Real-time Single-workstation Obstacle Avoidance Using Only Wide-field Flow Divergence

Ted Camus[1], David Coombs, Martin Herman[2], Tsai-Hong Hong
tedcamus@sensar.com, {dcoombs,mherman,thong}@nist.gov
National Institute of Standards and Technology
Intelligent Systems Division
Building 220 (Metrology) Room B-124
100 Bureau Dr Stop 8230
Gaithersburg MD 20899-8230

ABSTRACT

This paper describes a real-time robot vision system which uses only the divergence of the optical flow field for both steering control and collision detection. The robot has wandered about the lab at 20 cm/s for as long as 26 minutes without collision.The entire system is implemented on a single ordinary UNIX workstation without the benefit of real-time operating system support. Dense optical flow data are calculated in real-time across the entire wide-angle image. The divergence of this optical flow field is calculated everywhere and used to control steering and collision-avoidance behavior. Divergence alone has proven sufficient for steering past objects and detecting imminent collision. The major contribution is the demonstration of a simple, robust minimal system that uses flow-derived measures to control steering and speed to avoid collision in real time for extended periods. Such a system can be embedded in a general, multi-level perception/control system.

1 Introduction

Mobile robots that drive at reasonable speeds (*e.g.*, 20 cm/s indoors) must robustly sense and avoid obstacles in real time. Image motion provides powerful cues for understanding scene structure. Divergence of optical flow (the sum of optical flow derivatives in two perpendicular directions) is theoretically unaffected by camera rotation, so it gives a robust measure of scene structure for a moving observer. The robot system described here uses flow divergence to steer around obstacles while it attempts to achieve a goal (which for now is simply to drive straight ahead). When the obstacle avoidance is insufficient to avoid collision, the divergence data warn the robot of the impending collision. The robot stops, turns, and resumes wandering straight ahead in the new direction. These inte-

1. This research was conducted while the first author held a National Research Council Research Associateship at NIST. Current address: Sensar, 121 Whittendale Drive, Moorestown, NJ 08057. email: tedcamus@sensar.com
2 Current address: NIST, Information Access and User Interfaces Division, 100 Bureau Drive Stop 8940, Gaithersburg MD 20899-8940

grated behaviors have driven the robot around the lab at 20 cm/s for as long as 26 minutes without collision. Because this wandering behavior is already a real-time capability, there is promise that future increases in computational power will fuel development of both increasingly robust basic skills and additional behaviors for robots.

The simplicity of the system improves robustness and makes the system easier to extend. The system uses only a single framegrabber, a single processor, a single image stream, and a single low-level percept for all control functions. Simple robust filters are chosen in lieu of complex filters that require sensitive system modeling and synchronization. These filters are able to ignore momentary noise and artifacts that result from system module interactions, which enables modules to cooperate without delicate synchronization.

In addition, the obstacle avoidance system is extensible. Egocentric hazard maps are derived from divergence data, goals, and steering history. A composite hazard map is used to steer the vehicle. This design supports the use of multiple cues, which can be incorporated with additional hazard maps. Additional redundant or complementary sensing modes can be exploited by the existing framework.

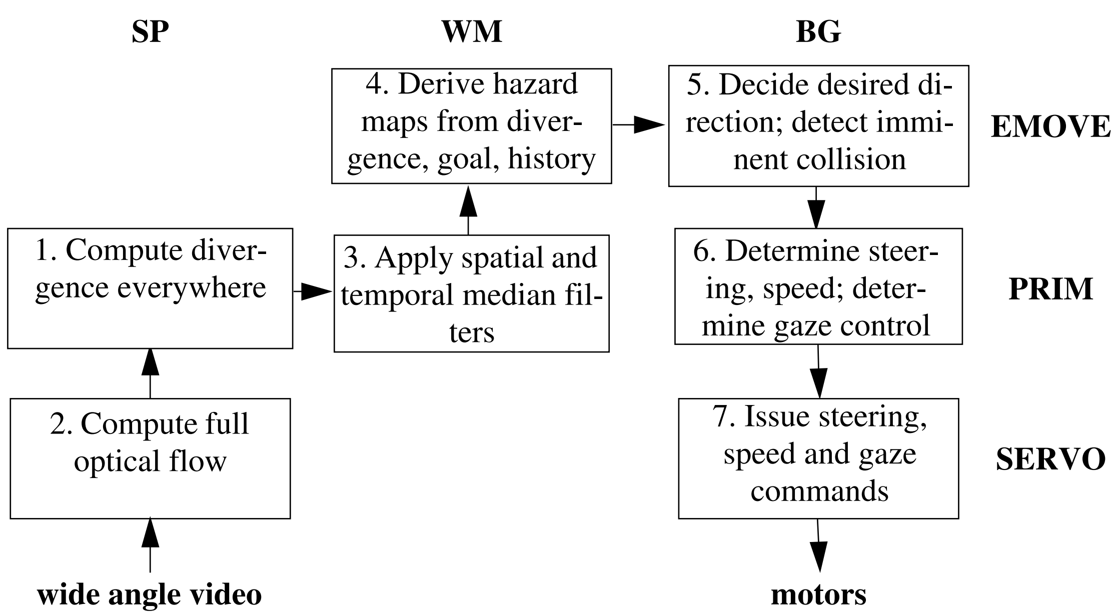


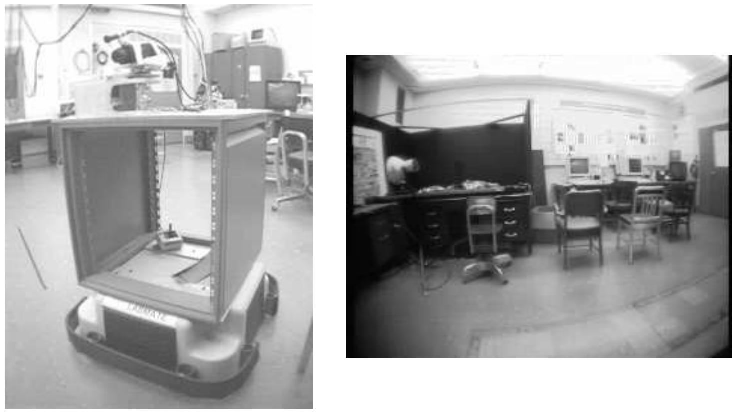
Figure 1 Obstacle Avoidance System Architecture

Our approach achieves real-time intelligent behavior by using minimalist visually-derived representations. In such representations, a minimal amount of information required to achieve the given task is extracted from the imagery [2][4]. The representations contain only task-relevant information (*i.e.*, relevant to obstacle avoidance), and the information is only represented in 2-D image coordinates. The control algorithms directly use observable image information represented in the 2-D image sequence; a 3-D reconstruction is not required [18][31]. Such an approach is particularly useful in closing control loops with vision at lower levels of a multi-level control system [1] (Figure 1). The limited body-centered data that are used are mapped into the local visual space for steering and speed control policies. A minimalist approach requires fewer calibrations, fewer scene hypotheses, and less computation. It is, therefore, simpler and faster.

The obstacle avoidance system we describe in this paper is designed in accordance with the Real-Time Control System (RCS) hierarchical architecture described in [1]. RCS decomposes goals both spatially and temporally to meet system objectives. The system monitors its environment with sensors and updates models of the states of the system itself and the world. Figure 1 describes the functionality of the obstacle avoidance system in the first three levels of the RCS hierarchy.

RCS is composed of three parallel legs, sensory processing (SP), world modeling (WM), and behavior generation (BG) that interact to control complex systems. The hierarchical levels run in parallel and are labelled, from highest to lowest: tribe, group, task, e-move (elemental-move), prim (primitive) and servo. The BG modules control physical devices. The WM modules supply information to both the BG hierarchy and the SP hierarchy. The WM modules maintain a database of system variables and filter and analyze data using support modules. The SP modules monitor and analyze sensory information from multiple sources in order to recognize objects, detect events and filter and integrate information. The world model uses this information to maintain the system's best estimate of the past and current states of the world and to predict future states of the world.

The testbed mobile robot is shown in Figure 2 (a).Video images are obtained from an onboard uncalibrated camera with a $115°$ field of view. The camera is mounted on a pan motor. The robot's view from this camera is shown in Figure 2 (b). Figure 1 sketches the obstacle avoidance system consisting of seven processing modules. The SP modules compute flow and divergence everywhere in the image. The WM modules apply spatial and temporal median filters to reduce momentary fluctuations in the divergence field. The BG



(a) Test mobile robot

(b) Robot's view of the lab through its 115-degree wide angle lens

Figure 2 Robot and Its View

modules use divergence to steer the robot around obstacles seen in the wide camera view and provide the body and gaze controllers with desired driving, steering, and gaze (in this case, simply pan) velocities. Using active gaze control, the camera is rotationally stabilized to reduce the magnitude of the flows in the image stream. When the camera points too far away from the heading, a saccade is made toward the heading. These saccades introduce momentary disturbances of the flow data, but the temporal median filter effectively eliminates disruptive effects. When divergence data indicate imminent collision ahead, the robot stops, turns away, and resumes wandering.

2 Full optical flow estimation

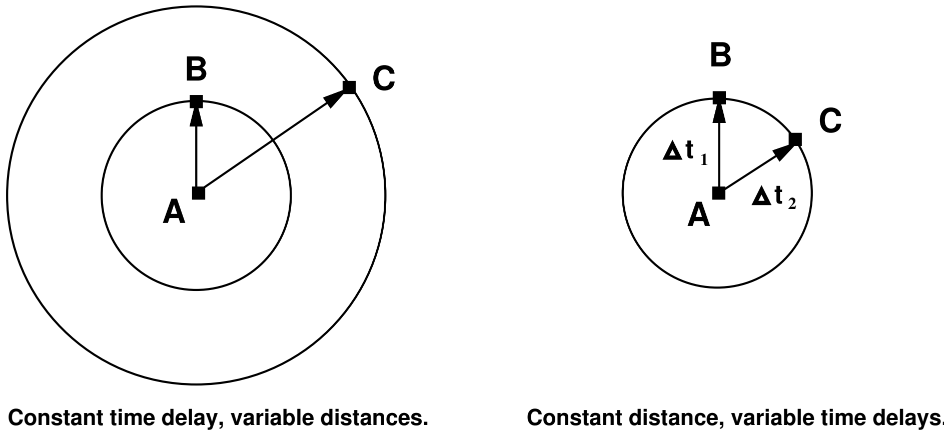
Robust, real-time optical flow is now becoming a practical means of robotic perception given new fast algorithms and increasingly faster hardware. Given that our entire system (flow, divergence, and body control) is implemented on a single workstation without the benefit of a real-time operating system, it is important to have sufficient processor idle time available to buffer the overhead of the operating system. Otherwise, the image capture frame rate could vary from frame to frame. Camus [7][8] describes a robust, real-time correlation-based optical flow algorithm which returns dense data even in areas of relatively low texture. This is the starting point of our implementation.

In correlation-based flow such as in [5][17][21], the motion for the pixel at $[x,y]$ in one frame to a successive frame is defined to be the displacement of the patch P_ν of $\nu \times \nu$ pixels centered at $[x,y]$, out of $(2\eta + 1) \times (2\eta + 1)$ possible displacements (where η is an arbitrary parameter dependent on the maximum expected motion in the image). 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 (such as the absolute difference between the two pixels' intensity values, E_1 and E_2 respectively), 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:

$$\forall(u, w) M(x, y;u,w) = \sum_{(i, j) \in P_\nu} \phi(E_1(i, j) - E_2(i + u, j + w)) \tag{1}$$

The actual motion of the pixel is taken to be that of the particular displacement, out of $(2\eta + 1) \times (2\eta + 1)$ possible displacements, with the maximum neighborhood match strength (or equivalently, minimum patch difference); thus this is called a "winner-take-all" algorithm.

This algorithm has many desirable properties. Due to the relatively large size of the matching window, the algorithm generally does not suffer from the aperture problem except in extreme cases [5][8], and it tends to be very resistant to random noise. Because 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 (except at motion boundaries). Therefore the resultant optical flow field tends to be relatively smooth, without requiring any additional smoothing steps. In gradient-based optical flow, on the other hand, noise usually results in direct errors in the basic optical flow measurements due to the sensitivity of numerical differentiation. Finally, because one optical flow vector is produced for each pixel of input (except for a small $\eta + \lfloor \nu/2 \rfloor$ border where flow may not be calculated), optical flow measurement density is 100 percent.



