Real-Time Obstacle Avoidance Using an MPEG-Processor-based Optic Flow Sensor

Norbert O. Stöffler, Tim Burkert, and Georg Färber
Institute for Real-Time Computer Systems
Technische Universität München
D-80333 München, Germany
stoffler@rcs.ei.tum.de, www.rcs.ei.tum.de/∼stoffler/

Abstract

This paper describes a vision system for obstacle detection in mobile robot navigation. The system uses an image processing board equipped with an MPEG motion estimation processor that calculates a robust optic-flow-like vector field in real-time. This field is then evaluated by algorithms running in software on the host PC. As the solutions to the general problem of structure and motion from optic flow are too instable for the use in this application, the typical constraints of mobile robotics are exploited, i.e. a reduced set of motion parameters and a known ground plane. Egomotion can then be reconstructed with robust one dimensional methods. A new criterion for obstacles that copes well with the noise properties of the motion field is introduced. For vectors belonging to obstacles the 3D information is reconstructed allowing not only qualitative detection of obstacles but quantitative path planning.

1. Introduction

The work presented in this paper is part of a research project toward the development of autonomous, vision guided, mobile robots. Such robots must robustly sense and avoid obstacles while driving through unknown environments. One possible approach that is not impaired by, but even exploits the movement of the camera is to use the *optic flow*. 3D motion of the camera is projected onto this 2D velocity field (or displacement field in the case of isochronous sampling using standard video cameras) on the image plane (x, y) (see Fig. 1).

If the displacement vectors are regarded to be equivalent to the velocity vectors (an acceptable assumption for realistic frame rates and velocities [1]) and the focal length of the camera is normalized to 1 (a mathematical convenience that does not restrict generality), the optic flow can be expressed by the following well known equation:

$$\begin{bmatrix}\Delta x\\ \Delta y\end{bmatrix} = \frac{1}{Z}\begin{bmatrix}T_z x - T_x\\ T_z y - T_y\end{bmatrix} + \omega_x \begin{bmatrix}xy\\ y^2+1\end{bmatrix} - \omega_y \begin{bmatrix}x^2+1\\ xy\end{bmatrix} - \omega_z \begin{bmatrix}-y\\ x\end{bmatrix} \quad (1)$$

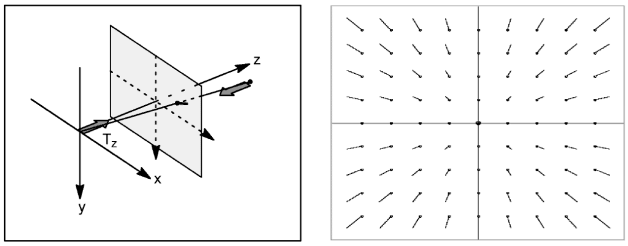

Figure 1. Coordinate system and optic flow

Before reconstructing the 3D parameters, this 2D vector field has to be estimated from the variation of brightness patterns in an image sequence I_t. The classical solutions to this problem include the evaluation of spatio-temporal derivatives, the tracking of single feature points and the correlation of small patches. The real-time optic flow calculation technique presented in section 2 is inspired by the similarity of those correlation techniques to the block-wise MPEG-1/2 motion compensation.

Many papers in the optic flow literature address the problem of the reconstruction of motion and depth parameters by solving Eqn. (1) for a set of vectors, some even perform segmentation of independently moving objects [1, 9, 10, 16]. Unfortunately, those complete solutions are computationally expensive and numerically instable. Graphically, this instability results from the similarity of optic flow fields generated by rotation and lateral translation [15]. The difference between those two field contributions is often hardly the magnitude of noise.

Published in: *Proc. 15th Int. Conf. on Pattern Recognition*, 161–166, Barcelona, E, Sept. 2000

Common approaches used by robotic researchers [3, 4, 5, 12] include the assumption of constraints to the motion parameters, for example to mere translation. Also qualitative detection without 3D reconstruction is proposed, avoiding the need of exact knowledge of the ego-motion.

The method for obstacle detection presented in section 3 exploits the constraints of the mobile robot application too, but works in presence of rotation, estimates the ego-motion, and recovers the 3D information of objects. It uses robust methods to cope with quantization noise and faulty vectors which are occasionally produced during the optic flow calculation.

