Pattern Recognition, Proc. 25th DAGM Symposium, Magdeburg, Germany, 2003, pp. 410-417, Springer-Verlag Berlin Heidelberg, ISBN 3-540-40861-4

Extraction of orientation from floor structure for odometry correction in mobile robotics

Christof Schroeter, Hans-Joachim Boehme, and Horst-Michael Gross

Fachgebiet Neuroinformatik,
Technische Universitaet Ilmenau
{christof.schroeter, hans-joachim.boehme, horst-michael.gross}@tu-ilmenau.de

Abstract. We are presenting a method for correcting odometry readings of a robot for increased accuracy of position estimation. Our method uses a simple pragmatic approach and exploits the distinct structure of the floor in our experimental area. By continually extracting orientation information from the floor view, we are able to correct the heading component of odometry, thereby eliminating the major source for position errors. Compared to other approaches the solution is computationally inexpensive. Our experiments show that by employing our correction method we are able to significantly increase position accuracy and consistently map paths up to several hundred meters.

1 Introduction

Self-localization is a basic task in any autonomous robot application. Knowledge of position is a pre-requisite to any sensible navigation behaviour. Most robots are equipped with some sort of odometry sensors (like wheel encoders) for measuring their own motion. Using dead-reckoning, these measurements can be used by the robot to localize itself relative to a reference point. Due to the nature of dead-reckoning the resulting position estimates are prone to a variety of errors, and accuracy is decreasing over time. In our PERSES project [4], we developed a simple approach for correcting the orientation component of odometry readings by exploiting the special fbor structure found in the environment we are using for development and experiments, a home depot.

This paper is structured as follows. In section 2 we motivate the use of our orientation correction method and discuss related topics such as scan matching, landmark navigation and SLAM. Section 3 shows in detail the implementation of our fbor image processing and correction of odometry data. Section 4 contains experimental results and proves the vast improvement in position estimation, enabling us to consistently map an area of 40 * 40m^2 without additional localization techniques.

2 Localization

In our PERSES project, we are developing an interactive artificial shopping assistant that will act as a guide and mobile information terminal for customers in a store environment. Current experimental platform is a RWI B21r robot. To be able to act autonomously, the robot must localize itself within its environment. The B21r is equipped

with wheel encoders to keep track of its own position by dead-reckoning. Odometry is widely used in robotics because it is inexpensive, allows high sampling rates and provides good short-term accuracy. However, the fundamental idea of integration of incremental motion information inevitably leads to the accumulation of errors. Error sources can be divided into 2 categories: systematic and non-systematic errors. Examples for systematic errors are inaccurate drive geometry (wheel diameters etc.) and finite encoder resolution, while non-systematic errors are introduced by uneven floors, slipping or external forces (interaction with external bodies) [1]. While some of these errors can be corrected by carefully adjusting system parameters [2], the overall effect of deviation in position estimation is unbounded if no external reference is used. Particularly, the accumulation of orientation error causes large position errors. An example is shown in Fig. 1 (left). Here the robot traveled a straight path of 30 meters, turning by 180° at the end and returning to the starting point. Due to increasing orientation error, the straight legs appear bent and the end point is virtually located 10 m from the start.

Advanced localization techniques have been an area of intensive research for a long time and a number of different methods have been developed.

2.1 Scan Matching

In scan matching, features are extracted from readings of range measuring sensors and matched to obtain displacement between measuring positions [6]. Most applications use laser scanners for their high accuracy and reliability. Sonar sensors have been used too, but results are generally less accurate due to the higher variance of sonar measurements. For robust results from matched scans this method still depends on a position hypothesis from odometry. Because we are strongly focusing on visual sensing and aiming towards a low-cost platform, regarding the potentially significant cost increase by equipping a robot with a 2D laser scanner, we are hesitant to rely on scan matching.

2.2 Landmark Navigation, Monte Carlo Localization

Another possibility of tracking the position of a robot is by localizing itself relative to known landmarks in the environment. Landmarks can be detected by range sensors or visual input. Usually the robot needs to maintain its position between landmark observations by odometry. Newer methods, like Monte-Carlo-Localization (MCL), perform probabilistic localization by estimating a probability distribution over the state space [3, 5]. This distribution is updated with motion (odometry) and environment observations. Both landmark navigation and MCL need a map of their environment. In order to build this map, knowledge of respective positions is needed, so either method is not fit for mapping previously unknown terrain.

