13th International Conference on Pattern Recognition
Applications and Robotic Systems
August 1996, Vienna, Austria

Real-Time Single-workstation Obstacle Avoidance Using Only Wide-Field Flow Divergence

Ted Camus, David Coombs, Martin Herman, Tsai-Hong Hong
National Institute of Standards and Technology
{tcamus,dcoombs,mherman,thong}@nist.gov; http://isd.cme.nist.gov/
Intelligent Systems Division, Building 220 Room B-124, Gaithersburg MD 20899

Abstract

A real-time robot vision system is described 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 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. In particular, the divergence of image flow (the sum of image flow derivatives in two perpendicular directions) is theoretically not affected by camera rotation, and can be used by a moving observer to navigate about an environment. 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 integrated 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 it 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 enable the system to ignore momentary noise and artifacts that result from module interactions, and this in turn 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, and 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.

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 represented in 2-D image coordinates only. The control algorithms directly use observable image information represented in the 2-D image sequence; a 3-D reconstruction is not required [17]. It is therefore simpler and faster. Such an approach is particularly useful in closing control loops with vision at lower levels of a multi-level control system[1] (Figure 1).

Figure 1 sketches the obstacle avoidance system. Video images are obtained from an on-board uncalibrated camera with a $115°$ field of view. The robot's view from this camera is shown in Figure 6(b). The images are subsampled and full flow is computed. Flow divergence is estimated and spatio-temporal median filters are applied to reduce momentary fluctuations in the divergence field. Hazard maps are derived from the divergence field, the previous steering decision, and the goal direction. A composite hazard map is used to steer the robot around objects as it drives in the goal direction. 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. The inputs to the body and gaze controllers consist of driving, steering, and gaze velocities.

2. Real-Time Control System (RCS)

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. It monitors its environment with sensors and updates models of the states of the system itself and the world. Figure 1 maps the functionality of the obstacle avoidance system into the first three levels of the RCS hierarchy.

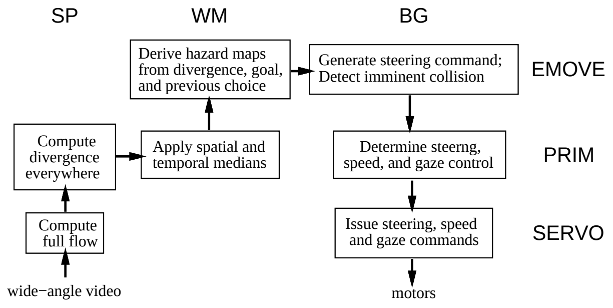
SP
WM
BG
Derive hazard maps from divergence, goal, and previous choice
Generate steering command; Detect imminent collision
EMOVE
Compute divergence everywhere
Apply spatial and temporal medians
Determine steerng, speed, and gaze control
PRIM
Compute full flow
Issue steering, speed and gaze commands
SERVO
wide−angle video
motors

Figure 1. Obstacle Avoidance Architecture

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. It maintains a database of system variables and filters and analyzes 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.

3. Full image flow estimation

Robust, real-time optical flow has become a practical means of robotic perception given new fast algorithms and increasingly faster scientific workstations. 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 [6] describes a robust, real-time correlation-based optical flow algorithm which returns dense data even in areas of low texture. This algorithm is the starting point of our new implementation.

In correlation-based flow such as in [5] the motion for the pixel at [x,y] in one frame to a successive frame is defined to be the determined motion 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). 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:

$$\forall(u, w) M(x, y; u, w) \qquad (1)$$

$$= \sum_{(i,j) \in P_\nu} \phi(E_1(i, j) - E_2(i + u, j + w))$$

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 (equivalently minimum patch difference); thus this is called a "winner-take-all" algorithm.

