Real-Time Computer Vision System for Mobile Robot

Stelian Persa*, Pieter Jonker

Pattern Recognition Group, Technical University Delft
Lorentzweg 1, Delft, 2628 CJ,The Netherlands

ABSTRACT

The purpose of this paper is to present a real-time vision system for position determination and vision guidance to navigate an autonomous mobile robot in a known environment. We use a digital camera, which provide ten times the video capture bandwidth then a USB, using FireWire interface. In order to achieve real-time image processing we use MMX technology to accelerate vision tasks. Camera calibration is a necessary step in 3D computer vision in order to extract metric information from 2D images. Calibration is used to determine the camera parameters by minimizing the mean square error between model and calibration points. The camera calibration we use here is based on several views of a planar calibration pattern, which is an easy to use and accurate algorithm. For position pose of the robot we use the corner points and lines, features extracted from the image, and matched with the model of the environment. The algorithm is as follows: first we compute an initial pose using the Ganatipathy's four point algorithm and we use this initial estimation as the starting point for the iterative algorithm proposed by Araújo in order to refine our pose.

Keywords: Real-time vision, calibration, position estimation, navigation, mobile robots

1. INTRODUCTION

In order to autonomously navigate and perform tasks, the robot needs to know its exact position and orientation; to localize itself. Localization is a problem concerning every autonomous vehicle. Different techniques have been developed to tackle this problem. These can be sorted into two main categories:

- Relative localization, which consist of evaluating the position and orientation through integration of information provided by diverse (usually encoder or inertial) information. The integration is started from the initial position and is continuously updated in time.
- Absolute localization, which is the technique that permits the vehicle to determine its position direct in the domain of motion. These methods usually rely on navigation beacons, active or passive landmarks, maps matching or satellite-based signals like Global Positioning System (GPS).

Due to the fact that the sensors suffer from inaccuracies (for example drift for gyroscopes), and the sensors readings are corrupted by noise, there is error in calculation of robot's position and orientation, which generally grows unbound with time. For this reason, the relative positioning alone can be use only a short period of time.

A method to correct this is to use for instance a vision system to correct the growing error. Hybrid systems that combine inertial tracking with position tracking from camera images seem most promising, because the inertial data can increase the robustness and computing efficiency of a vision system by providing a frame to frame prediction of camera orientation, while the vision system can correct for the accumulated drift of the inertial system.

Current general-purpose processors have been enhanced with new features explicitly dedicated to the efficient handling of multimedia (audio and video) data; in order to exploit the high intrinsic spatial parallelism of low-level image processing, a SIMD solution has been adopted. It can be observed that most of the image processing operators exhibits natural parallelism in the sense that the input image data required to compute a given area of the output is spatially localized. This high degree of natural parallelism exhibited by most of the low-level image processing operators can be easily exploited using SIMD parallel architectures or techniques. For MMX and Streaming SIMD implementation we have also considered the exploitation of the instruction level parallelism of the code and data reduction in order to fit into the internal processor cache.

* Correspondence: Stelian Persa. Other author information: S.P.; Email: stelian@ph.tn.tudelft.nl; WWW: http://www.ph.tn.tudelft.nl/~stelian; Telephone: 31 15 278 3727. P.J.; Email: pieter@ph.tn.tudelft.nl;

1.1 Previous work

With a plethora of different graphics and robotics applications that depend on motion-tracking technology for their existence, a wide range of interesting motion-tracking solutions have been invented. Surveys of magnetic, optical, acoustic, and mechanical robot positioning systems are available[1, 2]. Each positioning approach has limitations; most have a short range and are not suited for outdoor use. Hybrid systems attempt to compensate for the shortcomings of each technology by integrating multiple sensors to produce robust results. The system presented by Behringer[3] can achieve high output rate due to the simple image processing of matching silhouette, but it is not operable in an urban environment. The image processing system described by Harris[4] can track simple models like a model Hornet aircraft, cylindrical objects and patches on the ground, but will fail in an urban environment, where multiple objects with complex wire-frame models are present in the image. Schmid[5] best presents the problem of line matching, but unfortunately their approach uses stereo vision, and is very computation intensive.