2.3 SLAM

The chicken-and-egg problem of mapping and localization is adressed by a still relatively new method called Simultaneous Localization and Mapping (SLAM). In SLAM,

mapping and localization are not seen as separate tasks, but solved together. This is derived from the observation that the 2 problems depend on each other. The base of SLAM is an extended state space that contains position and map. A probability distribution over this common state space is maintained that converges with motion and observations [8]. Most successful applications are using laser scanners, while visual SLAM and adaption to arbitrarily large environments are subjects of ongoing research.

Here, our aim was to develop a pragmatic solution to consistent map building that is computationally efficient and does not need costly sensoric equipment. Furthermore, our experimental observations show that odometry errors of our experimental platform B21r mainly occur in the orientation component of odometry data. While drive range measurements are reasonably accurate, the growing and unpredictable error in orientation causes mapping to fail completely without a means of correction. Looking for a source of reference, we found the fbor pattern in our experimental area contains easily recognizable features that provide (partial) orientation information.

The fbor consists of square pieces of 30 cm by 30 cm. While the pieces themselves are only slightly textured, the lines between them show a strong contrast (Fig. 1 right). Orientation of these pieces is consistent over the whole store area. The idea of our method is to detect the main orientation of the lines between pieces and use them as reference for recalibrating odometry at each motion step.

We have experimented with deriving displacement directly from the fbor observations, but due to the narrow field of view we would need a very high update frequency for robustly finding corresponding points between subsequent pictures.

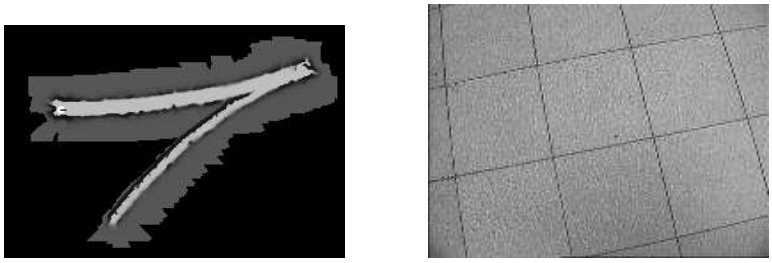

Fig. 1. left: the map shows the effects of heading error in odometry (light areas mark free space, dark areas mark obstacles), right: tiled structure on the floor of our experimental area, a home depot

3 Floor-based odometry correction

To obtain images of the fbor we use a camera dedicated to this purpose only. This camera is attached at a height of 1.40m and protruding, looking down vertically, so that it is seeing about 50cm of ground space in front. Images are captured at the relatively low resolution of 192 * 143 pixels for fast processing. We use grayscale images because color yields no further information about orientation.

3.1 Processing the floor image

Figure 2 (top left) shows that the image is distorted radially, which is visible in the slight curving of straight lines around the center of the image.

