Real-time obstacle avoidance using central flow divergence and peripheral flow

David Coombs, Martin Herman, Tsai Hong, Marilyn Nashman

{dcoombs, mherman, thong, mnashman}@nist.gov
National Institute of Standards and Technology
Building 220 Room B-124

ABSTRACT

The lure of using motion vision as a fundamental element in the perception of space drives this effort to use flow features as the sole cues for robot mobility. Real-time estimates of image flow and flow divergence provide the robot's sense of space. The robot steers down a conceptual corridor, comparing left and right peripheral flows. Large central flow divergence warns the robot of impending collisions at "dead ends." When this occurs, the robot turns around and resumes wandering. Behavior is generated by directly using flow-based information in the 2-D image sequence; no 3-D reconstruction is attempted. Active mechanical gaze stabilization simplifies the visual interpretation problems by reducing camera rotation. By combining corridor following and dead-end deflection, the robot has wandered around the lab at 30 cm/s for as long as 20 minutes without collision. The ability to support this behavior in real-time with current equipment promises expanded capabilities as computational power increases in the future.

Keywords: Active vision, real-time, robot mobility, navigation, image flow

1. Introduction

Mobile robots that drive at reasonable speeds (*e.g.*, 30 cm/s indoors) must robustly sense and avoid obstacles in real-time. Image motion provides powerful cues for understanding scene structure. Divergence of image flow (the sum of image flow derivatives in two perpendicular directions) is theoretically not affected by camera rotation, so it gives a robust measure of scene structure for a moving observer. In the direction of the camera's heading, divergence is inversely related to time-to-contact (T_c). The system described in this paper avoids obstacles while it wanders. It uses time-to-contact estimates derived from flow divergence to detect imminent head-on collision. This is combined with a centering behavior that compares left and right peripheral flows to steer the robot down a conceptual corridor [5]. When the corridor-following behavior drives the robot into a "dead end" in the "corridor," the central time-to-contact predictions warn the robot of the impending collision. The robot stops, turns, and resumes wandering, following a new "corridor." These integrated behaviors have driven the robot around the lab at 30 cm/s for as long as 20 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. A more complete description of our system is presented in [6].

Central Lens: 40° Peripheral Lens: 115°

Figure 1. Robot's View of the Scene

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][3]. The representations contain only task-relevant information, 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. This approach requires fewer calibrations, fewer scene hypotheses, and less computation. It is therefore simpler and faster [15].

In our system, video images are obtained from two onboard cameras. One of the cameras is above the other, and both are mounted on a single pan motor. Neither camera is calibrated. Central/peripheral visual sensing is achieved with narrow and wide lenses. The narrow central camera has a 40° field of view. It is tilted downward so that its visual axis intersects the floor 2 or 3 meters in front of the robot. The wide camera, mounted slightly below the narrow one, has a 115° field of view. The robot's view from these cameras is shown in Figure 1. The images from the two cameras are processed independently, and they are independently used to control behavior.

Our system consists of five processing modules. The first process computes normal flow, the component of mo-

tion perpendicular to edges in the images of the two cameras. The magnitude and quantized direction of the normal flow are presented to the second processing module, which computes divergence and time-to-contact in three overlapping windows of the central image. In the wide image, the process estimates maximum flow in two peripheral visual fields (left and right). Accuracy is sacrificed to achieve real-time performance. The resulting errors in normal flow estimates produce errors in divergence and time-to-contact measurements as well. The third process temporally filters maximum flow and T_c to reduce errors. Recursive estimation is used to update current flow and T_c estimates and to predict flow and T_c at the next sample time.

The fourth process uses flow to steer the robot down the conceptual corridor estimated by comparing peripheral flows in the wide camera. Using active gaze control, the cameras are rotationally stabilized so that their motion is approximately a translation. If the flow is larger on one peripheral side than the other, then objects in the scene are closer on that side, and the robot steers away. When the camera points too far away from the heading, a saccade is made toward the heading. When T_c predictions in the central camera indicate imminent collision in a "dead end" in the conceptual corridor, the robot stops, turns away, and resumes wandering. The inputs to the body and gaze controllers consist of driving, steering, and gaze velocities. The system is designed in accordance with the Real-Time Control System (RCS) hierarchical architecture [1][6].