1.2 The positioning system

In order to reach the requirements of centimeter-accuracy, we propose a positioning system fusing several sensors: accelerometers, gyroscopes, a compass, a video camera and differential GPS (DGPS) (Figure 1). The DGPS system uses relatively much power, is not very accurate (5-10 m typical error) and very slow (1Hz refresh). Therefore, it is used only for initializing the tracking system. The position of the camera can be determined by finding lines in the camera image and matching them with lines with known positions in the real world. The accuracy is roughly proportional to the pixel size of the camera, and can be of centimeter precision if there are matched lines at a meter distance.

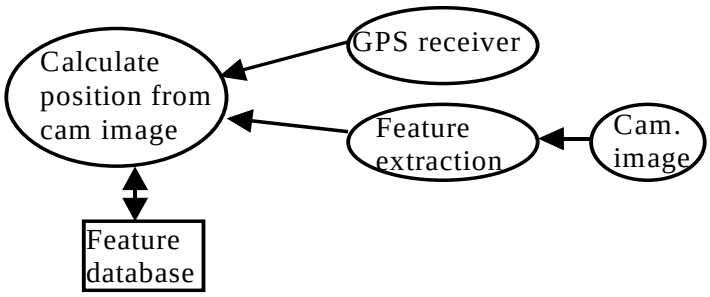

Figure 1. Integration of sensor data for position computation.

The matching process is relatively expensive (see 'line matching' below), and it is done in the backbone. Transmitting camera images to the backbone is expensive as well, and compression is not a good idea because the edges in the image will be blurred, hampering the line extraction. Instead we extract the lines from the images within the robot, and transmit only the lines found. In future when the computation power of a mobile robot will increase the matching task will be done also locally.

Accelerometers and gyroscopes are very fast and accurate, but due to their drift in these sensors, they have to be reset regularly, in the order of once per second depending on the drift of the gyroscopes and accelerometers. The sensors we use now have very low drift and theoretically can do with much lower reset rates, but this sensor is large and heavy. In the future we plan to use smaller and lighter sensors, but these have higher drift. We will discuss in this paper only the vision part of the system because this concerns the system improvement compared with other existing implementations.

The paper is organized as follows. Section 2 presents the camera calibration algorithm and position computation used. Section 3 describes image-processing algorithms used to support camera position computation. A method to speedup low-level image processing operators is presented in Section 4. Section 5 concludes the paper.

2. CAMERA CALIBRATION AND POSE

2.1 Camera calibration

At each time instant, the system receives one image of the scene and analyzes it in order to extract feature information: corner points and lines for pose computation. The accuracy of the 3D pose depends both on the accuracy of the estimated planar pose and the accuracy of the camera calibration.

Camera calibration is a necessary step in 3D computer vision in order to extract metric information from 2D images. Photogrammetric camera calibration needs a 3D calibration object with very good precision, and requires expensive setup. Self-calibration techniques do not use any calibration object. Only by moving the camera in a static scene, the rigidity of the scene provides in general two constrains, and correspondence between three images are sufficient to recover both internal end external parameters. While this approach is very flexible, it is not yet mature[6].

The camera calibration we use here is based on several views of a planar calibration pattern[7]. Given an image of the model plane we can estimate the homography $\mathbf{H} = [\mathbf{h_1}\ \mathbf{h_2}\ \mathbf{h_3}] = \lambda\mathbf{A}[\mathbf{r_1}\ \mathbf{r_2}\ \mathbf{t}]$, where A is the intrinsic matrix, λ an arbitrary scale factor and $[\mathbf{r_1}\ \mathbf{r_2}\ \mathbf{t}]$ the extrinsic matrix. Let M_i and m_i be the model and image points. The maximum likelihood estimation of **H** is obtained by minimizing the following functional: $\min_H \Sigma_i \|m_i - \bar{m}_i\|^2$, where $\bar{m}_i$ is computed based on the homography **H** (with $\mathbf{h}_I$ the i^{th} row of **H**):