(a) As the maximum pixel shift increases linearly, the search area increases quadratically.

(b) However, with a constant shift distance and variable discrete time delays, search over time is linear.

Figure 3 Time Delays and Distances

A significant limitation with the traditional correlation-based algorithm is that its time complexity grows quadratically with the maximum possible displacement allowed for the pixel [5][9] (Figure 3 (a)). 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:

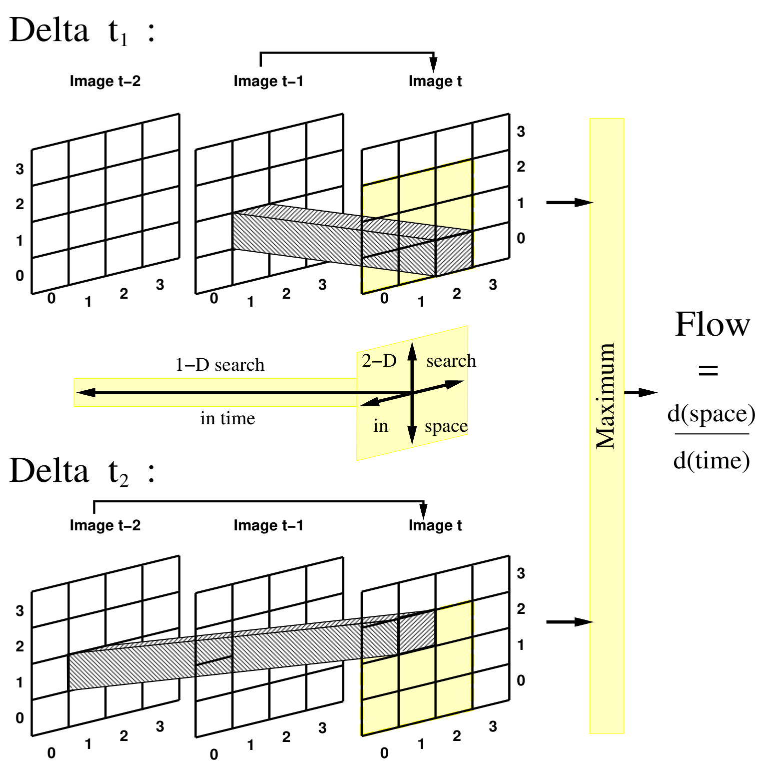
$$vel = \frac{\delta dist}{\delta time}.$$

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

$$\Delta \nu = \frac{\Delta d}{\delta t}, d \leq \eta .$$

However, we can easily see from Figure 3 (b) 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:

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



(a) The search area for the example pixel (1,1) in this example is shown in the current frame (Image t). This search begins from a previous frame (2 such previous frames are shown).

(b) The correlation match is the sum-of-absolute differences between the corresponding pixels' intensity values in a patch centered at the given pixel (a patch size of 7x7 is used in all examples; these patches are not shown for clarity).

(c) Optical flow is calculated to be the best-matching pixel shift divided by the corresponding frame delay.

Figure 4 Space-time correlation search of Camus' real-time optical flow algorithm

In this case, we generally prefer to keep δd as small as possible (for example a single pixel) in order to avoid the quadratic increase in search area. (Note, however, there is nothing preventing an algorithm based on both variable Δd and Δt.) This temporal matching concept is depicted in Figure 4. This time-space trade-off reduces a quadratic search in space into a linear one in time (at a cost of measurement precision), resulting in a very fast algorithm: optical flow can be computed on 32x64 images, quantizing flow speeds into 5 bins, at 35 frames per second on an 80 MHz Themis HyperSPARC[1] 10 computer. Although this algorithm is not as accurate as many other optical flow algorithms, it is generally superior in terms of computational efficiency [23].

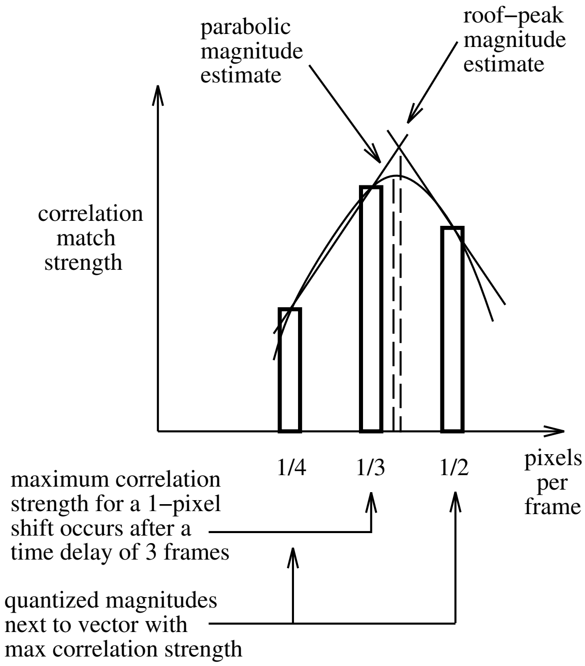
In our implementation, an original image of 256x512 pixels, captured with a 115° field-of-view camera, is subsampled to 32x64 pixels using a simple block-subsampling algorithm which averages an *N*x*N* block of pixels. This simple subsampling is very fast and

1. Certain commercial equipment, instruments, or materials are identified in this paper in order to adequately specify the experimental procedure. Such identification does not imply recommendation or endorsement by NIST, nor does it imply that the materials or equipment identified are necessarily best for the purpose.

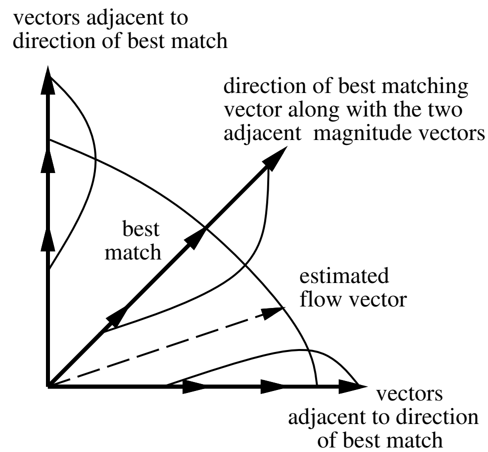
effective for this correlation-based algorithm (although perhaps not for other flow algorithms [8]). In this case, blocks of 8x8 pixels would be averaged into a single value. In practice, this requires more image bandwidth than is available using the framegrabber's VME bus connection to the Themis HyperSPARC 10 board's main memory. Therefore, the images are first subsampled by a factor of 2 on each side (yielding 1/4 the image data) using the framegrabber's own image scaling function. (This is the only function performed on the framegrabber other than capturing the images and transferring them to the host's memory.) Because the framegrabber's method of subsampling is not as effective as the one we use in software, the final subsampling by a factor of 4 on a side is performed on the host workstation itself.

The above algorithm returns *quantized* optical flow values. Although this is adequate for various robotic vision tasks [6][14][15][25], it is not sufficient for our application, because the calculation of divergence requires the ability to measure spatial derivatives of the optical flow. Because quantized optical flow is basically a step function, these derivatives do not exist. Consistent with theory, it was found that the divergence of the optical flow field could not be estimated successfully using only the quantized algorithm. Attempting to interpolate the optical flow field itself is not appropriate in this case because the quantizations are relatively coarse and the flow field is already dense. Smoothing the optical flow field would require extremely large masks and would, therefore, likely cover multiple objects simultaneously. This would be especially problematic in our case where a wide-angle lens is used and individual objects occupy only a small fraction of the visual field. Calculating a least-squares best fit to the correlation surface as in [3] was ruled out due to real-time performance requirements (see efficiency experiments in [22][24]). In order to satisfy our real-time requirements, a fast approximation method was used to derive continuously-valued flow fields.

To avoid a computationally expensive search for the true flow, the 2-dimensional interpolation is decomposed into two 1-dimensional interpolations; the first estimates the magnitude of a flow vector and the second estimates its precise angle. (The initial flow is returned as X and Y components. This is then converted to polar coordinates.) Both the directional component (consisting of one of eight possible directions or no motion) as well as the magnitude component (consisting of the time delay in frames) are quantized. The first one-dimensional interpolation is along the magnitude component of the flow. The



(a) Parabolic and roof methods



(b) Angular estimation of flow vector

Figure 5 Image flow interpolation

correlation match values for the best motion of a given pixel along with the match values for the "virtual" flow vectors of "adjacent" magnitudes in the same direction (i.e., of plus and minus one time delay in frames) are used. For example, in Figure 4 (c) assume that the shift of (0, 1) pixels with a delay of 2 frames (equivalently a motion of (0, 1/2) pixels per frame) has the best correlation match of any pixel shift. The correlation match values used for interpolation in this example are those that correspond to the pixel shifts of (0, 1) over 3 frames and (0, 1) over 1 frame, or "virtual" motions of (0, 1/3) and (0, 1/1) pixels per frame respectively. (If the peak correlation match magnitude is 1 pixel per frame, interpolation cannot be performed.) Roof interpolation is used to find the total time delay corresponding to the minimum correlation match value as shown in Figure 5 (a). Two lines are formed connecting the best correlation match strength and the match strengths corresponding to those two time delays which immediately bracket the former time delay. The negative value of the steeper of the two slopes is then substituted for the more gradual of the two. The abscissa of the intersection point of these two lines is taken as the new interpolated magnitude component of the flow.

The second interpolation is along the angular component of the flow (Figure 5 (b)). The first step is to calculate the match values for "virtual" flow vectors of neighboring vector directions but with the same magnitude as just calculated for that pixel. (Because we only calculate the correlation match values for eight directions of motion, this means that the motion vector with the best correlation match has two neighboring directions at $45°$ on either side.) This interpolation is not trivial because the magnitude of the motion along the diagonals is the $\sqrt{2}$ times that of motion along the north, south, east and west directions for a given velocity (time delay). In order to perform the second 1-dimensional interpolation, it is necessary to estimate the correlation match values of the neighboring directions at the *same* magnitude as the best matching flow vector. Although the roof interpolation was slightly more accurate than parabolic interpolation for finding a real-valued magnitude for a given optical flow vector, it was found to have the disadvantage of not returning as accurate a correlation match value estimate. A more accurate correlation match value estimate was instead found by calculating the coefficients of an interpolating parabola and taking the correlation match value at the same magnitude as found during the roof interpolation stage (Figure 5 (a)). The following formulas for the parabola coefficients can be derived given a parabola $ax^2 + bx + c$:

$$a = \frac{(y_2 - y_0) + (x_0 - x_2)(y_1 - y_0)/(x_1 - x_0)}{(x_2)^2 - (x_0)^2 + (x_0 - x_2)((x_1)^2 - (x_0)^2)/(x_1 - x_0)},$$

$$b = \frac{y_1 - y_0 - a((x_1)^2 - (x_0)^2)}{x_1 - x_0},$$

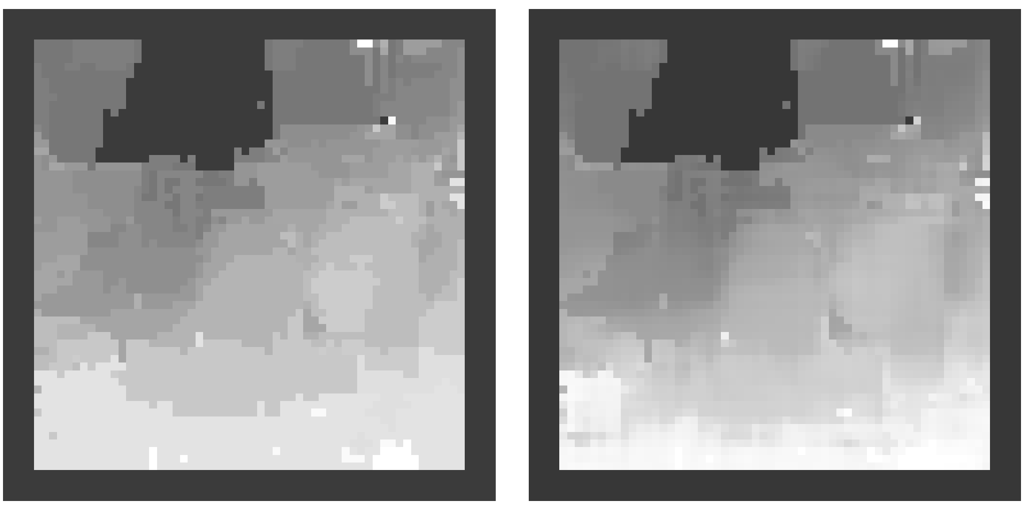
$$c = y_0 - a(x_0)^2 - bx_0,$$

where (x_0,y_0), (x_1,y_1) and (x_2,y_2) are the points used to fit the parabola.