2. Computation of Normal Flow

The system first computes the magnitude and quantized direction of normal flow in the image sequences from each of the two cameras mounted on a mobile robot. These flow values are computed in real-time on PIPE (Pipelined Image Processing Engine [10]. Normal flow is the component of image motion in the direction of the intensity gradient (*i.e.,* perpendicular to intensity edges). Motion information is extracted from the sequences using temporal and spatial derivatives [9]. For each camera, sequential images (video *fields* not frames) I_0, I_1, and I_2, are digitized and stored. To satisfy the continuity constraint for the gradient based flow computation, each image is filtered with a Gaussian function ($\sigma = 1$ pixel) using a 5×5 convolution kernel. The temporal derivative between images I_0 and I_2 is computed. The spatial derivatives are computed on image I_1 and, for every point in the image, the intensity gradient magnitude and orientation are computed using 3×3 spatial gradient operators [16]. The gradient direction, θ, of each point in the image is quantized into four directions: north-south, northeast-southwest, east-west, and southeast-northwest.

The normal flow $O_{\nabla I}$ is:

$$O_{\nabla I} = K\frac{\partial \mathrm{I}/\partial t}{\nabla I} \tag{1}$$

where K is a scaling factor that permits the result to be expressed in 8-bit integer format to meet hardware constraints. The gradient, ∇I, is used only if it is above a threshold to reduce the difference between the motion field and the corresponding optical flow [9]. K is chosen empirically to allow representation of an appropriate range of flows. It is chosen to handle central flows arising from obstacles between 1.5 and 2.5 meters from the camera at robot speeds up to 30 cm/s. Closer objects give rise to image flows too large to represent. Extremely small flows can be noisy. A different scale factor is used for peripheral flows.

The result of this process is a histogram of the quantized gradient directions, a list of image coordinates of each quantized direction, and the corresponding scaled flow values in each direction.

3. Maximum Flow in the Visual Periphery

The system estimates the maximum flows in the left and right peripheral visual fields of the wide-angle camera. As described in Section 7, the cameras are rotationally stabilized using active gaze control so that their motion is approximately a translation. For comparing clearance to obstacles on each side of the path, the maximum flow value in each peripheral receptive field is a function of the distance to the nearest object in that field [5]. The optical flow processing in each field is implemented in two parts: first, dense normal flow is estimated as described above, then the maximum magnitude flow in each receptive field is identified by examining the histogram of flows in each field. Spurious outlying high magnitude data are avoided by ignoring a small fraction of the highest magnitude flows (*e.g.*, 4%). The maximum flows are also recursively estimated and predicted (section 5.).

It should be noted that this implementation does not attempt to compute true range for objects in the image. To do so would require calibrating the focal length, the wide-angle lens distortion, and the vehicle velocity (both direction and magnitude).

4. Divergence and Time-to-contact

The system computes flow divergence in the central camera. 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 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:

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

$$O_y = \left(\frac{1}{Z}\right)(-T_y + yT_z) + \left(\left(1 + y^2\right)\omega_x - xy\omega_y - x\omega_z\right) \quad (3)$$

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[22]. The divergence of an optical flow field is defined as:

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

Note that

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

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

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

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

From equation (7),

$$\nabla^\circ(O_x, O_y) = 2\rho T_z \quad (8)$$

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). Time-to-contact (T_c) or nearness in time, can be estimated directly from divergence (Equation (8))

$$T_c = \frac{Z}{T_z} = 2/\nabla^\circ(O_x, O_y). \quad (9)$$

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 (9) suggests that T_c has only time as its dimension. The values of T_c over any significant area in the image represent the time needed to reach an object at distance Z with velocity T_z in the z direction. T_c is twice the inverse of divergence. Therefore, a family of simple fixed flow divergence templates can be applied to any image sequence to estimate divergence [14]. Each template is symmetrically divided into positive and negative halves (Figure 2). 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. Only normal flow values in the direction of the particular mask are considered for each mask. Since there are few accurate flow values in the implementation, a large window is needed to accumulate multiple samples. (The estimates are inaccurate particularly because they are only normal flow and their directions are quantized.) Each time-to-contact value has an associated confidence, which is the number of points in the window that have measurable flow values. In order to improve the consistency of the estimation of T_c, we apply a recursive least squares update procedure

