Max-Planck-Institut
für biologische Kybernetik
Arbeitsgruppe Bülthoff

Technical Report No. 13 Febuary 1995

Real-Time Optical Flow Extended in Time

Ted Camus & Heinrich H. Bülthoff

Abstract

Currently two major limitations to applying vision in real tasks are robustness in real-world, uncontrolled environments, and the computational resources required for real-time operation. In particular, many current robotic visual motion detection algorithms (optical flow) are not suited for practical applications such as segmentation and structure-from-motion because they either require highly specialized hardware or up to several minutes on a scientific workstation. In addition, many such algorithms depend on the computation of first and in some cases higher numerical derivatives, which are notoriously sensitive to noise. In fact the current trend in optical flow research is to stress accuracy under ideal conditions and not to consider computational resource requirements or resistance to noise, which are essential for real-time robotics. As a result robotic vision researchers are frustrated by an inability to obtain reliable optical flow estimates in real-world conditions, and practical applications for optical flow algorithms remain scarce. Algorithms based on the correlation of image patches have been shown to be robust in practice but are in general infeasible due to their computational complexity. This paper describes a space-time tradeoff to this algorithm which converts a quadratic-time algorithm into a linear-time one, as well as a method for dealing with the resulting problem of temporal aliasing, resulting in an algorithm that can run at over 6 frames per second on an 50 MHz Sun Sparcstation 20.

Most of the work presented in this techreport was done during the graduate studies of Ted Camus at Brown University under the supervision of Heinrich Bülthoff and Tom Dean. The work was supported in part by a National Science Foundation Presidential Young Investigator Award IRI-8957601 to Tom Dean, by the Air Force and the Advanced Research Projects Agency of the Department of Defense under Contract No. F30602-91-C-0041, and by the National Science foundation in conjunction with the Advanced Research Projects Agency of the Department of Defense under Contract No. IRI-8905436. Ted Camus' study at the Max-Planck-Institut für biologische Kybernetik at Tübingen during the summer of 1994 was supported by a grant from the Deutscher Akademischer Austauschdienst. Ted Camus' present address is: NIST, Intelligent Systems Division, MET220 B-124, email: `tac@cme.nist.gov`.

1 Introduction

Despite strong incentives to use vision as a powerful means of perception on mobile robots, and extensive mathematical analysis of computer vision, practical real-time robotic vision algorithms remain elusive, as recently noted by D. Touretzky et.al. ([TWR94]):

> "But the real problem with video cameras is that image processing is computationally expensive. Even something as simple as calculating real-time optic flow requires more processing power than is practical for a mobile robot. Yet optic flow is known to be computed in the early stages of mammalian vision. Such observations underscore the tremendous gulf that remains between today's digital computers and real nervous systems."

It is undeniably true that biological organisms are able to perform such tasks as motion detection, and most evidence from the field of computer vision (even to the present day) would tend to confirm the above statement that it might not seem possible to be able to compute optical flow, or point-wise visual motion detection, in real-time using computing power that was practical for a mobile robot. Nonetheless, [C93] proposed that optical flow could indeed be computed in real-time using computing power that is appropriate and practical for a mobile robot, that the optical flow thus produced would be robust, and that it would be sufficiently accurate to be used for certain robotic vision tasks. Here "real-time" is defined in the loose sense of being fast enough to be calculated on-line and be usable in some reactive system; on the order of 4-5 frames per second at a minimum. "Practical for a mobile robot" implies something on the order of a current low-end workstation or possibly a mid- to high-end personal computer. Unfortunately, in the past this has proven very difficult, since such the calculation of optical flow is traditionally computationally intensive, very sensitive to noise, and often inaccurate.

In fact currently the two major limitations to applying vision in real tasks are robustness in real-world, uncontrolled environments, and the computational resources required for real-time operation. In particular, many current optical flow algorithms are not suited for practical applications such as segmentation and structure-from-motion because they either require highly specialized hardware or up to several minutes on a scientific workstation. For example, [WM93] quotes 4 minutes on a Sun workstation and 10 seconds on a 128-processor Thinking Machines CM5 supercomputer. If computer processing power continues to grow at approximately 50% per year [PH94], then such an algorithm would take approximately 16 years for the 1000-fold increase in performance necessary for real-time rates on an ordinary workstation. In addition, many such algorithms depend on the computation of first and in some cases higher numerical derivatives, which are notoriously sensitive to noise. In fact the current trend in optical flow research ([BFBB92], [WM93]) is to stress accuracy under ideal conditions (either no noise or at best, modeling the noise as Gaussian, a questionable assumption), and not to consider computational resource requirements or resistance to noise, which are essential for real-time robotics. As a result robotic vision researchers are frustrated by an inability to obtain reliable optical flow estimates in real-world conditions, and practical applications for optical flow algorithms remain scarce.

Another trend is to only compute *qualitative* optical flow, such as *normal* flow [HS81]. Although normal flow can be computed much more easily than full optical flow and may be sufficient input for many useful tasks (e.g. [HA91], [CHHN95]) there will be many cases where full flow is preferable, if it can be computed efficiently and robustly.

Algorithms based on the correlation of image patches (e.g. [BLP89a]) have been shown to be robust in practice but are in general infeasible due to their computational complexity. This paper describes a space-time tradeoff to this algorithm which converts a quadratic-time algorithm into a linear-time one, as well as a heuristic for dealing with the resulting problem of temporal aliasing.

2 Optical Flow

The primary difficulty with computer vision is that a single 2-dimensional image can arise from effectively an infinite number of 3-dimensional scenes. A characteristic of image formation, one that must be well understood in order to attempt to understand the scene giving rise to a particular image, is the unavoidable loss of information in the imaging process. A single pixel is represented by a single intensity value, which itself is the result of many factors, such as the intensity, color, and location of the light source or sources, and the orientation, reflectance, and transparency of the objects in the scene, as well as the optical and electrical properties of the imaging device itself. Each of these effects may be understood individually, but they are mutually confounding; a single 2-dimensional image can correspond to an infinite number of 3-dimensional scenes.

Motion detection is an essential means of perception for an organism for determining the organism's own movement, the structure of the organism's environment, as well the motions of other organism's in its environment. Movement in images can be divided into *real* motion across the retina, and *apparent* motion, the perception of motion that arises when an object or pattern appears to instantaneously move from one place to another [RA86]. In the case of the latter, the *correspondence* problem must be solved [U79], that of matching points or features in one image with that of another. If motion across the retina is *real*, then it may be possible to detect motion using the spatial and temporal derivatives in an image, however, if the motion (or time delay) between successive frames is sufficiently small, then *apparent* motion begins to approximate *real* motion, thus it can be argued that studying the former is all that is necessary.

An object moving in 3-D space has a three dimensional velocity vector field $W(x, y, z)$. This 3-D motion field is projected onto the retina of an observer as the 2-D motion field $W_p(x, y)$. Unfortunately, this 2-D motion field may not be perceived directly since it is a purely geometrical concept; all that may be detected is some local measure of incident light at each point, $E(x, y)$. In a typical machine vision system, two images are taken by a camera separated by a discrete time interval Δt. What may be observed at each point is the changes in a point's intensity value $\Delta E(x, y)$ during this time interval. The optical flow is a vector field describing this intensity change by indicating the motion of features from one image to the other. An example of optical flow is shown in Figure 1.

By accurately computing this 2-D vector field, it is in principle possible to calculate three-dimensional properties of the environment, and quantities such as time-to-contact with an an observed object [Lee76]. Biological organisms make considerable use of optical flow, such as the detection of discontinuities [PRH81], and figure-ground discrimination [B81]. Motion detection is useful for autonomous mobile robot navigation [PBF89] as well as undersea navigation [NSY91]. [K86], [KvD78] discusses the decomposition of optical flow into curl, divergence and shear for the purposes of determining local shape of an object,

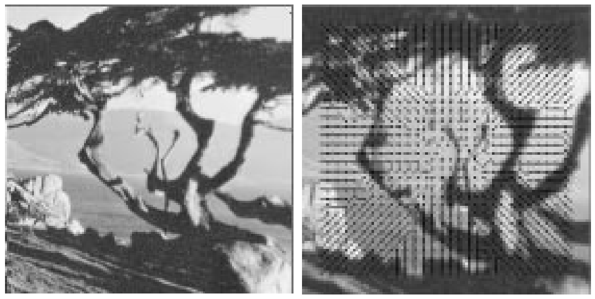


Figure 1: Optical flow "needles", overlaid onto the second image of an image sequence, indicate direction and magnitude of motion of pixels from the first (left) image to the second (right) image.

and [Reg86] presents evidence that the human visual system possesses specific detectors for similar types of basic motion. [TR93] describes an application for image coding. [MB90] covers experiments in segmentation, structure from motion, tracking and qualitative shape analysis. [NA89] discusses obstacle avoidance using only qualitative optical flow.

In general the optical flow will not be the same as the true 2-D projection of the 3-D motion field [VP87]. For example, a rotating, perfectly featureless sphere will not induce any optical flow, however the 2-D projection of its motion field is non-zero everywhere on the sphere except at the occluding boundaries. Conversely, if the sphere is stationary but a light source moves, the changes in shading will induce an optical flow field even though the motion field is zero everywhere ([Horn86]). However, for a sufficiently textured surface, the optical flow field will be arbitrarily close to the motion field.

