Max–Planck–Institut
für biologische Kybernetik
Arbeitsgruppe Bülthoff

Technical Report No. 14 February 1995

Calculating Time-to-Contact Using Real-Time Quantized Optical Flow

Ted Camus

Abstract

Despite many recent advances in optical flow research, many robotic vision researchers are frustrated by an inability to obtain reliable optical flow estimates in real-world conditions to apply for real-world tasks. Recently it has been demonstrated that robust, real-time optical flow is possible using only standard computing hardware [C93]. One limitation of this correlation-based algorithm is that it does not give truly real-valued image velocity measurements. Therefore, it is not obvious that it can be used for a wide range of robotics vision tasks. One particular application for optical flow is time-to-contact: based on the equations for the expansion of the optical flow field it is possible to compute the number of frames remaining before contact with an observed object. Although the individual motion measurements of this algorithm are of limited precision, they can be *combined* in such a manner as to produce remarkably accurate time-to-contact measurements, which can be produced at real-time rates, on the order of 6 frames a 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, USA. email: `tac@cme.nist.gov`.

1 Introduction

For robotic vision to be successful and practical in real-world environments, it must be robust, fast, and sufficiently accurate as appropriately defined for a given task. If an algorithm is not robust and only works on well-specified input, it is useless in the uncertainty of a noisy, real-world environment, regardless of any other desirable qualities it may possess. Similarly, an algorithm that may take many minutes to run is clearly inadequate for a reactive system, and must be run off-line or on expensive special-purpose hardware. Finally, a robotic vision algorithm must return correct measurements as appropriately defined for a given task. In some cases such as obstacle avoidance, *qualitative* computer vision may be sufficient (e.g. [NA89]). Of course more accuracy is preferable to less, however under no circumstances may the first two requirements of robustness and speed be sacrificed. If these first two criteria are satisfied however, accuracy may then be optimized.

Despite many recent advances in optical flow research, many robotic vision researchers are frustrated by an inability to obtain reliable optical flow estimates in real-world conditions to apply for real-world tasks. Recently [C93] demonstrated that robust, real-time optical flow is possible using only standard computing hardware. Although this algorithm was sufficiently fast and robust to be used in tasks such as instantiating fly-like control laws [War88] in a small mobile robot [Du94a], it still has the limitation of not returning truly real-valued image velocity measurements. Therefore, it is not obvious that it can be used for a wide range of robotics vision tasks. One particular application for optical flow is time-to-contact: based on the equations for the expansion of the optical flow field it is possible to compute the number of frames remaining before contact with an observed object [Lee76]. Although the individual motion measurements of this algorithm are of limited precision, they can be *combined* in such a manner as to produce remarkably accurate time-to-contact measurements, which can be produced at real-time rates.

2 Optical Flow for Real-Time Robotics

Many techniques for optical flow exist (see [BFB94] 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. D. Touretzky et.al. note *"Even something as simple as calculating real-time optic flow requires more processing power than is practical for a mobile robot."* ([TWR94]). 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], [WZ91], [L88]), Datacube ([N91], [LK93]), or custom image processors ([DW93]). One obvious reason calculating optical flow is so computationally intensive is that images are composed of thousands of pixels (which often motivates massively parallel implementations). Another reason is that optical flow is very sensitive to noise, and drastic steps (such as iterative smoothing [HS81], processing the images with multiple convolutions [WM93], etc.) are often needed to overcome this sensitivity. One option is to only calculate optical flow at areas of high contrast; for example [To92] 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 the purposes of tracking a moving object. Many applications of optical flow require dense outputs however, which is very difficult in areas of low contrast. For the purposes of real-time robotic vision, 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.

3 Robust, Real-time Optical Flow

In general it is not possible to determine the correct optical flow field given a pair of successive image frames. If certain assumptions are enforced, however, then the problem becomes well-posed, and can be satisfactorily solved in most cases. Pixel-shifting algorithms that select that motion which maximizes a match value (e.g., the sum of the absolute or possibly squared differences) such as [BLP89a], [A87a] can be robust in practice, but are often computationally expensive. A real-time optical flow algorithm described in [C94], [CB95] and summarized here makes use of a simple relationship between velocity, distance, and time:

$$velocity = \frac{\Delta distance}{\Delta time}. \qquad (1)$$

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

$$\Delta v = \frac{\Delta d}{\delta t}. \qquad (2)$$

However, this results in an algorithm that is quadratic in the range of velocities present [CB91]. 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). 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 $t - k$. 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 and the *temporal aperture problem* [C94], [CB95]; since our application of time-to-contact will assume constant motion, however, this effect does not concern us. The chosen motion for a given pixel is that motion which yields the best match value of all possible shifts (over both time and space).