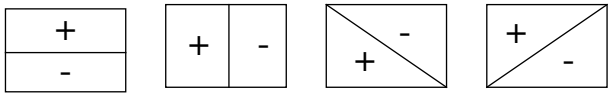

Figure 2. Flow Divergence Templates

5. Recursive Least Squares Estimation

One of the significant contributions of our approach is its use of a temporal model to estimate and predict T_c values. A linear model is maintained for the time-to-contact in each of the three windows:

$$T_c = a_1 + a_2 t \quad (10)$$

where t = 0, -1, -2,... For each measurement of time-to-contact, model parameters a_1 and a_2 are updated by a weighted recursive least squares computation with exponential decay [4][8][12][18]. This involves determining a_1 and a_2 such that the residual J is minimized:

$$J = \sum_{t=0}^{-n} \lambda^t w_t [T_c - (a_1 + a_2 t)]^2 \quad (11)$$

where $t = 0, -1, -2, \ldots, -n$; $t = 0$ is the present; λ is the forgetting factor; and w_t is the confidence of the t^{th} T_c measurement (the number of flow data points in the window).

In order to solve for a_1, a_2, the square root information filter (SRIF) algorithm [4] [12] is used. The SRIF provides an efficient, numerically stable, closed form solution to the least squares problem. It is also a recursive algorithm, i.e. the model is updated as new data becomes available without having to explicitly store old data.

Since the camera is approximately aligned with the robot's heading, the objects viewed in the estimation windows change as the robot steers away from obstacles. Recursive estimates derived over time are sensitive to the underlying data, which should ideally consist of the same sample points for the estimation period. However, the windows' sample data change constantly when the vehicle is turning, *i.e.,* old obstacles disappear from view and new obstacles become visible. The forgetting factor, λ, adjusts for this behavior by controlling the relative weighting of new and old data. λ is changed dynamically as a function of the control system's commanded angular velocity. In this way, the weight of past events is reduced when the robot turns faster and a new scene becomes visible. Similarly, past observations weigh more heavily when the robot moves straight ahead.

Statistical analysis of the recursive least squares predictions has produced a robust indicator for imminent collision. Figure 6 plots a history of time-to-contact predictions for straight line motion of the robot. The last entries in the plot are decreasing and approaching zero. However, although the general trend is downward, the progression is non-monotonic. No single time-to-contact datum provides sufficient confidence to conclude that collision is likely. There appears to be a correlation between collision, the time-to-contact value (a_1 in Equation [10]), the slope of the predicted line (a_2),and the confidence (w_0) which represents the number of normal flow pixels contributing to the current divergence estimate. A combination of short predicted time-to-contact ($a_1 < 0.07$), negative slope of T_c predictions ($a_2 < 0$), and at least a moderate number of current data ($w_0 > 20$) seem to reliably indicate imminent collision. a_1 represents a measure of time-to-contact, but because of repeated scaling operations, it is expressed only in terms of t units which are not calibrated to standard time units. We have empirically found that at least 10 T_c samples are necessary for reasonable accuracy of recursive estimates.

In our approach to filtering maximum flows, we use a temporal linear model similar to equation (11) in each receptor field. Again the forgetting factor λ is a constant that controls the degree of smoothing and the length of past history.

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 develop the ability to maneuver without collision using only visual image flow to sense the environment. The robot's behavioral goal is simply to drive forward, steering away from obstacles, and to stop and turn when it senses that collision is imminent.

The robot begins driving at 30 cm/s. Driving at a fixed speed allows the use of a customized lookup table for the computation of normal flow that is meaningful within the ranges of interest. The centering behavior uses the peripheral flows to steer the robot down its conceptual corridor. Indication of imminent collision (based on time-to-contact) in any of the central receptive fields causes the robot to stop, turn away and resume wandering. This is implemented with a finite state automaton, with a command associated with each state (Figure 3). Some state transitions are triggered by sensed events, and others merely provide command sequencing.

