An MPEG-Processor-based Robot Vision System for Real-Time Detection of Moving Objects by a Moving Observer

Norbert O. Stöffler and Zoltan Schnepf
Laboratory for Process Control and Real-Time Systems
Technische Universität München
D-80333 München, Germany
stoffler@lpr.ei.tum.de, www.lpr.ei.tum.de/∼stoffler/

Abstract

This paper describes a PC-based vision system that can be used to detect moving objects from a mobile robot. An image processing board equipped with an MPEG motion estimation processor calculates a sparse but robust optic flow in real-time. An algorithm to evaluate this kind of optic flow has been realized in software. It determines relevant motion parameters and a simple scene interpretation in terms of moving object regions. The image processing board and the algorithm are presented in some detail; the performance of the system is demonstrated by experiments.

1. Introduction

Using our experimental robot MARVIN (Mobile Autonomous Robot with VIsion-based Navigation, see Fig.1), we are currently working on the autonomous exploration of office-type environments. Primarily *vision* is used to build 3D world models [2]. Important cues to the interpretation of an observed scene can be obtained by the evaluation of motion. Moving objects (typically people in the above-mentioned environment) have to be distinguished from static parts of the scene on the one hand and have to be taken into account for dynamic motion planning on the other hand. This detection of moving objects has to take place during the motion of the robot itself.

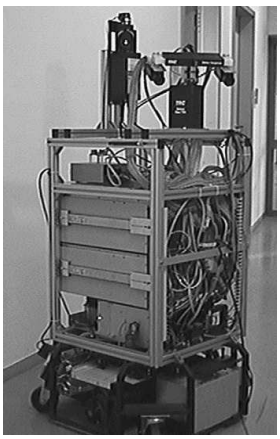

Figure 1. *MARVIN*

Three-dimensional motion is projected onto a two-dimensional velocity field (or displacement field in the case of isochronous sampling) on the image plane (x, y) of the camera. This vector field is commonly termed *optic flow* (or more precisely *image flow*). 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

$$F = \{\vec{f}_1 ... \vec{f}_i ... \vec{f}_n, \vec{f} = \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix}\}$$

can be calculated according to the following:

$$\begin{aligned} \vec{f}(x, y, Z, T_x, T_y, T_z, \omega_x, \omega_y, \omega_z) = \\ \frac{1}{Z} \begin{pmatrix} T_z x - T_x \\ T_z y - T_y \end{pmatrix} + \omega_x \begin{pmatrix} xy \\ y^2 + 1 \end{pmatrix} \\ -\omega_y \begin{pmatrix} x^2 + 1 \\ xy \end{pmatrix} - \omega_z \begin{pmatrix} -y \\ x \end{pmatrix} \end{aligned} \quad (1)$$

Before reconstructing the 3D motion parameters, this 2D vector field has to be estimated from the variation of brightness patterns on the projection plane.

Section 2 of this paper describes the architecture of an image processing system for real-time estimation of the optic flow that is motivated by the similarity of optic flow calculation and the so-called *motion compensation* defined by the MPEG video compression standards. As the properties of the resulting optic flow restrict the possibilities of further evaluation, Section 3 proposes techniques to reconstruct the relevant motion parameters and a simple scene interpretation.

Published in: *Proc. 14th Int. Conf. on Pattern Recognition*, 477–481, Brisbane, AU, Aug. 1998

2 Real-time optic flow sensor

The MPEG compression standards use the spatio-temporal redundancy in an image sequence I_t for bandwidth reduction. If possible, only displacement vectors are transmitted instead of the complete pixel information. Although the MPEG standards do not regulate how these vectors have to be computed, the state-of-the-art technique is correlation. As correlation is computationally expensive, specialized processors, so called *MEP*s (*Motion Estimation Processors*) have evolved from this area. A reference block (*RB*, 16x16 pel) from image I_r is compared with a search window (*SW*, 32x32 pel) in the image I_s. For all possible offsets $\mu, \nu \in \{0 \ldots 15\}$, a correlation-like value called *SAD* (*Sum of Absolute Differences*) is calculated; the minimum designates the best match, and its position defines the displacement vector:

$$SAD(\mu,\nu) = \sum_{\kappa=0}^{15} \sum_{\iota=0}^{15} |SW(\mu+\iota,\nu+\kappa) - RB(\iota,\kappa)| \quad (2)$$

$$SAD(\mu_{min},\nu_{min}) = \min_{\mu,\nu \in \{0\ldots15\}} SAD(\mu,\nu)$$
$$\Rightarrow \vec{f}_{pel} = \begin{pmatrix} \Delta x_{pel} \\ \Delta y_{pel} \end{pmatrix} = \begin{pmatrix} \mu_{min} - 8 \\ \nu_{min} - 8 \end{pmatrix} \quad (3)$$

If consecutive images are compared (i.e. $s - r = 1$), the resulting vector field 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 [5]. Resulting from their work, a commercial version is available, and meanwhile is used in various research projects [3, 4].