For example consider Figure 1 and Figure 2. Here we are trying to calculate the optical flow for pixel (1,1) at the current frame, Image T. Although only the pixel in question is shown, we would match a patch centered at the pixel in question when performing the motion search [BLP89a], with the match being the sum of absolute or possibly squared differences. In Figure 1 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 2 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.

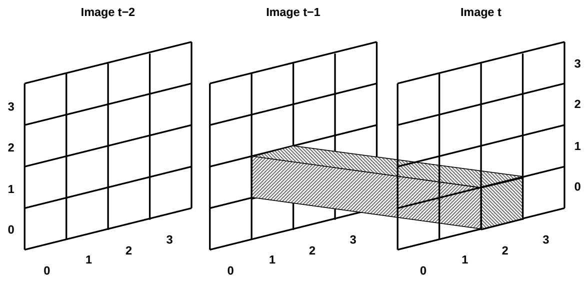

Figure 1: Visualization of motion from Image $T-1$: (1,1) to Image T: (2,0). This would be an optical flow of (1,-1) pixel/frames motion for pixel Image T: (1,1).

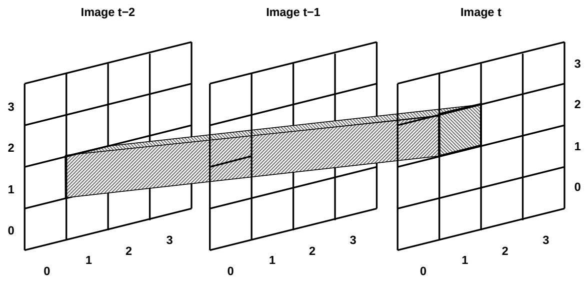

Figure 2: Visualization of motion from Image $T-2$: (1,1) to Image T: (1,2). This would be an optical flow of (0,1/2) pixel/frames motion for pixel Image T: (1,1).

The pixel-shift search space is fixed in the 2-D space of the current image, but is extended linearly in time. Since we perform a linear search in time instead of a quadratic search in space, real-time implementations are possible. Since the search is linear, the frame rate f is inversely proportional the slowest velocity detected, $1/S$ pixels/frame using a delay of S frames. Computing optical flow on 64x64 images on a 50 MHz Sun Sparcstation 20 yields $f * S = 64$. I.e., if 1/8 pixels/frame is the slowest velocity detected, optical flow can be calculated at 8 frames/second. However, if 1/4 pixels per frame is the slowest velocity detected, then up to 16 frames/second could be calculated. The exact value of S is application dependent. Certain applications such as time-to-contact can make use of even $S = 1$ and run at over double video rates alone, or at video rates including the processing for time-to-contact (see Section 5) .

4 Measurement Quantization

One disadvantage of the patch-matching approach is that the basic motion measurements are integer multiples of pixel-shifts. Although the algorithm discussed in this paper does calculate subpixel motions, it still computes velocities that are

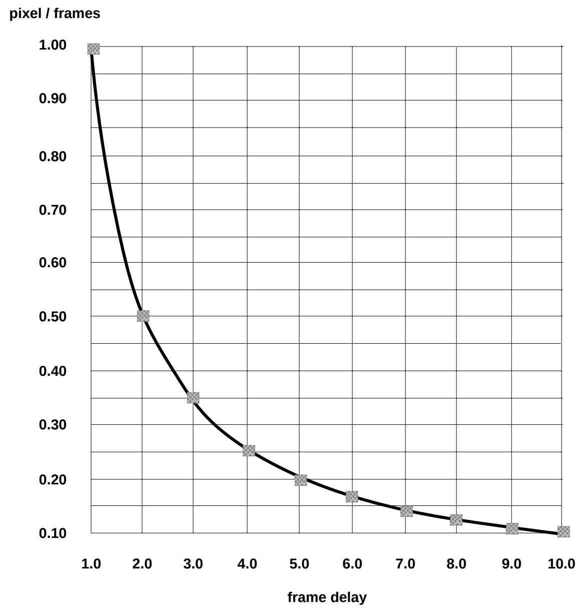

Figure 3: Only 1/(frame delay) velocity measurements can be detected in the current implementation.