The robot steers smoothly to a new desired heading with saturated negative visual feedback controls [7]. The desired heading, θ_v, is chosen to attempt to balance the peripheral flows and is represented in retinotopic (visual) coordinates.

$$\theta_v = k_h\left(O^L_{\nabla I} - O^L_{\nabla I} \right) \tag{12}$$

$O^L_{\nabla I}$ and $O^L_{\nabla I}$ are the left and right maximal normal flows in the peripheral visual fields. The gain k_h is chosen empirically to produce a sensible desired heading in the visual space. Desired change in heading, $\Delta\theta$, is then calculated, accounting for the current gaze angle, θ_g, with respect to heading.

$$\Delta\theta = \theta_v + \theta_g \tag{13}$$

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

$$\dot{\theta} = Saturate\left(k_s \cdot \Delta\theta \cdot \frac{1}{T_b}, s \right) \tag{14}$$

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. The angular velocity is saturated at $\pm s$ deg/s to limit the peak rotation rate to reasonable levels. There are three reasons for this limit. First, the latency

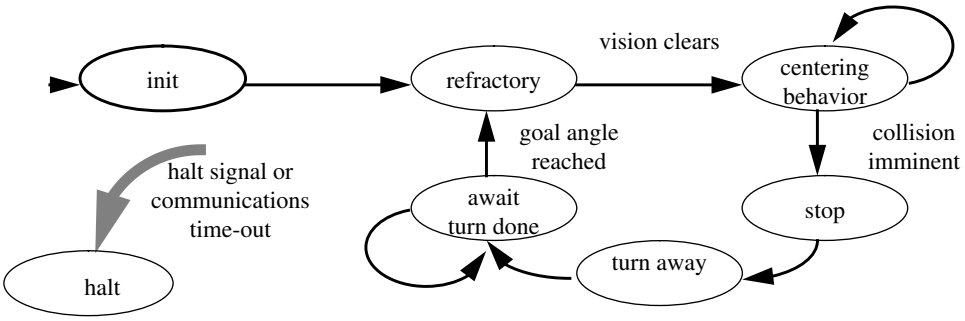

Figure 3. Body Control Automaton

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, since the controller might not be able to stop steering at the right time. Second, motion estimation suffers if the camera rotates too fast, since our computation of flow is based on the gradient method which limits motion to less than two pixels per frame. Poor flow measurements degrade T_c estimates. This impacts the quality of steering and stopping. Third, slower steering improves the temporal consistency of flow data for recursive estimation.

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 of image motion estimates. Also, in order for the centering behavior to work properly, the cameras should be translating approximately along the heading. In this way, the relationship between flow and range in the two peripheral fields will be the same. Therefore, the behavior and motor control systems must minimize 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 is not overstressed. Despite these precautions, gaze stabilization is imperfect and some data are contaminated.

When collision appears imminent, the robot stops and turns in the direction it had been steering when it detected collision. Experience suggests this strategy is too simple.

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 4). 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. With the cameras rotationally stabilized near the heading, the peripheral flow estimates can be interpreted as relative distances to obstacles.

It is important that the recursive estimators know when this quick phase motion occurs so they can reset since they will begin observing new sets of data.

8. Experiments and Results

Experiments with the obstacle avoidance system were conducted in a laboratory containing office furniture and robot and computing equipment. Furniture and equipment lined the walls and there was free space roughly 5 m by 3 m in the center of the lab. Office chairs provided obstacles.

In typical experiments, the robot began the trial at one end of the lab. Obstacles were placed in the cameras's views. The robot drove forward at 30 cm/s. As images were processed, time-to-contact was estimated. Figure 5 shows a sequence of images taken with the 40° field-of-view camera as the robot approached a stack of chairs. Figure 5a was taken at the beginning of the experiment, Figure 5b midway through the experiment, and Figure 5c shows the view when the robot detected imminent collision. Figure 6 plots the T_c recorded during such a trial against time, represented by sampling numbers. The time-to-contact dimension is not calibrated.