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 of the RBs or SWs is ambiguous or completely missing. Then the detection of a significant minimum according to Eqn. 3 fails.

Fig. 2 a 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, most vectors point to completely different directions. Unfortunately, this effect dominates in most office-type environments. Further evaluation of such a flow field is virtually impossible.

To solve this problem, we augmented a MEP with external circuitry that calculates an additional confidence value for each vector [7]. In the simplest case, this confidence value can be tested against a fixed threshold to sift out the noisy flow vectors. A drawback is the sparseness of the remaining field (see Fig. 2 b).

Our prototype system consists of an ISA image processing board containing the MEP, and a LINUX host PC (see Fig. 3).

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

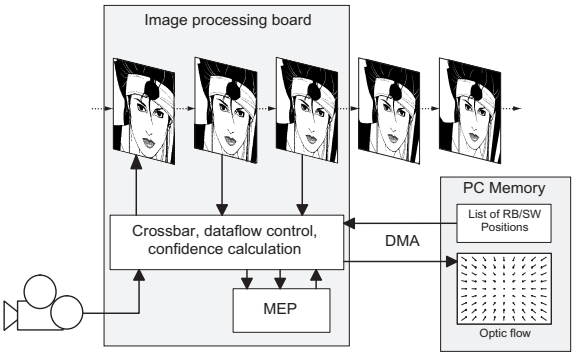

Figure 3. Optic flow sensor: System structure.

Three frame memories allow 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). For applications like tracking multi-

ple objects or big displacements (as typically induced by camera rotation) a fixed block-raster is not adequate. To achieve the highest possible flexibility, the coordinates of RBs and SWs can be randomly set by the software running on the PC for each single matching operation. A list of positions $\{(x_{RB_{pel}}, y_{RB_{pel}}, x_{SW_{pel}}, y_{SW_{pel}})_j\}$ is read via DMA, the list of results $\{(\Delta x_{pel}, \Delta y_{pel}, confidence)_i\}$ is written back by DMA again. This saves most of the CPU's processing power for the application software that evaluates the generated flow fields as described in the next section. For a more detailed description of the hardware see [7].

3 Detection of moving objects by optic flow segmentation

Many papers in the literature of optic flow address the problem of object segmentation and motion parameter reconstruction. Algorithms to calculate all five parameters (the absolute value of the translation vector gets lost during the 3D to 2D projection) were proposed for example by Prazdny [6] or Weng et al. [8]. Adiv presents an elegant approach to solve the segmentation problem for all five parameters: Objects are considered to consist of planar surfaces, so vectors can be clustered in an 8D (5D for the motion and 3D for the surface parameters) Hough space [1].

Applying these techniques to optic flows calculated by our sensor system produces no satisfying results, though. Responsible for the failure of these very general approaches is the numerical instability of the closed form solution to Eqn. 1 along with the strong quantization errors of the calculated vectors.

The reason for this failure can also be graphically deduced by the similarity of fields generated by mere lateral and mere rotational motion. Though all vectors are parallel in the translational case, and aligned with hyperbolas (according to Eqn. 1) in the rotational case, the difference has the same order of magnitude as the quantization effects. Fig. 4 demonstrates the problem for realistic camera parameters and constant depth.

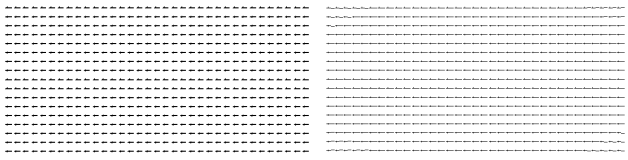

Figure 4. Optic flow: a) horizontal translation b) vertical rotation.

Therefore, a general solution to the complete motion recovery (i.e. segmentation of moving objects and determination of all 5 parameters for each object) seems impossible in our context.

More pragmatic approaches that cluster flow vectors along their 2D properties as for example proposed by Yamamoto et al. [9], produced better results, but are difficult to adapt for a moving observer and cannot be used for objects moving along the optical axis.