$$\bar{\mathbf{m}}_i = \frac{1}{\mathbf{h}_3^T \mathrm{M}_i}\begin{bmatrix}\mathbf{h}_1^T \mathrm{M}_i \\ \mathbf{h}_2^T \mathrm{M}_i\end{bmatrix}. \tag{1}$$

$$\begin{bmatrix}\mathrm{M}_i^T & 0^T & -u\mathrm{M}_i^T \\ 0^T & \mathrm{M}_i^T & -v\mathrm{M}_i^T\end{bmatrix}. \tag{2}$$

The nonlinear minimization is conducted with Levenberg-Marquardt algorithm and this requires an initial guess, which can be obtain by solving the equation $\mathbf{Lx}=\mathbf{0}$ using the singular value decomposition of matrix $\mathbf{L}^T\mathbf{L}$. In the equation, **x** is a vector containing the homography matrix elements (dimension 9) and **L** (see formula (2)).

Using the knowledge that $\mathbf{r_1}$ and $\mathbf{r_2}$ are orthonormal we have the following two basic constrains on the intrinsic parameters:

$$\begin{aligned}\mathbf{h}_1^T \mathbf{A}^{-T}\mathbf{A}^{-1}\mathbf{h}_2^T &= 0 \\ \mathbf{h}_1^T \mathbf{A}^{-T}\mathbf{A}^{-1}\mathbf{h}_1^T &= \mathbf{h}_2^T \mathbf{A}^{-T}\mathbf{A}^{-1}\mathbf{h}_2^T\end{aligned} \tag{3}$$

We have two constrains on the intrinsic matrix, but the intrinsic matrix has 5 unknowns. This means that we need at lest three images of the planar object in order to obtain a solution. We can observe that $\mathbf{B}=\mathbf{A}^{-T}\mathbf{A}^{-1}$ is a symmetric matrix defined by 6D vector, $\mathbf{b} = [B_{11}, B_{12}, B_{22}, B_{13}, B_{23}, B_{33},]^T$, then we have $\mathbf{h}_I^T\mathbf{B}\mathbf{h}_j = \mathbf{v}_{ij}\mathbf{b}$. From the previous two constrains we can form the matrix equation $\mathbf{Vb}=\mathbf{0}$, where **V** is a 2n x 6 matrix formed using the homography values from n images. The solution of the system equation is the eigenvector of $\mathbf{V}^T\mathbf{V}$ associated with the smallest eigenvalue. Once **b** is estimated, we can compute all camera intrinsic parameter of matrix **A**.

Maximum likelihood estimation

The above solution is obtained through minimizing an algebraic distance, which is not physically meaningful. We can refine it through maximum likelihood inference. We are given **n** images of a model plane and there are **m** points on the model plane. Assume that the image points are corrupted by independent and identically distributed noise. The maximum likelihood estimate can be obtained by minimizing the following functional:

$$\sum_{i=1}^{n}\sum_{j=1}^{m}\left\| m_{ij} - \hat{m}(\mathrm{A}, \mathrm{R}_i, \mathrm{t}_i, \mathrm{M}_j)\right\|^2. \tag{4}$$

where $m(A, R_i, t_i, M_j)$ is the projection of point M_j in image i. This is a nonlinear minimization problem, which is solved with the Levenberg-Marquardt algorithm as implemented in `Minpack`[8]. It requires an initial guess of A; $\{ R_i, t_i \mid i = 1::n \}$ which can be obtained using the technique described in the previous subsection.