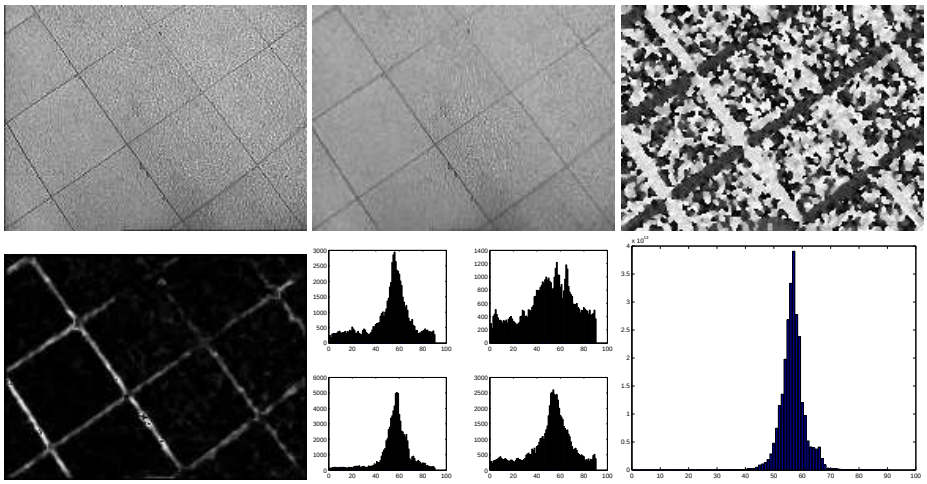

Fig. 2. top row from left to right: initial image as captured from the camera, image after radial correction, angle of local orientations (gray values coding orientation angles in a range of -90° to 90°), bottom row: power of local orientations (smoothed by box filter), histograms for the 4 sectors (see text for explanation), overall histogram showing strong peak at main orientation

To correct this, each pixels distance to the center of the image is increased by the formula

$$r_{new} = r_{old} * (1 + r_{old}^2 * k)$$

with an empirically determined radial correction factor $k = 2 * 10^{-6}$. This is a very simple approach to correcting camera abberation, but it's sufficient for our problem. Figure 2 (top center) depicts the corrected image.

The next step is to calculate local orientation for each pixel. This is done by applying an orientation tensor (inertia tensor method) as described in [7].

In detail, this consists of the following steps:

1. Bandpass filtering of the image by applying binomial filters of different sizes.

$$B_1 = image \otimes binom(3 * 3)$$

$$B_2 = B_1 \otimes binom(5 * 5)$$

$$B_3 = 3.0 * (B_1 - B_2)$$

2. Calculate local x- and y-gradients as difference of direct neighbours

$$\begin{aligned}\bigtriangledown B_3(x,y) &= \left(\frac{\partial B_3(x,y)}{\partial x}, \frac{\partial B_3(x,y)}{\partial y}\right)^T \\ &= (B_3(x-1,y) - B_3(x+1,y), B_3(x,y-1) - B_3(x,y+1))^T\end{aligned}$$

3. Calculate angle and power of local orientation at each image point

$$2\phi = atan\left(2 * \frac{\partial B_3}{\partial x} * \frac{\partial B_3}{\partial y}, \left(\frac{\partial B_3}{\partial x}\right)^2 - \left(\frac{\partial B_3}{\partial y}\right)^2\right)$$

$$power = \left(\left(\frac{\partial B_3}{\partial x}\right)^2 - \left(\frac{\partial B_3}{\partial y}\right)^2 + 4 * \frac{\partial B_3}{\partial x} * \frac{\partial B_3}{\partial y}\right) * \left(\left(\frac{\partial B_3}{\partial x}\right)^2 + \left(\frac{\partial B_3}{\partial y}\right)^2\right)^{-1}$$

The result of the orientation tensor are fields of angle and power of local orientation for each point of the image. As seen in Fig. 2 (top right, bottom left), at the edges of the tiles there are strong local orientations with angles aligned to direction of these edges while on the surface of the tiles the orientations are unaligned but with low power. In a histogram of local orientations, weighted with respective power, the maximum will show the robot orientation with respect to the main orientation of the fbor. While the orientation tensor output contains a range of -90° - +90° , the actual information that can be gathered is only an angle in the range 0° - 90° . Due to the square tiles images rotated by a multiple of 90° are indistinguishable. This corresponds to the histogram having 2 (dependent) maxima, one originating from vertical lines, the other from horizontal ones. Each angle k $<$ 0° corresponds to an angle k + 90° in the orientation tensor output, therefore all these values are simply mapped into the 0° - 90° range by adding 90°.

Situations may occur where the robot encounters objects lying in its path or steers near towards immobile objects, like e.g. goods shelfs. In such cases, it may happen that the view of the fbor camera is partly occupied by objects occluding the fbor, leading to unpredictable local orientations in a part of the image and disturbing the histogram. To avoid errors resulting from such a situation the image is split into 4 sectors (upper right, upper left, lower right, lower left) and histograms are evaluated for each sector separately. This is based on the expectation that in most cases an object will only cover one sector. The histograms are smoothed by a box-filter, then for each sector the main orientation is determined by the maximum in the sector-specific histogram. If one of these 4 values is outlying, the respective sector is discarded, the remaining sectors are merged into a common histogram by multiplication and the maximum of this histogram is regarded as main orientation in the image.