3.1 Determination of ego-motion

Because of the above problems, we introduce some simplifications to the general approaches that are inspired by the requirements of our application. Because the camera is mounted on a mobile robot with non-holonomic kinematics, only one translational and one rotational parameter remain. Without further restricting generality, but to simplify the equations, it is assumed that the camera is mounted horizontally. In camera coordinates, there is $T_x, T_y, \omega_x, \omega_z \equiv 0$ and the general equation of the optic flow (Eqn. 1) can be reduced to the following:

$$\vec{f_i} = \begin{pmatrix} \Delta x_i \\ \Delta y_i \end{pmatrix} = \frac{T_z}{Z_i} \begin{pmatrix} x_i \\ y_i \end{pmatrix} - \omega_y \begin{pmatrix} x_i^2 + 1 \\ x_i y_i \end{pmatrix} \quad (4)$$

As only two unknown variables ($\frac{T_z}{Z}$ and ω_y) remain, Eqn. 4 can be solved for each vector as follows:

$$\frac{T_z}{Z}_i = \Delta y_i \frac{x_i^2 + 1}{y_i} - \Delta x_i x_i \quad (5)$$

$$\omega_{y_i} = -\Delta x_i + \frac{x_i}{y_i} \Delta y_i \quad (6)$$

The first result $\frac{T_z}{Z}$ is a measurement of the distance of the corresponding 3D point and thus has an individual value for each vector (it is the reciprocal value of the *Time To Collision, TTC*).

The second result ω_y should correspond to the rotational velocity of the camera and therefore be identical for each vector. Thus, the rotation of the camera can be estimated by a simple mean value calculation:

$$\hat{\omega}_y = \bar{\omega_{y_i}} = \frac{1}{n} \sum_{i=1}^{n} (-\Delta x_i + \frac{x_i}{y_i} \Delta y_i) \quad (7)$$

Experiments on MARVIN in a real office-type environment have demonstrated the feasibility of the proposed estimation approach for ω_y. When employing the *median* instead of the simple *mean* value, the robustness of the calculated value against remaining faulty vectors can be increased further. Fig. 5 shows a comparison of the estimated $\hat{\omega}_y$ and the one calculated by the odometry of the robot.

Since the resulting vector length significantly exceeds the maximal vector length of the MEP, even for moderate turning rates of the robot ($\omega_y \approx 0.2 \frac{rad}{s}$ in Fig. 5), the estimated $\hat{\omega}_y$ must be used to bias the positions of the SWs accordingly.

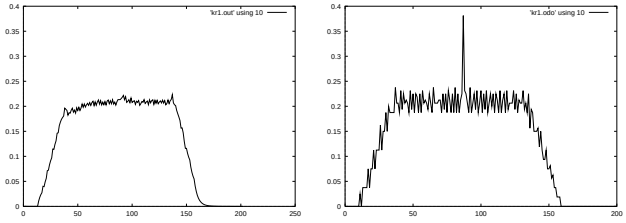

Figure 5. Comparison of a) the estimated $\hat{\omega}_y$ and b) the corresponding odometric value

Simply using the last $\hat{\omega_y}_{t-1}$ to calculate a bias vector

$$\vec{b} \quad = \quad -\hat{\omega_y}_{t-1} \begin{pmatrix} x^2+1 \\ xy \end{pmatrix}$$

transforms the limitation of the rotational velocity to a limitation of the rotational acceleration. Thus, a good estimate $\hat{\omega}_y$ is essential for the overall operation of the vision system.

3.2 Segmentation of moving objects

In the case of one or more moving objects in the observed scene the determination of the rotation has to be modified. The ω_{y_i} of vectors belonging to a moving object no longer coincide with the ω_y of the camera. Segmentation can be performed by clustering vectors according to their ω_{y_i}. A simple cluster algorithm has proven to be sufficient in the experiments. Vectors are sorted by increasing values of ω_{y_i}, so

$$\forall \mu < \nu \rightarrow \omega_{y_\mu} \leq \omega_{y_\nu}$$

Then they are combined to form a set of clusters $\{C_1 \dots C_j \dots C_m\}$ as long as they satisfy the criterion $\omega_{y_{i+1}} - \omega_{y_i} \leq \epsilon$. Therefore $C_j = \{\vec{f_i} \mid \omega_{y_{min_j}} \leq \omega_{y_i} \leq \omega_{y_{max_j}}\}$.

These m clusters build the first hypotheses for the segmentation of the optic flow. Unfortunately, not each cluster corresponds to one moving object, because the assumptions that led to Eqn. 4 are only true for a restricted camera motion, i.e. for the vectors belonging to the background. So in this step, only a classification *background* versus *not background* can be made for each cluster. Good results can be achieved by declaring the cluster $C_{j=b}$ as background that contains the set of vectors with the largest spatial variance ξ. This approach is inspired by the observation that the background is the only "object" not corresponding to a compact image region.