Many techniques for optical flow exist (see [BFBB92] and [LV89] for reviews and discussions of several techniques). Although these techniques can perform very well for certain sequences of images, there are very few that are currently able to support real-time performance. Authors rarely report the computational time needed for their algorithms; when they do, it is on the order of many minutes per frame ([WM93]), or require specialized hardware such as a Connection Machine ([BLP89a], [DN93], [SU87], [WZ91], [L88]), Datacube ([LK93], [N91]), custom image processors [DW93], or PIPE [KSL85], [WWB88]. Techniques which can run in real-time often impose strict restrictions on the environment; [HB88] presents a technique that can segment in real-time for tracking purposes, but requires that a textured object be moving in front of a relative textureless background. We desire a technique that computes a more general-purpose optical flow field, in real-time on a standard desktop workstation, without imposing strict restrictions on the input.

Techniques for optical flow can be divided into *gradient-based*, *velocity-tuned filter based*, and *correspondence-based* motion, each described in turn.

2.1 Gradient-Based Optical Flow

A common technique for computing optical flow is to assume that the total spatial and temporal derivatives of the image brightness remain constant. For small motions, constant ambient illumination, these assumptions are more or less true except for pathological situations such as occluding boundaries. While detecting occluding boundaries is an important application for optical flow, for the moment we will only consider the general case.

¿From [HS81], assume that the brightness of a given point in an image is constant:

$$\frac{dE}{dt} = 0. \tag{1}$$

We wish to find the velocity $v = (u, w) = (\frac{dx}{dt}, \frac{dy}{dt})$.

After expanding equation 1 we have:

$$\frac{\partial E}{\partial x}u + \frac{\partial E}{\partial y}w + \frac{\partial E}{\partial t} = 0. \tag{2}$$

(Note that equation 2 does not contain second and higher order terms; these vanish as $\delta t \to 0$. Since δt is often a significant fraction of a second, it is questionable to ignore these terms, thus yielding second-derivative methods such as [UGVT88], which we will mention later. For now we will ignore this issue.)

The spatial derivatives $\frac{\partial E}{\partial x}$ and $\frac{\partial E}{\partial y}$ and the temporal derivative at an image point $\frac{\partial E}{\partial t}$ can be estimated using two or more images by using for example finite or central difference methods [LV89]. This leaves two unknowns u and w, motion along the X and Y axes respectively, with only one constraint, equation 2. Only the motion along the direction of the gradient $(\frac{\partial E}{\partial x}, \frac{\partial E}{\partial y})$ is available. This is known as the *aperture problem* [MU81], [NS88] and is illustrated in Figure 2. The left of Figure 2 shows an instance of the *strong aperture problem* [BLP89b]. If intensity variations are one-dimensional (i.e. no curvature of the intensity isocountours), and no endpoints or distinguishing markers are visible (as if the contour were viewed through a small aperture), then the motion of a point on a line is ambiguous; point A could have moved to point A or point A' for example. Under these conditions (when intensity variation is one-dimensional) the problem as stated is *ill-posed* without further assumptions and the correct motion cannot be determined. When dealing with spatial gradients, the aperture problem manifests itself in the fact that first derivatives are only a planar approximation to the curvature at a given point. Generally however natural image sequences do have sufficient intensity variations. The right of Figure 2 shows a translating curved contour. Here, the motion of point A may be disambiguated by comparing the motion of nearby points as well; points B and C match the correct motion of point A, but that of points D and E do not. If there are two-dimensional intensity variations, then the aperture problem is known as the *weak aperture problem*. However, this particular solution requires a non-local mechanism, and in this case an assumption of local rigidity, which is not necessarily true in general, but may be approximately true for most cases.

Given that nearby points on moving objects are likely to have similar three-dimensional velocities, their two-dimensional velocity projections should also have similar velocities. The problem can then be formulated as minimizing the sum of the errors in the equation for the rate of change of image brightness and the deviation from smoothness in the optical flow [HS81]. The optical flow field is then determined by minimizing a cost functional L ([HS81], [KWM89]):

$$L(\dot{x}, \dot{y}) = \int\int \{[E_x\dot{x} + E_y\dot{y} + E_t]^2 + \lambda[(\frac{\partial \dot{x}}{\partial x})^2 + (\frac{\partial \dot{x}}{\partial y})^2 + (\frac{\partial \dot{y}}{\partial x})^2 + (\frac{\partial \dot{y}}{\partial y})^2]\} dx\, dy$$

In this equation λ is a parameter which weighs the smoothness of the motion field relative to the error in the intensity constraint equation (2). [KWM89] proposes implementing such a smoothness constraint in a neural network and relates it to the primate's visual system.

In contrast to finding dense flow using a smoothness constraint, [Hil84] proposes calculating optical flow at points along the contours in the image. Here contours are extracted from the image using Marr and Hildreth's edge detector [MH80] and required that the velocity field be smooth only along a contour and not across it. This method does not blur the optical flow field at discontinuities,

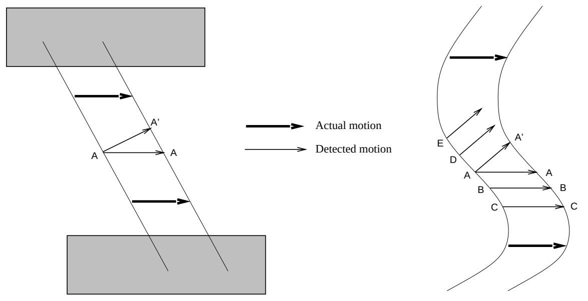


Figure 2: The *strong aperture problem* is illustrated on the left; only motion *normal* to a translating straight contour may be determined. If there are curvature or intensity variations, then we have the *weak aperture problem*, which can be satisfactorily solved if certain assumptions are enforced.

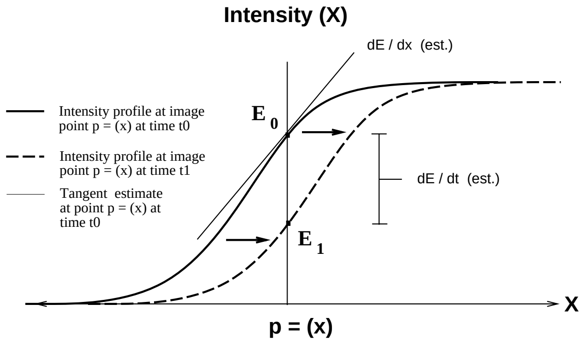
but only gives sparse measurements (only along the contours) and may confuse object boundary contours with other kinds of edges, such as those due to changes in shape or reflectance.

[SAH91] represents the problem probabilistically to account for the confounding effects of image noise, low contrast regions, multiple motions, lighting changes. A two-dimensional probability distribution is used that reflects directional uncertainty in the motion estimates. Areas of high contrast are treated as more reliable than areas of low contrast, subject to an asymptotic maximum, rather than effectively normalizing contrast when combining velocity information of a given point with that of nearby points.

A description of a general gradient-based method for a one-dimensional image is seen in Figure 3. Here a one-dimensional image intensity profile is shown in the vicinity of a given point p at coordinate (x). At time t_0 the image intensity at point p is E_0, and at time t_1 at point p it is E_1. An estimate $\frac{\delta E}{\delta t}$ of the temporal derivative $\frac{\partial E}{\partial t}$ can be provided using a forward-difference method: $\frac{\delta E}{\delta t} = \frac{E_1 - E_0}{t_1 - t_0}$. The spatial derivative $\frac{\partial E}{\partial x}$ can be similarly estimated using the pixels adjacent to pixel p.

[DN93] calculates gradient-based optical flow using the multipoint technique for solving partial differential equations. Here the least-squares solution for equation 2 is solved in a local NxN neighborhood, under the assumption that local velocity is locally constant. This can run at several frames per second on a Connection-Machine 2 (depending on the size of the neighborhood N); see also [DNS92].

[BJ94] minimizes an objective function consisting of three terms: one, the standard optical



Intensity (X)
dE / dx (est.)
Intensity profile at image point p = (x) at time t0
Intensity profile at image point p = (x) at time t1
Tangent estimate at point p = (x) at time t0
E 0
dE / dt (est.)
E 1
X
p = (x)

Figure 3: Gradient-based Optical Flow

flow intensity constraint equation; two, the differences with the optical flow in neighboring pixels; and three, the difference with a parametric planar patch estimate of the optical flow in regions of approximately constant intensity. Robust error norms are used to reject outliers in each error term.

[WWB88] calculates the motion of edges by differentiating their Gaussian-convolved spatio-temporal activation profiles with the PIPE pipelined image processor at 15 frames per second. This method is very fast and can be robust, but only gives normal flow at edges, not the true flow.