Problems may occur when the fbor structure is widely obscured by large objects or strong light/shadow contrasts introduce misleading edges. Ideally this will result in the histogram having no strong maximum, in which case no correction will be applied until the fbor is visible again. However, situations may occur where objects in the image lead to a wrong orientation. Since we only use this correction approach during map building by joysticking, the human operator is responsible for avoiding such critical situations.

3.2 Re-calibrating odometry

Given the true orientation of the robot, we can now correct odometry readings. In our architecture, sensor readings from wheel encoders are already translated into a position in low-level processing and sent to the high-level software as (x,y,ϕ)-triplets. These low-level routines always use the orientation measured by odometry and therefore calculate wrong (x,y)-positions when the true orientation is different. This means application level software must not only adjust the orientation component, but also calculate the true position change between 2 odometry data readings and update position. To this purpose, the last odometry reading is stored together with the corrected position and the angle difference between odometric orientation and visual orientation is maintained continually. When receiving a new odometry position, the difference between current and previous believed position is calculated as the driven path, rotated by the known odometry orientation error and added to the last corrected position. Correction over 2 steps is shown in Figure 3. The method simplifies by implying all the rotation error that occurs in one step is aquired at the starting point of this particular step. This seems sufficient for very small steps (updates are done with a distance of a few cm only). Experimentation with more sophisticated but computationally expensive models only showed marginal differences.

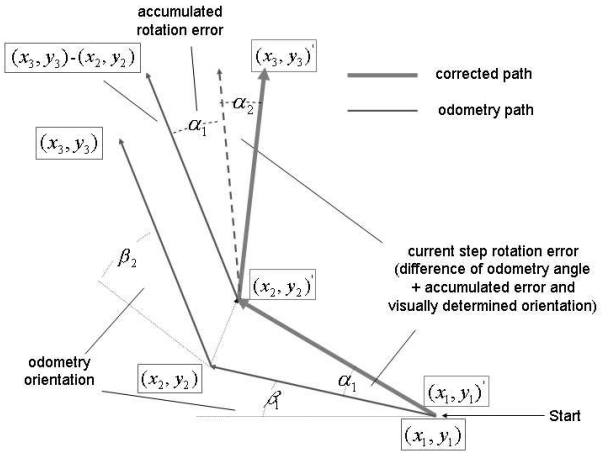

Fig. 3. *Principle of correcting odometry with reference orientation*

Now with a good estimate of the true position of the robot, we can build maps by generating local maps from sonar range readings and incorporating them into a global map. For this purpose we use the grid map approach and update formula from [9].

4 Results

We tested our method by driving closed loops of different size in our home depot experimental area. By exactly returning to the starting point, we were able to determine the error in localization.

In the first experiment (Fig. 4 top row) we drove through 2 hallways with an overall path length of about 55 meters. Without correction the end position yielded by odometry was about 7 m from the real position (left). With correction (right), the difference between real and estimated position is about 10 cm in either x- and y-direction of the reference frame. After driving the same loop back in the opposite directon the error still was no larger than 12 cm in either direction. The second pair of maps (Fig. 4 bottom row) shows an experiment with a slightly longer path of about 120 m. Here without correction the error was more than 12 m. With correction, however, the position after driving that path was 8 cm in x- and 30 cm in y-direction, and after driving the same path back to the starting point again, it was 30 cm and 40 cm. Overall repeated experiments show that we can assume a total position difference of about 20 cm for each 50 m of driven path. These remaining errors are introduced by inaccuracies in the visual orientation as well as from errors in odometry range measurement.