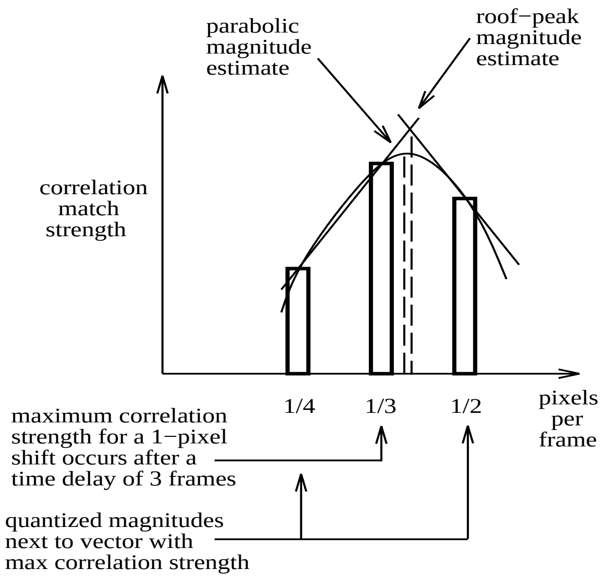
parabolic magnitude estimate
roof−peak magnitude estimate
correlation match strength
1/4
1/3
1/2
pixels per frame
maximum correlation strength for a 1−pixel shift occurs after a time delay of 3 frames
quantized magnitudes next to vector with max correlation strength

Figure 2: Parabolic and roof estimation methods.

One limitation with the traditional correlation-based algorithm is that its time complexity grows quadratically with the maximum possible displacement allowed for the pixel [5][7]. 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. Note the simple relationship between velocity, distance and time: $vel = (\delta dist)/(\delta time)$. Normally, in order to search for variable velocities, we keep the inter-frame delay δt constant and search over variable distances (pixel shifts): $\Delta\nu = (\Delta d)/(\delta t), d \le \eta$. However, 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: $\Delta v = (\delta d)/(\Delta t)$. In this case, we generally prefer to keep δd as small as possible in order to avoid the quadratic increase in search area. This time-space trade-off results in a very fast algorithm: optical flow can be computed on 32x64 images (subsampled from 256x512), calculating 5 speeds per frame, at up to 35 frames per second on a 80 MHz HyperSPARC[1] computer.

The above algorithm returns *quantized* optical flow values. Although this is sufficient for various robotics vision tasks [6][10][11], it is not sufficient for our application since the calculation of divergence requires that the spatial derivatives of the optical flow can be measured. Because the quantized optical flow is basically a step function, these derivatives do not exist. Smoothing the optical flow field would require extremely large masks and would therefore likely cover multiple objects simultaneously. This would be especially problematic since a wide-angle lens is used and individual objects do not occupy more than a small fraction of the visual field. Calculating a least-squares best fit to the correlation surface such as in [3] was ruled out due to real-time performance requirements.

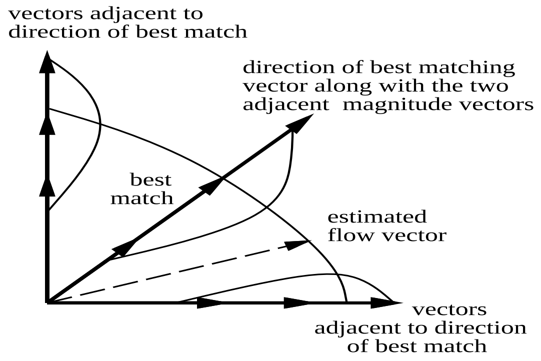
vectors adjacent to direction of best match
direction of best matching vector along with the two adjacent magnitude vectors
best match
estimated flow vector
vectors adjacent to direction of best match

Figure 3: Angular estimation of flow vector.

In order to avoid a computationally expensive search for the true flow, the 2-dimensional search space was decomposed into two 1-dimensional searches, the first estimating the magnitude of the flow vector and the second estimating the precise angle of the true flow vector. (The actual flow is returned as the X and Y components of the flow. Converting to polar coordinates can be done in the usual way.) Both the directional component as well as the magnitude component of the flow are quantized. The first one-dimensional interpolation is along the magnitude component of the flow. The correlation match values for the best motion of a given pixel along with the match values for the flow vectors of "adjacent" magnitudes in the same direction (i.e., of plus and minus one time delay in frames) are used. Roof interpolation is used to find the total time delay corresponding to the minimum correlation match value as shown in Figure 2. Two lines are formed with the best correlation match strength and the match strengths corresponding to those two time delays which immediately bracket the time delay with the best match. Substituting the steeper of the two slopes (in an absolute sense) for the more gradual of the two results in a single intersection of the two lines; the abscissa of this point is taken as the new interpolated magnitude component of the flow.

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.