One solution to the aperture problem of equation 2 is to assume that the total temporal derivative of the spatial gradient is zero:

$$\frac{d\nabla E}{dt} = 0$$

Expanding this equation leads to two constraints:

$$\frac{\partial^2 E}{\partial x^2}v_1 + \frac{\partial^2 E}{\partial x \partial y}v_2 + \frac{\partial^2 E}{\partial x \partial_t} = \frac{\partial^2 E}{\partial y \partial x}v_1 + \frac{\partial^2 E}{\partial y^2}v_2 + \frac{\partial^2 E}{\partial y \partial_t} = 0$$

This approach is taken in [Nag87], [UGVT88], and assumes that the displacements being measured are less than one half of a cycle of the highest spatial frequencies in the image, otherwise there will be aliasing. Although this approach can in theory give very precise measurements, one major problem is that numerical differentiation of this sort is very sensitive to noise in the input (for example cf. [KMB+91]). This is particularly true if the spatial derivative is small (i.e. the slope is flat) at the point in question, in which case a constant amount of noise has a greater detrimental effect on the numerical differentiation. Methods based on second or higher derivatives are potentially even more susceptible to noise as the problems associated with numerical differentiation are even worse. [To92a] calculates velocity at areas of sufficient contrast in the image using the technique of [UGVT88] at about 2-3 frames per second on a Sun 4/330. For his tracking application, dense measurements are not needed, nor is it necessary for highly accurate measurements. Any less than a full 8 bits of information per pixel may make second-derivative techniques unsuitable however [To92b]. In practice for robustness, [UGVT88] implements two variants of regularization techniques to create a smooth optical flow field. Smoothing the image can help attenuate the effects of noise, however smoothing the basic optical flow measurements by imposing a smoothness constraint or regularization such as in [HS81] can smooth over discontinuities in the motion field and introduce inaccuracies, as well as increase the computational cost. One option for improving performance for these methods is to only calculate optical flow where the spatial derivatives are large (i.e. the gradient is steep), such as is discussed in [BFB94], however this can result in a sparse motion field which may be inadequate for some problems. It is desirable to find a method of calculating optical flow that is less sensitive to noise in the imaging process, gives a dense output, and is computationally efficient.

2.2 Velocity-Tuned Filter Optical Flow

One class of optical flow techniques represents motion in terms of spatial and temporal frequencies. From [Heg87], two-dimensional patterns translating in the image plane occupy a plane in the spatio-temporal frequency domain:

$$\omega_t = u\omega_x + v\omega_y$$

where ω_t, ω_x, ω_y are the temporal and spatial frequencies of the motion respectively. This approach assumes that several points in spatio-temporal space can be detected in the moving surface. If, however, the aperture problem exists, then the points in spatiotemporal space occupy a line, and the plane is not fully determined. To solve this problem, [WA85] combines ten oriented sensors into opponent-motion pairs, discarding the smaller response of the pair. In theory, the output of any two remaining sensors (within 90 degrees of the actual motion) is sufficient to determine the actual motion, since that would provide two linearly independent components of the velocity vector. [SS85] describes *elaborated Reichardt detectors*, in which the point receptive fields of the original Reichardt detectors [R57] are replaced with spatial filters, and stresses the need for effective voting rules in differentiating between different sensors' outputs. [AB85] describes the construction of phase-independent separable spatiotemporally oriented opponent-motion filters, and suggests using the ratios between multiple sensors for contrast-invariance; although higher contrast will increase the *absolute* outputs of the motion sensors, the *ratios* of their outputs should remain constant. [AB85] also shows the relationship between energy models and Reichardt detectors.

[WM93] first convolves the image with a set of linear, separable spatiotemporal filters and applies the brightness constancy equation (equation 1) to each. The resulting over-determined set of equations is then solved using a *total least squares* or *orthogonal regression* technique.

[Fl92] describes a phase-based method which describes component velocity in terms of the motion of level phase contours in the output of band-pass velocity-tuned filters. This technique is essentially a differential method applied to phase rather than intensity [BFBB92]. The full 2-dimensional velocity field is determined by fitting a linear velocity field to the component velocities. Although these methods can produce accurate results ([BFB94], [WM93]), methods based on velocity-tuned or spatiotemporal filters can be extremely computationally intensive, requiring up to several minutes or even hours on a scientific workstation. Thus without some powerful special-purpose hardware, they are generally not suitable for real-time robotic vision.

2.3 Correlation-based Optical Flow

In general it is not possible to determine the correct optical flow field given a pair of successive image frames, due to the aperture problem. If certain assumptions are enforced, however, then the problem becomes well-posed, and can be satisfactorily solved in most cases. A relatively noise-resistant method of calculating optical flow is described in [BLP89a], [BLP89b] and summarized here. We will assume that the maximum possible displacement for any pixel is limited to η in any direction. The actual value of η depends on the expected velocities of the pixels in the image plane. This is shown in Figure 4.

Since we are generally concerned with the flow of rigid-bodied objects, it is usually the case that any given pixel has the same velocity as those of

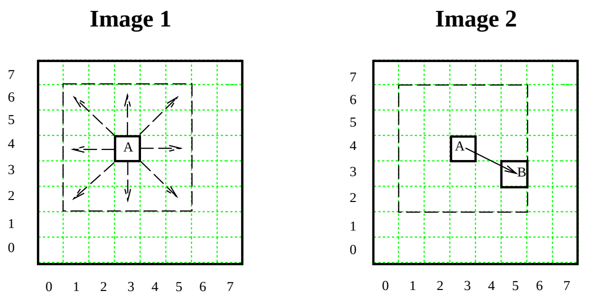


Figure 4: In image 1, the search space for Pixel A for $\eta = 2$ is shown. The correct motion from image 1 to image 2 can be displayed by the vector AB.

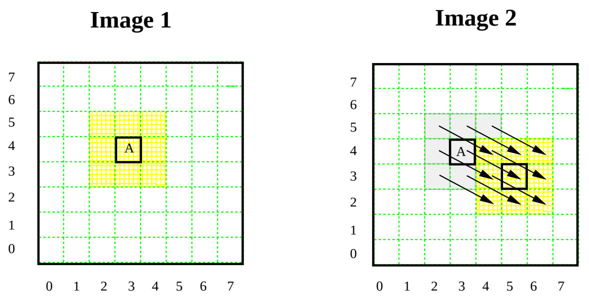


Figure 5: Optical flow for pixels adjacent to a given pixel is assumed constant due to the rigid-body assumption. In this figure, $\nu = 3$.

its neighbors. We assume that these pixels are in a square neighborhood, or window, of size ν centered around the given pixel. That is, it is assumed that the motion vectors for pixels adjacent to a given pixel will be similar, as shown in Figure 5. All examples in this paper use a value $\nu = 7$.

The motion for the pixel at [x,y] is defined to be the determined motion of the patch of $\nu * \nu$ pixels centered at [x,y], out of $(2\eta + 1) * (2\eta + 1)$ possible displacements. We determine the correct motion of the patch of pixels by simulating the motion of the patch for each possible displacement of [x,y] and considering a match strength for each displacement. If ϕ represents a matching function which returns a value proportional to the match of two given features, then the match strength M(x,y;u,w) for a point [x,y] and displacement (u,w) is calculated by taking the sum of the match values between each pixel in the displaced patch P_ν in the first image and the corresponding pixel in the actual patch in the second image:

$$\forall u, w : M(x, y; u, w) = \sum \phi(E_1(i, j) - E_2(i + u, j + w)), (i, j)\epsilon P_\nu$$

A possible function for ϕ is the absolute difference between the two pixels' intensity values; another is the squared difference of their respective intensity values. In these cases a lower value indicates a better match. Moravec [Mor77] uses a variant of normalized cross correlation in stereo matching. Although this has the advantage of insensitivity to contrast, the sum of absolute differences measure used in most examples in this paper was found to be very insensitive to the image contrast. In addition, experiments using normalized cross-correlation did not produce good results.

In an ideal situation, where we have only fronto-parallel motion, no noise, constant ambient illumination, and a translation of an integer number of pixels (within our given range) we would expect the pixels of the two patches to match perfectly. The actual motion of the pixel is taken to be that of the particular displacement, out of $(2\eta+1)*(2\eta+1)$ possible displacements, with the maximum neighborhood match strength (equivalently minimum patch difference); thus this is called a "winner-take-all" algorithm. This is repeated for each pixel in the image, independently of the other pixels, with the exception that motion is not computed along a $\eta + \lfloor \nu/2 \rfloor$ border at the edges of the image since the matching operator would have to access points that are not available in the image itself. With a large neighborhood, ties are relatively rare and are decided arbitrarily.

This algorithm has many desirable properties. Due to the two-dimensional scope of the matching window, this algorithm generally does not suffer from the aperture problem except in extreme cases [BLP89b], and tends to be very resistant to random noise. In fact the algorithm's "winner-take-all" nature does not even require that the calculated match strengths have any relation whatsoever to what their values should theoretically be, it is only necessary that their relative ordering remains correct. For example, a change in illumination between frames would certainly affect the individual match strengths, but need not change the best matching pixel shift. Conversely, any noise in a gradient-based method usually directly results in errors in the basic optical flow measurements. In the case of a change in illumination, the image intensity constraint equation does not apply since total image intensity does not remain constant. In addition, since the patch of a given pixel largely overlaps with that of an adjacent pixel, match strengths for all displacements for adjacent pixels tend to be similar, and so the resultant optical flow field tends to be relatively smooth, without requiring an additional smoothing step. Finally, since one optical flow vector is produced for each pixel of input (excepting the small $\eta + \lfloor \nu/2 \rfloor$ border), optical flow measurement density is 100 percent. For these reasons, the basic correlation-based algorithm tends to be robust in practice, thus satisfying the first criteria for practical robotic vision.