Dealing with radial distortion

Up to now, we have not considered lens distortion of a camera. However, a desktop camera usually exhibits significant lens distortion, especially radial distortion. In this section, we only consider the first two terms of radial distortion. The reader is referred to Ref. 9, 12 for more elaborated models. Based on the reports in the literature[9, 10, 11], it is likely that

the distortion function is totally dominated by the radial components, and especially dominated by the first term. It has also been found that any more elaborated modeling not only would not help (negligible when compared with sensor quantization), but also would cause numerical instability[10, 11].

As the radial distortion is expected to be small, one would expect to estimate the other five intrinsic parameters, using the technique described above, reasonable well by simply ignoring distortion. One strategy is then to estimate k1 and k2 after having estimated the other parameters, which will give us the ideal pixel coordinates (u; v). Then we have two equations for each point in each image:

$$\begin{bmatrix} (u-u_0)(x^2+y^2) & (u-u_0)(x^2+y^2)^2 \\ (v-v_0)(x^2+y^2) & (v-v_0)(x^2+y^2)^2 \end{bmatrix} \begin{bmatrix} k_1 \\ k_2 \end{bmatrix} = \begin{bmatrix} \breve{u}-u \\ \breve{v}-v \end{bmatrix}. \tag{5}$$

Given m points in n images, we can stack all equations together to obtain in total 2mn equations, or in matrix form as **Dk** = **d**, where **k** = [k1; k2] ^{T}. The linear least-squares solution is given by

$$\mathrm{k} = \left(\mathrm{D}^T\mathrm{D}\right)^{-1}\mathrm{D}^T\mathrm{d}. \tag{6}$$

Once k1 and k2 are estimated, one can refine the estimate of the other parameters by solving (4) with $m(A, R_i, t_i, M_j)$ replaced by distorted coordinates computed using k1 and k2. We can alternate these two procedures until convergence.

2.2 Pose computation

Pose estimation is an essential step in many machine vision and photogrammetric applications including robotics and 3D reconstruction, and is also known as exterior camera calibration. It addresses the issue of determining the position and orientation of a camera with respect to an object coordinate frame.

For position pose of the mobile robot we use the corner points and lines extracted from the image and matched with features stored in a database. In order to be able to perform the pose using only one camera we need rigid objects in scene. The algorithm is as follows: first we compute an initial pose using the Ganapathy[13] four point algorithm, and we use this initial estimation as the starting point for the iterative algorithm proposed by Araújo[14] in order to refine our pose.

The linear formalism proposed by Ganapathy in the case of four coplanar points, start by computing the 3x3 homography matrix. Based on the computed homography, the exterior camera parameters are linearly derived. This four points can be easily selected as the points who belongs to the same facet. After this we have a first camera position estimation. The iterative algorithm proposed by Araújo, which is an improved Newton method, further refines this pose.
If the initial guess is not too far from the true solution, the residuals are small, and can be approximated by a first-order expansion. The algorithm performs iteratively the following steps:

- Project the model according to the current pose estimate.
- Check the stop conditions based on projection error and number of steps.
- Compute the error vector and the Jacobian matrix.
- Solve the resulting system: delta = Jacob \ error.
- Update the rotation matrix and translation vector.

3. IMAGE PROCESSING TASKS

The efficiency of image processing systems depends on the quality of the images, the basic models and evaluation algorithms, but also on the representation of the data. At the beginning of such a system the image is only represented as a matrix of certain values like grey-level or color. The task of the system is now to obtain from this matrix an application dependent representation of the image content so that a user or a subsequent process can take advantage of this information. Feature Extraction (FE) is used to extract important image information, i.e. to suppress redundant information or neglect information, which is not used in the following processes.

Edges approximation by straight lines is also called line detection. For the approximation of the edges by straight lines many different approaches are possible like merging, splitting or split&merge algorithms. The critical point is to find the breakpoints or corners, which lead to the best approximation.

