

Visual Odometry for the Autonomous City Explorer

Tianguang Zhang¹, Xiaodong Liu¹, Kolja Kühnlenz^{1,2} and Martin Buss¹

¹Institute of Automatic Control Engineering (LSR)

²Institute for Advanced Study (IAS)

Technische Universität München

D-80290 Munich, Germany

Email: {tg.zhang, kolja.kuehnlenz, m.buss}@ieee.org

Abstract—The goal of the *Autonomous City Explorer (ACE)* is to navigate autonomously, efficiently and safely in an unpredictable and unstructured urban environment. To achieve this aim, an accurate localization is one of the preconditions. Due to the characteristics of our navigation environment, an elaborated visual odometry system is proposed to estimate the current position and orientation of the ACE platform. The existing algorithms of optical flow computation are experimentally evaluated and compared. The method based on pyramidal Lucas-Kanade algorithm with high-speed performance is selected. Based on the optical flow in 2D images, the camera ego-motion is estimated using image Jacobian matrix and least squares method. The kinematic model is set up to map the camera ego-motion to the robot motion. To eliminate systematic errors, a novel system calibration approach is proposed. Finally the odometry system is evaluated in experiments.

I. INTRODUCTION

The Autonomous City Explorer (ACE) project [1] develops a robot (see Fig. 1) that can autonomously navigate in an unstructured urban environment and find its way through interaction with humans. To achieve an efficient and safe operation in natural populated environments, an accurate localization is one of the most important preconditions.

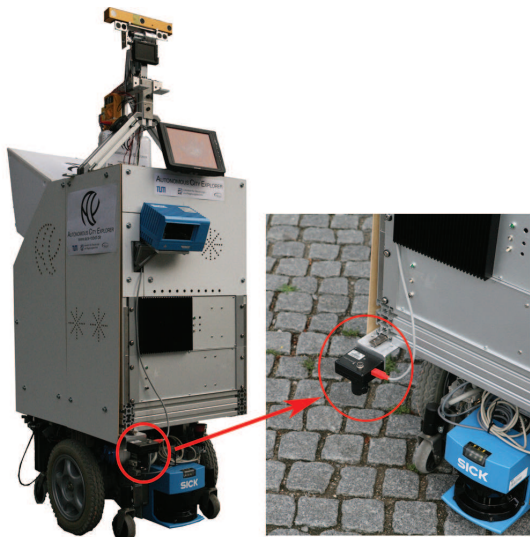


Fig. 1: ACE robot and the camera for visual odometry

Angular encoders on the wheels of the robot platform are normally used for odometry. But if ACE moves on a

ground which is not flat, e.g. on cobblestone sidewalk, or has sands on it, the wheels will slip. The encoders cannot provide accurate information any more. Thereby visual data is fused with the information from angular encoders on the wheels to support the localization.

How to locate the position and orientation of a moving object using visual information has long been a research focus of the computer vision community. Campbell et al. [2] designed a model using monocular camera mounted at the robot's front, viewing the front of the robot and the ground. The optical flow vectors are divided into three parts: a dead zone near the horizon is defined; the vectors above this area are used to calculate the robot rotation, while the vectors below that are used to estimate the robot translation. Similar to the model in [2], the monocular camera in [3] only focuses on the locally planar ground. The translation and the rotation are calculated together. Because of our application environments, both models are not sufficient for our system. Utilizing Harris corners detection and normalized correlation, Nister et al. presented a system [4] that provided an accurate estimation but worked relatively slow. In [5] salient features are tracked continuously over multiple images. Then the differences between features that denote the robot's motion are computed. An inertial measurement unit (IMU) is also employed in [6] to estimate the orientation. In [7], utilizing the task-sharing abilities of the operating system, the problem of synchronization across multiple frame-grabbers can be solved. In order to have a better efficiency, a *sum of absolute differences* (SAD) algorithm is used here, but the accuracy is not perfect.