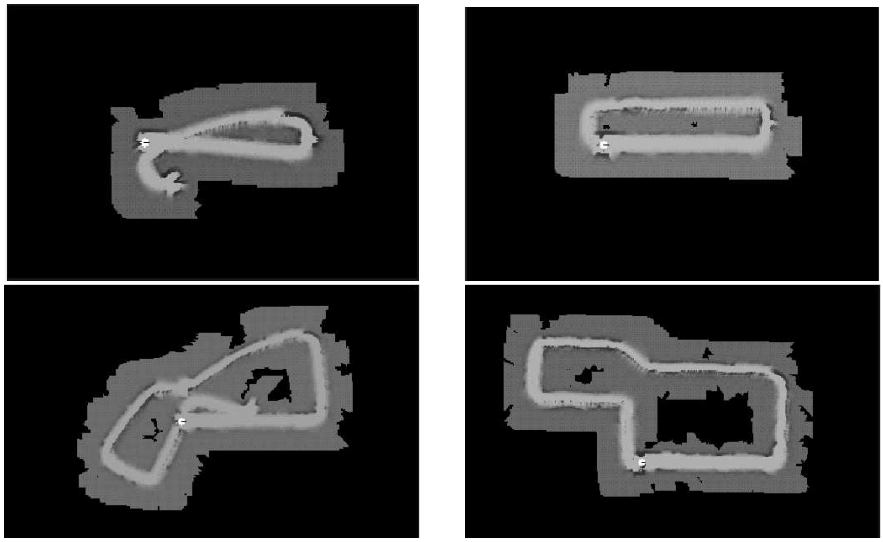

Fig. 4. Comparison of maps with uncorrected and corrected odometry, top left: closed path of 55 m length, without correction, right: same with correction, bottom left: closed path of 120 m without correction, right: same with correction

Finally Figure 5 shows the result of mapping an area of about 30% of the whole store. Here, after a total path length of 350 m, the effects of the remaining position error become visible in the map (2 neighbouring hallways appear too near together leaving no room for the goods shelf between). The total error was 0.9 m in both directions. To avoid the accumulating of errors, we still need to re-calibrate robot position after driving some hundred meters. To this purpose, we will choose some reference points and measure their respective positions, so we can set exact position in regular intervals when crossing such a point while joysticking the robot around. Once a map is generated, we do not rely on correct odometric information anymore because we can then employ another localization method based on that map as explained in section 2.2.

Fig. 5. *Here we mapped a significant part of the home depot area*

5 Conclusions

We developed and tested an approach to vision-based odometry correction by extracting orientation from fiducial images. The implementation exploits the strong structure we find in our experimental area, but with specific pre-processing it may be able to adapt to other recognizable patterns. Although we cannot completely negate the long-term effect of inaccurate odometry, we showed that it is possible to build maps without the necessity of further sensors or special preparation of the environment. Our future interest, however, lies in visual SLAM (Simultaneous Localization And Mapping) and we expect to be able to build consistent maps without the need for odometry correction and to gain flexibility in arbitrary environments for our robots.

References

1. J.Borenstein, H.R.Everett, and L.Feng. Where am I? - Sensors and Methods for Mobile Robot Positioning. Technical report, University of Michigan, 1996.
2. J.Borenstein and L.Feng. Measurement and correction of systematic odometry errors in mobile robots. *IEEE Trans. on Robotics and Automation*, 12(5), 1996.
3. F.Dellaert, D.Fox, W.Burgard, and S.Thrun. Monte carlo localization for mobile robots. In *Proc. of the 1999 IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 1999.
4. H.-M.Gross and H.-J.Boehme. Perses - a vision-based interactive mobile shopping assistant. In *Proc. of the IEEE Intl Conf. on Systems, Man and Cybernetics (IEEE-SMC 2000*, pp 80–85, 2000.
5. H.-M.Gross, A.Koenig, H.-J.Boehme, and C.Schroeter. Vision-based monte carlo self-localization for a mobile service robot acting as shopping assistant in a home store. In *Proc. of the 2002 IEEE/RSJ Intl. Conf. on Intelligent Robots and System (IROS2002)*, pp 265–262, 2002.
6. J.-S.Gutmann and C.Schlegel. Amos: Comparion of scan matching approaches for self-localization in indoor environments. In *Proc. of the 1st Euromicro Workshop on Advanced Mobile Robots (EUROBOT '96)*, 1996.
7. B.Jaehne. *Practical Handbook on Image Processing for Scientific Applications*. CRC Press LLC, Boca Raton, Florida, 1997.
8. M.Montemerlo, S.Thrun, D.Koller, and B.Wegbreit. FastSLAM:A factored solution to the simultaneous localization and mapping problem. In *Proc. of the AAAI Natl. Conf. on Artificial Intelligence*, 2002.
9. Hans Moravec. Sensor fusion in certainty grids for mobile robots. *AI Magazine*, 9(2):61–77, 1988.