Section 4 explains some important implementation details and section 5 shows the results of the obstacle detection and exemplary navigation.

2. Real-time optic flow sensor

According to the MPEG-1/2 standards, reduction of the temporal redundancy in an image sequence I_t is achieved by replacing pixel information by reference vectors to other images. For real-time encoding, specialized processors, so called *MEP*s (*Motion Estimation Processors*) have been developed, which calculate these reference vectors by block-matching. A reference block (*RB*, 16×16 pel) from image I_{rb} is compared with a search window (*SW*, 32×32 pel) in the image I_{sw}. For all possible offsets $\mu, \nu \in \{-8 \dots 7\}$, a correlation-like value called *SAD* (*Sum of Absolute Differences*) is calculated; the minimum designates the best match, and its position defines the reference vector:

$$SAD(\mu,\nu) = \sum_{\kappa=-8}^{7} \sum_{\iota=-8}^{7} |SW(\mu+\iota,\nu+\kappa) - RB(\iota,\kappa)|$$
$$SAD(\Delta x_{pel}, \Delta y_{pel}) = \min_{\mu,\nu \in \{-8 \dots 7\}} SAD(\mu,\nu) \qquad (2)$$

If consecutive images are compared (i.e. $sw - rb = 1$), the resulting set of vectors

$$\mathcal{F} = \{\boldsymbol{f}_i : \boldsymbol{f} = \begin{bmatrix}\Delta x \\ \Delta y\end{bmatrix}, i \in \{1 \dots n\}\}$$

can be regarded as optic flow.

The idea to use one of those extremely optimized MEPs for the generation of optic flow is not new. In particular Inoue et al. describe the integration of a MEP into their image processing transputer network [8]. Resulting from their work, a commercial version is available, and meanwhile is used in various research projects [2, 7].

A problem that several researchers report is that the optic flow generated by such a correlation processor can become very noisy. This happens when the image structure inside the *RB*s or *SW*s is ambiguous or completely missing. Then the detection of a significant minimum according to Eqn. (2) fails and faulty vectors are calculated. Fig. 2a illustrates the problem. The flow was generated by a linear forward movement of the camera, so all vectors should intersect in a single point, the *FOE* (*Focus Of Expansion*). Due to local lack of structure, many vectors point to completely different directions. Unfortunately, this effect dominates in most indoor environments. Further evaluation of such a flow field is virtually impossible.

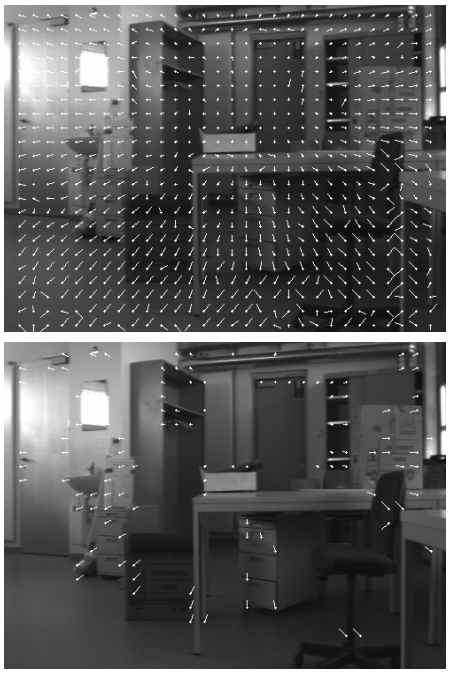

Figure 2. Optic flow, generated by a MEP: a) complete, b) sifted.

To solve this problem, we augmented the MEP with external circuitry that calculates an additional confidence value for each vector. In the simplest case, this confidence value can be tested against a fixed threshold to sift out the faulty flow vectors Fig. 2b.