Our visual odometry is an incremental, online estimation of robot motion by analyzing a sequence of images from an onboard camera. In this paper, an elaborated concept is described and implemented with high video frame rate. Using SAD and *pyramidal Lucas-Kanade* (PLK) algorithms, optical flows in 2D images are calculated. Then the translation and rotation of the camera are computed from these optical flows. A geometry model denoting the relation between the camera and the robot is established and the kinematic modeling of ACE as well as its non-holonomic characteristics are also regarded, so that the localization of the robot can be deduced from the position of the camera. Moreover, a novel system calibration procedure is proposed to deal with systematic errors. At last, the implementation is integrated

into the whole project, utilizing an appropriate observer, a Kalman-filter, for data fusion. It is a distinctly local, low-latency approach that facilitates closed-loop motion control and highly accurate dead reckoning to help ACE determine the precise position and orientation.

The paper is organized as follows: Firstly, in Section II, the complete system including the hardware and our algorithms are introduced. In Section III, we estimate the camera ego-motion. The kinematic modeling, system calibration and data fusion are established in Section IV. Experimental results are shown in Section V. Conclusions and future work are given in Section VI.

II. SYSTEM OVERVIEW

ACE is equipped with various sensors which can be considered for this localization task. However, they have major drawbacks with respect to the experimental scenario:

- The navigation route partly consists of ground with sands and cobblestone (see Fig. 1), on which the angular encoders on the wheels cannot provide reliable information. Driving onto/off sidewalks also causes serious wheel sliding.
- Since ACE interacts with humans frequently and also navigates in very crowded pedestrian areas, the measurement data from stereo cameras and lasers facing forwards are not directly usable.
- Moving on poor ground conditions, such as on cobblestone sidewalk, ACE bumps during the navigation, which causes a catastrophic performance of gyroscopes.

To avoid disturbances of the independently moving entities such as passers-by, the camera for visual odometry is mounted in the front right of ACE and gazes downwards. The optical axis is perpendicular to the ground as shown in Fig. 1 and Fig. 2.

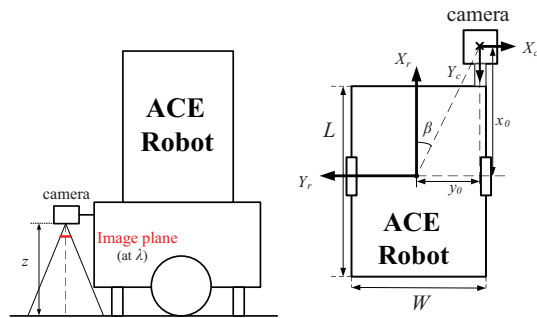


Fig. 2: Profile (left) and topview (right) sketches of the visual odometry configuration

A. Hardware Description

Our visual odometry system is equipped with a high-speed camera and a 1394b PCI-express adapter which are used to capture and transfer the images to the vision processing computer on ACE. A Dragonfly Express camera (Point Grey Research Inc.) is used, equipped with a normal wide-angle lens, which can work at 200 fps at a resolution of 640×480

pixels. We use a lens with a focal length of 4.8 mm and a view angle of 60° . The distance (z) between the image plane and the ground plane is 32 cm . The vision processing computer is equipped with an AMD Phenom 9500 Quad-Core processor with a basic frequency of 2.2 GHz and 4 GB memory.

B. General Concept

The general concept of our visual odometry is composed of six parts:

- 1) After being captured from camera, images come to the camera calibration process using the camera calibration toolbox [8].
- 2) After image undistortion, optical flow computation algorithm is implemented to calculate the interest points velocity in the image plane.
- 3) The camera ego-motion is estimated utilizing the image Jacobian matrix and least squares method.
- 4) A kinematic model is established, in order to transform the camera motion into the robot motion.
- 5) Two scenarios are designed to correct the systematic errors, which is called system calibration.
- 6) The visual odometry results are combined with the conventional odometry data using a Kalman-filter, enabling ACE to localize itself with a better accuracy.

Parts 1, 4 and 5 are operated only once, while the others are executed between each two successive frames captured sequentially.

III. CAMERA EGO-MOTION ESTIMATION

In this section two different optical flow algorithms, in combination with image Jacobian matrix and least squares optimization method are implemented to estimate the camera ego-motion.

A. Optical Flow Algorithms

In this visual odometry system two optical flow computation algorithms, *sum of absolute differences* (SAD) and *pyramidal Lucas-Kanade* (PLK) are implemented. The computation speed and accuracy are compared and the one with a better performance is selected.

1) *SAD*: SAD is a kind of block matching algorithm using absolute difference as criteria of similarity. The original block in the first image and the matching block in the second image should have the minimum SAD value [9].