These formulas are used to estimate the correlation match strengths for neighboring "virtual" flow vectors that correspond to the same magnitude as that of the best matching flow vector; these three parabolic interpolations are shown in Figure 5 (b). A final parabolic interpolation is then calculated using these three estimated correlation match strengths, and is also shown in Figure 5 (b).

(a) frame 25 of approach to two chairs

(b) frame 35



(c) quantized flow (pixel brightness indicates image motion magnitude)

(d) interpolated flow

Figure 6   Quantized and interpolated optical flow results

An example of the optical flow output of this algorithm is shown in Figure 6. The top part of this figure shows two 256x256 pixel images, spaced ten images apart, of a typical image sequence resulting from a robot's own forward motion. The bottom of this figure shows grey-level images indicating the magnitude of the optical flow vectors at each point; brighter pixels correspond to faster motion. Figure 6 (c) shows the output of the quantized optical flow algorithm on these ten images (subsampled to 64x64 pixels in size). This can be compared to the results of the interpolated optical flow algorithm, seen in Figure 6 (d).

The quantized optical flow field (Figure 6 (c)) contains a total of only 19 discrete motion magnitudes, whereas the interpolated optical flow field (Figure 6 (d)) yields a continuously-valued flow field, which, when quantized to 8-bits for display, contains 144 separate motion magnitudes. Although the original quantization boundaries are still visible in many places, this interpolated optical flow algorithm represents a greatly improved motion vector resolution over the quantized optical flow algorithm and will prove sufficient for our purposes. Also, note that in both cases the flow estimates are dense even on the relatively weakly-textured laboratory floor.

When compared to computing flow without interpolation, the use of two one-dimensional interpolations cuts the frame rate approximately in half: real-valued optical flow can be computed on 32x64 images, calculating and interpolating 5 speeds per frame, at 17 frames per second on an 80 MHz Themis HyperSPARC 10 computer. In our obstacle-avoidance application, the flow is run at only about 4 Hz. This consumes from 20-25% of the processor's total time and allows the entire obstacle avoidance system to run comfortably on a single workstation with a consistent frame rate and about a 20% processor idle time to buffer unexpected operating system events.

3 Spherical Pinhole Camera Model

Most authors assume a simple planar pinhole camera model in deriving equations. We will argue that the $115°$ wide-angle camera used in our system more closely approximates a spherical pinhole camera model than the traditional planar pinhole model. The local area at any point (x,y) on the spherical imaging surface can be approximated by a tangent planar surface with $(x,y) = (0,0)$ at the tangent point. In the next section, we will exploit this observation to use properties of image flow divergence which hold at $(x,y) = (0,0)$ in a planar pinhole imaging model. This tangent planar surface approximation extends the results for $(x,y) = (0,0)$ in a planar pinhole coordinate system to all points (x,y) in a spherical pinhole coordinate system, as long as all calculations performed are limited to a local area around the given (x,y).

An image from the $115°$ wide-angle camera appears in Figure 7 (a). This image is visibly distorted, as expected with a wide-angle camera. To show that the wide-angle camera more closely approximates a spherical pinhole camera model, we warp the wide-angle im-

(a) Actual wide-angle image.

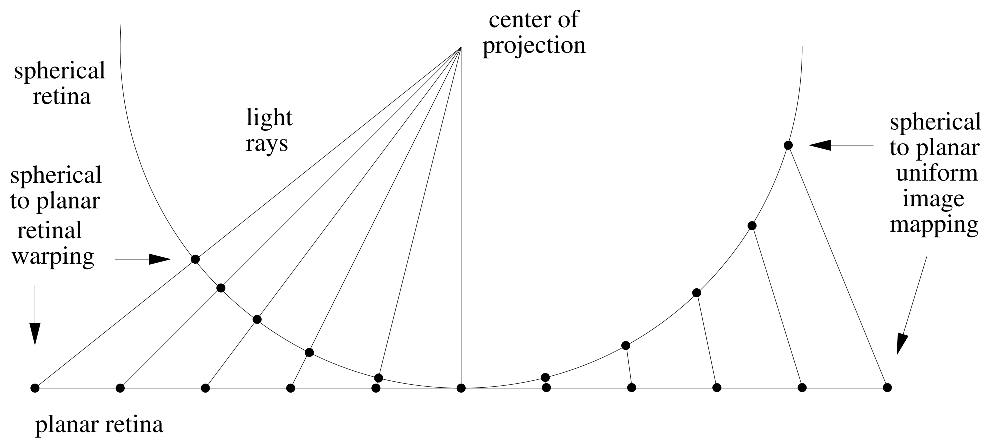
(b) Same image warped to a planar image assuming that the wide-angle camera approximates a spherical pinhole camera model.

Figure 7 Wide-angle Optics

age into a planar pinhole camera image under the spherical pinhole camera assumption. If this assumption is true, the warped image should appear relatively flat and undistorted.

There are two steps to this process. First, we transform a spherical pinhole camera retina into a planar pinhole camera retina, *i.e.*, we warp an image created from a spherical pinhole camera into an equivalent image as it would appear if it were created from a traditional planar pinhole camera; this is shown in Figure 8 (a). However, we are not given true spherical retinas as imaging surfaces. We are given flat, distorted images. Thus, prior to this transformation, we must convert the flat, distorted image into its equivalent spherical retinal representation, as shown in Figure 8 (b). Note that this latter mapping is uniform and is equivalent to simply "rolling" a flat planar representation onto a spherical retina. Thus, the complete sequence involved to map a spherical pinhole model image (presented in the form of a flat, distorted image) is to take its spherical imaging surface equivalent (shown in Figure 8 (b)) and map it to its planar imaging surface equivalent (shown in Figure 8 (a))

The result of this mapping/warping process on Figure 7 (a), using a focal length equivalent of 150 pixels (given an image width of 256 pixels), results in Figure 7 (b), which is clearly much less distorted than the former image. From experiments such as these, we conclude that our 115-degree wide-angle camera is better approximated by a spherical pin-



(a) Above left: warping a spherical pinhole camera retina to a planar pinhole camera retina as a function of equal angle from the center of projection

(b) Above right: uniformly mapping a spherical pinhole camera image to a flat image as a function of equal linear distance from the center of the imaging surface

Figure 8 Spherical projection camera model

hole camera model than a more traditional planar pinhole camera model. As described previously, this fact is exploited in the next section.

4 Divergence for Obstacle Avoidance

Process 2 (Figure 1) computes flow divergence. Divergence of flow can be used to estimate time-to-contact (T_c). Both theory and implementation are discussed here as well as considerations for employing T_c for obstacle detection by a moving robot.

The equations for the x and y components of optical flow (O_x, O_y) due to general camera motion (arbitrary translation and rotation) in a stationary environment given a planar projection coordinate system are

$$O_x = (1/z)(-T_x + xT_z) + (xy\omega_x - (1 + x^2)\omega_y + y\omega_z) \quad (2)$$

$$O_y = (1/z)(-T_y + yT_z) + ((1+y^2)\omega_x - xy\omega_y - x\omega_z) \tag{3}$$

where z is the depth of the object in the environment relative to the camera, and (T_x, T_y, T_z) and $(\omega_x, \omega_y, \omega_z)$ are the translational and rotational motion of the environment relative to the camera [35]. This camera frame is a right-handed coordinate system centered behind the image plane with the positive *X* direction extending leftward, positive *Y* upward, and positive *Z* forward. The divergence of an optical flow field (parameterized by image coordinates (x,y) is defined as:

$$\nabla^\circ(O_x, O_y) = \frac{\partial O_x}{\partial x} + \frac{\partial O_y}{\partial y}. \tag{4}$$

Note that

$$\frac{\partial O_x}{\partial x} = \frac{\partial \rho}{\partial x}(-T_x + xT_z) + \rho T_z + y\omega_x - 2x\omega_y \tag{5}$$

$$\frac{\partial O_y}{\partial y} = \frac{\partial \rho}{\partial y}(-T_y + yT_z) + \rho T_z + 2y\omega_x - x\omega_y \tag{6}$$

where $\rho = 1/z$. From equations (4) through (6), at $(x,y) = (0,0)$:

$$\nabla^\circ(O_x, O_y) = \frac{\partial \rho}{\partial x}(-T_x) + 2\rho T_z + \frac{\partial \rho}{\partial y}(-T_y) \;. \tag{7}$$

This can be rewritten in terms of the gradient $\left(\frac{\partial z}{\partial x}, \frac{\partial z}{\partial y}\right)$ as

$$\nabla^\circ(O_x, O_y) = \frac{2T_z}{z} + \frac{1}{z^2}\left(\frac{\partial z}{\partial x}, \frac{\partial z}{\partial y}\right) \cdot (T_x, T_y) \tag{8}$$

$$\nabla^\circ(O_x, O_y) = \frac{2T_z}{z} + \frac{1}{z}F \cdot (T_x, T_y) \tag{9}$$

where $F = \frac{1}{z}\left(\frac{\partial z}{\partial x}, \frac{\partial z}{\partial y}\right)$ is the surface gradient [10]. As stated, this equation holds for a *planar* coordinate system at $(x,y) = (0,0)$. Because we are modeling our wide-angle camera using a spherical coordinate system, the spherical imaging surface at every point (x,y) can be approximated by a tangent planar surface with $(x,y) = (0,0)$ in that tangent plane's local coordinate system. A precise derivation of the consequences of this assumption is future work. The local *Z* axis is the ray through the point (x,y). Equation (9) now holds for every point in the image of the wide-angle camera that is well-approximated by a spherical pinhole camera model. From equation (9),

$$\nabla^\circ(O_x, O_y) = \frac{2T_z}{z} \tag{10}$$

whenever the imaged surface is perpendicular to the local Z axis, or the gradient of the surface, F, is perpendicular to the transverse velocity (T_x, T_y), or the transverse velocity is zero.

As will be explained below, we use relatively large image masks for computing divergence. In our lab experiments, these masks, when projected out into the 3-D scene, usually cover a scene area 1 to 4 square meters. There are often many surfaces, at different orientations, in such an area in our scenes. We believe that when the surface gradients, expressed in the local camera coordinate system, are averaged in such an area, the average gradient tends to be very small. Given two (nearly planar) surface segments of the same size in 3-D space, the segment with large gradient (i.e., steep slope in the local coordinate system) will project into a much smaller area of the image than the segment with small gradient (i.e., which is nearly perpendicular to the line of sight). Therefore, surfaces with small gradients should tend to contribute more to the "average surface gradient" within the mask area than surfaces with large gradients. Because of these effects, the first term in the right-hand-side of equation (9) usually dominates the second term. For surfaces near the direction of motion, (T_x, T_y) is small. This further reinforces the domination of the first term.Therefore, we can estimate time-to-contact directly from equation (10).

Divergence is a particularly useful measure for obstacle avoidance during visual navigation because it is invariant under the rotational motion of the sensor that is inevitable due to imperfect stabilization.

