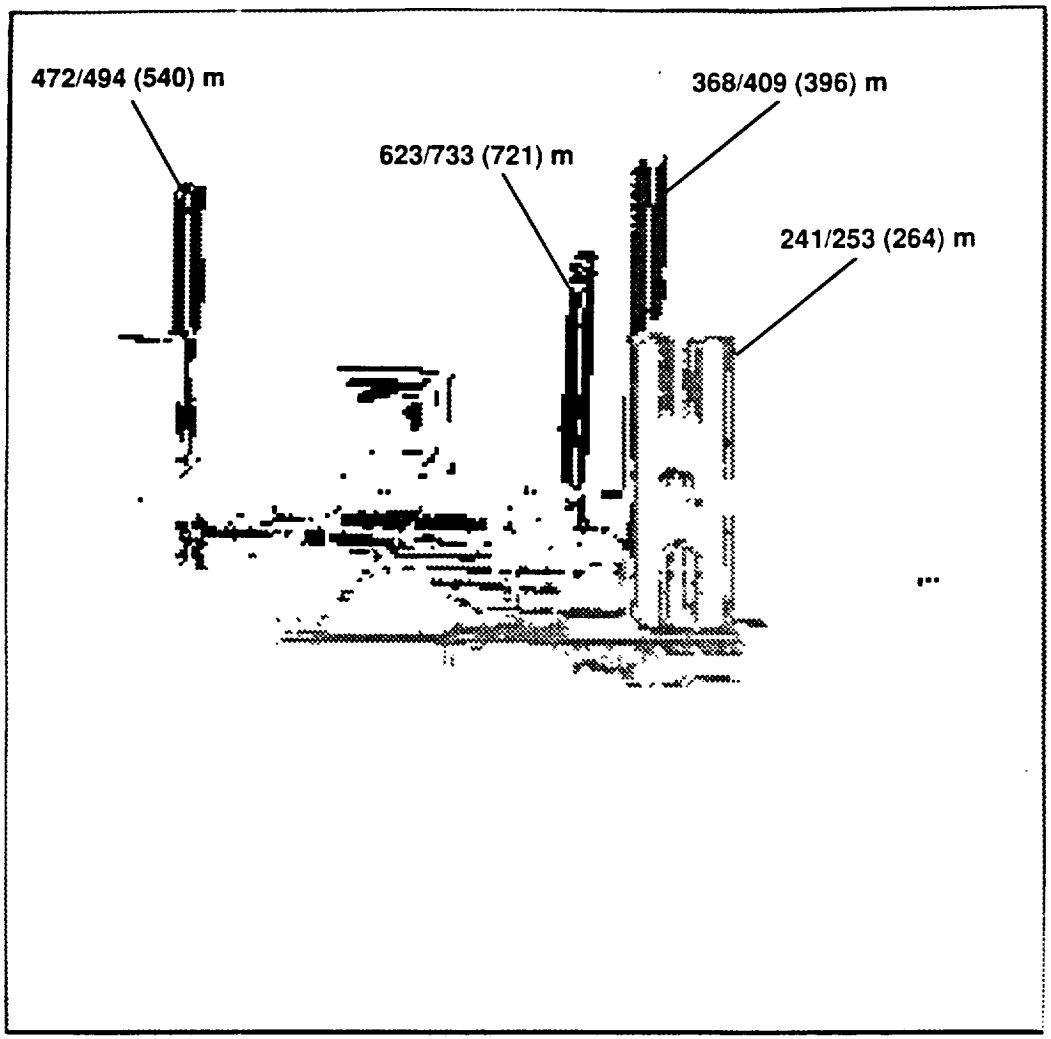

Figure 27. FSN depth image based on gradient imagery: with expand = 1, $Gth = 2.5$, $Rth = 0.5$, z_{min}/z_{max} = 100/800 m, l_p = 1 pixel (–6 dB), side = 2 pixels, v = 10 m/sec, R_f = 1 frames/sec, scale = 800, N = 16.

3. Develop methods to group close-by equal-gray-level pixels into larger objects, thus increasing the robustness of the algorithm.

4. Develop better tests for depth allocation to pixels or to objects (the gray-level variance/average over all frames is currently in use).