[A87a] and [G87] prefilter the images with band-pass filters and employ a coarse-to-fine matching strategy in which initial coarse estimates at one spatial scale are fed into the next higher resolution level. Although sub-pixel motions can be calculated by finding the minima of a sum-of-squared-difference with a quadratic approximation to the intensity surface, this usually requires an iterative procedure which increases computation time. Furthermore, although matching methods such as these can do well with translating images, they are less effective when there are multiple velocities present [BFB94]; in any event hierarchal techniques are not inconsistent with the methods described in this paper.

[AP93] computes optical flow only along a single dimension; often, the vector sum of two one-dimensional optical flow vectors is sufficiently *qualitatively* similar to the two-dimensional optical flow to be used in certain robotic vision tasks. Using Green's theorems, the divergence of the optical flow field can be estimated by integrating the optical flow normal to a closed contour. By using the estimated divergence of the flow field, time-to-contact is calculated with errors reported on the order of 10% [AP93]. The simplicity of this approach makes it a plausible candidate for biological implementations.

[Hub94a] uses the correlation of 2-dimensional image patches along a single dimension ([AP93], [BLP89a]) to simulate the navigation of a fly in a synthetic maze using optical flow, at a rate of about one frame per second using 512x512 images [Hub94b]. A genetic algorithm is used to train the artificial fly's optomotor response to turbulent air conditions to prevent collisions with the maze's walls.

The independence of one pixel's chosen displacement from all other pixels' displacements motivates massive parallel implementations such as the one implemented on the connection machine [BLP89a]. But even these massively parallel implementations are only "close-to-real-time", on the order of seconds per frame. Here we define "real-time" in the loose sense of being fast enough to be

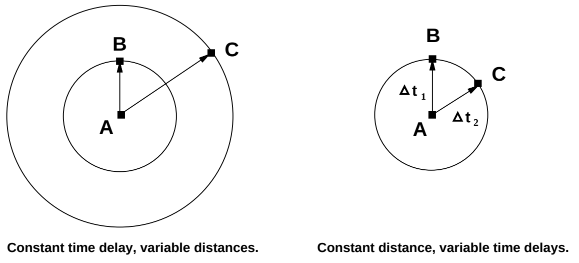

B
C
A
B
C
Δt_1
A
Δt_2
Constant time delay, variable distances.
Constant distance, variable time delays.

Figure 6: As the maximum pixel shift increases linearly, the search area increases quadratically. However with a constant shift distance and variable discrete time delays, search over time is linear.

calculated on-line and be usable in some reactive system; on the order of 4-5 frames per second at a minimum.

It is possible to perform the bulk of the computations in customized silicon chips; one such (partial) implementation has been done in CMOS [C91]. [DW93] calculates structure-from-motion using the same basic shift-match-winner-take-all algorithm implemented on the Image Understanding Architecture simulator; they report an estimated 0.54 seconds per frame using a maximum possible displacement $\eta = 20$. [LK93] calculates optical flow (for the purposes of tracking) within a limited radius (+/- 2 pixels vertically, +/- 3 pixels horizontally) using a 7x7 correlation window at 10 Hz on 128x120 images using the Datacube MaxVideo 200.

Certainly, without the luxury of custom image processors, other techniques must be used for practical operation with conventional serial computers due to the search-based nature of the optical flow algorithm itself, which is described next.

3 Linear-Time Optical Flow

Since correlation-based optical flow algorithms (e.g. [BLP89a]) satisfy the first criteria for practical robotic vision, that of robustness, we will use it as a starting point. The second criteria for practical robotic vision is computational efficiency. This section will analyze the computational complexity of the traditional correlation-based optical flow algorithm and develop a space-time tradeoff to create a very fast, linear-time optical flow algorithm.

3.1 Time-Space Dimensional Reduction

One limitation with the traditional correlation-based algorithm described in Section 2.3 is that its time complexity grows quadratically with the maximum possible displacement allowed for the pixel; see the left of Figure 6. Intuitively, as the speed of the object being tracked doubles, the time taken to search for its motion quadruples, because the area over which we have to search is equal to a circle centered at the pixel with a radius equal to the maximum speed we wish to detect.

However, note the simple relationship between velocity, distance and time:

$$velocity = \frac{\Delta distance}{\Delta time}.$$

Normally, in order to search for variable velocities, we keep the inter-frame delay δt constant and search over variable distances (pixel shifts):

$$\Delta v = \frac{\Delta d}{\delta t}.$$

However, we can easily see from Figure 6 that doing so results in an algorithm that is quadratic in the range of velocities present. Alternatively, we can keep the shift distance δd constant and search over variable time delays:

$$\Delta v = \frac{\delta d}{\Delta t}. \tag{3}$$

In this case, we generally prefer to keep δd as small as possible in order to avoid the quadratic increase in search area. Thus, in all examples δd is fixed to be a single pixel. (Note however, there is nothing preventing an algorithm based on both variable Δd and Δt's). Since the frame rate is generally constant, we implement "variable time delays" by integral multiples of a single frame delay. Thus, we search for a fixed pixel shift distance $\delta d = 1$ pixel over variable integral frame delays of $\Delta t \in \{1, 2, 3, ...S\}$. S is the maximum time delay allowed and results in the slowest motion calculated, $1/S$ pixel/frames. For example, a $1/k$ pixel/frames motion is checked by searching for a 1-pixel motion between the current frame t and frame $t-k$. Thus our pixel-shift search space is fixed in the 2-D space of the current image, but has been extended linearly in time. As before, the chosen motion for a given pixel is that motion which yields the best match value of all possible shifts.

For example consider Figure 7 and Figure 8. Here we are trying to calculate the optical flow for pixel (1,1) at the current frame, image T. In Figure 7 the optimal optical flow for pixel (1,1) from image $T-1$ to image T is calculated to be a pixel shift of (1,-1). This is only a temporally local measurement however; it may not be the final chosen motion. In Figure 8 the same search is performed, except using image $T-2$ as the first image. In this case the calculated motion happens to be a pixel shift of (0,1) pixels over 2 frames, equivalently (0,1/2) pixel/frames motion. If the maximum time delay $S = 2$, then the procedure would stop here, otherwise we would continue processing frames until finally the optical flow from image $T-S$ to image T was calculated as well. The best of all these motions is taken to be the actual motion.

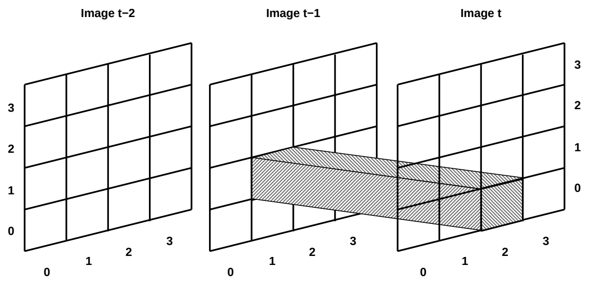


Figure 7: Visualization of motion from image $T-1$: (1,1) to image T: (2,0). This would be an optical flow of (1,-1) pixel/frames motion for pixel image T: (1,1).

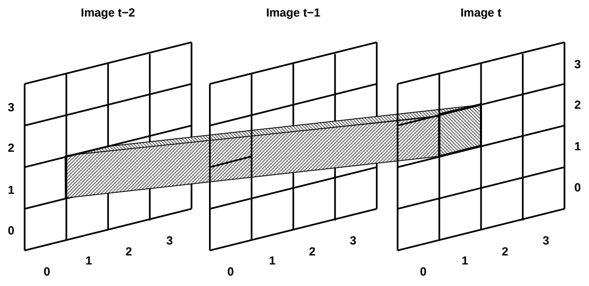


Figure 8: Visualization of motion from image $T-2$: (1,1) to image T: (1,2). This would be an optical flow of (0,1/2) pixel/frames motion for pixel image T: (1,1).

For a given velocity of $1/\delta t$ pixel/frames, we assume that motion is constant for t frames in order to register a cumulative motion of one pixel. Failure of this assumption can result in temporal aliasing, discussed in more detail later. [P90] calculates the normal velocity of edges using temporal-delay sensors which are summed in a later stage to determine actual motion. The images are first smoothed with a Gaussian to reduce the chances of aliasing due to multiple edges appearing at a single sensor.

This time-space tradeoff reduces a quadratic search in space into a linear one in time, resulting in a very fast algorithm for computing optical flow. This partially satisfies the second criteria for practical robotic vision, that of real-time performance. Performance issues, particularly hardware considerations, are discussed in detail in [C94b].