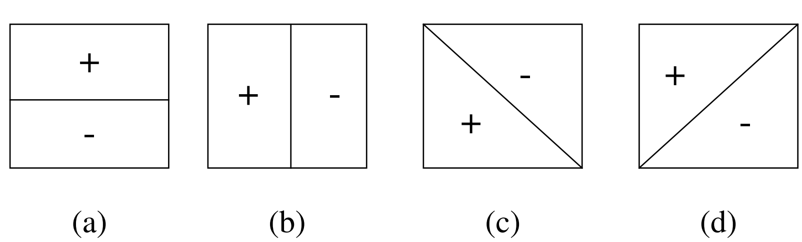


Figure 9 Flow divergence templates

Equation (10) suggests that *divergence* has only time as its dimension. The values of *divergence* over any significant area represent the inverse of time-to-contact, i.e., the time needed to reach an object at depth Z with velocity T_z in the Z direction. A family of simple fixed flow divergence templates can be applied to any image sequence to estimate divergence [28]. Each template is symmetrically divided into positive and negative halves (Figure 9). Flow divergence is calculated by convolving the template with a window in the flow image and computing the sum of the optical flow derivatives in perpendicular directions. In particular, the convolution of the first two such templates may be performed extremely quickly using a *box filter* as described in [7][8]. In order to improve the consistency of the divergence estimates, we apply temporal and spatial median filters to the individual divergence values (process 3 in Figure 1).

5 Simple Robust Filters

Median filtering performed on dense two-dimensional data can use fast running-histogram methods such as in [19] if the dynamic range of the data and desired quantization resolution of the median value can be specified. The algorithm described in [19] was intended for finding the true median and reduces an *O(nm)* complexity algorithm to approximately *O(n)* per pixel for an $n \times m$ filtering window, where $n < m$. It can, however, be generalized to the separable median [26] which approximates a two-dimensional true median filter by the successive application of two orthogonal one-dimensional median filters: first the median of each row of a two-dimensional window is computed, and then, the median of these row-medians is returned as the separable median. This transformation reduces the time complexity from *O(n)* for the running-histogram method alone to approximately constant time for the combined method. To the authors' knowledge, this is the first time that this combined, approximately constant time median filtering algorithm has been used in image analysis. Because of this, its utility will be explored further.

Exploiting this combined approach assumes that there are only a limited number of histogram bins, which would not be the case with floating-point data. In that case one could modify the algorithm to first quantize the data into a histogram of 256 bins (for example). The true median could then be found by performing a quicksort-partitioning [30] on that bin which contains the midpoint of the histogram for that pixel's local window. In our case, however, the divergence data are already quantized to 8-bit values (necessary in order to reduce the data bandwidth in the previous system's implementation [11]) so this extension is not necessary.

Although the separable median filter is not guaranteed to find true median values, it reduces noise almost as well as a true median filter [26]. By definition, the true median is greater than or equal to half the data in the data set and less than or equal to the other half of the data. The separable median, however, is greater than or equal to all points less than or equal to those row-medians which are less than or equal to the separable median. By definition, the separable median is greater than or equal to 50% of the row medians, therefore, the separable median is greater than or equal to 25% of the data. Similarly, the separable median is guaranteed to be less than or equal to 25% of the data. Its relationship to the rest of the data is unknown.

The "breakpoint" of an estimator is defined as the smallest fraction (or percent) of contamination of a data set that can result in an arbitrarily incorrect value [32]. In the case of least-sum-of-squares (a non-robust estimator), that percent tends to 0% for large n (i.e., a single outlier can cause an arbitrarily incorrect result). In the case of the true median filter, that percent is 50%. In the case of the separable median filter, that percent is 25%, given our arguments above. Although this means that the separable median filter is not as robust as the true median filter, it still performs extremely well in most cases.

The current system first performs an 11×17 (height by width) spatial median filter on the divergence estimates (The filter's proportions roughly paralleled the proportions of the 256×512 image field). Because our robot operates in a 2-D world (*i.e.,* its motion is constrained to two degrees of freedom: speed and steering), we use the 11×17 filter to produce a single horizontal array of divergence values, which is centered on the middle row of each image.

The 1-D divergence arrays are stored in a history buffer of 1-D arrays of spatially-filtered divergence estimates. The dimensions of the temporal filter are 11 pixels in space (horizontally within each array) and 11 frames in time (across arrays). Because of the temporal median filter's width, the filtered divergence information has a latency of 5 frames or about 1.25 seconds. This effect is offset, however, by the ability of the system to sense obstacles at a range of up to several meters.

Both the separable and true median filters have the desirable property of preserving horizontally and vertically aligned edges. This means that unlike many other averaging or smoothing filters, these filters have no temporal hysteresis. Unlike the true median filter, however, the separable median filter has the additional desirable property of preserving corners [26]. Preserving corners can be advantageous in ordinary image processing but is especially valuable in our application, because an erosion of an object's full spatial or temporal extent could create the illusion of an open space and cause a collision. This corner preservation can also prevent distinct objects from blurring together [29], which could be equally undesirable because some valid pathways may otherwise falsely appear to be blocked.

When performing the separable temporal median filter, the order of the two 1-D median operations can make a difference in some cases [29]. By performing the spatial 1-D median filter first, we slightly emphasize spatial coherence over temporal continuity.

6 Driving Control

Processes 4 and 5 (Figure 1) control driving and gaze. The robot's task is to avoid obstacles while achieving mobility goals. In general, such goals might be specified by coordinates in a map, features that uniquely identify a location, or simply features that satisfy a precondition required for the next subtask (*e.g.*, the mobility goal might be positioning the robot to pick up an object.) Ideally, the robot would survey the visual data to identify the direction nearest its desired path that is also a safe direction in which to travel.

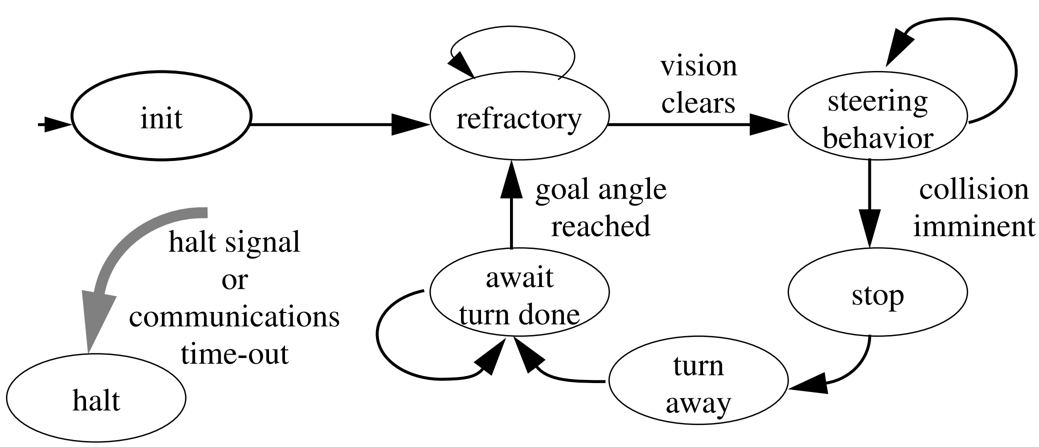


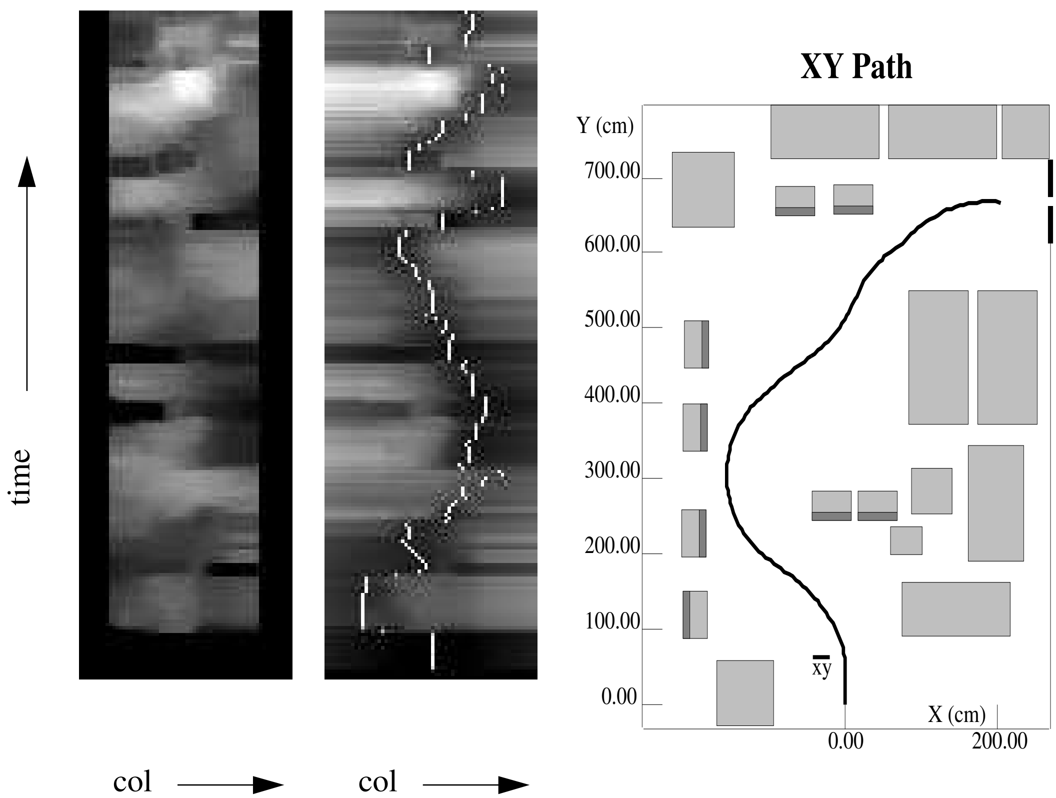
Figure 10 Body control automaton

In these experiments, the goal is to maneuver without collision using only flow divergence to sense the environment. The robot's behavioral goal is simply to drive forward, steering away from obstacles in its path, and to stop and turn when it senses that collision is imminent. The robot drives at up to 20 cm/s. Speed is regulated to keep visual data (*i.e.*, flow and divergence estimates) within measurable ranges and to avoid detrimental behavior that results when high rates of speed and steering occur simultaneously. The steering policy uses the sensed flow divergences to steer around obstacles while attempting to steer toward the provided goal direction. (In these experiments, the goal direction was always straight ahead for simplicity.) Indication of imminent collision in the central quarter of the camera's divergence data causes the robot to stop, turn away and resume wandering. This sequencing is implemented with a finite state automaton, with a command associated with each state (Figure 10). Some state transitions are triggered by sensed events, and others merely provide command sequencing. For instance, the system remains in the refractory state until visual motion lingering from the brisk turn is no longer seen by the visual system. An interesting consequence of this behavior is that the robot in the refractory state will remain still while a person moves in its view.

The robot steering and collision detection improve when the robot turns relatively slowly for two reasons: (1) temporal consistency of the spatial locus of samples for divergence estimation is improved, and (2) accuracy of motion estimates themselves is improved. Therefore, the behavior and motor control systems are designed to reduce rotation of the cameras. This is accomplished by stabilizing the cameras with active motor commands and by limiting rotation of the body so the gaze stabilization system is not overstressed. Despite these precautions, gaze stabilization is imperfect and some data are contaminated. However, the edge-preserving spatio-temporal median filtering effectively discards intermittent unusable data

The steering policy is implemented using hazard maps derived from flow divergence, the desired goal direction, and the target heading θ' that was previously selected by the steering policy. Each hazard map is a 1-dimensional horizontal array that encodes the "risk" associated with each possible steering direction. Traces of median-filtered flow divergence maps and composite hazard maps are shown in Figure 11 (b) and (c). The target heading selected in each hazard map is highlighted in white. The path of the robot in this

(a) Robot's view of the gauntlet of office chairs before the trial.



(b) Trace of median-filtered flow divergence maps (brighter intensity indicates larger divergence).

(c) Trace of hazard maps with selected goal direction highlighted in white.

(d) XY path trace beginning at (0,0).

Figure 11 Trace of Robot's View

trial appears in (d) for the gauntlet of office chairs seen in (a) from the robot's viewpoint before the trial began.