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 $\{1/1, 1/2, 1/3, .., 1/S\}$ pixels per frame equivalent motion, Figure 3. Although this harmonic series is not equivalent to the traditional linear set of motions $\{1, 2, 3, ..., \eta\}$ pixels per frame since it provides greater precision at slower velocities, it does seem more suitable for many vision tasks since the harmonic series of motions more closely models the effects of perspective projection (see [C94], [CB95]).

However, the basic measurements are still quantized in velocity magnitude. 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]), remarkable accuracy may still be achieved in the context of calculating time-to-collision despite these deficiences.

5 Calculating Time-to-Contact

A primary use of optical flow in robotic vision is collision detection, in particular time-to-contact, sometimes pessimistically referred to as time-to-collision or time-to-crash [Lee76], [T90], [AP93]. Using only optical measurements, and without knowing one's own velocity or distance from a surface, it is possible to determine when contact with a visible surface will be made. Figure 5 shows six frames from a collision sequence. A small mobile robot began about about a meter and a half away from a chair with a sweatshirt draped over the chair so that actual contact could be made with the visual target without damaging the robot or camera. The translation rate was about 5 cm per second, and the frame rate was about 5 frames per second, so each frame represents about 1 cm of actual translation. (All measurements are subject to the limits of the accuracy of the robot's gears and the reliability of UNIX's timing commands.) The camera's lens has a field of view of 60 degrees, however only the central 256x256 pixels of an original 512x512 image were used in subsampling to 64x64. The camera's lens was not refocused as the robot approached the target.

Note in particular the very low contrast of the sequence; gradient techniques which suggest calculating flow at grey-level "corners" would not find any such high-contrast locations in these images. Although it may be possible to construct a time-to-collision detector using contrast-sensitive detectors such as Reichardt detectors [R57] if such detectors were consistent throughout the visual field [AP93], typical office environments are likely to have widely varying spatial structures, so this cannot be guaranteed. Contrast-sensitive elementary motion detectors may be sufficient for very small (and lightweight) insects such as the fly [Bor90], [EB93] since such a small insect can afford a wide margin of error when landing on a surface due its very high strength-to-weight ratio, contrast-sensitive motion detectors are clearly insufficient for mobile robots since an error in collision detection could seriously damage the robot.

Actual contact with the sweatshirt covering the chair was made somewhere between frames 141 and 142 (nominally 141.5). Thus we would want our time-to-contact algorithm to determine, at every point during the sequence, that the contact point is around frame 141.5 (i.e. the current frame number plus the current time-to-contact in frames). For this sequence the slowest velocity searched for, $1/S$, was set to 1/10 pixel/frames.

Figure 4 describes the optical geometry (the

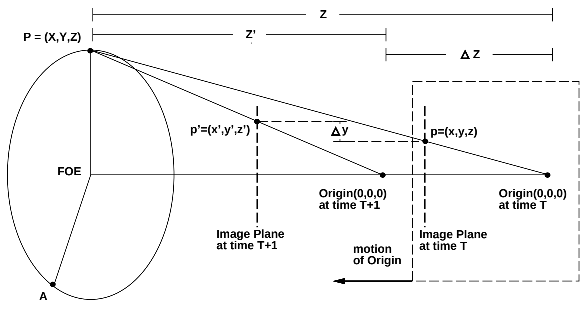

Figure 4: Optical geometry for time-to-contact. See text for details.