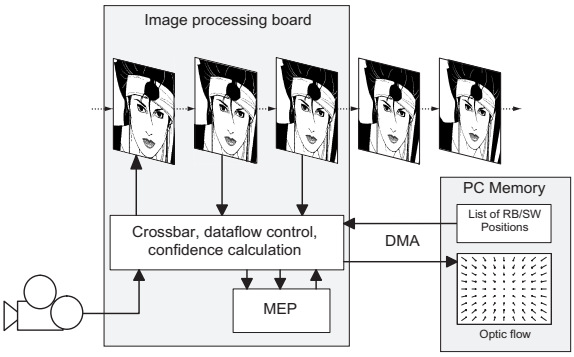

Figure 3. System structure

Our prototype system consists of an ISA image processing board containing the MEP, and a LINUX host PC (see Fig. 3). A ring-buffer of three frame memories allows the

comparison of two images and simultaneous acquisition of the next image into the third memory. Depending on the utilization of the MEP internal pipeline, up to 525 vectors can be calculated per frame (PAL, 25 Hz). To achieve the highest possible flexibility, the coordinates of *RB*s and *SW*s can be randomly set by the software running on the PC for each single matching operation. This feature is exploited by the implementation described in section 4. For a more detailed description of the hardware see [13].

3. Reconstruction of motion parameters and depth

The approach pursued in this work is to simplify the problem by allowing only two degrees of freedom for the camera motion. In this case Eqn. (1) can be solved for each vector individually, and robust one dimensional techniques can be used for the determination of the ego-motion. This simplification is possible in several applications, for example fixed pan-tilt surveillance cameras. When the camera is mounted on a mobile robot with non-holonomic kinematics the motion can also be decomposed into a translation T along the trajectory of the vehicle and a rotation ω around the vertical axis. To keep the formulas simple, it is also assumed that the camera is mounted with its x-axis horizontally and the y-z-plane in parallel to the heading of the vehicle. Describing the pitch-angle ρ of the camera by the two trigonometric shorthands $\alpha = cos(\rho)$ and $\beta = sin(\rho)$ Eqn. (1) is reduced to

$$\boldsymbol{f} = \frac{T}{Z}\boldsymbol{e} + \omega\boldsymbol{h} \tag{3}$$

with $\boldsymbol{e}$ representing the "expansive" part of the field contributed by translation and $\boldsymbol{h}$ representing the hyperbolic part contributed by rotation:

$$\boldsymbol{e} = \begin{bmatrix} \alpha x \\ \alpha y + \beta \end{bmatrix}, \boldsymbol{h} = \begin{bmatrix} \alpha(x^2+1) - \beta y \\ \alpha x y + \beta x \end{bmatrix}$$

This equation can be solved for each vector:

$$\omega = \frac{\alpha x \Delta y - (\alpha y + \beta)\Delta x}{\alpha\beta y^2 - \alpha^2 y + \beta^2 y - \alpha\beta} \tag{4}$$

$$\frac{T}{Z} = \frac{\Delta x - \omega(\alpha(x^2+1) - \beta y)}{\alpha x} \tag{5}$$

The first result ω should correspond to the rotational velocity of the vehicle and therefore be identical for each vector. Thus, the rotation could be estimated by a simple least square fit over all ω_i as proposed e.g. in [4]. As even the sifted flow field sometimes contains faulty vectors (for example due to cyclic patterns in the images), this can yield poor results in practice. Better performance is achieved by using robust mode estimators with high breakdown points. The *shortest half window* mode estimator (*shorth*, [11]) can tolerate up to 50 % outliers and performs very well in the experiments. Also it is computationally inexpensive:

$$\begin{aligned} \xi_j &: \xi_j \le \xi_{j+1}, j \in \{1 \dots n-1\} \\ \delta_j &= (\xi_{j+\lfloor \frac{n}{2} \rfloor} - \xi_j) \\ \delta_m &= \min_{j \in \{1 \dots n - \lfloor \frac{n}{2} \rfloor\}} \delta_j \\ \mathrm{shorth}(\xi_j) &= \frac{1}{2}(\xi_m + \xi_{m+\lfloor \frac{n}{2} \rfloor}) \qquad (6) \\ \hat{\omega} &= \mathrm{shorth}(\omega_i) \qquad (7) \end{aligned}$$

A foreground/background separation technique that can be used to get an estimate for ω even in the presence of large moving objects (larger than 50% of the field) has already been published in [14].