1. The merging algorithm considers each pixel along the edges are sequentially, and group them to a straight line as long as the line fit the edge well enough. If they do not, the line ends and a new breakpoint is established. A disadvantage of this approach is its dependency on the merging order: starting from the other end of the edge would probably lead to different breakpoints.
2. Splitting algorithms divide recursively the edges in (normally) two parts, until the parts fulfill some fitting conditions. Considering an edge consisting of a sequence of edge pixel P 1 ; P 2 ; : : :; Pn; P 1 and Pn as the end points or nodes are joined by an arc. For each pixel on that arc, the distance to the edge is calculated. If the maximum distance is larger than a given threshold the edge segment is divided into two new segments at the position where the maximum distance was found
In our implementation we use the corner and line detection as implemented in FEX[18], and are sometime called Förstner corner and line detection.

3.1 GIS database

We need a GIS database in the backbone for the line matching of the tracking system. We discuss how the GIS database is constructed, and which 3D data structure has use in order to allow quick response to latency-sensitive queries from tracking system. The first, rough version of the GIS database has been constructed semi-automatically, using photos from the air and ground, DGPS measurements, 2D digital maps from the cadaster, and manual steering of the processes (Figure 2). In contrast with common 2.5D GIS databases, our database has a full 3D structure.

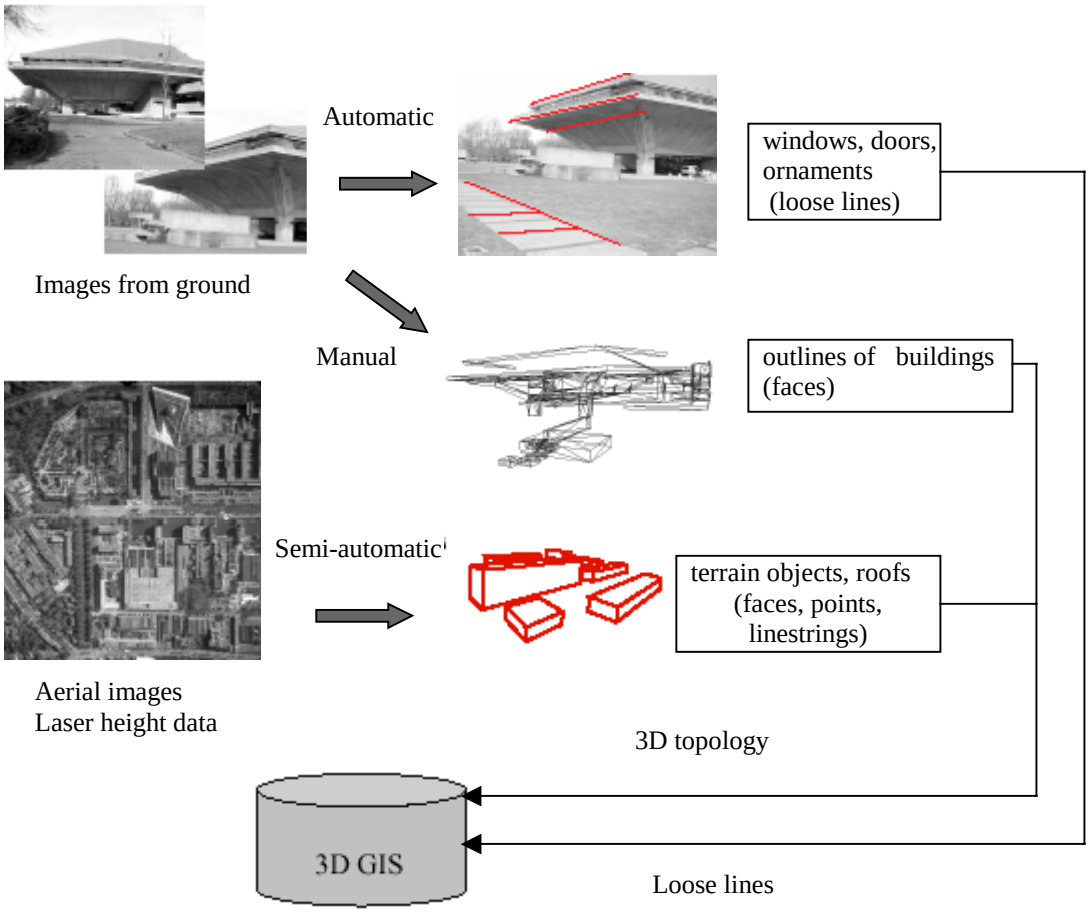

Figure 2. Semi-automatic construction of the database; Photos taken from the ground are used to reconstruct the rough outlines of the buildings. These outlines are improved automatically with additional lines. Placement, alignment and height of buildings on a larger scale is semi-automatically based on images and laser height data taken from the air.

The rough outlines in the database are enriched with more lines, which are extracted from camera images. This enrichment can be done before the database is used for tracking, but also during use, by processing images from the camera on the robot. The database is stored within Oracle 8. Line matching is optimally served if only data especially relevant for the current position of the user is extracted from the database. Therefore, quick response to database queries

(in the order of 1 second) is essential. To speed up especially response time, we decided to use our own functionality and 3D data structure, instead of oracle's spatial data engine.

3.2 Line matching

Matching the lines from the database with the lines extracted from the camera images is essential to find the position and orientation of the camera (Figure 3)

Figure 3. Lines extracted from the camera images (left) and from the GIS database (right). Matching those lines gives an accurate estimation of the camera position and orientation.

This matching is difficult:
• Many lines in the database will certainly be invisible, for instance lines at the back of a building. These lines have to be removed to improve the matching process. These hidden lines can be found with existing hidden surface algorithms. However, the camera viewpoint is only approximately known, and therefore we cannot be completely sure which lines are visible. Depending on the stability of the matching process, lines of which it is not certain whether they are visible may be kept or removed. If kept, a false match may cause a precise but incorrect position estimation. If removed, the result will be some missed matches and a degraded position estimation.
• A related problem is that there will be lines invisible that were expected to be visible, for instance because there is a car in front of it, or because there is a tree in front of it that is not in the database.
• Lack of resolution in the database; For instance, the exact locations of nearby tiles may not be stored, missing opportunities to accurately estimate the position.
A number of these problems still have to be resolved.

4. INTEL MMX AND STREAMING SIMD INSTRUCTION FOR ACCELERATING IMAGE PROCESSING TASKS

Intel's MMX Technology[15, 16] adds several new data types and 57 new instructions specifically designed to manipulate and process video, audio and graphical data more efficiently. These types of applications often use repetitive loops that, while occupying 10 percent or less of the overall application code, can account for up to 90 percent of the execution time. A process called Single Instruction Multiple data (SIMD) enables one instruction to perform the same function on multiple pieces of data.

The recent Intel Pentium III processor now provides sufficient processing power for many real-time computer vision applications. In particular, the Pentium III includes integer and floating point Single Instruction Multiple Data (SIMD) instructions, which can greatly increase the speed of computer vision algorithms. The Streaming SIMD Extensions expand the SIMD capabilities of the Intel® Architecture. Previously, Intel® MMX™ instructions allowed SIMD integer operations. These new instructions implement floating-point operations on a set of eight new SIMD registers. Additionally, the Streaming SIMD Extensions provide new integer instructions operating on the MMX registers as well as cache control instructions to optimize memory access. Applications using 3D graphics, digital imaging, and motion video are generally well suited for optimization with these new instructions. Multiprocessor Pentium III systems are relatively inexpensive, and provide similar memory bandwidths and computational power as non-Intel workstation costing significantly more money.

Block matching is essential in motion estimation and optical flow. Equation (7) is used to calculate the Sum of Absolute Differences (also referred to as the Sum of Absolute Distortions), which is the output of the block-matching function.

$$SAD = \sum_{i=0}^{15}\sum_{j=0}^{15}\left|V_n(x+i, y+j) - V_m(x+dx+i, y+dy+j)\right|. \quad (7)$$

Figure 3 illustrates how motion estimation is accomplished; *dx* and *dy* are candidate motion vectors. Motion estimation is accomplished by performing multiple block matching operations, each with a different dx,dy. The motion vector with the minimum SAD value is the best motion vector.