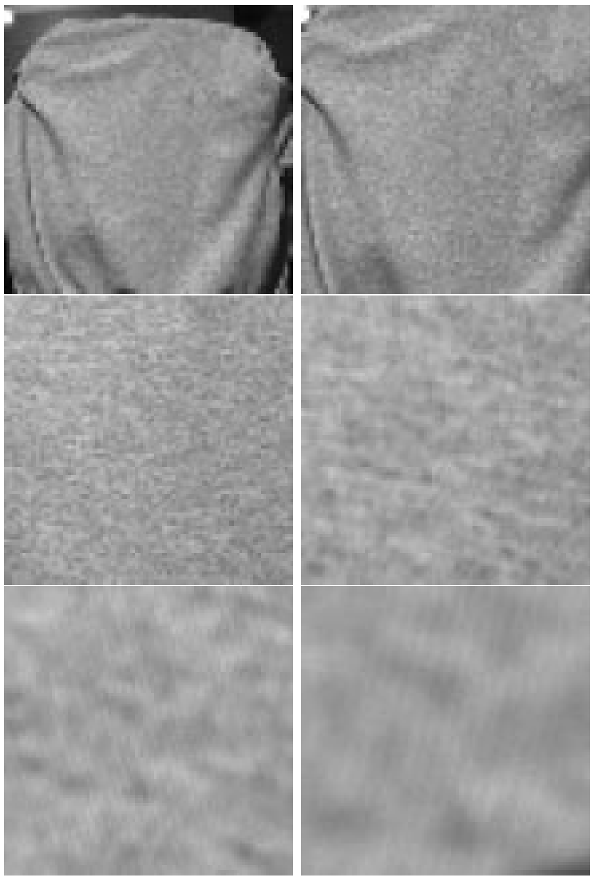

Figure 5: Images of the chair (in row-major order) at frames 12, 41, 96, 126, 134, and 139. Note the extremely low contrast of the sequence.

classic reference for the image-plane coordinate system is [LP80]). A point of interest P at coordinates (X,Y,Z) is projected through the focus of projection centered at the origin of the coordinate system (0,0,0). P is fixed in physical space and does not move. The origin/focus of projection, however move forward with a velocity $\frac{dZ}{dt}$. If the camera is facing the same direction as the direction of motion, then this direction is what is commonly known as the focus of expansion (FOE), since it is the point from which the optical flow diverges. (In the case of mobile robotics, it is quite reasonable to assume that the camera is in the same direction as the direction of translation; we do not have to solve the general case of egomotion in order to exhibit useful behaviors.) The image plane is fixed at a distance z in front of the origin; for convenience, we set $z = 1$. (The actual value of z would depend on factors such as the focal length of the camera.) This image plane moves along with the origin. P projects onto point p in this plane. As the image plane moves closer to P, the position of p in the image plane changes as well. Using equilateral triangles:

$$\frac{y}{z} = \frac{y}{1} = \frac{Y}{Z}$$

Differentiating with respect to time [where $\dot{a}$ represents the time derivative $\frac{da}{dt}$ for a given variable a]:

$$\dot{y} = \frac{\dot{Y}}{Z} - Y(\frac{\dot{Z}}{Z^2})$$

Since P is immobile, set $\dot{Y} = 0$ and substituting (yZ) for Y:

$$\dot{y} = -y(\frac{\dot{Z}}{Z})$$

Finally divide by y and take the reciprocals of both sides:

$$\frac{y}{\dot{y}} = -\frac{Z}{\dot{Z}} = \tau \qquad (4)$$

The quantity τ is known as the time-to-contact [Lee76]. Note that the left hand side contains purely optical quantities, and that knowledge of τ does not give any information about distance or velocity per se, but only of their ratio. The left-hand side of equation 4 gives us a method for calculating time-to-contact: for a camera heading in the same direction as the FOE, pick a point in the image, and divide its distance from the FOE by its divergence from the FOE.

This requires first calculating the FOE. Ideally, we could simply take the intersection of any two optical flow vectors, but due to measurement errors this would be inaccurate, and in the case of this particular algorithm it certainly would not work, since the velocity measurements are limited to only eight directions (and zero).

This algorithm can take advantage of the X-Y component representation of the optical flow vectors. In the case of forward translational motion and a single object filling the field of view, the FOE may be calculated by averaging the X and Y components of all the optical flow vectors, with each component being assigned a unit pixel value regardless of its actual magnitude, and treating the average as an offset from the line of sight, which is assumed to be straight forward. Since motion is quantized into eight directions, horizontal and vertical motion is considered to be plus or minus one unit motion in either the X or Y directions and zero units for the other, and diagonal motion is considered to be plus or minus one unit in both the X and Y directions. The vector components are assigned unit values for robustness, *i.e.*, vectors close to the FOE are not penalized because their actual magnitudes are very small.

From equation 4 and Figure 4 we can see that the expected divergence of any point along a circle of a given radius should be equal; e.g., given the assumption of a flat surface and a direct approach the divergence of any point A should be equal to that of point P. Therefore, it is justifiable to average the optical flow measurements along the circumference of any circle of a given radius, centered at the FOE, to get a single real-valued measurement for $\dot{y}$ in equation 4. The number of individual measurements used in this paper is 4 times the radius in pixels, i.e. 4 measurements for a one-pixel radius, 8 for a 2 pixel radius, up to 124 for a maximum 31-pixel radius. In this case, each "individual" measurement is in fact the weighted averaged of the four nearest pixels to that actual real-coordinate point in the image. Quantity y in equation 4 is then simply the radius of that particular circle, and then finally a single measurement of time-to-contact is calculated by $\tau = y/\dot{y}$. The number of independent time-to-contact measurements available is only limited by the image size, and the position of the FOE.