The second interpolation is along the angular component of the flow. We wish to calculate the corresponding match values for flow vectors of neighboring vector directions but of the same magnitude as just calculated for that pixel. (Since we only calculate the correlation match values for eight directions of motion, this means that each neighboring direction is $45°$ from that motion vector with the best correlation match.) This interpolation is not trivial since the magnitude of the motion along a diagonal is $\sqrt{2}$ times that of motion along a row or a column for a given velocity (time delay). In order to perform the second 1-dimensional interpolation it is necessary to estimate the correlation matches values of the neighboring direction at the *same* magnitude as the best matching flow vector. Although the roof interpolation was slightly more effective 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. The following formulas for the parabola coefficients were derived given a parabola $ax^2 + bx + c$:

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

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

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

Once these coefficients are calculated, it is possible to estimate the correlation match strength at any point (magnitude) in any given direction. In particular, the correlation match strengths measured at the same magnitude as the best correlation match for that pixel may be estimated at those two angles (out of eight measured) which bracket the best matching motions direction; this allows the simple additional one-dimensional interpolation to provide the interpolated angle of motion. Figure 3 shows this graphically; thin lines represent parabolic interpolations of three given correlation match values.

This double one-dimensional interpolation calculation cuts the frame rate approximately in half: real-valued optical flow can be computed on 32x64 images, calculating and

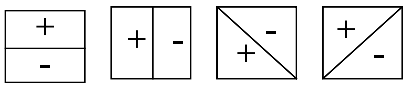

Figure 4. Flow Divergence Templates

interpolating 5 speeds per frame, at up to 17 frames per second on a 80 MHz HyperSPARC computer. In practice, the flow is run at only about 4 Hz. This consumes from 20-25% of the processor's total time and allows the entire system to run easily on a single workstation with a consistent frame rate and about a 20% processor idle time to buffer unexpected operating system events.

4. Divergence for Obstacle Avoidance

Divergence of the flow field is computed in the central band of the wide-field camera. Divergence can be used to qualitatively estimate time-to-contact (T_c). Both theory and implementation are discussed here as well as considerations for employing T_c qualitatively estimated from divergence 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 are

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

$$O_y = (1/Z)(-T_y + yT_z) + \left((1+y^2)\omega_x - xy\omega_y - x\omega_z\right)$$

where Z is the depth of the object in the environment relative to the camera, (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. The divergence of an optical flow field is defined as: $\nabla\circ(O_x, O_y) = \frac{\partial O_x}{\partial x} + \frac{\partial O_y}{\partial y}$ whenever the imaged surface is a mostly perpendicular surface *or* the gradient of the imaged surface is perpendicular to the transverse velocity (T_x, T_y). In our experiments, the values of (T_x, T_y) are qualitatively equal to zero. Divergence can qualitatively estimate directly Time-to-Contact [8]

$$\nabla\circ(O_x, O_y) = 2 \cdot T_z/Z . \qquad (2)$$

This measurement is particularly useful for obstacle avoidance during visual navigation because divergence is invariant under the rotational motion of the sensor that is inevitable due to imperfect stabilization.Equation (2) suggests that *divergence* has only time as its dimension. The values of *divergence* over any significant area represent the inverse of the time needed to reach an object at distance Z with velocity T_z in the z direction. Therefore, a family of simple fixed flow divergence templates can be applied to any image sequence to estimate divergence [15]. Each template is symmetrically divided into positive and negative halves (Figure 4). Flow divergence is calculated by convolving the template with a window in the flow image and computing the sum of the image 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 [6]. In order to improve the consistency of the divergence estimates, we apply temporal and spatial median filters to the individual divergence calculations.

5. Simple Robust Filters

Medians performed on dense two-dimensional data can use fast running-histogram methods if the dynamic range of the data and desired resolution of the median can be specified [12]. That algorithm was intended for finding the true median and reduces an $O(nm)$ complexity algorithm to approximately $O(n)$ per pixel for a $n \times m$ filtering window where $n < m$. It can however be generalized to the separable median [14] reducing its complexity from $O(n)$ to approximately constant time. This approach assumes that there are only a limited number of 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 256 bins and then use quicksort-partitioning [16] to find the true median within the bin which is known to contain it. In our case however, divergence data were quantized to 256 parts in order to reduce data bandwidth so this extension is not necessary. Although the separable median is not guaranteed to find the true median, the effects of the separable median in otherwise reducing noise is almost as good as the true median [14].

The current system first performs a 11x17 (row times column) width spatial median filter, with the height of the filter smaller due to the flat rectangular images used. A second filtering is performed, with a dimension of 11 in space and 11 in time. Since it would be undesirable to have a delay in a real-time system, we simply localize the temporal median filter such that its leading edge includes the current data point but no future data points.