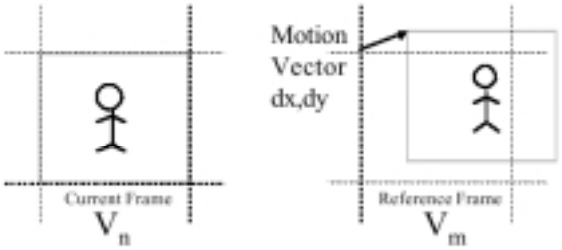

Figure 3. Block matching

Streaming SIMD Extensions provide a new instruction, `psadbw`, that speeds up block matching. The code to perform this operation is given below. This code has been observed to provide a performance increase of up to twice that obtained when using MMX™ technology.

```
psad_top: // 16 x 16 block matching
// Do PSAD for a row, accumulate results
        movq mm1, [esi]
        movq mm2, [esi+8]
        psadbw mm1, [edi]
         psadbw mm2, [edi+8]
// Increment pointers to next row
         add esi, eax
        add edi, eax
// Accumulate diff in 2 accumulators
        paddw mm0, mm1
         paddw mm7, mm2
        dec ecx // Do all 16 rows of macroblock
        jg psad_top
// Add partial results for final SADvalue
         paddw mm0, mm7
```

An interesting problem is how to measure the execution time of MMX routine. This could be done using the information from RDTSC (real time stamp counter), which contains the cycle counter. The detailed description of the RDTSC counter may be find in Ref. 17. The measured timing is approximate and depends on many factors as OS overheads, number of processes running, cash situation if MMX code contains read/write instructions, etc. Various conditions, as cache warming prior reading or writing from/to the same memory blocks, a particular write strategy implemented in the processor and L2 cache, most significantly affect the performance. For that reasons we need to carefully consider the results and run multiple test and average out the results excluding the values far from the mean, which may occur due to a particular running condition.

In Figure 4 we show a sample application to compute real-time optical flow. The flow algorithm we utilize is correlation-based, using color video images to enhance the correspondence accuracy. To speed up the flow estimation, we use a 1D search pattern and only estimate the flow in regions where motion is detected. The camera is ADS Pyro WebCam with a FireWire interface that ensures a 400Mbps bandwidth, which is approximately 40 times higher that on USB. We use the Unibrain driver and SDK, which allows us to modify in real-time parameters like exposure (shutter, gain and iris), color (hue and saturation) and basic parameters (brightness, sharpness and gamma).

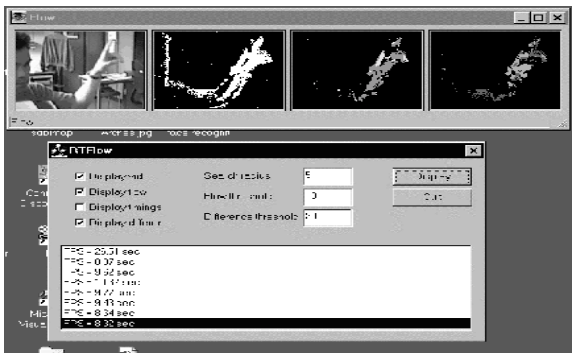

Figure 4. Optical Flow computation using Streaming SIMD instruction

5. CONCLUSIONS

The system described above has been implemented and tested on modified Nomad robot available in our lab. All software (robot's control, sensors, GPS and camera measurement acquisition and processing) is programmed in C++. The real-time implementation allows the position and attitude estimation to be updated every 30ms.

The DGPS is not very accurate (5-10 m typical error) and very slow (1Hz refresh). Therefore, it is used only for initializing the tracking system. The position of the camera can be determined by finding lines in the camera image and matching them with lines with known positions in the real world. The accuracy is roughly proportional to the pixel size of the camera, and can be of centimeter precision if there are matched lines at a meter distance.