The current implementation only calculates a maximum velocity of one pixel per frame along any of the four cardinal directions (NEWS). Along the diagonals, the maximum velocity is $\sqrt{2}$ pixels/frame, however when the actual motion for points along a circle of a given radius exceeds one pixel per frame (for the current implementation), the optical flow in the NEWS directions saturates, yielding slower than actual optical flow, and thus a longer than actual time-to-contact. For the current problem, we will simply not consider measuring the time-to-contact for radii whose averaged motion results in greater than one pixel per frame. Even when only 2.5 frames away from collision, there are enough valid measurements for an accurate time-to-contact calculation.

In practice we limit the velocity search to some lower bound, these slow velocities may be slower than can be detected. However, the slowest velocity that is detected may give a better match than no motion, thus these pixels' velocities may be rounded up to $1/S$ pixels/frame, yielding a lower than actual time-to-contact for that radius. Thus it is often useful to set some lower velocity threshold below which time-to-contact measurements will not be considered, since they are likely to be inaccurate. Rather than setting a threshold on the individual measurements, we set a threshold on the averaged measurement for the entire circle. This allows the slowest velocities to participate in the distribution of velocities for a circle of a given radius. Although there currently is no automatic mechanism for setting this threshold, values from 1.6 to 1.8 times the slowest velocity checked for in the image have been successful. In addition, experiments show that there is a point where increasing the threshold has little effect on the measurements (for those frames where the object fills the field of view). Thus it could be possible to adaptively set this threshold. Even at the beginning of the sequence when only a few measurements are above threshold, the time-to-contact measurement is quite reliable. Together these two thresholds eliminate measurements that are outside the optimal range of $\{1/1, 1/2, ..., 1/S\}$ pixel/frames.

By calculating an estimate of time-to-contact along the circumferences of circles of multiple radii, we can compute several independent estimates, and take some best-fit measure among them. In this case a robust maximum-likelihood estimate is used [H81]. Figure 6 shows the time-to-contact measurements for each radius for several frames in the collision sequence. Each measurement results from applying equation 4 to the averaged motion estimate for that given radius. The top vertical line represents the 1.0-pixel/frames upper threshold, the bottom vertical line represents the lower-bound threshold of .183 pixel/frames, the dotted horizontal line represents the initial estimate, and the solid horizontal line