The second solution to Eqn. (3), $\frac{T}{Z}$, still depends on the depth of the corresponding 3D point and thus has an individual value for each vector (it is the reciprocal value of the so called *Time To Collision*, *TTC*). To get also an estimate for the translation T, additional assumptions have to be made. In the case of a camera mounted on a mobile robot, beside the pitch angle also the height h of the camera above the ground plane is known. If the robot moves on an infinite ground plane without any obstacles, the Z coordinate can be calculated for each point in the image plane:

$$Z = \frac{h}{\alpha y + \beta} \tag{8}$$

Of course this postulate cannot be made for a system which is intended to detect obstacles. But as a vehicle always has a minimum braking distance, it is reasonable to assume the absence of obstacles in the area directly in front of the robot. This area is called *ground window* in the following and the set of vectors inside this window is $\mathcal{G}$.

By inserting Eqn. (8) into Eqn. (5) one could get a value T for each vector. But, using this method, the influence of the pixel quantization would vary over the image plane. Also remaining faulty vectors pollute the estimation. Therefore, we propose a different technique: First, the optic flow field is derotated using $\hat{\omega}$:

$$\boldsymbol{d} = \boldsymbol{f} - \hat{\omega}\boldsymbol{h} \tag{9}$$

Each derotated vector $\boldsymbol{d}$ should then coincide with an epipolar line through the *FOE* and thus be collinear with $\boldsymbol{e}$. In practice it deviates a little from this epipolar line due to the quantization error, or completely in case of a faulty vector. The first category of errors should be corrected as good as possible, the second category should be detected and eliminated from $\mathcal{F}$. For the first purpose, the projection of each vector $\boldsymbol{d}$ on its epipolar line is used for the calculation of $\frac{T}{Z}$:

$$\frac{T}{Z} = \frac{\boldsymbol{e}^\top \boldsymbol{d}}{\|\boldsymbol{e}\|^2} \tag{10}$$

For the detection and elimination of faulty vectors the distance of the vector from the epipolar line can be calculated and thresholded:

$$\mathcal{F}^* = \{\boldsymbol{f}_i : |\frac{\boldsymbol{e}^\top \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \boldsymbol{d}}{\|\boldsymbol{e}\|}| < \lambda \wedge \frac{T}{Z} > 0\} \tag{11}$$

Then $\hat{T}$ is determined using the *shorth* estimator again, which allows up to 50% obstacles even inside the ground window:

$$\hat{T} = \mathrm{shorth}(\frac{\boldsymbol{e}_i^\top \boldsymbol{d}_i}{\|\boldsymbol{e}_i\|^2} Z_i), \boldsymbol{d}_i \in \mathcal{G} \cap \mathcal{F}^* \tag{12}$$

Now Z could be calculated for each vector and the distance from the ground plane could be used to detect obstacles. Another criterion would be the ratio between the vector length and the length of a predicted ground vector, as proposed in [5]. Both approaches have the drawback, that the impact of pixel quantization is stronger for short vectors. Again, a better criterion for obstacles is

$$\mathcal{O} = \{\boldsymbol{f}_i : \frac{\boldsymbol{e}_i^\top \boldsymbol{d}_i}{\|\boldsymbol{e}_i\|} - \frac{\hat{T}}{Z_i}\|\boldsymbol{e}_i\| > \tau\} \tag{13}$$

Both criteria according to Eqn. (11) and Eqn. (13) calculate *distances in image coordinates*, which can be compared to fixed thresholds λ and τ in the magnitude of the pixel quantization.

To get rid of single bogus detections which have passed sifting and elimination, we also apply a simple morphological operation to $\mathcal{O}$. For the remaining vectors $\boldsymbol{f} \in \mathcal{O}$ finally the Z-coordinate is calculated to get a 3D obstacle point.

4. Implementation

As a standard video camera is used, only fields (not frames) can be evaluated due to the interlacing problem. Additional considerations have to be made when implementing an algorithm according to section 3 using the optic flow sensor presented in section 2.

4.1. Placement

Placing the *RB*s along a regular grid in the image plane would result in detailed surveillance of the area close in front, but only few information in the distance. Besides, the calculation of $\frac{T}{Z}$ according to Eqn. (10) becomes instable for small values of $\|\boldsymbol{e}\|$. Thus, *RB*s should not be placed inside an area below a threshold for $\|\boldsymbol{e}\|$, which results in a practically blind spot in the upper middle of the image plane (that graphically can be considered the intersection of a cone around the *FOE* with the image plane). Fig. 4 shows the chosen optimized placement of the vectors and the position of the ground window. Unfortunately this non-regular placement of the *RB*s prevents the optimal use of the MEPs internal pipeline, reducing the number of possible vector calculations to 350 (and even a little less in practice due to the latencies of the LINUX operating system). Also morphological operations become more complicated but can be accelerated by a precalculated lookup table containing neighborhood information.