The separable median as well as the true median filter both have the desirable property of preserving horizontally and vertically aligned edges, which means that unlike many other averaging or smoothing filters there is no temporal hysteresis. Unlike the true median however, the separable median has the additional desirable property of preserving corners [14]. Preserving corners is especially valuable in time-space or other plots, since an erosion of an object's full spatial and/or temporal extent could create the illusion of open space and cause a collision.

6. Driving Control

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 (*i.e.,* 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. 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 for-

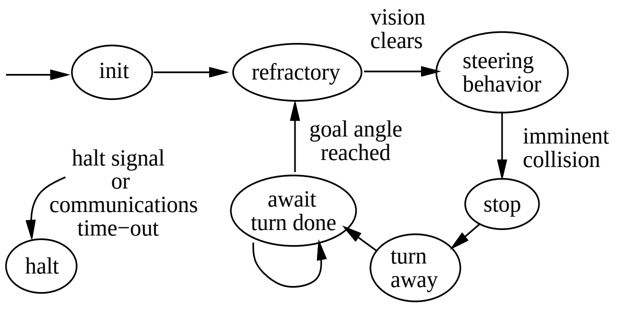

Figure 5. Body Control Automaton

ward, 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. 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 simply straight ahead.) Indication of imminent collision in the central region of the 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 5). Some state transitions are triggered by sensed events, and others merely provide command sequencing.

A view of the robot is provided in Figure 6 (a). It consists of two cameras mounted on a hollow steel cage attached to a TRC Labmate platform. Although both cameras were used in [8], only the top wide-angle camera is used in the current system. The robot's tether, consisting of video cables, the connection to the gaze motor controller, and a serial communications line to the Labmate can also be seen.

The steering policy is implemented using hazard maps derived from flow divergence, the desired goal direction, and the target heading, θ_v, previously selected by the steering policy. Each hazard map is a 1-dimensional vector that encodes the "risk" associated with each possible steering direction.

One hazard map is derived from the divergence data (a 64-element wide vector per sample interval) that indicates obstacles and also encodes the cost of crossing "ridges" in the divergence vector, starting from the previously selected heading. Similarly, another hazard map is derived from the desired goal direction and the previously selected heading (accounting for the gaze angle). This map is roughly a trough centered mid-way 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 adding the component hazard maps. The steering policy chooses the direction of minimum hazard in the composite 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. A composite hazard map is shown in Figure 6(c) and the resulting path of the robot appears in (d) for the gauntlet of office chairs seen in (b) from the robot's viewpoint before the trial began.

When a new desired heading is chosen, the robot steers smoothly to it with saturated negative visual feedback controls [9]. Desired change in heading, $\Delta\theta$, is then calculated, accounting for the current gaze angle, θ_g, with respect to heading: $\Delta\theta = \theta_v + \theta_g$. The steering control policy is simply a saturated steering velocity proportional to the desired heading: $\dot{\theta} = Saturate(k_s \cdot \Delta\theta \cdot 1/T_b, s)$. The gain k_s 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 per second rather than degrees per 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). The angular velocity is saturated at $\pm s$ degrees per second to limit the peak rotation rate to reasonable levels.

The robot steering and collision detection improve when the robot turns relatively slowly for two reasons: (1) temporal consistency of spatial samples, and (2) accuracy