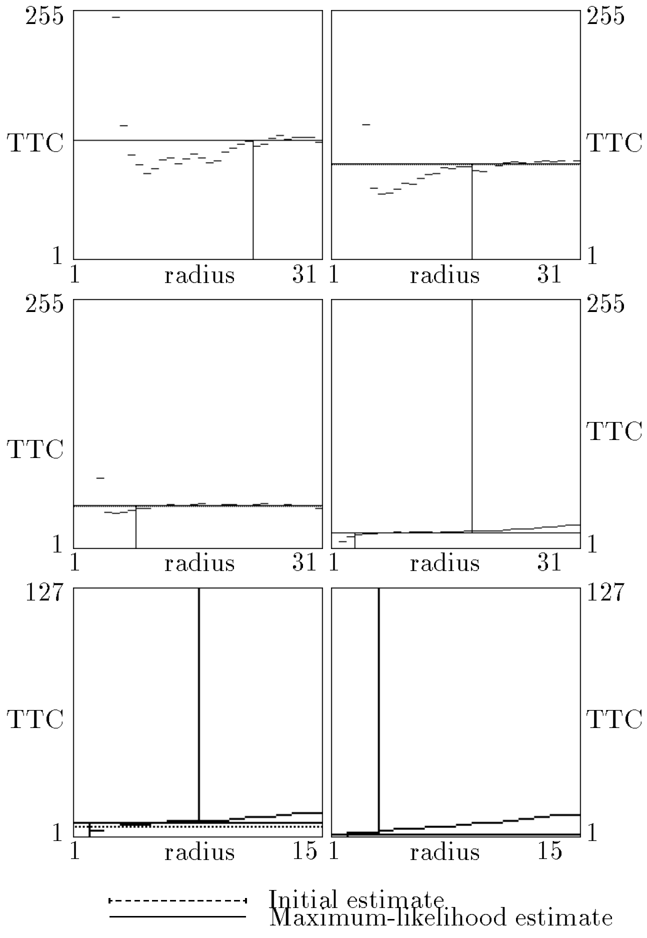

Figure 6: Time-to-contact estimates (in row-major order) for frames 12, 41, 96, 126, 134, and 139. Lower-bound threshold is .183 pixel/frames. The plots for the last 2 frames have only a few valid measurements less than 1.0-pixel/frames, so for enhanced visibility the lower left-hand quarter is zoomed in by a factor of 2.

represents the final maximum-likelihood estimate for time-to-contact.

Examining this figure, there is generally a drop in the time-to-contact measurements corresponding to motions below the lower-bound threshold, due to the "rounding up" of the slowest velocities near the FOE. The location of this sudden drop could possibly be used to adaptively set the value of the lower-bound threshold, if it were known that all measurements corresponded to a single surface. The remaining valid measurements are in strong agreement with one another, meaning that the time-to-contact measurements are highly redundant and are valid regardless of their distance from the FOE. In addition, this property is true at almost all time points in the image sequence.

Figure 7 left shows a plot of these best-fit

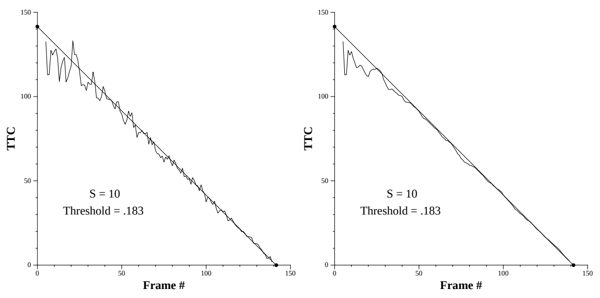

Figure 7: Results of time-to-contact experiment for the collision sequence. Left figure is a plot of instantaneous time-to-contact vs. frame number. Right figure is similar except that each value represents the average of the current and previous 7 instantaneous time-to-contact measurements.

time-to-contact measurements against the current frame number. The actual contact point is somewhere between frames 141 and 142, so for the purposes of this graph was arbitrarily set to 141.5, represented by the solid line from (0,141.5) and (141.5,0). The jagged nature of this plot suggests aliasing due to "jumps" of the velocity measurements at various positions in the image from $1/S$ (or $\sqrt{2}/S$) pixels/frames to $1/(S-1)$ (or $\sqrt{2}/(S-1)$) pixels/frames as the target is approached, or some similar artifact, but this has not been confirmed. In any event, we can take an average of the last n time-to-contact measurements as the final measure of time-to-contact; in this paper $n = 8$. The final result is shown in Figure 7. Even with the fast linear-time optical flow algorithm, collision detection time is dominated by the computation of optical flow; calculating time-to-contact is roughly equivalent in CPU time to calculating a single speed (time delay) of optical flow, or about 64 frames per second given the optical flow input. For this example the optical flow and time-to-contact calculations were computed offline and can be computed at about 6 frames per second on a 50 MHz Sun Sparcstation 20.

Most of the measurements in the last 100 frames are within a frame of this value, and the average is 140.86 with standard deviation 0.91 frames. The average for the last 50 frames is 141.43, with standard deviation 0.52 frames. In addition, measurements are valid up until 2 frames before contact (invalid measurements are clearly labeled as such by the algorithm). Clearly, integrating across space and time more than compensates for the non-real-valued measurements of the linear-time optical flow algorithm (at least for this example).

This is not the only way to approach this problem. Another option is to average the motions across the entire image, rather than along individual circles of a given radius, and then a best fit among these measurements. Performing the averaging in two steps, however, enables us to set the thresholds we described above and conveniently eliminate points that yield measurements that fall outside the optimal range of $\{1/1, 1/2, ..., 1/S\}$ pixel/frames. It also allows us to visualize the optical flow algorithm as a function of distance from the radius. Finally, placing points that are a constant distance from the FOE into a single equivalence class yielding a single measurement results in a dimensional reduction which results in finding the maximum-likelihood estimate of only up to 31 measurements (the maximum radius used for a 64x64 image) instead of up to 4096 individual points (minus those in the border area). This makes the computation time for finding the maximum-likelihood estimate negligible, given the usual fast convergence.