A typical trace of the robot's path using obstacle avoidance and wandering behaviors is plotted in Figure 7. The lab was lined with chairs forming a free space roughly 5 m by 3 m. As the dead-reckoned trace shows, the robot traversed most of the space in these five minutes of recorded data. Variability in steering and stopping depends in part on the textures visible from a particular approach. Stopping distances may vary just because the angle of approach varies. As time-to-contact and peripheral flows are computed, the robot moves forward toward open areas while centering itself in the open space. Upon detection of an imminent collision, it turns to avoid the obstacle and continues its wan-

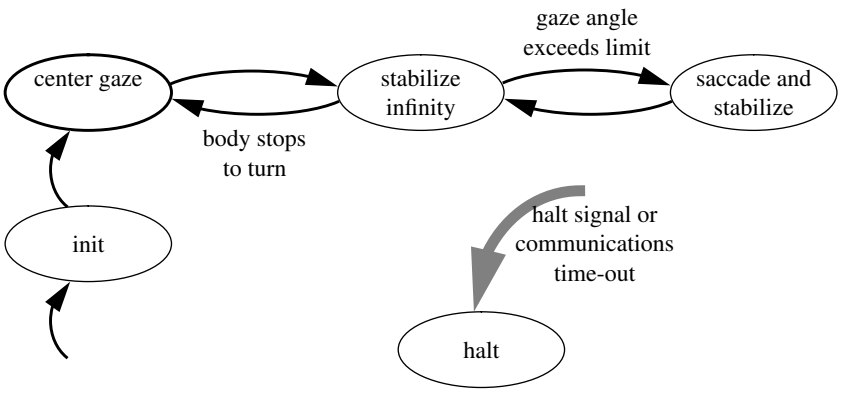

Figure 4. Gaze Control Automaton

(a) (b) (c)

Figure 5. Sequence of Images As Robot Approaches Obstacles

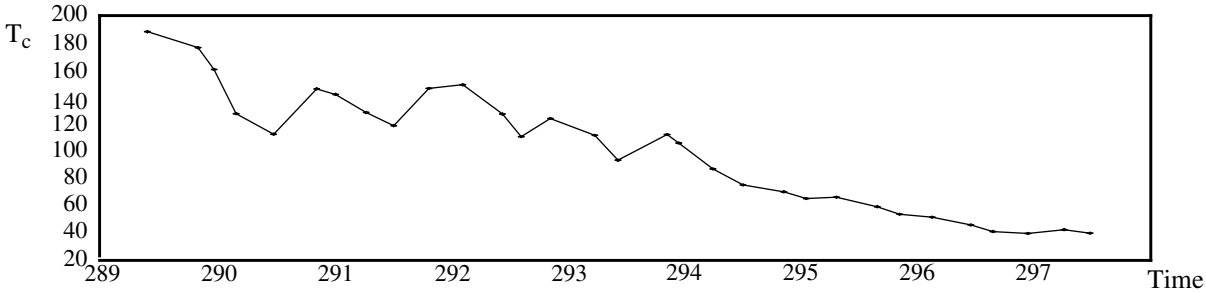

Figure 6. Time-to-contact versus Time

dering behavior. The system has run successfully for up to 20 minutes.

There are three primary factors that lead to system failure. First, there is a variable time delay between detection of an event and the behavioral response to this event. The variability is a function of intercommunication delays caused by processes (eye and robot controls) running part of the system in a UNIX environment. The effect of this variability can delay the robot from reacting quickly enough to avoid an obstacle. Second, the flow computation assumes that there will be sufficient texture in the field of view from which to compute flow. If this is not the case, no flow is detected and the robot collides with the obstacle. The third cause of failure is the rectangular geometry of the robot base and the presence of bumpers which trigger stopping upon contact. There are instances where the robot is turning to avoid an obstacle, and in so doing, brushes its bumper against a nearby object.

9. Discussion

Some researchers [11][19][20] have proposed using divergence or flow derivatives for visual cues, but they do not provide real-time implementations of these ideas. Nelson and Aloimonos [13][14] used directional flow divergence for stop-and-look obstacle avoidance (not real-time smooth driving). Also, their system did not simultaneously handle forward and lateral obstacle avoidance, under both translational and rotational robot motion. Finally, their environments were much simpler than ours and they did not demonstrate extensive robust behavior over extended periods of time.

Corridor following behaviors using wide field peripheral flow have been demonstrated with [5] and without [17] gaze stabilization. The choice depends on several factors. As noted, a rotationally stabilized visual frame may improve temporal integration of data. In addition, image flows may be interpreted directly as proximity estimates because the camera is approximately translating.