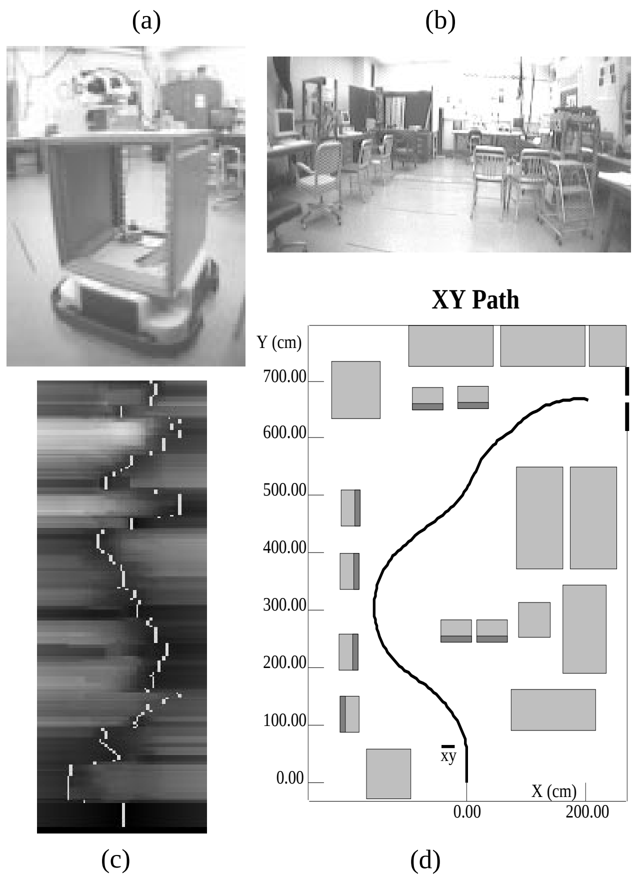

Figure 6: (a) View of the robot; (b) robot's view of the gauntlet of office chairs at the start of the trial; (c) hazard map (with time increasing upward); (d) XY path trace beginning at (0,0), showing obstacles in the lab.

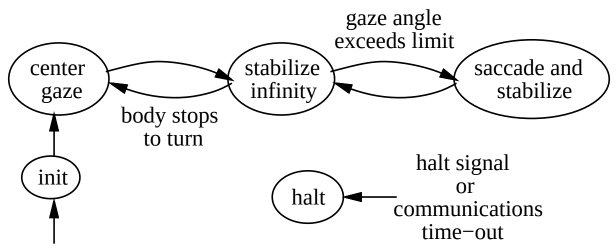

Figure 7. Gaze Control Automaton

of motion estimates. Therefore, the behavior and motor control systems 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 poor data.

7. Gaze Control

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 7). 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, θ_g, from the robot's heading and snaps the camera back to the heading if the limit is exceeded.

The saccades that perform the quick-phase return to realign gaze with the robot's heading briefly produce extremely large image 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 and robot and computing equipment. Furniture and equipment lined the walls and there was free space 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 6 for an example of the robot's view of the lab.) Three set of experiments where performed. (1) "Crash tests" evaluated the system's ability to detect obstacles and warn of imminent collision. (2) The "gauntlet trials" tested the robot's ability to maneuver around obstacles in its environment while traveling across the lab. (3) Wandering trials tested the robot's ability to move about for extended periods of time.

8.1. Crash tests

Initial experiments tested the robot's ability to detect obstacles and warn of imminent collision. A row of chairs was placed across the far end of the lab and the robot drove straight toward it at fixed speeds. The system detected objects (at divergence levels above the noise level) at ranges up to 6 m (the maximum testable distance in the lab) at forward speeds ranging from 0.1 m/s to 0.8 m/s. The divergence signal arising from an object rose reasonably smoothly as the object was approached, and the object continued to be visible until the robot approached very near. Based on these trials, an imminent collision function was derived for robot speeds up to 0.8 m/s.

8.2. Gauntlet trials

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 are shown in Figure 6. In Figure 6 (d) each inflection point in the curve represents an evasive turn. The robot first 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. An MPEG of the second half of this sequence, along with simultaneous flow and divergence estimates, is available at the first author's WWW address. In these trials the robot traveled at 20 cm/s and steered at a maximum rate of $8°$/s.

Gaze stabilization contributed considerably to the effectiveness of the system by reducing the magnitude of the image flows while the robot was steering. In control trials without gaze stabilization, analysis of the data showed that image 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 see obstacles as it completed evasive maneuvers. The simple memory of the steering policy and the spatio-temporal edge-preserving median filtering of divergence served to commit the robot to a single course around an obstacle until it had cleared.

8.3. Wandering trials

The third experiment was a test of 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, 13 trials were run and data were collected. 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 8. A moderate length path of about 7 minutes is shown entirely in Figure 9. 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

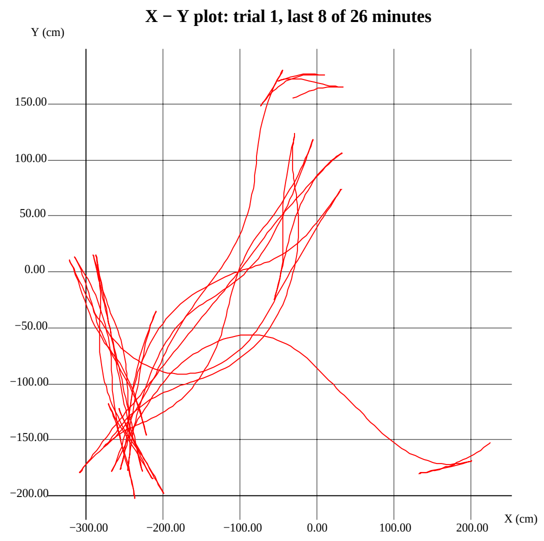

Figure 8: Wandering trial XY path.