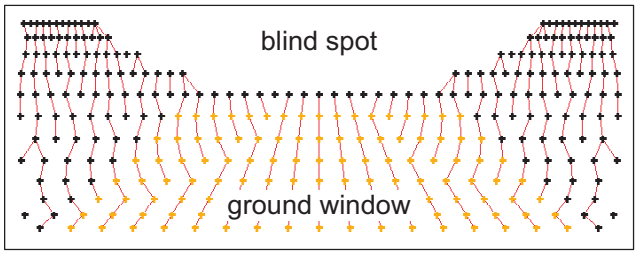

Figure 4. Optimized placement of the vectors

4.2. Biasing

As the vector length generated by realistic motion of the robot significantly exceeds the maximal vector length that can be calculated by the MEP, the positions of the *SW*s have to be biased against the positions of the *RB*s. This is achieved by using the current motion estimates $(\hat{\omega}, \hat{T})$ to predict the field for the next operation cycle using Eqn. (3). Each *SW* is then centered around the center of the corresponding *RB* plus a bias vector $\boldsymbol{b} = \boldsymbol{f}(\hat{\omega}, \hat{T})$ (see Fig. 5). This biasing technique transforms the limitations on the velocities to limitations on the accelerations.

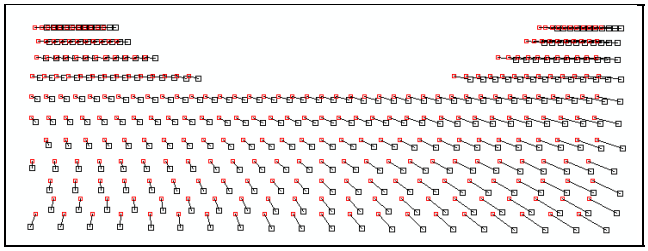

Figure 5. Search window bias

4.3. Rate control

Another problem is, that at realistic velocities the translational component $\frac{T}{Z}\boldsymbol{e}$ of most of the vectors is too short to significantly exceed the pixel quantization error. Unfortunately it is not possible to just skip $(r-1)$ images in order to reduce the frame rate to $\frac{1}{r}$, which would increase the vector length. The reason is, that the field is dominated by the rotational component $\omega \boldsymbol{h}$, which has to be estimated at maximal rate.

Our solution is to vary the length l of the hardware ringbuffer (Fig. 3): I_{rb} is repeatedly "frozen" and compared to the next r images according to Fig. 6. Using this method, no image is lost and it is possible to estimate $\hat{\omega}$ with every frame, but determine $\hat{T}$ and obstacles at a randomly chosen lower rate.

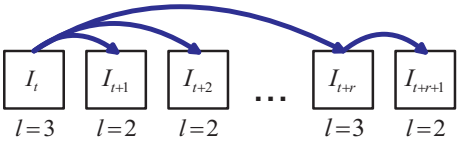

Figure 6. Rate control

5. Obstacle avoidance

Fig. 7 shows some snapshots from the experiments. Vectors corresponding to the detected obstacle points are marked with black squares, the image in the middle shows a rotation $\omega \approx 30°/\text{s}$.

Figure 7. Obstacle detection examples

To generate a local 2D obstacle map for robot navigation, the 3D points are transformed into the robot coordinate system. Fig. 8 corresponds to the bottom image of Fig. 7; obstacle points belonging to the chair and the railing can be seen. The magnitude of the accuracy of 3D reconstruction is 10 cm, which is by far sufficient for obstacle avoidance.