5. Write a simulation program that can simulate each item of concern separately and under controlled conditions.

6. Analyze performance and compare experimental results for various versions of the algorithm.

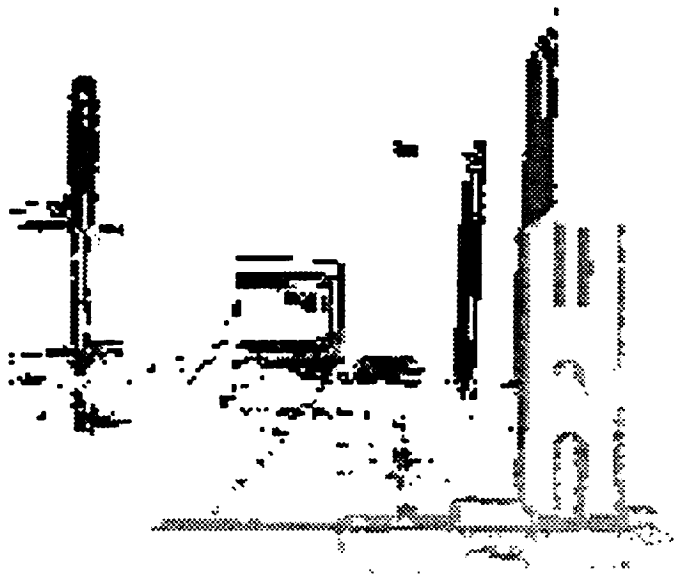

Figure 28. FSN depth image based on gradient-as-a-test imagery: with expand = 0, Gth = 2.5, Rth = 0.5, z_{min}/z_{max} = 100/800 m, l_p = 1 pixel (–6 dB), side = 2 pixels, v = 10 m/sec, R_f= 1 frames/sec, scale = 800, N = 16.

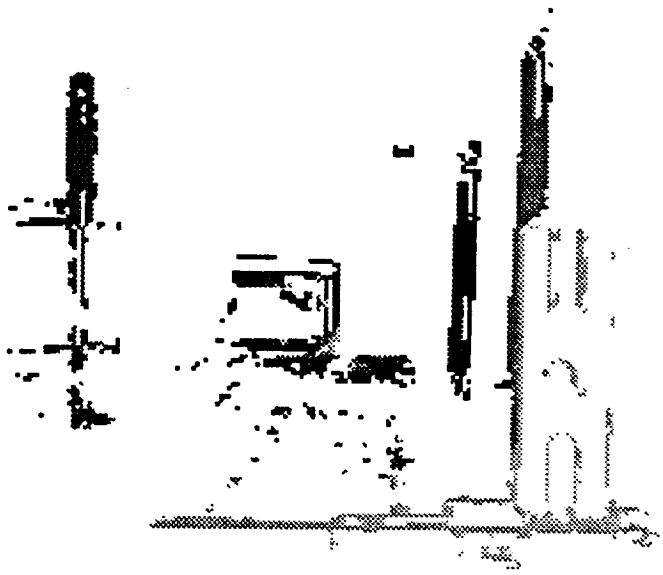

Figure 29. FSN depth image based on gradient-as-a-test imagery: with expand = 1, Gth = 2.5, Rth = 0.5, z_{min}/z_{max} = 100/800 m, l_p = 1 pixel (–6 dB), side = 2 pixels, v = 10 m/sec, R_f= 1 frames/sec, scale = 800, N = 16.