One factor in producing real-time performance is that the original images are generally subsampled from 256x256 pixels to 64x64 pixels in size. Ideally, some optimal filtering procedure would be used to produce these reduced-resolution images, however this is generally not possible in a real-world situation. This is because the original images must be subsampled on the framebuffer itself in order to reduce the framebuffer to host computer memory bandwidth requirements. The method of subsampling is not known, but due to limited hardware on the framebuffer itself it is likely to be a simple technique such as a box filter which performs an unweighted average of the pixels within the filter area, i.e., most likely a 256x256 to 64x64 reduction is performed by an unweighted averaging of each subsequent 16x16 block of pixels. Since this system is over five years old, normal advances in framebuffer technology would be expected to greatly increase this rate. For the moment however, we will simply have to deal with this loss in resolution. For this reason as well, all images in this paper that are originally greater than 64x64 in resolution are subsampled to 64x64 using a box filter, since it is likely the technique used by simple framebuffers (to use a more sophisticated technique would give an unrealistic advantage to synthetic and high-resolution images).

When viewing such a subsampled image sequence, it is clear that considerable aliasing is occurring (see [FvD+90] for details on the problem of aliasing). In fact the aliasing is so bad one wonders how the algorithm can work at all. Figure 9 provides an example of this problem. On the left, a dark pixel moves between two CCD sensors. In particular, this dark pixel is split between both sensors during frame 1. This split cannot be represented in an actual image of the given resolution; however, the CCD sensor must provide a unique intensity for each pixel location. Thus in the actual image the pixel's intensity is "split" between both pixel locations. An image sequence $\{0, 1, 2\}$ would result in a flickering effect as the dark pixel passed between the two positions. For an algorithm measuring spatial derivatives, this could result in inaccurate measurements. The linear-time correlation algorithm, however, does not measure these types of spatial derivatives. Instead, it performs pixel matching across space and time. In this example, a perfect pixel match can be made between frame 0 and frame 2 (a motion of 1/2 pixel/frames), completely avoiding the difficulties associated with frame 1.

This algorithm has been successively used on many real and synthetic image sequences for a variety of real-time robotic vision tasks ([CB91], [C93], [C94a]). [Du94a] reports being able to use this optical flow algorithm to instantiate some fly-like control laws ([War88]) in a small mobile robot. With the camera mounted on top of the robot, the image information from the camera is fed through the optical flow routine and the motion information is then used to directly control the robot. With a frame rate of 4-5 frames per second and the robot moving at 4-5 cm per second, the robot is able to successfully maneuver through an unmodified office environment. Recently, [Du94b] has also been able to use the algorithm to play the game of tag with a cardboard target, and even with a slow-moving person, using only optical flow for pursuit, docking, and escape.

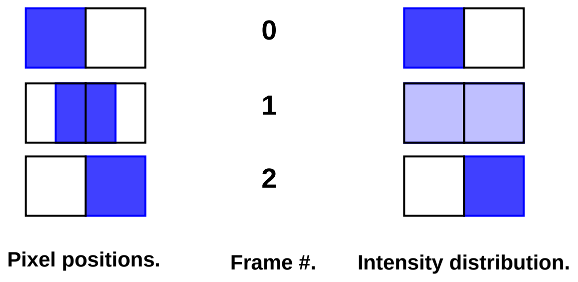


Figure 9: A dark square moves across a light background. As the square crosses a pixel boundary, its intensity is "split" between the two adjacent pixels in the image.

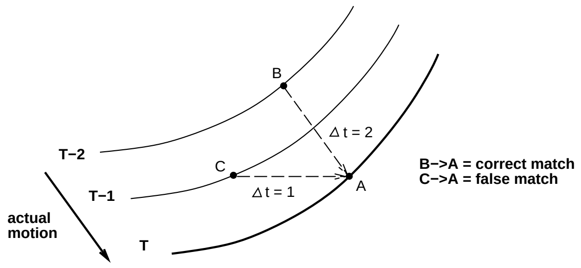


Figure 10: The Temporal Aperture Problem. The correct motion is $B \rightarrow A$, however if our window in time is too narrow, an incorrect motion $C \rightarrow A$ may be detected, because the contours are similar at points C and A.

3.2 Temporal Aperture Problem

The standard aperture problem has already been discussed in Section 2.1. In the temporal domain, it takes on a slightly different form. In Figure 10 there is a contour translating to the lower-right at a certain speed, such that it requires $\delta t = 2$ time units to translate one pixel. We expect a match between points A at time T and point B at time $T-2$ to be very close. However, we get another "close" match between points A at time T and point C at time $T-1$. Globally, this motion is not correct, however motion is calculated independently at each time delay, and locally this false motion has a better match than no motion for that time delay.

If the contour shown extended infinitely with no end markers or distinguishing illumination changes, then we would have an instance of the *strong aperture problem* [BLP89b], which in principle cannot be solved. If there is at least some variation of the isocontour and/or textural differences, then we have an instance of the *weak aperture problem*, which can be solved but requires a non-local mechanism. In our case of an area-based correlation algorithm, our non-local mechanism is the matching window, which has a finite (but usually sufficient) spatial extent.

In the case of the standard aperture problem, the motion of a contour may be ambiguous when viewed through a small aperture (or equivalently when our matching window is too small). In the case discussed above, the incorrect motion of the contour may be detected when our search "window" through time is too narrow. Thus we call this the *temporal aperture problem*. By sufficiently extending our search through time and picking the best match across time as well as space, we can in general avoid this problem. However, there is one special circumstance where this approach fails due to temporal aliasing, discussed next.

3.3 Temporal Aliasing

Previously, we have seen that we may solve the aperture problem for most cases by taking the best match value for a translating pixel in both time as well as space. Unfortunately, this does not always give the correct result. Examine Figure 11. Pixel A translates to pixel B and then to pixel C, as if some object was executing a tight turn. In this diagram, it might be entirely possible that the A $\Rightarrow$ C motion may yield a better match than B $\Rightarrow$ C. If this is true, then based on a best-match policy, we would erroneously choose A $\Rightarrow$ C as the correct motion.

One "obvious" solution is to assume that the faster motions are correct.

It works for this example, and one can argue that the potential for aliasing is only limited by the sampling rate. Therefore, it might seem that the motion B $\Rightarrow$ C was correct, based on temporal aliasing arguments. This solution was presented in [CB91], and was a reasonable solution at the

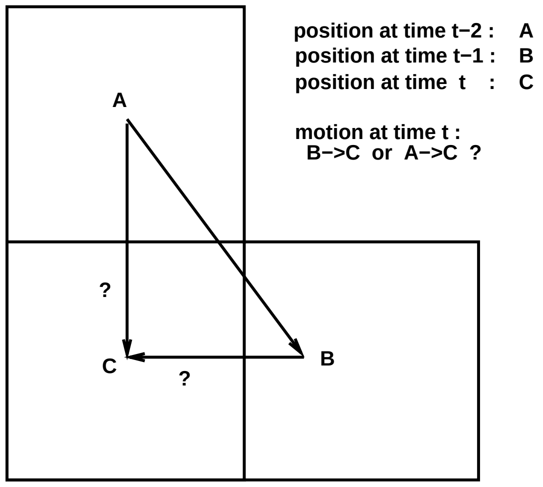


Figure 11: Temporal Aliasing. The correct motion is $B \rightarrow C$, however $A \rightarrow C$ *may* give a better match. Our solution to the Temporal Aperture Problem would fail for such a case.

time, because available computational power limited practical motion detection to a small number of temporal delays. Now however, delays of as much as $\delta t = 20$ frames are practical and have been successfully tested. In these cases, the "fastest first" rule [CB91] too often selects an incorrect faster motion over a slower, better matching motion. The Temporal Aperture Problem is an example of this phenomenon.

Our solution to this problem is the following: in cases where the best velocity match is not in the same direction as the fastest motion for that time frame, then we have an inconsistency. Either 1) the slower motion is due to temporal aliasing or 2) the faster motion is due to the temporal aperture problem. To resolve the inconsistency, we perform a second search over the multiple independent measurements, starting from the best matching velocity and continuing until the fastest velocity. Among these, we select as the correct motion the velocity with the best matching value subject to either a temporal consistency constraint, or the motion satisfies the intent of the "fastest-first" rule. In the former case, we assume that a motion has temporal support if it the motion at time t is "similar" to that at time $t+1$. To do this, we perform a "look-ahead" of one frame, and we define "similar" to be motion in the same direction (i.e. we allow for slight speed variations). In the latter case, if the motion is in the same direction as the fastest motion in the image, then we accept it.