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 mode 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 [13][18] have proposed using divergence or flow derivatives for visual cues, but they do not provide real-time implementations of these ideas. Nelson and Aloimonos [15] used directional flow divergence for stop-and-look obstacle avoidance (not real-time smooth driving). Their environments were much simpler than ours and they did not demonstrate extensive robust behavior over extended periods of time.

Duchon and Warren [10] demonstrated flow and flow-derived time-to-contact for free wandering at 5 cm/s as long as 5 minutes. Their most robust steering strategy was balancing peripheral flows (i.e., "corridor-following"). However, this strategy is not easily adapted to goal-oriented behavior.

Coombs *et al* [8] also used flow to implement "corridor-following" and used divergence to detect imminent collision. Our work achieves similar results using divergence alone and is therefore not limited to "corridor-following." Our system supports goal-directed behavior while providing local obstacle avoidance. The method of optical flow described in this paper has been shown to detect obstacles as far away as 6 meters under good conditions where the flow returned from the PIPE was practically limited to a range of about 1 to 2.5 meters due to the difficulty of detecting edges of far away surfaces. The range of our system is even more remarkable given the coarse resolution 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 [8].

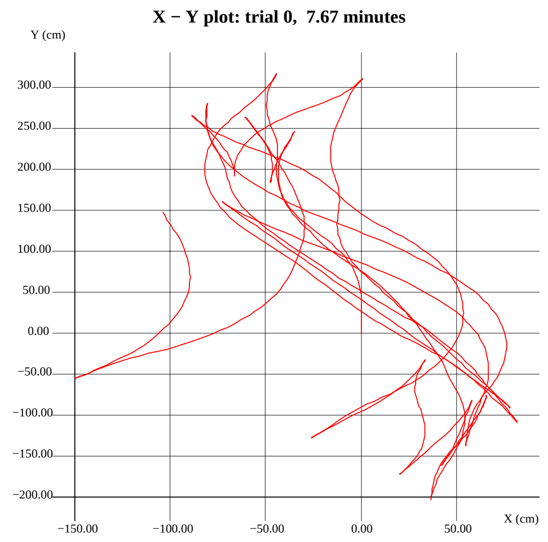

Figure 9: Wandering trial XY path.

System performance depends on many factors. Underlying the divergence estimates are image flow measurements. Although divergence is theoretically unaffected by camera rotation, rotation contributes directly to image flow. The system calculates image 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 image 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 reorient the camera to the robot's heading are 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 modules to cooperate without tight coordination.

The primary cause of system failure consists of 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 since 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 operating system (OS) events, it is likely that a system making use of real-time OS facilities could use this available CPU time to process a larger image.

It has been argued that there are computational advantages in keeping the search radius of the optical flow algorithm as small as one pixel [6] and keep the frame rate high. It should be noted that because images are subsampled from 256x512 pixels to 32x64 pixels in size, a single pixel shift at the new coarser scale is equal to an 8 pixel shift at the original resolution. In addition, since sub-pixel flows are detected, a magnitude of 1/2 pixels per frame corresponds to a 4 pixel shift at the old 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. In this application we can easily modify the search space so that faster velocities are detected only in the horizontal directions where the greater flows occur. This would allow faster turning velocities without saturating the flows but only linearly increase the computational time used.

10. Conclusions

A robot system is presented that uses only real-time motion 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 using only flow divergence.

The paper describes how flow divergence is computed in real-time to provide the robot's sense of space and how steering, collision detection, and camera gaze control cooperate to avoid obstacles while the robot attempts 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 for the first time 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 Quantized Optical Flow", to appear in *The Journal of Real-Time Imaging* (special issue on Real-Time Motion Analysis), Academic Press, 1996.

[7] 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, p.268-276, Nov. 1991

[8] D. Coombs, M. Herman, T. 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.

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

[10] 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.

[11] 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.

[12] 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

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

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

[15] 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.

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

[17] 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.

[18] 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.