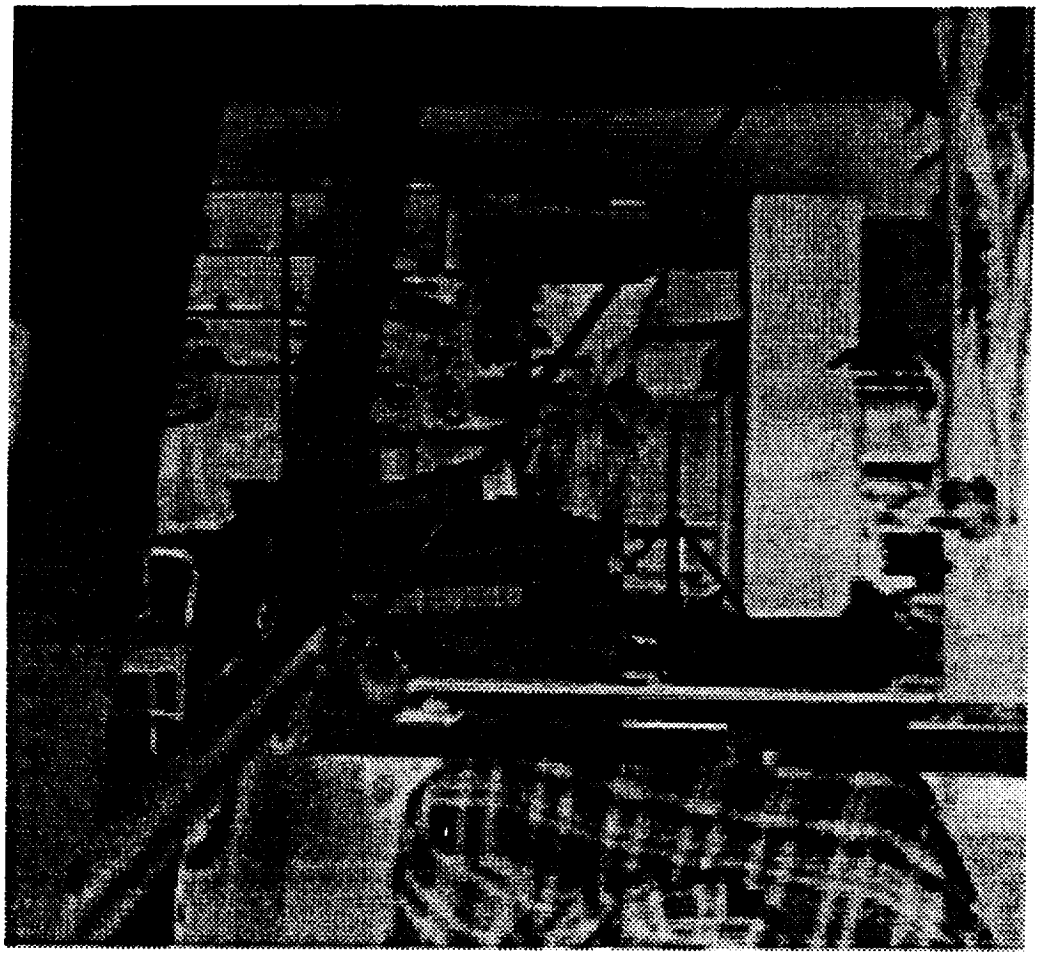

Figure 30. First SRI image.

Figure 31. Depth image of SRI set based on gradient imagery: with z_{min}/z_{max} = 10/120 m, Gth = 3.0, Rth = 0.4, l_p = 1 pixel (–6 dB), side = 2 pixels, v = 10 m/sec, expand = 0, R_f = 32 frames/sec, scale = 30.76, N = 3.

Figure 32. Depth image of SRI set based on gradient-as-a-test imagery: expand = 0, z_{min}/z_{max} = 10/120 m, Gth = 3.0, Rth = 0.4, l_p = 1 pixel (–6 dB), side = 2 pixels, v = 10 m/sec, R_f = 32 frames/sec, scale = 30.76, N = 32.