This paper described a positioning and attitude estimation system based on several sensors, which are corrected based on a vision system. The method takes advantage of both types of sensors by fusing the information from inertial and GPS sensors with information from a camera sensor. Real-time implementation of the system on our mobile robot platform allowed to show that it works well.

ACKNOWLEDGMENTS

This work was done in the framework of Ubiquitous Communications Program (www.ubicom.tudelft.nl).

REFERENCES

1. Youngblut C, Johnson RE, Nash SH, Wienclaw RA, Will CA., "Review of Virtual Environment Interface Technology", *Internal report P-3186*, Institute for Defense Analyses (IDA), Alexandria, VA, 1996, Available Internet: http://www.hitl.washington.edu/scivw/IDA/.
2. Borenstein J, Everett HR, Feng L., "Where am I? Sensors and Methods for Mobile Robot Positioning", *Technical Report*, University of Michigan, 1996, Available Internet: http://www-personal.engin.umich.edu/~johannb/position.htm.
3. Behringer R., "Registration for outdoor augmented reality applications using computer vision techniques and hybrid sensors", *Proc. IEEE Virtual Reality*, 244-251, 1999.
4. Harris C., "Tracking with Rigid Models", In A. Blake, Yuille (Eds), *Active Vision*, pp.59-74. MIT Press, 1992.
5. Schmid C, Zisserman A., "The geometry and matching of lines and curves over multiple views", *International Journal on Computer Vision*, 40 (3), pp.199-234, 2000.
6. S. Bougnoux, "From projective to euclidean space under any practical situation, a criticism of self-calibration", *In Proc. of 6th International Conference on Computer Vision*, pp.790-796, 1998.
7. Z. Zhang, "Flexible Camera Calibration by Viewing a Plane from Unknown Orientations", *International Conference on Computer Vision*, 1999.
8. J. More, "The levenberg-marquardt algorithm, implementation and theory", In G. A. Watson, editor, *Numerical Analysis, Lecture Notes in Mathematics* 630. Springer-Verlag, 1977.
9. D. C. Brown, "Close-range camera calibration", *Photogrammetric Engineering*, 37(8), pp.855–866,1971.

10. R. Y. Tsai, “A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf tv cameras and lenses”, *IEEE Journal of Robotics and Automation*,3(4), pp.323–344, Aug. 1987.
11. G. Wei and S. Ma, “Implicit and explicit camera calibration: Theory and experiments”, *IEEETransactions on Pattern Analysis and Machine Intelligence*, 16(5), pp.469–480, 1994.
12. J. Weng, P. Cohen, and M. Herniou, “Camera calibration with distortion models and accuracy evaluation”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(10), pp. 965–980,Oct. 1992.
13. S. Ganapathy, “Decomposition of Transformation Matrices for Robot Vision”, *Proceedings of Int. Conf. On Robotics and Automation*, pp. 130-139, 1984.
14. H. Araújo, R.L. Carceroni, C.M. Brown, “A Fully Projective Formulation to Improve the Accuracy of Lowe's Pose-Estimation Algorithm”, *Computer Vision and Image Understanding*, pp. 227-238, May 1998.
15. Intel Corporation, *Intel Architecture MMX Technology Developer’s Manual*, Intel Corporation 1997. Available at http://www.intel.com.
16. Intel Corporation, *MMX Technology Programmers Reference Manual*, Intel Corporation 1997. Available at http://www.intel.com.
17. Intel Corporation, *Using the RDTSC Instruction for Performance Monitoring*, Available at http://developer.intel.com/drg/pentiumII/appnotes/RDTSCPM1.HTM.
18. Förstner, W., “A Framework for Low Level Feature Extraction”, *Band 802 der Reihe LNCS*, pp. 383-394. Springer, 1994.