Also a short history of obstacle sets is fused to increase accuracy and robustness. This is achieved by transforming older sets of obstacle points into the current robot coordinate system. The parameters $\Delta x_{robot}, \Delta y_{robot}, \Delta\varphi_{robot}$ for this transformation are calculated from the optic flow by integrating the estimated motion parameters $\hat{\omega}, \hat{T}$. Obstacles

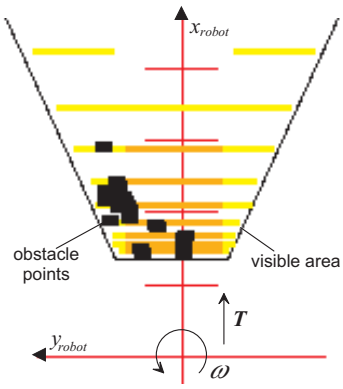

Figure 8. Obstacle map in 2D robot coordinates

are then fused by increasing probabilities in an occupancy grid.

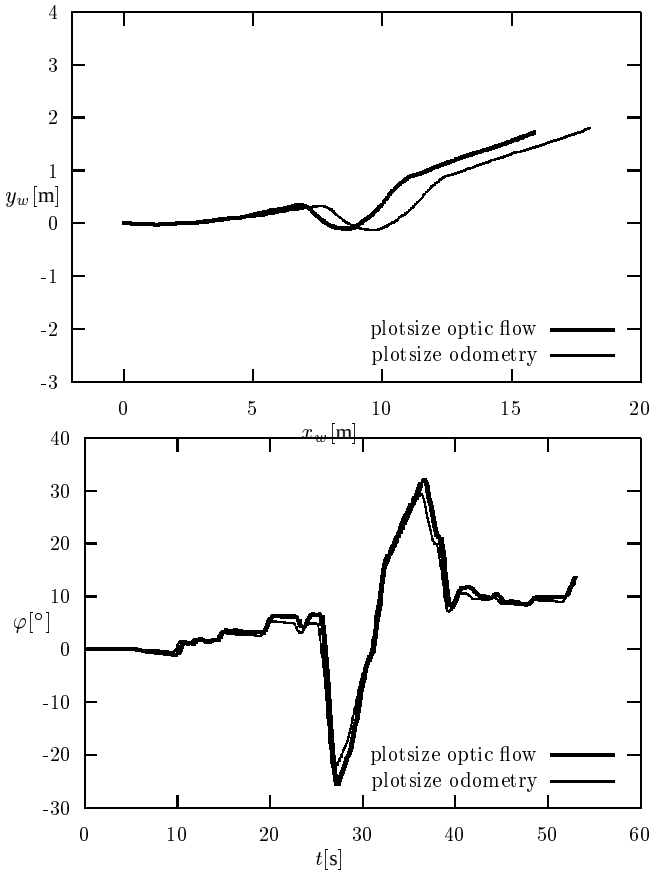

Figure 9. Trajectory and heading of the robot

Robot control is currently performed by a very simple navigation algorithm inspired by the dynamic window approach [6], which periodically (typically every 200 ms) chooses the best trajectory in terms of distance to obsta-

cles and closeness to the next way point. For fast evaluation the set of possible trajectories (circular lanes of the robots width) is precalculated for each possible turn radius $R = \frac{T}{\omega}$ and the set of corresponding map elements is stored in lookup tables. Fig. 9 depicts the result of the navigation experiment, from which the bottom snapshot of Fig. 7 was taken. The "way point" was fixed at $x_{robot} = 5\,\mathrm{m}$ and $y_{robot} = 2\,\mathrm{m}$, providing an "unreachable goal" on the left side. Together with the obstacles in the occupancy grid this results in the behavior to follow a wall on the left.

At the beginning of this experiment, the robot followed the bow of the railing. After approximately 7 m it detected the chair, rounded it, and then returned to the railing again. The figures show the absolute position of the robot in the world coordinate system as well as the heading φ over time. Both plots compare values calculated by the odometry with values calculated by integrating the motion parameters from optic flow. The drift is small enough to use the optic flow values for local calculations like the history fusion.

6. Conclusion

We have presented a real-time vision system consisting of a MEP-based optic flow sensor and a set of robust evaluation techniques to do obstacle detection for a mobile robot. The robot MARVIN (Mobile Autonomous Robot with Vision-based Navigation, see Fig. 10) currently uses the system for navigating through the corridors of our laboratory at speeds of $T = 0.4\,\mathrm{m/s}$ and $\omega_{max} = 30^\circ/\mathrm{s}$. The typical pitch-angle of 25° delivers obstacles from a minimal distance of 1.5 m to a maximal distance of 3.5 m in the central area and 8 m in the peripheral areas.