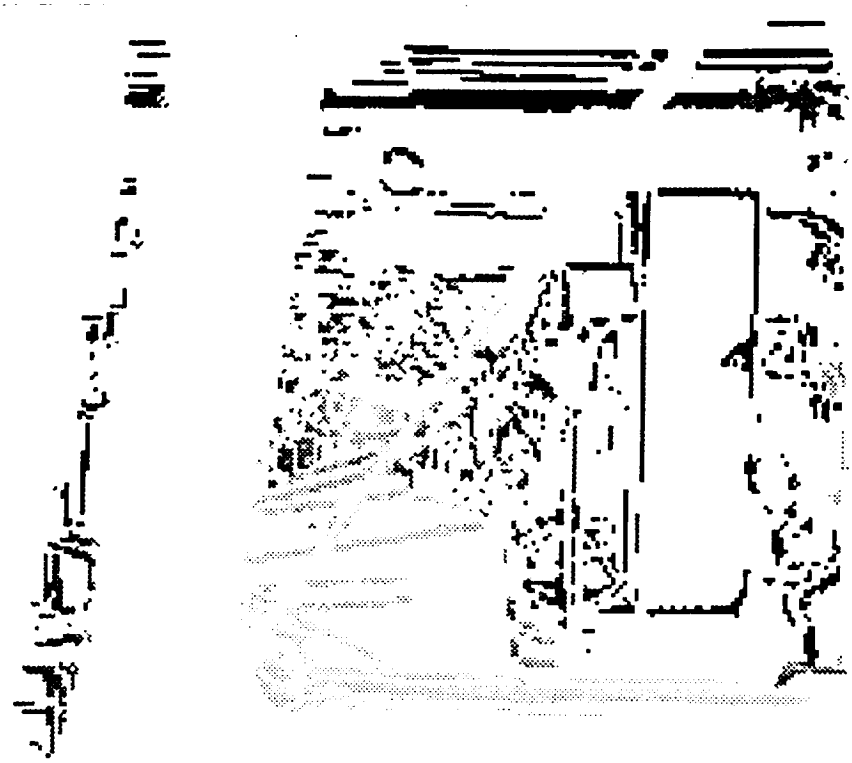

Figure 33. Depth image of SRI set based on gradient imagery: expand = 1, z_{min}/z_{max} = 30/250 m, Gth = 3.0, Rth = 0.4, l_p = 1 pixel (–6 dB), side = 2 pixels, v = 10 m/sec, R_f = 32 frames/sec, scale = 30.76, N = 32.

REFERENCES

1. Reed, I. S.; R. M. Gagliardi; and H. M. Shao: Application of Three-Dimensional Filtering to Moving Target Detection. *IEEE Trans. Aerosp. and Electronic Sys.*, vol. AES-19, no. 6, Nov. 1983, pp. 898-905.

2. Watson, A. B.; and A. J. Ahumada, Jr.: Model of Human Visual-Motion Sensing. *J. Opt. Soc. Am*, A, vol. 2, no. 2, Feb. 1985, pp. 322-341.

3. Porat, B.; and B. Friedlander: A Frequency Domain Approach to Multiframe Detection and Estimation of Dim Targets. *ICASSP*, Apr. 1987

4. Reed, I. S.; R. M. Gagliardi; and L. B. Stotts: Optical Moving Target Detection with 3-D Matched Filtering. *IEEE Trans. Aerosp. and Electronic Sys.*, vol. AES-24, no. 4, July 1988, pp. 327-335.

5. Stotts, L. B.; I. S. Reed; and R. M. Gagliardi: Air Vehicle Detection by Using an Ensemble of Moving-Picture Images. *NOSC TR-1163*, Jan. 1987.

6. Kendall, W. B.; and W. J. Jacobi: Passive Electro-Optical Sensor Processing for Helicopter Obstacle Avoidance. *Space Computer Corporation*, Santa Monica, Calif., Dec. 1988.

7. Bracewell, R.: *The Fourier Transform and Its Applications*, McGraw Hill, 1965, pp. 69-90.

8. Papoulis, A.: *Systems and Transforms in Optics*. McGraw Hill, 1969.

9. Wozencraft, J. M.; and I. M. Jacobs: *Principles of Communication Engineering*. Wiley, 1965, pp. 229-231.

10. Nahi, N.E.: *Estimation Theory and Applications*. Wiley, 1969.

NASA
National Aeronautics and Space Administration

Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA TM-102802		

4. Title and Subtitle
Velocity Filtering Applied to Optical Flow Calculations

5. Report Date
August 1990

6. Performing Organization Code

7. Author(s)
Yair Barniv

8. Performing Organization Report No.
A-90108

10. Work Unit No.
505-66-51

9. Performing Organization Name and Address
Ames Research Center
Moffett Field, CA 94035-1000

11. Contract or Grant No.

13. Type of Report and Period Covered
Technical Memorandum

12. Sponsoring Agency Name and Address
National Aeronautics and Space Administration
Washington, DC 20546-0001

14. Sponsoring Agency Code

15. Supplementary Notes

Point of Contact: Yair Barniv, Ames Research Center, MS 210-9, Moffett Field, CA 94035-1000
(415) 604-5451 or FTS 464-5451

16. Abstract

Optical flow is a method by which a stream of two-dimensional images obtained from a forward-looking passive sensor is used to map the three-dimensional volume in front of a moving vehicle. Passive ranging via optical flow is applied here to the helicopter obstacle-avoidance problem. Velocity filtering is used as a field-based method to determine range to all pixels in the initial image. This report expands on the theoretical understanding and performance analysis of velocity filtering as applied to optical flow and presents experimental results.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Matched filtering, Optical flow Shift-and-add algorithm 3-D Fourier transform filtering Velocity filtering	Unclassified-Unlimited Subject Category – 04

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	60	A0

NASA FORM 1626 OCT 86

For sale by the National Technical Information Service, Springfield, Virginia 22161