Although not guaranteed to be correct, this heuristic has been thoroughly tested on many nat-

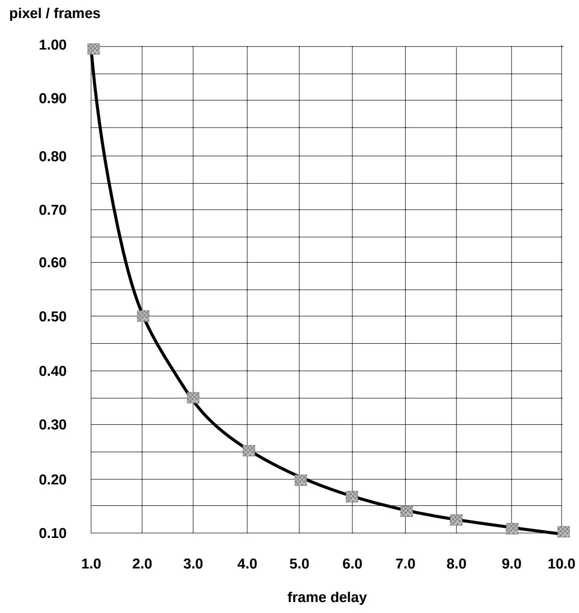


Figure 12: Only 1/(frame delay) velocity measurements can be detected in the current implementation.

ural and synthetic image sequences and seems to perform quite well. Slow motion is accommodated by the temporal consistency constraint, and faster motions are supported by the modified "fastest-first" rule. Note that this additional search is limited to the number of velocities searched over, i.e. it is still a strictly linear algorithm. In addition, we limit our look-ahead to only one frame. Although techniques that use considerable temporal support can perform quite well [BFBB92], in real-time robotics it is not acceptable to impose a latency of a large number of images before a result is produced. The lookahead of a single image used by this algorithm is a relatively modest requirement.

4 Harmonic Search Intervals

We have seen that by performing searches in time instead of space we can, in theory, convert a quadratic time algorithm into a linear one. In this section we will more closely examine this point.

One disadvantage of the traditional algorithm is that it computes image velocities that are integral multiples of pixel shifts [BLP89a,b] : {1,2,3,...,η} pixels per frame. Although the algorithm discussed in this paper does calculate sub-pixel motions, it still computes velocities that are basically a ratio of integers (generally with the numerator equal to one pixel), not a truly real-valued measurement. Given $\delta d = 1$ and discrete frame delays $\Delta t = \{1, 2, 3, ..., S\}$ equation 3 yields

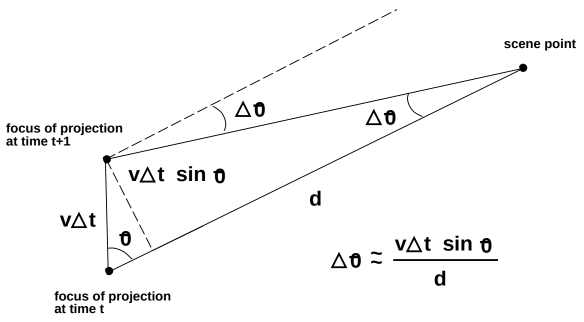

scene point
$\Delta\theta$
$\Delta\theta$
focus of projection at time t+1
$v\Delta t \sin\theta$
d
$v\Delta t$
θ
$\Delta\theta \approx \frac{v\Delta t \sin\theta}{d}$
focus of projection at time t

Figure 13: [From Nalwa 93] Change in visual angle is inversely related to depth.

$\{1/1, 1/2, 1/3, ..., 1/S\}$ pixels per frame equivalent motion, Figure 12. An immediate consequence of this is that sub-pixel motions are now detectable, correcting a deficiency of the original algorithm. Although the linear set of velocities may seem more intuitive than the harmonic series, in fact the latter is often much more suited to the types of motion found in real vision problems.

One example is shown in Figure 13 [Nal93]. If the velocities in the image are small, we can approximate

$$\Delta\theta \approx \frac{v\Delta t \sin\theta}{d} \tag{4}$$

where θ is the visual angle (although visual angle is a measure used with spherical perspective projection, rather than planar perspective projection which is more appropriate for a machine vision system, they are approximately the same near the line of sight where $\tan\theta \approx \theta$. Although the former has the advantage of being independent of any coordinate system, the latter more accurately characterizes a machine vision system (see also [T91], Appendix). The motion of a point in an image for a moving camera with line of sight orthogonal to the direction of motion is inversely proportional to a point's distance from the focus of projection, and can be used to determine depth; for example see Figure 14. This algorithm, which computes velocities inversely proportional to the discrete frame delays $\Delta t = \{1, 2, 3, ...S\}$ is therefore very well suited to computing depth via motion parallax. It would be much more awkward to attempt to calculate motion parallax using a linear set of motions $\{1, 2, 3, ..., \eta\}$ pixels per frame. Although there is a sine factor in equation 4, note that sine "degrades gracefully" with small deviations from 90 degrees, e.g., a 10 degree deviation from orthogonal motion to the line of sight is a factor of 0.985, and a 20 degree deviation (quite large for even a moderately

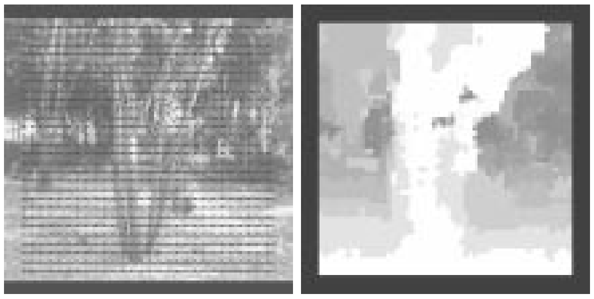


Figure 14: Depth from motion parallax. Camera is translating left, with the resulting optical flow and grey-level shaded relative depth map. Lighter areas indicate objects closer to the camera.

calibrated mobile robot) still yields 0.94. Even a very large 30 degree deviation from the orthogonal yields only a .866 factor. To complete the argument, even if the deviation was a ridiculous 45 degrees (i.e. the robot was heading as much parallel to the line of sight as orthogonal to it!) the factor is still only 0.707. Thus, we do not need anywhere near perfect orthogonal motion to make practical use of depth from motion parallax.

It should be noted that it is not necessary to limit the measured velocities to a strict harmonic sequence. Given that the motion to be detected lies within our slowest and fastest measurements (1/1 and $1/S$ pixels per frame), and in the absence of bad spatial aliasing, we need only make measurements as precisely as we need. For example, we could detect a range of velocities $v = \{1/1, 1/2, 1/4, 1/8, .., 1/(2^{\lfloor log_2 S \rfloor})\}$ pixels per frame. In effect, this means that we have the option of detecting up to a quadratic range of velocities, again at only a linear-time cost. Of course, the resolution within these ranges is reduced by the same factor, but this may be a useful trade-off for a particular application. Conversely, it is generally not possible for the traditional algorithm to "skip" pixels, since the actual motion may be missed.

5 Summary

The space-time tradeoff discussed in Section 3.1 allowed the development of an optical flow algorithm that was linear in the range of velocities detected, rather than quadratic as in the traditional algorithm. Unfortunately, dealing with time in this new manner introduces the problems of temporal aliasing and the temporal aperture problem, which were addressed by adding heuristic temporal antialiasing to the linear-time optical flow algorithm. Adding time to the search space also had some unexpected beneficial side effects, as noted in Section 4.

One disadvantage of the patch-matching approach is that the basic motion measurements are integer multiples of pixel-shifts. Although the linear-time algorithm discussed in this paper does calculate sub-pixel motions, it still computes velocities that are basically a ratio of integers (generally with the numerator equal to one pixel), not a truly real-valued measurement. Although calculating real-valued optical flow measurements may be possible by using interpolation, this remains future work. In addition, the angles computed in the current implementation are only the eight nearest-neighbor pixels for eight possible angles of motion (plus the possibility of no motion). Increasing angular accuracy may also be possible by interpolating pixels, however this too remains as future work. Although it is not always necessary to have accurate optical flow to perform such functions as obstacle avoidance ([NA89]), [C94b] demonstrates that despite these deficiences remarkable accuracy may still be achieved in the context of calculating time-to-collision.

6 Conclusions and Future Work

The current implementation of the algorithm still suffers from many limitations. One example is the limited angular accuracy due to the standard rectangular tessellation of images; this can be clearly seen in Figure 1. This can be overcome in principle by shifting the input image by a half-pixel in either the X or the Y direction, and performing the search over these new pixels as well. For applications that demand increased angular accuracy, it may be worth the additional computational cost. This approximation can be performed by either interpolation, or exactly in cases when subsampling is done by shifting the input array appropriately. A similar form of interpolation via pixel-shifting could be used to increase the accuracy of the magnitudes of the motions detected.

The capturing of off-object pixels is not an issue in many types of qualitative computer vision where precise boundary detection is not needed, nor is it an issue in the case of time-to-contact with a single surface. In cases were more precise boundary detection is desired however, one option is to zoom in on boundaries using a pyramid technique such as in [BBH+89]. Figure 15 shows the optical flow algorithm run on the same SRI tree sequence as in Figure 14, except that the image is zoomed by a factor of 2, and shows the outline of the branches much more clearly. One issue with

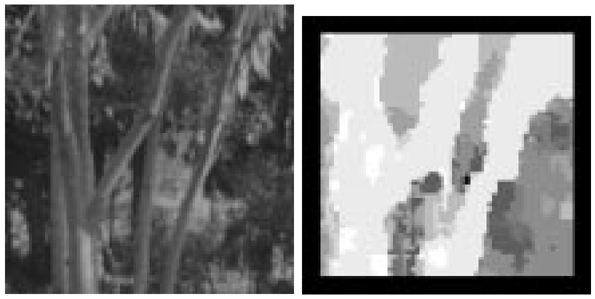


Figure 15: Depth from motion parallax, using image zoomed by a factor of 2. Camera is translating left, with the resulting optical flow and grey-level shaded relative depth map. Lighter areas indicate objects closer to the camera.

this type of processing is that the image motion for such a zoomed image is double that of the normal sequence; a method of dynamically adjusting the range of velocities detected would be useful in this case.