A hazard map is derived from the divergence map. During each steering control cycle it encodes not only the present divergence data, but also the past steering choices of the robot to help the robot commit to passing an obstacle on one side without switching back and forth. The hazard map is a 64-element wide map per sample interval that indicates obstacles and also encodes the cost of crossing local maxima in the divergence map, starting from the previously selected heading. Imagine the divergence map as a cross-section of a topographic map, such as the divergence map trace of Figure 11(b). Obstacles are indicated by hills and open spaces by valleys. In the previous cycle, the steering policy selected a heading value and committed to one of the valleys in the divergence map. In the present cycle, a different valley may appear deeper, but the cost of changing course to pass on the other side of an obstacle that has not yet been cleared is represented by the cost of crossing the hill to reach the valley on the other side.

Similarly, another hazard map is derived from the desired goal direction and the previously selected heading (accounting for the gaze angle). This map appears roughly as a shallow trough centered midway between the previously selected heading and the goal heading, which has the effect of drawing the selected heading back to the goal direction in the absence of obstacles in this path.

These hazard maps are combined into a single hazard map by summing the component hazard maps. The steering policy chooses the direction of minimum hazard in the composite hazard map, with a preference for directions nearest the previously selected heading in case of a tie. The result in general is that if *any* sensing mode shows strong evidence of danger in some direction, it is unlikely that direction will be chosen.

Summarizing, the image-based heading ψ is a heading angle in image space, the gaze angle ϕ is relative to the robot's current heading θ, and from these the new target heading θ' is calculated.

$$\theta' = \theta + \phi + \psi \tag{11}$$

When a new desired heading θ' is chosen, the robot steers smoothly to it with saturated negative feedback controls [13]. Desired change in heading, $\Delta\theta$, is simply the difference between the desired heading and current heading.

$$\Delta\theta = \theta' - \theta \tag{12}$$

The steering control policy uses a saturated steering velocity proportional to the desired heading.

$$\dot{\theta} = Saturate\left(k_s \cdot \Delta\theta \cdot \frac{1}{T_b}, s\right) \tag{13}$$

The gain k_s (usually <1) determines how quickly the steering is servoed to the desired heading. Time is normalized to seconds by dividing by the body control cycle time, T_b. Thus angular velocity is expressed in degrees/s rather than degrees/cycle. For instance, setting $k_s = 0.3$ will command a velocity that would reduce the error by 30% in the next control cycle (assuming nearly instantaneous acceleration of the steering motor). The angular velocity is saturated at ±*s deg/s* (*e.g.*, 6 *deg/s*) to limit the peak rotation rate to reasonable levels. There are three reasons for this limit. First, the latency of robot command execution is quite large, and the command cycle is not entirely uniform. Therefore, it is possible to overshoot the desired heading if the rotational velocity is too high, because the controller might not be able to stop steering at the right time. Second, motion estimation suffers if the camera rotates too fast, because our computation of flow is based on a correlation method which limits motion to less than one pixel per frame. Poor flow measurements degrade divergence estimates. This impacts the quality of steering and stopping. Third, slower steering improves the temporal consistency of flow data for divergence estimation. Although the median filtering is quite robust to disturbances, cleaner data produce smoother estimates over time. The second and third issues would be of no concern if gaze were perfectly stabilized, but stabilization is not perfect, and the residual camera rotation is correlated with the steering rate.

Like steering, the robot's speed is regulated to maintain all systems within their operating ranges. The robot's top speed is set at 20 cm/s based on empirical trials in showing that divergence estimation was compromised at faster driving speeds when the robot steered at the rates necessary to negotiate the lab. (The crash tests reported later in this paper were run at speeds up to 80 cm/s, but these tests used no steering.) Impairment of divergence estimation was proportional to steering rate. Below the top speed, the robot attempts to keep visual data within measurable ranges. Robot speed is increased when observed flows are too near noise levels, and speed is decreased when flows are too large for the estimation system to observe. Specifically, the speed is increased to keep the maximum observed flow estimate above a set value (50% of the maximum flow estimate that can be detected by the system). This ensures that most flows are large enough to be within the operating range of the flow estimator. Similarly, robot speed is decreased when too many flow estimates (50%) are saturated (have the maximum value the estimator can detect).

7 Gaze Control

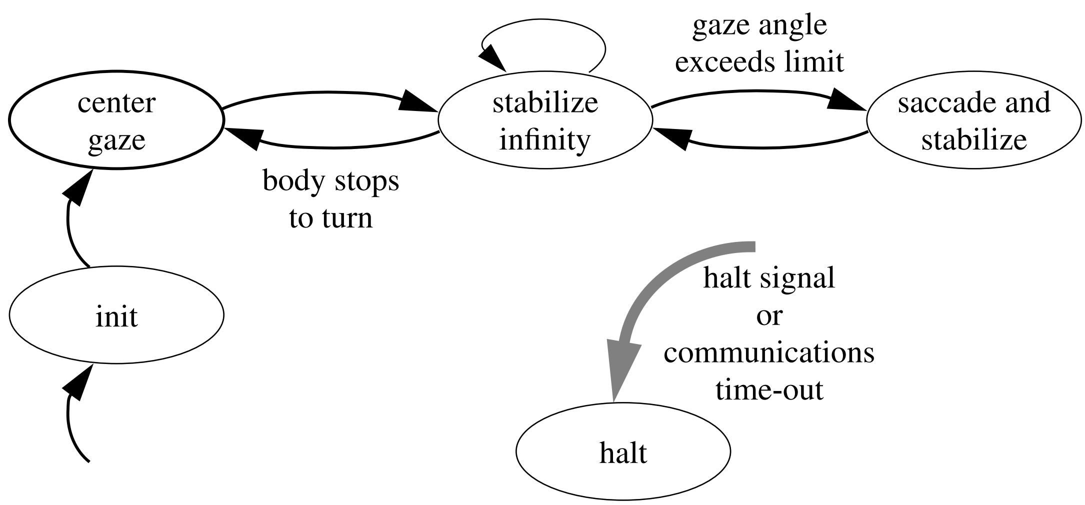


Figure 12 Gaze control automaton

The nonlinear gaze control is a *nystagmus*, a repetitive eye motion of slow phase rotations punctuated by quick phase rapid returns. It is also implemented as a finite state automaton (Figure 12). The camera is rotated at velocity $\dot{\phi} = -\dot{\theta}$ to counter the body rotation and stabilize the camera images. The gaze control also checks the deviation of the gaze angle, ϕ, from the robot's heading and snaps the camera back to the heading if the limit is exceeded (limit = 12^o).

The nystagmus quick phase that returns gaze to the vehicle's heading turns the gaze beyond the current heading in the direction the robot is turning: $\phi = k_g\dot{\theta}$. This heuristic attempts to reduce the number of quick phases required by putting gaze a little bit ahead of the vehicle heading, rather than always lagging behind it.

The saccades that perform the quick-phase return to realign gaze with the robot's heading briefly produce extremely large optical flows. These large flows often are encountered by the flow estimator. Although the resulting divergence estimates are unusable, the edge-preserving spatio-temporal median filtering effectively discards them, providing only the divergences observed preceding and following the saccade.

8 Experiments and Results

Experiments with the obstacle avoidance system were performed in a laboratory containing office furniture, robot, and computing equipment. Furniture and equipment lined the walls, and free space was roughly 7 m by 4 m in the center of the lab. Office chairs provided obstacles. In addition, there was some space leading to doors in two corners of the lab. In all experiments, a single camera with a $115°$ field of view was used. Only the half height band in the center of the image was processed. (See Figure 11 (a) for an example of the robot's view of the lab.) Three sets of experiments were performed: (1) "crash tests" evaluated the system's ability to detect obstacles and warn of imminent collision; (2) the robot was forced to run a gauntlet bristling with office chairs to evaluate its ability to avoid obstacles while traveling through the room; (3) wandering trials tested the robot's ability to move about for extended periods of time.

8.1 Crash tests

(a) frame 0

(b) frame 30

(c) frame 60

(d) frame 70

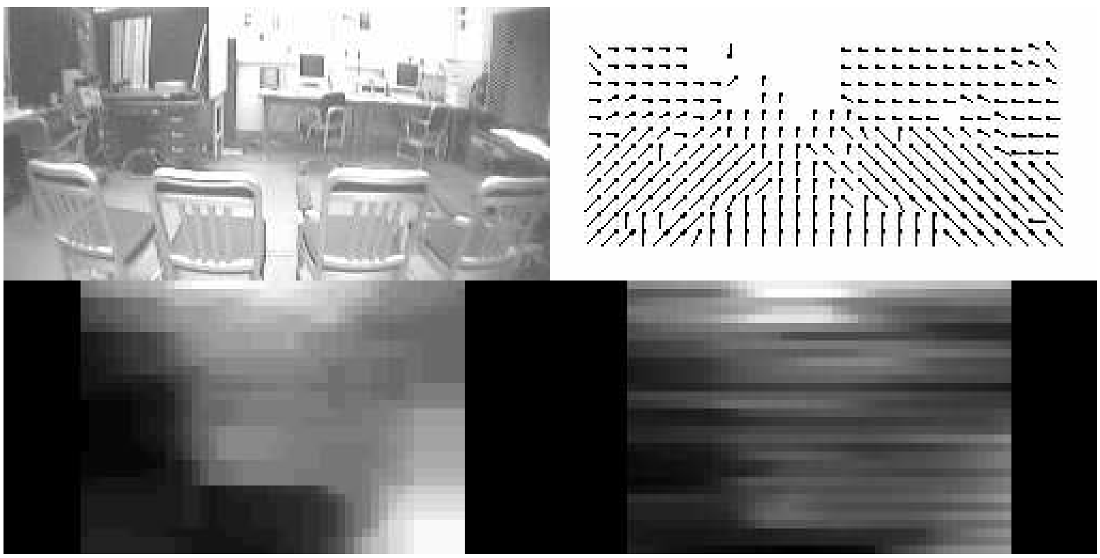
Figure 13 Frames from a 20 cm/s approach to chairs 4 meters distant

Initial experiments tested the robot's ability to detect obstacles and warn of imminent collision. To test the robot's ability to detect objects across the wide visual field, a row of chairs was placed across the far end of the lab. The robot drove straight toward the chairs at fixed speeds (see Figure 13). The system should not only detect obstacles at a reasonable distance, but should also continue to detect obstacles as they are approached until they are no longer visible in the bottom of the image. To evaluate this property, the robot did not attempt to stop to avoid collision, but rather drove into the chairs until the bumper triggered the systems to shut down due to collision.

Examples are shown in Figure 14 (20 cm/s), Figure 15 (30 cm/s), Figure 16 (40 cm/s), and Figure 17 (50 cm/s), all beginning at a distance of 4 m from the chairs. The upper left quarter (a) of each frame is the original 256x512 pixel image, taken from a 115° wide-angle lens, subsampled to 128x256 pixels. Results of processing the image are shown clockwise (b-d). The upper right quarter (b) shows the optical flow "needle" plot calculated from a 32x64 subsampled version of the original image with the needle diagram itself subsampled to 16x32 for clarity. The lower right quarter (c) of each frame shows a time-

(a) intensity frame 64: original 256x512 pixel image, taken from a 115° wide-angle lens, subsampled to 128x256 pixels

(b) optical flow “needle” plot: calculated from a 32x64 subsampled version of the original image, with the needle diagram itself subsampled to 16x32 for clarity



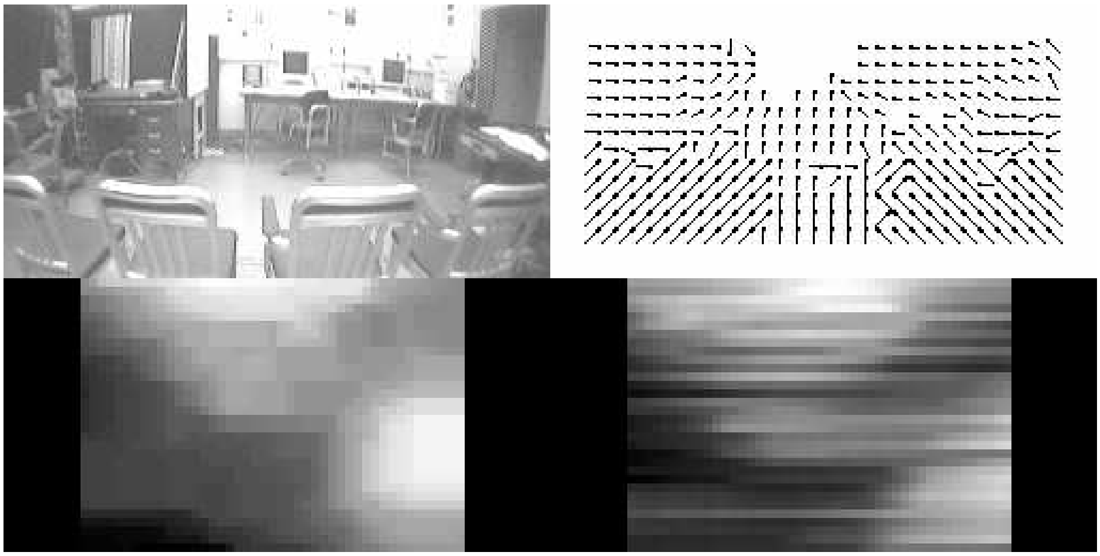
(d) median-filtered divergence: the same divergence plot but median filtered in time (11 frames) as well as in space (11 pixels). This filtered 1-D map of divergence estimates is used at each frame to control the robot’s steering.