SAD is expressed in assembly language with *AT&T* [10] syntax under Linux system to attain a fast computational speed – 70Hz . The size of images is 640×480 pixels and the 360×360 pixels around the principal point are chosen as interest area, which is divided into 36 groups. Each group consists of 3×3 windows containing 20×20 pixels each as shown in Fig. 3. In each group we set a threshold to eliminate some windows whose optical flow values seem not to be accurate. The average optical flow value of remaining windows in each group is computed and could be seen as a valid optical flow value of this group. It can be considered in a way that optical flows of 36 feature

points have been computed in total with a better accuracy. This alignment also benefits the efficiency of the assembly language programming.

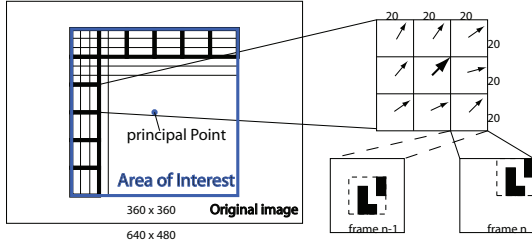


Fig. 3: Sketch of points of interest selection

2) *PLK*: Lucas-Kanade [11] is a classic differential optical flow technique. One advantage of the PLK algorithm is its ability to handle a large pixel motion exactly with local sub-pixel accuracy. The pyramidal implementation of the Lucas-Kanade algorithm [12] works robustly and efficiently. 36 sets of optical flow results are obtained with this algorithm in the same way as by the SAD algorithm.

Selection of the integration window size and pyramid levels depends on the speed of ACE. If ACE is running at its lowest speed of 0.2 m/s and the camera works at a frequency of 30 Hz , the pixel motion between two successive frames is $13.8 \text{ pixels/frame}$. If ACE is at its highest speed 0.8 m/s , the pixel motion increases to $55.3 \text{ pixels/frame}$. Due to hardware limitations frame jumping occurs occasionally, resulting in a long time interval between two frames. Then the inter-frame pixel motion in the image is even larger. The number of pyramid levels is set to 3 and the size of the integration windows is 7×7 . This PLK method is able to handle a maximum pixel motion of $(2^{3+1} - 1) \times 7 = 105 \text{ pixels/frame}$, which is robust enough against most frame jumping situations. Compared to SAD, although the latter can work at about 70 fps , the maximum pixel motion it can handle is only 8 pixels/frame , which is not adequate, if ACE is running faster than 0.4 m/s . Moreover, the SAD algorithm cannot work precisely, if ACE moves with a curved trajectory.

B. Camera Velocity Computation

After the optical flow computation, divided by the time interval between successive images, the velocities of the 36 points of interest in the image plane are acquired. The image Jacobian matrix \mathbf{J} is applied to determine the camera velocity vector $\dot{\mathbf{r}}_c$ in 3D world according to the corresponding pixel velocity $\dot{\mathbf{f}}_p$ in 2D image plane [13]

$$\dot{\mathbf{f}}_p = \mathbf{J} \cdot \dot{\mathbf{r}}_c. \quad (1)$$

Normally there are 6 unknown components (6 DOF) in the camera velocity vector. Because the displacement of the camera in the vertical direction Δz is much smaller than the distance between the camera and the ground z and the sum of Δz is almost zero, it can be assumed that the ground is a

flat plane. The ACE-platform moves without any roll- and pitch angle (ω_x and ω_y), which can be expressed as:

$$\lim_{T \rightarrow \infty} \sum_{t=0}^T \Delta z = 0; \quad \omega_x \approx 0; \quad \omega_y \approx 0. \quad (2)$$

Under this assumption only 3 components of $\dot{\mathbf{r}}_c$ are considered: the translation in x- and y-directions as well as the orientation around z axis. Thus, (1) reduces to:

$$\begin{bmatrix} \dot{u}_1 \\ \dot{v}_1 \\ \vdots \\ \dot{u}_{36} \\ \dot{v}_{36} \end{bmatrix} = \begin{bmatrix} \frac{\lambda}{z} & 0 & -v_1 \\ 0 & \frac{\lambda}{z} & u_1 \\ \vdots & \vdots & \vdots \\ \frac{\lambda}{z} & 0 & -v_{36} \\ 0 & \frac{\lambda}{z} & u_{36} \end{bmatrix} \cdot \begin{bmatrix} T_{cx} \\ T_{cy} \\ \omega_{cz} \end{bmatrix} \quad (3)$$

where u_i, v_i ($i = 1, \dots, 36$) are the positions of the 36 points of interest along x- and y-direction in the image plane; \dot{u}_i, \dot{v}_i are the corresponding velocity components; T_{cx} , T_{cy} and ω_{cz} are the camera velocities and angular velocity in the camera coordinate system; λ represents the focal length of the camera and z is the distance between the image plane and the ground plane, which is assumed to be constant.