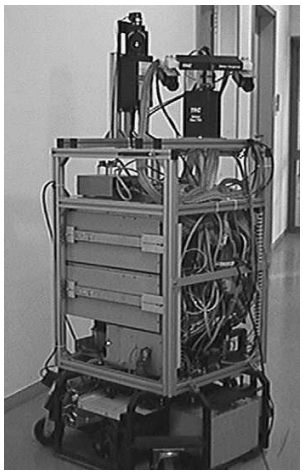

Figure 10. MARVIN

In similar robotic systems, only qualitative scene descriptions are derived from optic flow, which allow only qualitative navigation approaches [2, 3, 4, 12]. Since the result of our obstacle detection approach is a local occupancy grid, standard robot navigation techniques can be implemented.

References

[1] G. Adiv. Determining Three-Dimensional Motion and Structure from Optical Flow Generated by Several Moving Objects. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 7(4):319–336, July 1985.

[2] G. Cheng and A. Zelinsky. Real-Time Visual Behaviours for Navigating a Mobile Robot. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'96)*, pages 973–980, Osaka, J, Nov. 1996.

[3] D. Coombs, M. Herman, T. Hong, and M. Nashman. Real-time obstacle avoidance using central flow divergence and peripheral flow. In *Proc. 5th Int. Conf. on Computer Vision*, pages 276–283, Cambridge, MA, USA, June 1995.

[4] A. Dev, B. J. A. Kröse, and F. C. A. Groen. Navigation of a mobile robot on the temporal development of the optic flow. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'97)*, pages 558–563, 1997.

[5] W. Enkelmann. Obstacle detection by evaluation of optical flow fields from image sequences. In O. Faugeras, editor, *Proc. 1st European Conf. on Computer Vision (ECCV'90)*, volume 427 of *Lecture Notes in Computer Science*, pages 134–138, Antibes, F, Apr. 1990. Springer-Verlag.

[6] D. Fox, W. Burgard, and S. Thrun. The Dynamic Window Approach to Collision Avoidance. *IEEE Trans. on Robotics and Automation*, 4(1), Mar. 1997.

[7] M. Inaba, K. Nagasaka, F. Kanehiro, S. Kagami, and H. Inoue. Real-Time Vision-Based Control of Swing Motion by Human-form Robot using the Remote-Brained Approach. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'96)*, pages 15–22, Osaka, J, Nov. 1996.

[8] H. Inoue, T. Tachikawa, and M. Inaba. Robot Vision System with a Correlation Chip for Real-time Tracking, Optical Flow and Depth Map Generation. In *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA'92)*, pages 1621–1626, 1992.

[9] H. C. Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293(5):133–135, 1981.

[10] L. Matthies, R. Szeliski, and T. Kanade. Kalman Filter-based Algorithms for Estimating Depth from Image Sequences. *Int. J. Computer Vision*, 3(3):209–236, 1989.

[11] P. Meer. Robust Techniques for Computer Vision. Tutorial at the 14th Int. Conf. on Pattern Recognition, Brisbane, QLD, AU, Aug. 1998.

[12] J. Santos-Victor and G. Sandini. Visual-Based Obstacle Detection: A purposive approach using the normal flow. In U. R. et al., editor, *Intelligent Autonomous Systems*. IOS Press, 1995.

[13] N. O. Stöffler and G. Färber. An Image Processing Board with an MPEG Processor and Additional Confidence Calculation for Fast and Robust Optic Flow Generation in Real Environments. In *Proc. Int. Conf. on Advanced Robotics (ICAR'97)*, pages 845–850, Monterey, California, USA, July 1997.

[14] N. O. Stöffler and Z. Schnepf. An MPEG-Processor-based Robot Vision System for Real-Time Detection of Moving Objects by a Moving Observer. In *Proc. 14th Int. Conf. on Pattern Recognition*, pages 477–481, Brisbane, Australia, Aug. 1998.

[15] J. Weng, N. Ahuja, and T. S. Huang. Optimal Motion and Structure Estimation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 15(9):864–884, Sept. 1993.

[16] J. Weng, T. S. Huang, and N. Ahudja. Motion and Structure from Two Perspective Views: Algorithms, Error Analysis, and Error Estimation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 11(5):451–476, May 1989.