5.1 Example: Low-precision Image Data Input

Real-time robotic vision has two essential requirements, real-time performance and robustness in real-world situations. The former condition has been addressed by the development of a real-time optical flow algorithm. The second condition demands that algorithms be extremely robust against noise. In particular optical flow methods based on numerical differentiation are notoriously sensitive to noise. The following example demonstrates the algorithm's degradation of performance for the time-to-contact experiment of Figure 7 using very low precision images.

The intensity value of each pixel in a greyscale image is generally represented by an unsigned integer in the range of 0-255, i.e. an 8-bit integer. The following example demonstrates the result of the time-to-contact algorithm when run on images reduced to only 2 bits of precision. The extremely limited precision of these images is evident in Figure 8. In fact, the vast majority of the image sequence actually only uses 2 colors, corresponding to grey levels 128 and 192 respectively, and are thus effectively 1-bit bitmaps. Clearly, with only 2 grey levels present, techniques based on numerical differentiation cannot be used. The results for the collision sequence with effectively 1-bit input are shown in Figure 9. For this input the computed time-to-contact value is at worst about within a factor of 2 of the correct value and becomes more accurate as contact approaches. At 90 frames from contact (approximately 90 cm. from the target) the algorithm produces a time-to-contact 70% of actual, and at 50 frames away it is 85% of actual. At 40 frames from contact the algorithm is within three frames of actual, and at 25 frames from contact it is within a single frame. The algorithm maintains this accuracy up until two frames before contact (same as that for the full-precision sequence), at which point it diverges slightly. The algorithm demonstrates a graceful degradation of performance on images with intermediate bit precision, as well as images degraded with varying levels of noise [C94]. As with the full-precision example, the optical flow and time-to-contact calculations were computed offline and can be computed at about 6 frames per second for 64x64 images and $S = 10$ on a 50 MHz Sun Sparcstation 20. The fact that such accuracy is possible given such poor input data and coarsely quantized optical flow suggest relatively simple hardware implementations, in terms of modest computational power, low-quality CCD camera, and low-precision frame-buffer requirements.

An important point here is that we are not merely using some "quick and dirty trick" to determine time-to-contact, nor are we tracking the motion of feature points, which might be undesirable due to the difficulty of extracting reliable feature points (and assuming that there are enough such points in the image for subsequent processing). This particular application of time-to-contact on effectively binary data does not explicitly attempt to perform the type of binary correlation matching as done in [Ni84], which autocorrelates the sign of a convolution with the Laplacian of a Gaussian with the image, although the similarities are interesting. This application is using the same exact optical flow and time-to-contact algorithms as before, just on very poor image data.

5.2 Example: Low-precision Optical Flow Input

The previous section tested the robustness of the optical flow algorithm by using very low-precision images as input data. This section will test the robustness of the time-to-contact algorithm itself by using very-low precision optical flow input.

Biological implementations of collision detection, such as the spatial integration of simple correlation-type motion detectors found in the fly do suggest that useful collision avoidance can be performed using imprecise input [EB93] (although the fly does not appear to explicitly calculate time-to-contact [BB88]). For example, [AP93] computes optical flow only along a single dimension;