(c) divergence plot (top row is current): time-space plot of the 1-D divergence estimates, median-filtered in space (11x17 pixel window centered on the middle row of each image). Brighter areas indicate higher divergence and thus closer objects (automatically contrast-adjusted for better clarity).

Figure 14 Frame 64 of a 20 cm/s approach to chairs 4 m distant

space plot of the 1-D divergence estimates, median-filtered in space (11x17 pixel window centered on the middle row of each image) but not in time. In these plots, brighter areas indicate higher divergence and thus closer objects (each individual plot is automatically contrast-adjusted for better clarity). In these plots, time moves upward. Thus, the 1-D divergence map for the current frame is found at the top of the sub-image with the 31 previous frames displayed in addition. The lower left quarter (d) of each frame shows the same divergence plot but median filtered in time (11 frames) as well as in space (11 pixels). The second filtered 1-D map of divergence estimates is used at each frame to control the robot’s steering. In the examples, the divergence due to imminent collision with the chairs can be clearly seen. The divergence due to a nearby stepladder and some cables that were suspended from the ceiling on the right side of the lab (visible in Figure 11 and Figure 13)

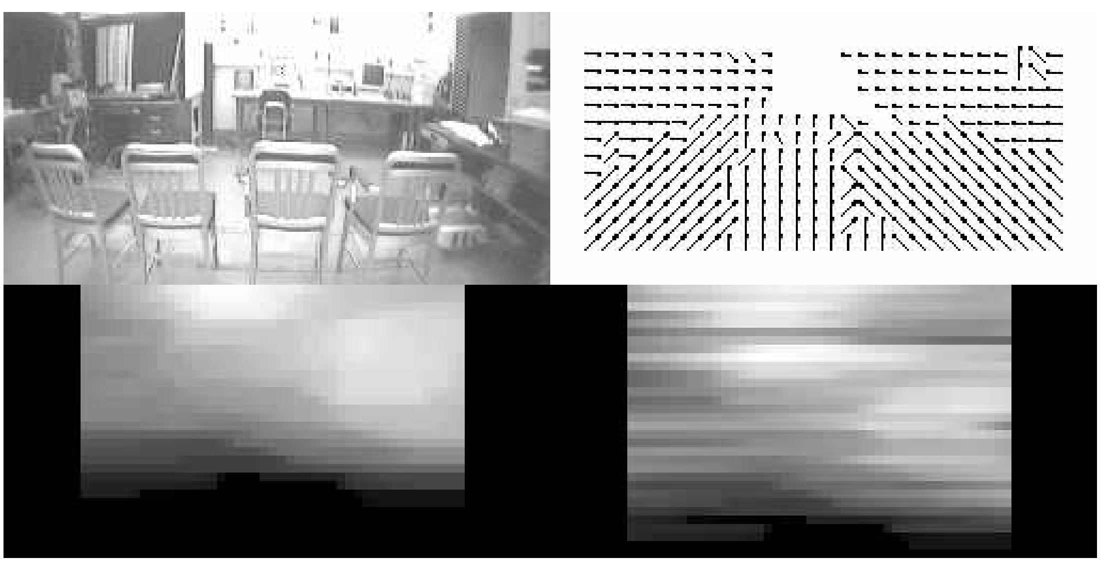
(a) intensity frame 49 (b) optical flow “needle” plot



(d) median-filtered divergence (c) divergence plot (top row is current)

Figure 15 Frame 49 of a 30 cm/s approach to chairs 4 m distant

(a) intensity frame 36 (b) optical flow “needle” plot

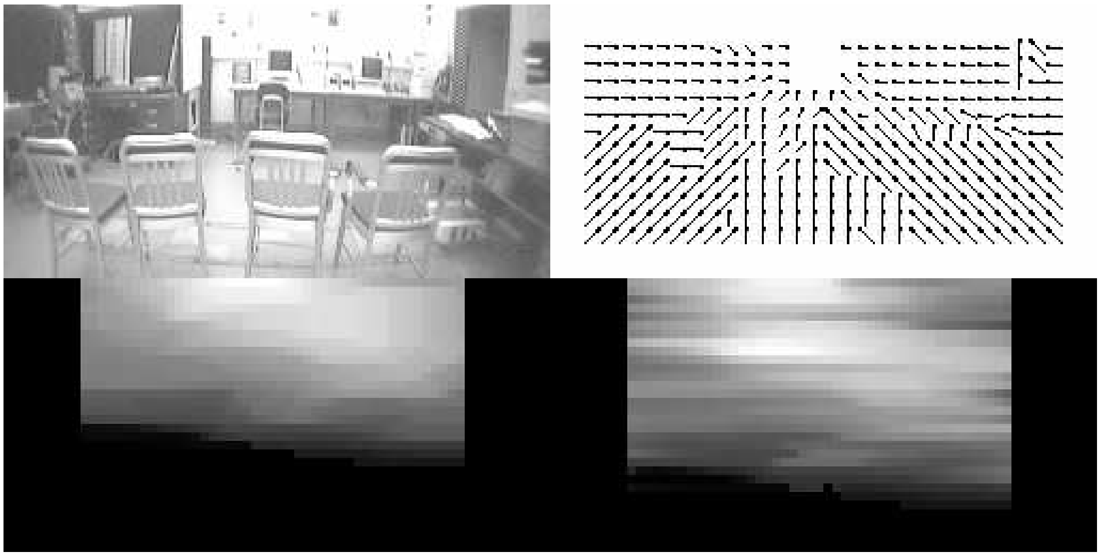


(d) median-filtered divergence (c) divergence plot (top row is current)

Figure 16 Frame 36 of a 40 cm/s approach to chairs 4 m distant

can also be seen in the history of the time-space divergence plots. Because of the temporal median filter’s width, the filtered divergence information has a latency of 5 frames or

(a) intensity frame 30 (b) optical flow "needle" plot



(d) median-filtered divergence (c) divergence plot (top row is current)

Figure 17 Frame 30 of a 50 cm/s approach to chairs 4 m distant

about 1.25 seconds. As shown in the figures, however, this effect is offset by the ability of the system to sense objects well in advance.

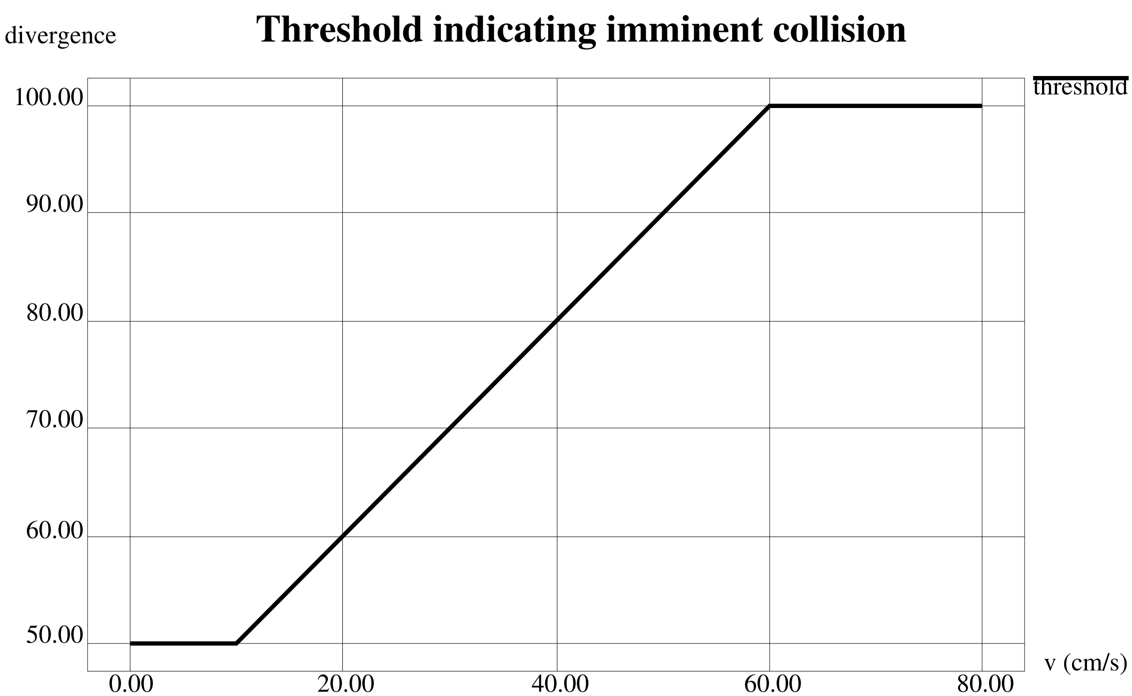
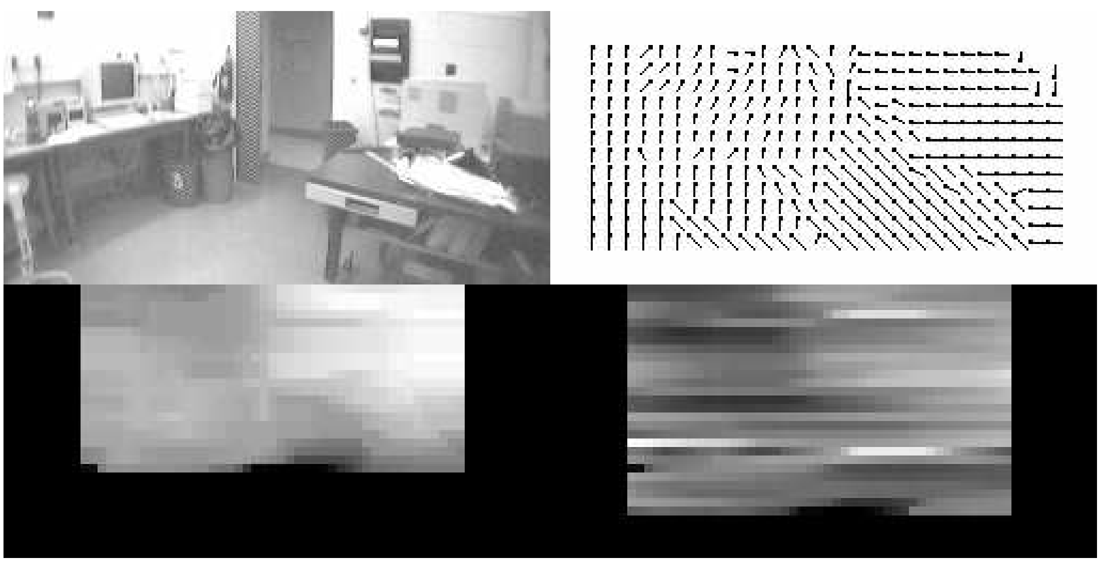


Figure 18 Space collision threshold as a function of robot speed

The system is able to detect objects (at divergence estimates above the noise level) at ranges up to 6 m (the maximum testable in the lab) at forward speeds ranging from 0.1 to 0.8 m/s. The divergence estimate arising from an object rises reasonably smoothly as the object is approached, and the object continues to be visible until the robot is very near to it. These results represent a considerable improvement in both range and persistence of detection over the results reported in [11], in which objects were detected in the narrow range of 1 to 2.5 m from the vehicle. Based on these trials, an imminent collision function was derived for robot speeds up to 0.8 m/s (Figure 18).