A system that does not rotationally stabilize the camera must cope with flows due to the camera rotation as well as its translation. The system of Santos-Victor *et al*. [17] works despite the rotational flows that could obscure the translational flows (which express the scene structure) because the situation is adequately constrained. The most important constraint is that the rotational flow magnitudes are small compared to the translational flows. This requires that the vehicle (to which the camera is fixed) turn slowly. The definition of slow depends on the visually apparent speed of the vehicle, since it depends on the image flows due to translation.

In practice most flow estimators have limited spatiotemporal range, and rapid camera rotation could easily contribute a large share of the image flows observed by a translating and rotating observer. This would significantly degrade the resulting steering of a system using flow estimators with limited dynamic range. The demonstration system presented here employs a derivative-based flow estimation technique whose dynamic range is fairly limited. Minimizing the rotational flows provides the best estimates of translational flows. Even small camera rotations can produce rotational flows that are large compared to translational flows.

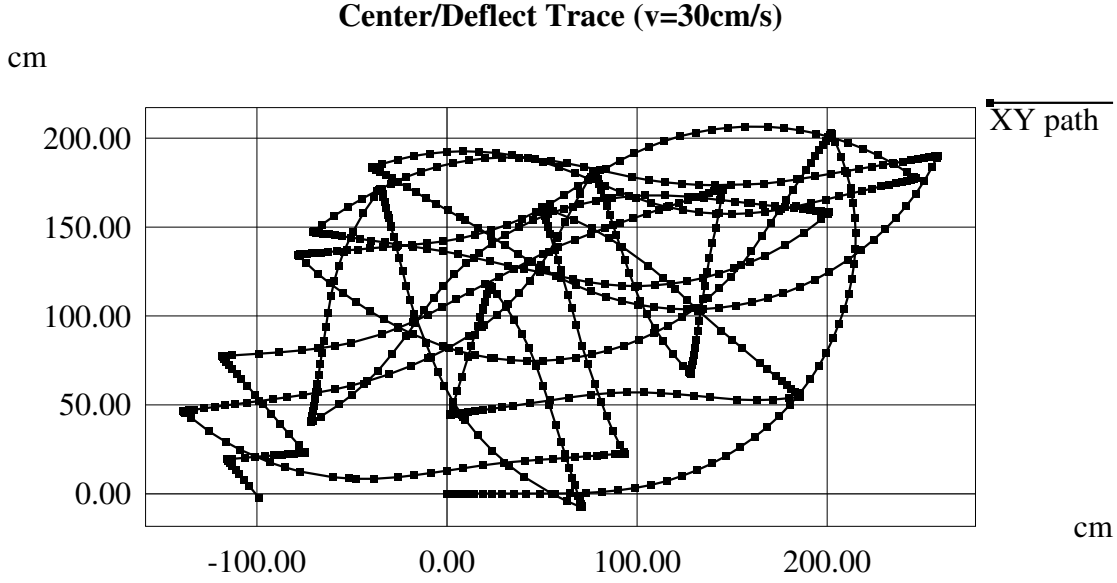

Figure 7. Robot's path

Corridor following systems suffer from a gaping central blind spot. The present work combines central and peripheral vision to avoid collision over the entire field for vision-based mobility.

System performance depends on many factors. Underlying the instantaneous T_c estimates are image flow measurements. Although divergence is theoretically unaffected by camera rotation, rotation contributes directly to image flow. We calculate image flow using the gradient method, which, like all techniques, has limited spatiotemporal sensitivity. In particular, large flows are not estimated accurately, so fast camera rotation can corrupt the image flow estimates on which the divergence and T_c estimates rely.

In addition, image flow depends on the field of view and scene texture. The obstacle must appear in the field of view in order to be detected. Since the normal flow computation performs best on textured scenes, the system fails in the absence of texture. Further, the real-time estimation of image flow on PIPE suffers from limited resolution in flow estimates. The translational components of flow which indicate the scene structure are very small near the heading direction, where T_c is directly related to divergence. Therefore T_c estimates are noisy, and it becomes necessary to detect trends in the data since we cannot base behavioral decisions on any single datum.