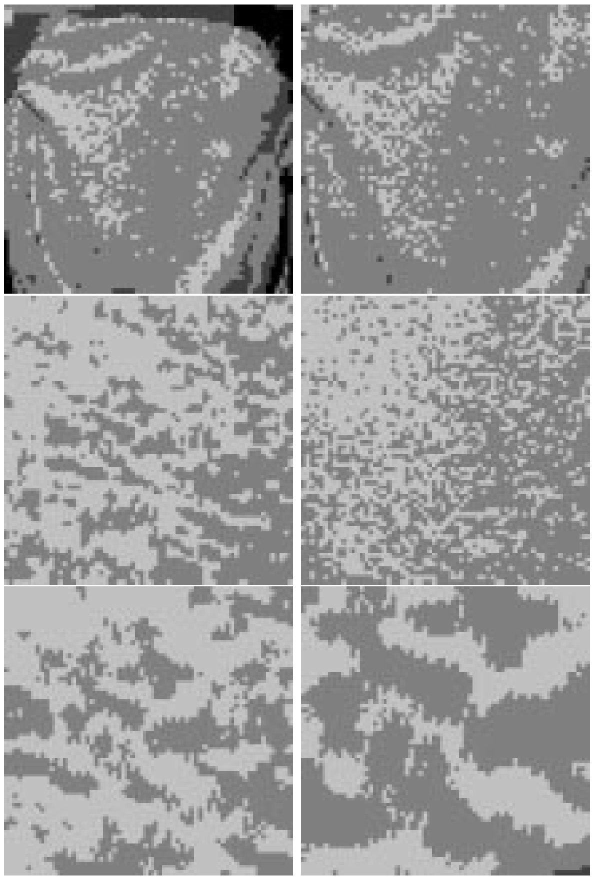

Figure 8: Image of the chair (in row-major order) at frames 12, 41, 96, 126, 134, and 139 reduced to only 2 bits of precision. The majority of the images actually only use 2 grey levels, and are effectively 1-bit bitmaps.

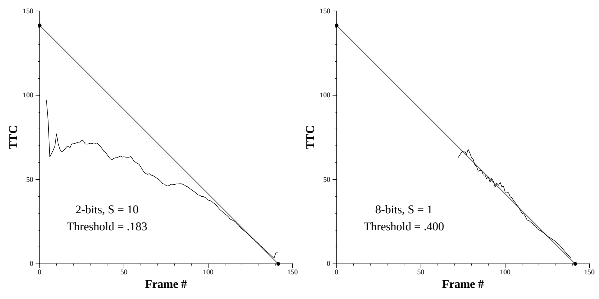

Figure 9: Results of time-to-contact experiment for the collision sequence with (left) 2-bit precision images, $S = 10$, lower threshold = .183, and (right) full precision images, $S = 1$, and lower threshold = .400. For the graph on the right, valid measurements did not begin until about halfway through the sequence.

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 intergrating 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 results of running the optical flow algorithm with $S = 1$ (i.e., calculating only zero-or-one pixels/frame motion along the NEWS directions and $\sqrt{2}$ pixels/frame along the diagonals) on normal image input and the time-to-contact algorithm with this optical flow output is shown on the right of Figure 9. Here the lower-bound threshold was arbitrarily set to .400 pixels/frame. (Ideally, in the future this threshold can be adaptively set as discussed earlier, although it would be more difficult with such limited velocity measurements.) Most time-to-contact estimates in this graph are within 2 frames of the actual value of time-to-contact. Estimates are valid up until 2 frames before contact, however in this example there were no valid time-to-contact estimates until about halfway through the 142 frame sequence. The optical flow with $S = 1$ and the standard time-to-contact algorithm can be computed together at over 30 frames/second for 64x64 images on a 50 MHz Sun Sparcstation 20.

A lesson learned is that instead of relying on the accuracy of individual measurements, each one of which is very difficult to obtain and is sensitive to noise, it may be better to use some combination of individual measurements, each one of which is robust but not necessarily very precise.

6 Future Work

The Time-to-Contact and FOE calculations made several assumptions, such as constant unidirectional motion, and a flat object that completely fills the field of view. In addition, the sweatshirt used as a target, although solid-colored, is not as textureless as many objects found in a typical office, such as a black filing cabinet. Subsequent work will examine these issues in turn.

7 References

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

[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

[BB88] A. Borst, S. Bahde, Spatio-Temporal Integration of Motion, *Naturwissenschaften* 75, p.265-267, 1988

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

[Bor90] A. Borst, How Do Flies Land ?, *BioScience* 40(4): 292-299, April 1990

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

[C94] 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

[CB95] T. Camus, H. Bülthoff, Real-Time Optical Flow Extended in Time, MPI für biologische Kybernetik 13, February 1995

[Du94] 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.

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

[EB93] M. Egelhaaf, A. Borst, Motion Computation and Visual Orientation in Flies, *Comp. Biochem. Physiol.*, 104A(4):659-673, 1993

[H81] P. Huber, *Robust Statistics*, Wiley, New York, 1981

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

[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

[LP80] H. Longuet-Higgins, K. Prazdny, The Interpretation of a Moving Retinal Image, *Proc. of the Royal Society of London*, B 208:385-397,1980

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

[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

[Ni84] H. Nishihara, Practical Real-Time Imaging Stereo Matcher, *Optical Engineering* 23(5):536-545 Sept./Oct. 1984

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

[T90] J. Tresilian, Perceptual Information for the Timing of Interceptive Action, *Perception* 19:223-239, 1990

[To92] 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

[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

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

[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