8.2 Gauntlettrials

(a) intensity frame 28 (b) optical flow "needle" plot

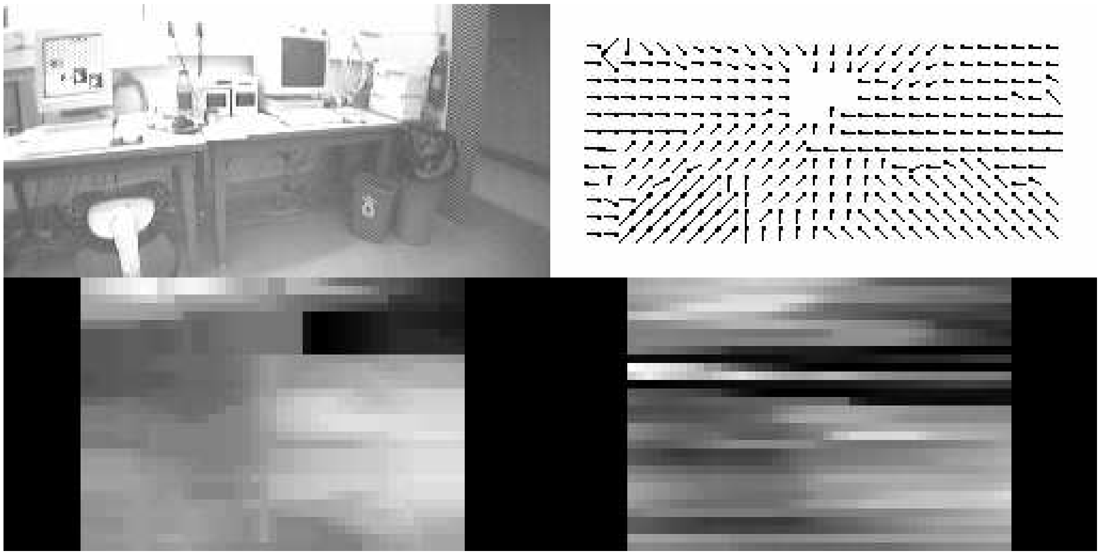


(d) median-filtered divergence (c) divergence plot (top row is current)

Figure 19 Frame 28 of a 20 cm/s obstacle-avoidance test run

The robot ran a gauntlet of office chairs to demonstrate the system's ability to avoid obstacles while traversing the lab. The lab setup and results for a run are shown in Figure 11. The robot deflected left to avoid the chairs blocking its path and then continued traversing the room, deflecting to the right to avoid the opposite row of chairs. After deflecting right, the robot's path was partially blocked by the end of a desk, as shown in Figure 19, causing the robot to deflect left. After the robot deflected left, its closest obstacle was a chair as shown in Figure 20. The robot then deflected right and headed toward the door

(a) intensity frame 43 (b) optical flow "needle" plot



(d) median-filtered divergence (c) divergence plot (top row is current)

Figure 20 Frame 43 of a 20 cm/s obstacle-avoidance test run

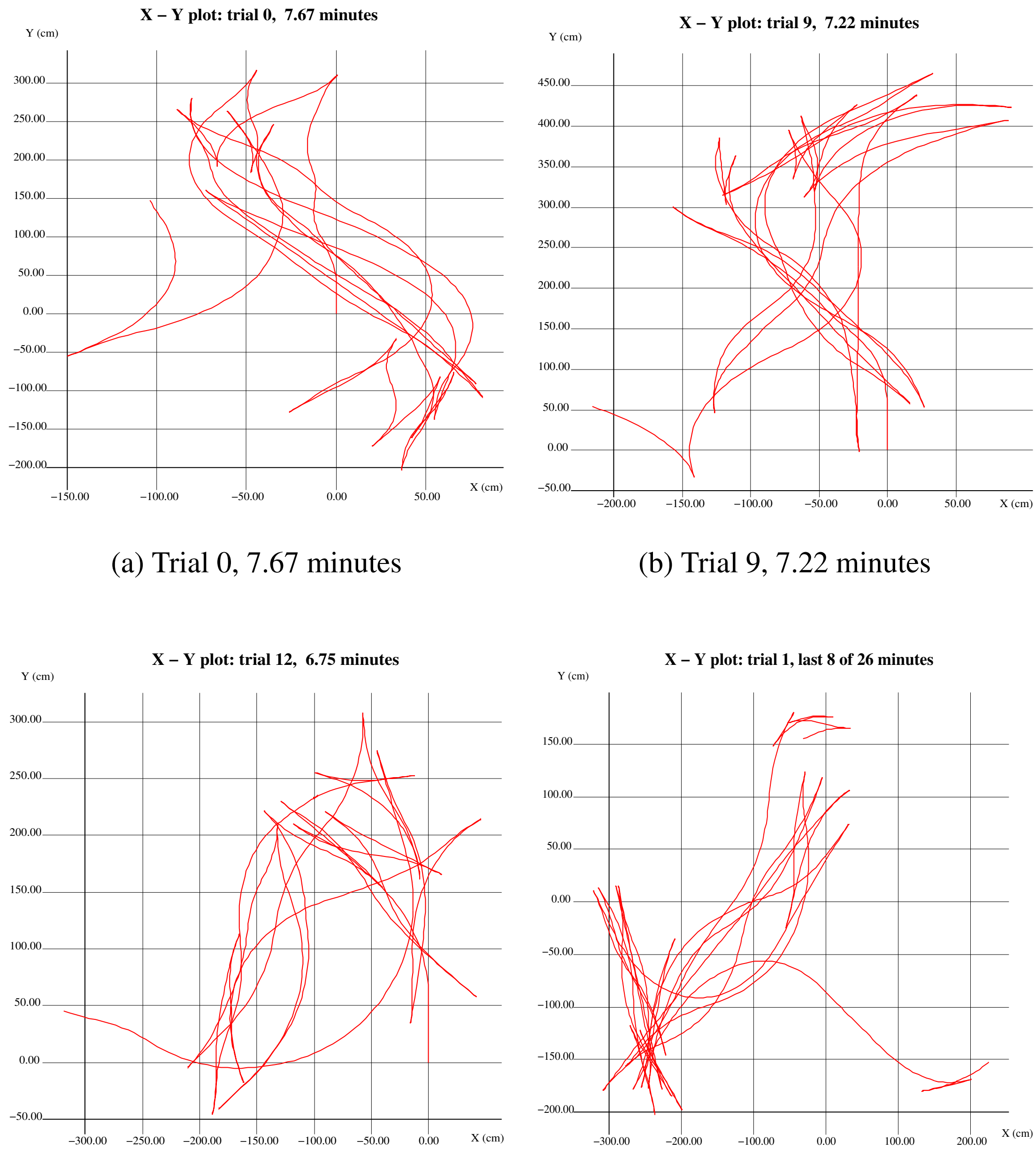
(the robot's complete path is shown in Figure 11 (d)). In these trials, the robot traveled at 20 cm/s and steered at a maximum rate of 8 degrees.

Gaze stabilization contributed considerably to the effectiveness of the system by reducing the magnitude of the optical flows while the robot was steering. In control trials without gaze stabilization, analysis of the data showed that optical flows observed while the robot was steering routinely exceeded the range of the flow estimation system. The resulting corrupted data rendered obstacles "invisible," and the robot consequently failed to continue seeing obstacles as executed evasive maneuvers.

The simple memory of the steering policy, together with the spatio-temporal edge-preserving median filtering of divergence and assigning a cost to "crossing ridges" in the divergence data (discussed in Section 6), served to commit the robot to a single course around an obstacle until it was cleared. In control trials without these features, the robot was sometimes lured back toward an obstacle by lower hazard estimates on the far side of the obstacle.

An MPEG video of an example test run may be found in the *hypertext portion of this paper*. It demonstrates the relationships between the image, the optical flow, the divergence estimates, and the time-filtered divergence estimates which control the robot's steering behavior.

8.3Wanderingtrials



(a) Trial 0, 7.67 minutes

(b) Trial 9, 7.22 minutes

(c) Trial 12, 6.75 minutes

(d) Trial 1, last 8 of 26 minutes

Figure 21 Wandering trial paths: x-y plots

The third set of experiments tested duration. In the wandering trials, the robot was permitted to wander about the relatively uncluttered lab one day while the authors prepared a report of the present work. Throughout the day, the authors ran 13 trials and collected data.

In these trials, the robot was started toward open space from various locations in the lab. The longest trial lasted 26 minutes. The path of the robot in the final 8 minutes of this trial is shown in Figure 21 (d). Three moderate length paths of about 7 minutes are shown entirely in Figure 21 (a-c). The mean trial length was roughly 7.1 minutes and the median length was 6.75 minutes. While the robot generally drove back and forth along similar paths, it also often worked its way out of such limit cycles. These results were achieved with extremely simple behavior control. More sophisticated behavior control making use of various mechanisms (*e.g.*, an explicit notion of segmented objects, adaptation) to derive or interpret hazard maps can be expected to shorten the time to escape such situations. The robot also covered a considerable fraction of the lab's open space in the longer trials. The failure modes that most commonly terminated these trials with collision are discussed in Section 9. While this performance falls far short of the ideal of limitless collision-free (however crude) mobility as a base of competence, it is promising enough to be considered as a low-level competence in a goal-directed mobile robot system.

9 Discussion

Some researchers [20][33][34] have proposed using divergence or flow derivatives for visual cues, but did provide real-time implementations of these ideas. Nelson and Aloimonos [27][28] used directional flow divergence for stop-and-look obstacle avoidance (not real-time smooth driving). Their environments were simplified, and they did not demonstrate extensive robust behavior over extended periods of time.

Duchon, *et al.* [14][15][16] demonstrated flow and flow-derived time-to-contact for free wandering at 5 cm/s for as long as 5 minutes. In a second implementation, speeds of up to 30 cm/s have been achieved, and unrecorded trials have lasted up to 25 minutes without collision. Their most robust steering strategy was balancing peripheral flows, *i.e.,* comparing left and right peripheral flows to steer the robot down a conceptual corridor (referred to as corridor-following, flow-balancing, centering, *etc.*). However, the "corridor-following" technique is not well-suited to goal-oriented behavior. In [16] target chasing and flow balancing are combined by summing the egocentric heading changes dictated by both systems. It is possible for this strategy of combining behaviors to result in taking a heading that is dangerous because there is no way for behaviors to eliminate all dangerous headings from consideration.

Coombs, *et al.*[11][12] also used flow to implement “corridor-following” and used divergence to detect imminent collision. The present work achieves similar results using divergence alone and is, therefore, not limited to “corridor-following.” The present system supports goal-directed behavior while providing local obstacle avoidance. The method of estimating optical flow described in this paper has been shown to detect obstacles as far away as 6 meters under good conditions. On the other hand, the flow returned from the PIPE image processing computer used in [11][12] was limited to a range of about 1 to 2.5 meters due to the difficulty of detecting edges of distant surfaces. The range of our system is even more remarkable given the coarse resolution (32x64) of the images used. In addition, our system implements both wide-angle and narrow-angle camera functions using only one wide-angle camera and a single framegrabber, unlike the two cameras and video channels used in [11][12].

System performance depends on many factors. Underlying the divergence estimates are optical flow measurements. Although divergence is theoretically unaffected by camera rotation, rotation contributes directly to optical flow. The system calculates optical flow using a correlation method, which, like all techniques, has limited spatiotemporal sensitivity. In particular, large flows are underestimated, so fast camera rotation can corrupt the optical flow estimates on which the divergence estimates rely. Similarly, differential measures such as flow and divergence are inherently susceptible to noise.

Our system relies on gaze stabilization and robust data filters to cope with these problems. Rotational stabilization of the camera reduces flow magnitudes to manageable levels. The brief disturbances introduced by saccades that re-orient the camera to the robot’s heading are effectively ignored by the spatial and temporal median filters that also suppress noise (in contrast to a non-robust smoothing filter which would be affected). This enables the sensing and behavior modules to complement one another without requiring tight coordination among them.