[BFBB93], [NG92], [VG92], [SAH91] make use of confidence measures in calculating optical flow. Conversely, the "winner-take-all" nature of this algorithm makes no use of confidence measures whatsoever, instead relying on the algorithm's inherent smoothness in producing a 100% dense output. If dense measurements are required, this fact may be viewed as a strength; if only sparse measurements are acceptable, it may be viewed as a weakness. In fact, there would be little computational savings in calculating optical flow only at areas of high contrast such as edges [Hil84] or corners ([Nag87]) since the pipelined nature of the box-filter generally assumes that optical flow is calculated at all points in the image to be efficient [C94b]. Although it may be possible to use the match strengths as a measure of confidence, it seems more beneficial to use the match values for interpolation of velocities. In this way all velocities become more accurate, rather than throwing out "bad" matches.

Although the field of computer graphics is mostly concerned with the opposite of computer vision, that is the process of converting 3-dimensional scenes into two-dimensional images, concepts from one field are very often applicable to the other. For example, spatial resampling techniques can be used in the construction of pyramids [F86]. [Burt81] presents a Hierarchal Discrete Correlation technique which can very efficiently approximate a Gaussian kernel. It remains to be seen how useful antialiasing techniques such as in [SS93] are applicable to the problems described in this paper. In addition, there would be a tremendous potential for using commercial graphics hardware to assist computer vision algorithms should a common ground be found. Special-purpose MPEG encoders/decoders, or special-purpose instructions found on general-purpose CPU's) are the first examples of this.

Ultimately, it is likely that accurate and robust structure-from-motion will make use of both depth from stereo as well as depth from motion, as in [TGS91]. The process of correspondence in optical flow can be similar to that of binocular stereo, except that in the former case the matching is across time, and in the latter the matching is across space [Rock75]. Both applications can make use of various methods for correspondence (issues of correspondence and image matching are discussed in [HS93]), however tracking across space is not equivalent to tracking across time. In particular, the former can make use of the *epipolar constraint* in matching across space. The latter's matching across space and time can make use of the *velocity equation* in matching across both space and time. Of course, stereo can make use of spatiotemporal coherence as well.

The major barrier to applying this algorithm as a generic optical flow "black box" for robotic vision is that it must be carefully adapted to the specific task in order to produce quantitatively accurate results, as in the case of time-to-contact. It would be pointless to attempt to measure the performance of this particular algorithm *per se* using the methods used in [BFBB92] and [WM93]; since the individual optical flow vectors are quantized in magnitude and direction, the average error for a single pixel would be very poor. But note that optical flow is only valuable when used in the context of a specific task, such as obstacle avoidance [MBLB91] or time-to-contact, and the performance of this algorithm when applied to the latter has been shown to be exceptionally good. However, it remains to be seen if such performance can be equaled for other robotic vision tasks. Given the algorithm's inherent robustness, computational efficiency, and demonstrated *potential* accuracy, there is a good chance for success. Ideally, a more general procedure for adapting this real-time optical flow algorithm to specific robotic vision tasks will someday emerge.

7 References

[A87a] P. Anandan, *Measuring Visual Motion from Image Sequences*, PhD Thesis, COINS TR 87-21, University of Massachusetts at Amherst, 1987

[AB85] E. Adelson, J. Bergen, Spatiotemporal Energy Models for the Perception of Motion, *Journal of the Optical Society of America*, 2(2):284-299, 1985

[AP93] N.Ancona, T.Poggio, Optical Flow from 1D Correlation: Application to a Time-To-Crash Detector, *Fourth International Conference on Computer Vision* p.209-214,1993

[B81] H. Bülthoff, Figure-Ground Discrimination in the Visual System of Drosophila melanogaster, *Biological Cybernetics 41* p.139-145, 1981

[BBH+89] P. Burt, J. Bergen, R. Hingorani, R. Kolezynski, W. Lee, A. Leung, J. Lubin, H. Shvaytser, Object Tracking With a Moving Camera, *IEEE 1989 Workshop on Visual Motion*, p.2-12, March 1989

[BFBB92] J. Barron, D. Fleet, S.S. Beauchemin, T. Burkitt, Performance of optical Flow Techniques, *Proceedings of the IEEE CVPR*, p.236-242, 1992

[BFB94] J. Barron, D. Fleet, S.S. Beauchemin, Performance of Optical Flow Techniques, *International Journal of Computer Vision*, 12(1):43-77, 1994

[BJ94] M. Black, A. Jepson, Estimating Multiple Independent Motions in Segmented Images using Parametric Models with Local Deformations, *IEEE Workshop on Motion of Non-Rigid and Articulated Objects*, Austin, Texas, Nov. 1994

[BLP89a] H. Bülthoff, J. Little, T. Poggio, A Parallel Algorithm for Real-time Computation of Optical Flow, *Nature* 337(6207):549-553, 9 Feb 1989

[BLP89b] H. Bülthoff, J. Little, T. Poggio, A Parallel Motion Algorithm Consistent with Psychophysics and Physiology, *IEEE 1989 Workshop on Visual Motion*, 165-172, March 1989

[Burt81] P. Burt, Fast Filter Transforms for Image Processing, *Computer Graphics and Image Processing* 16:20-51, 1981

[Burt83] P. Burt, Fast Algorithms for Estimating Local Image Properties, *Computer Vision, Graphics, and Image Processing* 21:368-382, 1983

[C91] T. Camus, Accelerating Optical Flow Computations in VLSI, unpublished report, Brown University January 1991

[C93] T. Camus, Thesis Proposal, Brown AI Memo 93-1005, October 1993

[C94a] T. Camus, Real-Time Optical Flow, SME Technical Paper MS94-176, MVA/SME Applied Machine Vision June 1994, Minneapolis Minnesota

[C94b] T.Camus, *Real-Time Optical Flow*, PhD Thesis, Brown University Technical Report CS-94-36, 1994

[CB91] T. Camus, H. Bülthoff, Space-Time Tradeoffs for Adaptive Real-Time Tracking, Mobile Robots VI, William J. Wolfe, Wendall H. Chun ed., *Proc. SPIE 1613*, p.268-276, Nov. 1991

[CHHN95] D. Coombs, M. Herman, T. Hong, M. Nashman, Real-Time Obstacle Avoidance Using Central Flow Divergence and Peripheral Flow, in submission and National Institute of Standards and Technology technical report.

[DN93] A. Del Bimbo, P. Nesi, Optical Flow Estimation on the Connection Machine 2, p.267-274, *Workshop on Computer Architectures for Machine Perception*, New Orleans Louisiana, IEEE Computer Society Press, Los Alamitos CA, 1993

[DNS92] A. Del Bimbo, P. Nesi, J. Sanz, Optical Flow Computation Using Extended Constraints, Department of Systems and Informatics Tech Report DSI-RT 19/92, Faculty of Engineering, University of Florence, Italy, 1992

[Du94a] A. Duchon, Robot Navigation from a Gibsonian Viewpoint. 1994 *IEEE International Conference on Systems, Man and Cybernetics*

San Antonio, Texas. October 2-5, 1994. IEEE, Piscataway, NJ, pp 2272-2277.

[Du94b] A. Duchon, Ecological Robotics. in preparation, 1994.

[DW93] R. Dutta. C. Weens, Parallel Dense Depth from Motion on the Image Understanding Architecture, *Proceedings of the IEEE CVPR*, p.154-159, 1993

[F86] K. Fant, A Nonaliasing, Real-Time Spatial Transform Technique, *IEEE Computer Graphics and Applications* 6:71-80, 1986

[Fl92] D. Fleet, *Measurement of Image Velocity*, Kluwer Academic Publ., Norwell MA, 1992

[FvD+90] J. Foley, A. vanDam, S. Feiner, J. Hughes, *Computer Graphics Principles and Practice*, Addison-Wesley, Reading MA, 1990

[G87] F. Glazer, *Hierarchal Motion Detection, PhD Dissertation*, COINS TR 87-02 University of Massachusetts at Amherst, 1987

[HA91] L. Huang, Y. Aloimonos, Relative Depth from Motion Using Normal Flow: An Active and Purposeful Solution, *IEEE Workshop on Visual Motion*, p.196-204, 1991

[Hil84] E. Hildreth, *The Measurement of Visual Motion*, MIT Press, Cambridge MA

[Horn86] B. Horn, *Robotic Vision*, MIT press, Cambridge MA, 1986

[HB88] I. Horswill, R.Brooks, Situated Vision in a Dynamic World: Chasing Objects, *Seventh National Conference on Artificial Intelligence*, p.796-800,1988

[Heg87] D. Heger, Optical Flow from Spatiotemporal Filters, *Proceedings of the IEEE ICCV*, p.181-190, 1987

[HS81] B. Horn, P. Schunck, Determining Optical Flow, *Artificial Intelligence* 17:185-203, August 1981

[HS93] R. Haralick, L. Shapiro, *Computer and Robot Vision II*, Addison-Wesley, Reading, MA, 1993

[Hub94a] S. Huber, video presentation, From Animals to Animats, *Third International Conference on Simulation of Adaptive Behavior*, Brighton, G.B., 8-12 August 94

[Hub94b] S. Huber, personal communication.