Sensorimotor interactions affect overall performance. There are significant latencies in the sensing, estimation, and control modules. These modules are distributed and to some extent asynchronous. Although in general the latency equals the 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. Normal flow estimates are produced at 7.5 Hz. In parallel, the divergence estimation task runs at a mean rate of 7 Hz, though the rate varies with the density of the normal flow data. On the same processor, the recursive estimator executes in 12 ms. The robot accepts speed and steering commands at about 3 Hz. Hence, at a robot velocity of 30 cm/s, visual data become available about every 5 cm of robot travel and steering is adjusted about every 10 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. 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 actually effecting a robot command, means it is difficult for the control (and sensing) systems to know precisely what the robot is doing. Consequently, the systems are designed to require only approximate knowledge of the robot's current motion state if they use any at all.

10. Conclusions

A demonstration system is presented that uses only real-time motion cues to wander while avoiding obstacles in a lab containing office furniture and robot and computing equipment. The robot has wandered around the lab at 30 cm/s for as long as 20 minutes without collision. To our

knowledge, this is the first such demonstration of real-time wandering using only image motion cues.

The paper describes how flow, divergence of flow, and maximal flows are computed in real-time to provide the robot's sense of space, and how steering, collision detection, and camera gaze control together accomplish safe wandering. The major contribution of this work is in developing and demonstrating this integrated system that robustly coordinates real-time behaviors in a complex environment.

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 and used as the sole perceptual cues to safely move about a complex environment in real-time for extended periods.

This work includes the integration of several visual mobility behaviors, including forward obstacle avoidance, lateral obstacle avoidance, and camera gaze control. It also demonstrates the integration of foveal/peripheral visual processing, image motion processing, retinal 2-D representations and behavior control. The manner of integration presented here is unique and represents an important contribution in integration methods. The approach requires fewer calibrations, fewer scene hypotheses, and less computation than standard techniques. It is therefore simpler and faster.

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

[4] G. Bierman. *Factorization Methods for Discrete Sequential Estimation*. Academic Press. New York, 1977.

[5] D. Coombs and K. Roberts. "Centering behavior using peripheral vision", In *Proc. of CVPR 1993, the IEEE Conference on Computer Vision and Pattern Recognition*, New York, June 15-17, 1993.

[6] D. Coombs, M. Herman, T. Hong, M. Nashman, "Real-Time Obstacle Avoidance Using Central Flow Divergence and Peripheral Flow", NIST Internal Report 5605, Gaithersburg, MD, February 1995.

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

[8] G. Goodwin and K. Sin. *Adaptive Filtering, Prediction and Control.* Prentice-Hall. Englewood Cliffs, New Jersey, 1984.

[9] B.K. Horn and B. Schunck. "Determining Optical Flow", *Artificial Intelligence Journal*, 17, 1981.

[10] E. Kent and M. Shneier and R. Lumia. "PIPE (Pipelined Image-Processing Engine)", *Journal of Parallel and Distributed Computing*, 2:50-78, 1985

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

[12] C. Lawson and R. Hanson. *Solving Least Squares Problems*. Prentice-Hall. Englewood Cliffs, New Jersey, 1974.

[13] 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.

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

[15] 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.

[16] A. Rosenfeld and A. Kak. *Digital Picture Processing*, Vol. 1, Second Edition, Academic Press, 1982.

[17] J. Santos-Victor, G. Sandini, F. Curotto, and S. Garibaldi. "Divergent stereo for robot navigation: Learning from bees", *Proceedings, CVPR 1993, IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (New York, NY, June 15-17) pp. 434-439, 1993.

[18] H. Schneiderman and M. Nashman. "Visual Tracking for Autonomous Driving", *IEEE Transactions on Robotics and Automation, special issue on Perception Based Real-World Navigation*, in press.

[19] 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.

[20] 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.

[21] A. Verri and T. Poggio. "Against quantitative optical flow", In *Proc. First International Conference on Computer Vision*, London, England, pp. 171-180, June 8-11, 1987.

[22] G.S. Young, T.H. Hong, M. Herman, and J.C.S. Yang, "New visual invariants for obstacle detection using optical flow induced from general motion," *Proc. IEEE Workshop on Applications of Computer Vision*, Palm Springs, CA, November 30-December 2, 1992, pp. 100-109.

Acknowledgments

The authors thank Henry Schneiderman for his help with the recursive estimation.