Optical flow also depends on the field of view and scene texture. Although the optical flow algorithm used requires very little texture given adequate lighting [7][8], naturally the obstacle must appear in the field of view in order to be detected. Even with a wide-angle ($115°$) lens, not all obstacles will be visible.

There are two main causes of collision. The first case involves a collision with an obstacle in one of the lower corners of the full-sized image. Currently, due to real-time requirements, only a central 256-pixel band of the 512-pixel height image is used in calculating flow. Grabbing the bottom 256 rows of the 512 rows available would enable these objects to be seen. However, lowering the visual band taken from the full field of view is undesirable because this also lowers the top edge of the image and thus limits the maximum visual range of the system. Given that the current image size allows for about a 20% idle time to buffer OS events, it is likely that a more refined system (perhaps making use of real-time OS facilities) could use this available CPU time to process a larger image.

The second most common cause of collision involves a very narrow object, such as the back of a chair-viewed edge-on; such an object does not have enough spatial extent to trigger a reaction from the system. Because very wide images are being subsampled (from a width of 512 pixels down to 64 pixels), these objects may be only a few pixels wide in the image and they are therefore difficult to detect. This is especially true given the spatial filtering performed to increase robustness.

A sensible extension would be construction of a local map of space to help prevent such collisions, though this would introduce new problems, such as maintaining the map's integrity. Our hazard maps with the steering policy's limited hysteresis provides a simple local spatiotemporal map that has proved sufficient to carry the robot around most obstacles in its path without introducing a complex spatial memory or world map. Dealing with very narrow objects remains future work.

It has been argued that there are computational advantages in keeping the search radius of the optical flow algorithm as small as one pixel [7][8] and keeping the frame rate high. It should be noted that because images are subsampled from a size of 256x512 pixels down to 32x64 pixels, a single pixel shift at the new coarser scale is equal to an 8 pixel shift at the original resolution. In addition, because sub-pixel flows are detected, a magnitude of 1/2 pixels per frame corresponds to a 4 pixel shift at the original resolution, 1/4 pixels per frame corresponds to a 2 pixel shift, *etc*. Even so, when the robot is rotating, the optical flow velocities can be extremely high. (Adding to this effect is the fact that the camera is mounted about 30 centimeters in front of the rotational axis; each "rotation" of the robot results in combination of a rotation and a translation of the camera in inertial space.) In this application, we could easily modify the search space to detect faster velocities in only the horizontal directions, where the greater flows occur. This would allow the vehicle to turn faster without saturating the computed flows, with only a linear increase in the computational time.

In the previous version of the optical flow algorithm [7][8], the correlation match values are not saved as they are calculated, because only the best matching correlation match is needed. Instead, as each one is calculated, it is compared with the current best match and saved if it is the new best match or discarded if it is not. In the new version described here, the correlation match values are interpolated, and thus all match values are saved. Although this only requires 553 KB of memory for current parameters, cache thrashing could become a performance bottleneck for higher image and velocity resolutions. Future versions will be streamlined to reduce this effect.

Sensorimotor interactions also affect overall performance. There are significant latencies in the sensing, estimation, and control modules shown in Figure 1. These modules are only loosely synchronized. Although in general the latency for each module equals its cycle time, in some cases the latency is a bit greater. The modules produce and consume data at various rates, and the interactions of the unequal cycle times have considerable consequences. Flow and divergence estimates are produced approximately every 260 ms (just under 4 Hz). The robot accepts speed and steering commands at about 3 Hz. At a robot velocity of 20 cm/s, visual data become available about every 5 cm of robot travel and steering is adjusted about every 7 cm. To avoid losing valuable data, especially time-critical impending-collision indications, the behavior controller runs at 20 Hz, evaluating any fresh data and writing appropriate steering and speed commands. These commands are only single-buffered, so only the most recent command is read by the robot controller when it is ready for a new one. This minimizes the overall latency in the system. However, it also means the behavior controller does not know exactly which command is being executed unless the robot controller sends an acknowledgment that identifies the accepted command. This condition, coupled with considerable latencies in executing a robot command, means it is difficult for the control (and sensing) systems to know precisely what the robot is doing, short of installing high-speed low-latency feedback sensors. Consequently, the systems are designed to require only approximate knowledge of the robot's current motion state if they use any at all. Instead, for instance, robust data filters are able to ignore momentary noise and artifacts that result from system module interactions, and this enables modules to cooperate without delicate synchronization.

10 Conclusions

A robot system is presented that uses only real-time image flow divergence to avoid obstacles while driving toward a specified goal direction (straight ahead in this demonstration) in a lab containing office furniture and robot and computing equipment. The robot has wandered around the lab at 20 cm/s for as long as 26 minutes without collision. To our knowledge, this is the first such demonstration of real-time smooth wandering for extended periods of time using only flow divergence.

The paper describes how flow divergence is computed in real-time to provide the robot's sense of space and also how steering, collision detection, and camera gaze control are used together to avoid obstacles while attempting to drive in the specified goal direction. The major contribution is the demonstration of a simple, robust, minimal system that uses flow-derived measures to control steering and speed to avoid collision in real time for extended periods.

Although image motion has long been considered a fundamental element in the perception of space, attempts to use it in real-world mobility tasks have always been hampered by problems such as noise, brittleness, and computational complexity. We demonstrate that robust image motion cues can be extracted using a single ordinary UNIX workstation to safely move about a complex environment in real-time for extended periods.

These results demonstrate that real-time robot vision and control can be achieved with careful implementations on ordinary computing platforms and environments. Similarly, an extensible framework can combine simple robust components in a manner than minimizes requirements for tight synchronization.

11 References

[1] J. Albus. "Outline for a Theory of Intelligence," *IEEE Transactions on Systems, Man and Cybernetics*, 21(3):473-509, 1991.

[2] J. Aloimonos, I. Weiss, and A. Bandyopadhyay, "Active Vision," *International Journal of Computer Vision* 1: 333-356, 1988.

[3] P. Anandan, "A Computational Framework and an Algorithm for the Measurement of Visual Motion", *International Journal of Computer Vision*, 2:283-310, 1989

[4] D. Ballard and C. Brown, "Principles of Animate Vision," *CVGIP: Image Understanding*, 56(1):3-21, 1992.

[5] H. Bülthoff, J. Little, T. Poggio, "A Parallel Algorithm for Real-time Computation of Optical Flow", *Nature*

337(6207):549-553, 9 Feb 1989

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

[7] T. Camus, “Real-Time Quantized Optical Flow”, *IEEE Workshop on Computer Architectures for Machine Perception*, Como, Italy, September 18-20, IEEE Computer Society Press, Los Alamitos CA, 1995

[8] T. Camus, “Real-Time Quantized Optical Flow”, *Journal of Real-Time Imaging* (special issue on Real-Time Motion Analysis), Vol.3, pp.71-86, Academic Press, 1997.

[9] T. Camus, H. Bülthoff, “Space-Time Trade-offs for Adaptive Real-Time Tracking”, *Mobile Robots VI*, William J. Wolfe, Wendall H. Chun ed., Proc. SPIE 1613, pp.268-276, Nov. 1991

[10] R. Cipolla and A. Blake. “Surface orientation and time to contact from image divergence and deformation,” In *Proc. of Computer Vision-ECCV '92, the Second European Conference on Computer Vision*, Santa Margherita Ligure, Italy, May 19-22, 1992, pp. 187--202.

[11] D. Coombs, M. Herman, T.-H. Hong, and M. Nashman. “Real-time Obstacle Avoidance Using Central Flow Divergence and Peripheral Flow,” In *Proc. of ICCV 1995, the Fifth International Conference on Computer Vision*, Cambridge, Massachusetts, June, 1995.

[12] D. Coombs, M. Herman, T.-H. Hong, and M. Nashman, “Real-time Obstacle Avoidance Using Central Flow Divergence and Peripheral Flow,” *IEEE Transactions on Robotics and Automation,* 14(1):49--59, 1998.

[13] R. Dorf. *Modern Control Systems,* Addison-Wesley, 1980.

[14] A.P. Duchon and W.H. Warren, “Robot Navigation from a Gibsonian Viewpoint,” *Proc., SMC 1994, IEEE International Conference on Systems, Man, and Cybernetics.* (San Antonio, TX, October 2-5) pp. 2272-2277, 1994.

[15] A. Duchon, W. Warren, and L. Kaelbling, “Ecological Robotics: Controlling Behavior with Optical Flow”, pp. 164-169, *Proceedings of the Seventeenth Annual Conference of the Cognitive Science Society*, Pittsburgh, PA July 22-25, 1995. Johanna D. Moore and Jill Fain Lehman, eds. Lawrence Erlbaum Associates, Mahwah, NJ.

[16] A. Duchon, W. Warren, and L. Kaelbling, “Ecological Robotics”, *Adaptive Behavior*, 6(3/4):473--507, 1998.

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

[18] M. Herman, M. Nashman, T.-H. Hong, H. Schneiderman, D. Coombs, G.-S. Young, D. Raviv, A. J. Wavering, “Minimalist Vision for Navigation”, In *Visual Navigation: From Biological Systems to Unmanned Ground Vehicles,* Y. Aloimonos ed., Lawrence Erlbaum Associates, Mahwah, NJ, 1997, 275-316.

[19] T. Huang, G. Yang, G. Tang, ``A Fast Two-dimensional Median Filtering Algorithm”, *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 1:13-18, Feb 1979.

[20] J. Koenderink and A. van Doorn., “Optic Flow,” *Vision Research*, 26(1):161-180, 1986.

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

[22] H. Liu, *A General Motion Model and Spatio-Temporal Filters for 3-D Motion Interpretations*, Ph.D. Thesis, Technical Report CAR-TR-789 and CS-TR-3525, University of Maryland, 1995.

[23] H. Liu, T.-H. Hong, M. Herman, T. Camus, R. Chellapa, “Accuracy vs. Efficiency Trade-offs in Optical Flow Algo-

rithms", *Computer Vision and Image Understanding*, Vol. 72, No. 3, pp. 271-286, Academic Press, December 1998.

[24] H. Liu, T.-H. Hong, M. Herman, R. Chellapa, "Accuracy vs. Efficiency Trade-offs in Optical Flow Algorithms", P*roceedings of the Fourth European Conference on Computer Vision*, Cambridge, England, April 1996.

[25] J. McCann, *Neural Networks for Mobile Robot Navigation*, Masters Thesis, Brown University, February 1995.

[26] P. Narendra, "A Separable Median Filter for Image Noise Smoothing", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 3(1):20-29, Jan 1981.

[27] R. Nelson and Y. Aloimonos. "Using Flow Field Divergence for Obstacle Avoidance in Visual Navigation," In *Proc. DARPA Image Understanding Workshop*, pp. 548-567, April 1988.

[28] R. Nelson and Y. Aloimonos. "Obstacle Avoidance Using Flow Field Divergence," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(10):1102-1106, October 1989.

[29] T. Nodes, N. Gallagher, "Two-Dimensional Root Structures and Convergence Properties of the Separable Median Filter", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 6:1350-1365, Dec 1983.

[30] W.H.Press, S.A. Teukolsky, W.T.Vetterling, B.P. Flannery, *Numerical Recipes in C* (second edition), Cambridge University Press, 1992

[31] D. Raviv and M. Herman, "Visual Servoing from 2-D Image Cues," In *Active Perception*, Y. Aloimonos, ed., Lawrence Erlbaum Associates, Hillsdale, NJ, pp. 191-226, 1993.

[32] P. Rousseeuw, A. Leroy, *Robust Regression and Outlier Detection*, John Wiley & Sons, Inc., 1987

[33] W.B. Thompson and J.K. Kearney, "Inexact vision," *Proceedings of Workshop on Motion: Representation and Analysis* (Charleston, SC, May 7-9, 1986), pp. 15-21.

[34] M. Tistarelli and G. Sandini, "On the advantages of Polar and Log-polar Mapping for Direct Estimation of Time-to-Impact from Optical Flow," *IEEE-PAMI*, April, 1993.

[35] G.S. Young, T.H. Hong, M. Herman, and J.C.S. Yang, "New Visual Invariants for Terrain Navigation Without 3-D Reconstruction." *International Journal of Computer Vision*, Vol. 28, No. 1, June 1998, 45-71.