[K86] J.J. Koenderink, Optic Flow, *Vision Research* 26(1):161-180, 1986

[KMB+91] C. Koch, A. Moore, W. Bair, T. Horiuchi, B. Bishofberger, J. Lazzaro, Computing Motion Using Analog VLSI Vision Chips, An Experimental Comparison Among Four Approaches, *IEEE Workshop on Visual Motion*, p.312-324, 1991

[KSL85] E. Kent, M. Shneier, R. Lumia, PIPE: Pipelined Image Processing Engine, *Journal Parallel and Distributed Computing* 2, P. 50-78, 1985

[KvD78] J.J. Koenderink, A.J. van Doorn, How an Ambulant Observer can Construct a Model of the Environment from the Geometrical Structure of the Visual Flow, *Kybernetic* 1978, Oldenbourg, Muenchen

[KWM89] C. Koch, H.T. Wang, B. Mathur, Computing Motion in the Primate's Visual System, *Journal of Experimental Biology*, 146:115-139,1989

[Lee76] D. Lee, A Theory of Visual Control of Braking Based on Information about Time-to-Collision, *Perception* 5:437-459, 1976

[L88] J.Little, Integrating Vision Modules on a Fine-Grained Parallel Machine, in *Machine Vision*, Academic Press, p.57-96, 1988

[LK93] J. Little, J. Kahn, A Smart Buffer for Tracking Using Motion Data, p.257-266, *Workshop on Computer Architectures for Machine Perception*, New Orleans Louisiana, IEEE Computer Society Press, Los Alamitos CA, 1993

[LV89] J. Little, A. Verri, Analysis of Differential and Matching Methods for Optical Flow, *IEEE 1989 Workshop on Visual Motion*, 173-180, March 1989

[Mor77] H. Moravec, Towards Automatic Visual Obstacle Avoidance, *5th IJCAI*, p.584, 1977

[MBLB91] H. Mallot, H. Bülthoff, J. Little, S. Bohrer, Inverse Perspective Mapping Simplifies Optical Flow Computation and Obstacle Detection, *Biological Cybernetics*, 64:177-185,1991

[MB90] D. Murray, B. Buxton, *Experiments in the Machine Interpretation of Visual Motion*, MIT Press, Cambridge MA, 1990

[MH80] D. Marr, E. Hildreth, Theory of Edge Detection, *Proceedings of the Royal Society of London*, B:207 p.187-217, 1980

[MU81] D. Marr, S. Ullman, Directional Sensitivity and Its Use in Early Visual Processing, *Proc.R.Soc.Lond.* B 211, p.151-180, 1981

[Nag87] H-H Nagel, On the Estimation of Optical Flow: Relations between Different Approaches and Some New Results, *Artificial Intelligence* 33:299-324, 1987

[N91] R.Nelson,Qualitative Detection of Motion by a Moving Observer, *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, p.173-178,1991

[NA89] R. Nelson, J. Aloimonos, Obstacle Avoidance Using Flow Field Divergence, *IEEE PAMI-11* no. 10, p.1102-1106, October 1987

[Nal93] V. Nalwa, *A Guided Tour of Computer Vision*, Addison-Wesley, Reading, Massachusetts, 1993

[NG92] S. Negahdaripour, V. Ganesan, Simple Direct Computation of the FOE, *Proceedings of the IEEE CVPR*, p.228-235, 1992

[NS88] K. Nakayama, G. Silverman, The Aperture Problem-I. *Vision Research* 28(6):739-746, 1988

[NSY91] S. Negahdaripour, A. Shokrollahi, C.H. Yu, Optical Sensing for Undersea Robotic Vehicles, *Robotics and Autonomous Systems* 7:151-163, 1991

[P90] J. Perrone, Simple Technique for Optical Flow Estimation, *Journal of the Optical Society of America*, 7(2):264-278, 1990

[PBF89] J-M Pichon, C. Blanes, N. Franceschini, Visual Guidance of a Mobile Robot Equipped with a Network of Self-Motion Sensors, *SPIE Vol. 1195 Mobile Robots IV*, p.44-53, 1989

[PH94] D. Patterson, J. Hennessy, *Computer Organization and Design: the Hardware/Software Interface*, Morgan Kaufmann, San Mateo CA, 1994

[PRH81] T. Poggio, W. Reichardt, K. Hausen, A Neuronal Circuitry for Relative Movement Discrimination by the Visual System of the Fly, *Naturwissenschaften* 68, p.443-446, 1981

[R57] W. Reichardt, Autokorrelationsauswertung als Funktionsprinzip des Zentralnervensystems, *Z. Naturforsch.* 12b:447-457, 1957

[Rock75] I. Rock, *An Introduction to Perception*, Macmillan Publishing Co., New York 1975

[RA86] V. Ramachandran, S. Anstis, The Perception of Apparent Motion, *Scientific American*, June 1986; also in compilation The *Perceptual World*, p.139-151,

[Reg86] D. Regan, Visual Processing of Four Kinds of Relative Motion, *Vision Research* (26)1:127-145, 1986

[S91] A. Singh, *Optic Flow Computation A Unified Perspective*, IEEE Computer Society Press, Los Alamitos CA, 1991

[SAH91] E. P. Simoncelli, E. H. Adelson, D. J. Heeger, Probability Distributions of Optical flow, *Proceedings of the IEEE CVPR*, p.310-315, May 1991

[SS85] J. van Santen, G. Sperling, Elaborated Reichardt Detectors, *Journal of the Optical Society of America*, 2(2):300-321, 1985

[SS93] A. Schilling, W. Strasser, EXACT: Algorithm and Hardware Architecture for an Improved A-Buffer, *ACM SIGGRAPH*, p.85-91, 1993

[SU87] A.Spoerri,S.Ullman,The Early Detection of Motion Boundaries, *International Conference on Computer Vision*, p.209-218, 1987

[T91] J. Tresilian, Empirical and Theoretical Issues in the Perception of Time to Contact, *Journal of Experimental Psychology: Human Perception and Performance*, 17:3 p.865-876 ,1991

[TGS91] M. Tistarelli, E. Grosso, G. Sandini, Dynamic Stereo in Visual Navigation, *Proceedings of the IEEE CVPR*, p.186-193, 1991

[To92a] S. Toelg, Gaze Control for an Active Camera System by Modeling Human Pursuit Eye Movements, *SPIE Intelligent Robots and Computer Vision XI*, p.585-598, Nov. 1992

[To92b] S. Toelg, personal communication.

[TR93] S. Tubaro, F. Rocca, Motion Field Estimators and Their Application to Image Interplation, in *Motion Analysis and Image Sequence Processing*, 1993, Kluwer Academic Publishers, Norwell MA

[TWR94] D. Touretzky, H. Wan, A. Redish, Neural Representation of Space in Rats and Robots, in J.M. Zurada and R.J. Marks, eds., Computational Intelligence: Imitating Life. *Proceedings of the symposium at the 1994 IEEE World Congress on Computational Intelligence* IEEE Press, 1994

[U79] S. Ullman, *The Interpretation of Visual Motion*, The MIT Press, Cambridge MA, 1979

[UGVT88] S. Uras, F. Girosi, A. Verri, V. Torre, A Computational Approach to Motion Perception, *Biological Cybernetics* , 60:79-87, 1988

[VG92] J.A. Vlontzos, D. Geiger, A MRF Approach to Optical Flow Estimation, *Proceedings of the IEEE CVPR* , p.853-856, 1992

[VP87] A. Verri, T. Poggio, Against Quantitative Optical Flow, *Proceedings of the IEEE ICCV* , p.171-180, 1987

[WA85] A. Watson, A. Ahumada Jr, Model of Human Motion Sensing, *Journal of the Optical Society of America* , 2(2):322-342, 1985

[War88] W. Warren Jr., Action Modes and Laws of Control for the Visual Guidance of Action, in *Complex Movement Behavior: The motor-action controversy* , p.339-380, O. Meijer and K. Roth eds., Elsevier Science Pubs., B.V. (North-Holland), 1988

[WM93] J. Weber, J. Malik, Robust Computation of Optical Flow in a Multi-Scale Differential Framework, *Fourth International Conference on Computer Vision*, p.12-20, 1993

[WWB88] A. Waxman, J. Wu, F. Bergholm, Convected Activation Profiles: Receptive fields for Real-Time Measurement of Short-Range Visual Motion, *Proceedings of the IEEE CVPR*, p.717-723, 1988

[WZ91] J. Woodfill, R. Zabih, An Algorithm for Real-Time Tracking of Non-Rigid Objects, *Ninth National Conference on Artificial Intelligence*, p.718-723,1991