Since, as in the live experiments only the first field of each frame is used, the variance ξ in x direction is much more meaningful than in y direction and can be used exclusively:

$$\xi_j \quad = \quad \sigma^2(x_i)_{\vec{f_i} \in C_j}$$

$$\xi_b = \max_{j \in \{1 \dots m\}} \xi_j \quad \Rightarrow \quad C_b =: background$$

Taking into account only the vectors of C_b, the estimate for the camera rotation can again be calculated according to Eqn. 7. Fig. 6 demonstrates the performance of the background detection statistically using typical image sequences containing one to several moving objects. In this experiment, the camera is not moving, i.e. the estimated $\hat{\omega}_y$ should be 0. Each dot corresponds to one flow field F. The abscissa depicts the percentage of flow vectors belonging to the background, the ordinate the estimated ω_y (here in $\frac{deg}{s}$). If more than 50% of the background is visible, the median of all ω_{y_i} coincides with the actual ω_y. Below this percentage, the median of the background cluster C_b still delivers a satisfying result.

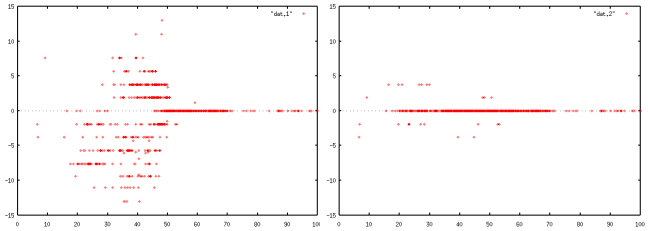

Figure 6. a) Median of all ω_{y_i}, b) Median of $\omega_{y_i} \in C_b$.

If the size of the background candidate does not exceed a certain threshold, no statement can be made at all and no estimate is produced in this case.

In closed loop operation on MARVIN, the accuracy of the estimated rotation is sufficient to calculate suitable SW-bias vectors for a robust system operation.

3.3 Object segmentation

According to the aforementioned considerations, the set of vectors $O = F \setminus C_b$ belongs to the *foreground*, i.e. to moving objects. To designate volatile image regions that should be excluded from static scene reconstruction, this set is sufficient. But, for navigation purposes, a further clustering of this set to individual objects $\{O_1 \dots O_l\}$ is desirable. This can be achieved by a spatial clustering of the vectors $\vec{f_i}$. For this purpose, the same cluster algorithm as described in Section 3.2 is applied first to $x_{i,\vec{f_i} \in O}$. The resulting Clusters $O_{x_k} = \{\vec{f_i} \mid \vec{f_i} \in O \wedge x_{min_k} \leq x_i \leq x_{max_k}\}$ can then again be clustered along $y_{i,\vec{f_i} \in C_{x_k}}$. As in the typical application, moving objects (people) never appear on top of each other, this second clustering step is skipped in the real-time implementation. Instead, the image regions belonging to moving objects are represented by their bounding boxes $\{(x_{min_k}, y_{min_k}, x_{max_k}, y_{max_k})\}$. By keeping track

Figure 7. Segmentation results (PAL, first field only)

of those bounding boxes in a list, hypotheses of object regions can also be maintained when the detection fails for a short time, i.e. when a moving object temporarily stops.

Fig. 7 shows samples from a typical image sequence used in the experiments. The camera (i.e. the robot) is moving along a slight left turn while a person crosses the scene. Vectors not belonging to the background are robustly detected and clustered. The bounding boxes can, of course, only encompass those parts of the moving objects where optic flow vectors have been calculated and have passed the sifting process.

4 Conclusion and further work

We have presented a low cost but efficient image processing system that is able to calculate a sparse optic flow in real-time (up to 525 vectors per frame). Further, a practical algorithm to detect moving objects by segmentation of the flow field was proposed and some experimental results were presented. The system is currently used in closed loop experiments on an autonomous, vision-guided robot.

Further work will be concerned with improvements to the robustness of the system. Applying Kalman filtering to the estimation of the camera rotation can further reduce the sensitivity to noise and allow a prediction of ω_y even when the background detection temporarily fails. For the vectors belonging to the background, the additional parameter $\frac{Z}{T_z}$ (i.e. the *Time To Collision*) can be calculated. By examining this parameter, static obstacles could also be detected and taken into consideration for collision avoidance.

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, Jul 1985.

[2] D. Burschka, C. Eberst, and C. Robl. Vision Based Model Generation for Indoor Environments. In *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA'97)*, pages 1940–1945, 1997.

[3] 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, Nov 1996.

[4] 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, Nov 1996.

[5] 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.

[6] K. Prazdny. Determining the Instantaneous Direction of Motion from Optical Flow Generated by Curvilinearly Moving Observer. *Computer Graphics and Image Processing*, 17:238–248, 1981.

[7] 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.

[8] 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):451–479, May 1989.

[9] S. Yamamoto, Y. Mae, Y. Shiray, and J. Miura. Realtime Multiple Object Tracking Based on Optical Flows. In *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA'95)*, volume 3, pages 2328–2333, May 1995.