Eq. 3 is an over-determined system. Least squares method [14] is utilized in the form of pseudo-inverse matrix to obtain an optimal solution of the camera ego-motion.

IV. ROBOT LOCALIZATION

Using the camera velocity $\dot{\mathbf{r}}_c$ acquired in the last section, the precise position and orientation of ACE will be obtained after setting up kinematic models, system calibration and data fusion.

A. Kinematic Modeling

Since the camera is mounted on ACE rigidly, the relative position between the camera and the robot does not change. Any actuated motion of the robot results in a movement of the camera relative to its original position. Furthermore, when ACE is moving, it has no lateral velocity perpendicular to its heading direction. Similar to a unicycle, ACE has only a longitudinal velocity along the forward direction and an angular velocity around the vertical axis relative to itself in robot coordinates. But it does have translations in both x- and y-directions in world coordinates due to its current angular velocity. It can be considered a non-holonomic system [15] with constraints, which eases the discrimination of redundant results from optical flow. The final state of the system depends on the values of the momentary velocities along its trajectory. Based on the characteristics of non-holonomic systems, the momentary velocity of ACE $\dot{\mathbf{r}} = [T_{rx}, \omega_{rz}]^T$ can be transformed to the instantaneous camera velocity $\dot{\mathbf{r}}_c = [T_{cx}, T_{cy}, \omega_{cz}]^T$ at each time step (see. Fig. 4)

$$\begin{aligned} T_{cx} &= -\omega_{rz} R \cos \beta = -\omega_{rz} x_0 \\ T_{cy} &= -T_{rx} - \omega_{rz} R \sin \beta = -T_{rx} - \omega_{rz} y_0 \\ \omega_{cz} &= -\omega_{rz}, \end{aligned} \quad (4)$$

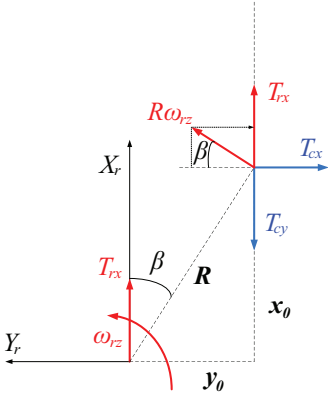


Fig. 4: Velocity transformation based on characteristics of non-holonomic systems

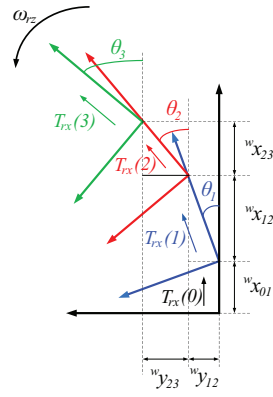


Fig. 5: Moving trajectory in world coordinate

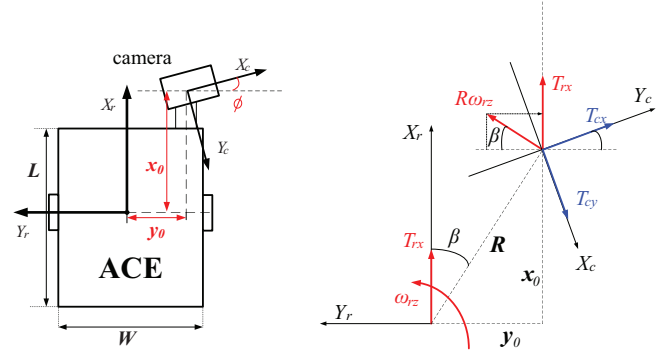


Fig. 6: Camera mounted with an angle error (left) and corresponding coordinates relationship (right)

where x_0 and y_0 are the distances between the gravity center of ACE and principal point in the image plane (Fig. 6). We reformulate (4) into:

$$\dot{\mathbf{r}}_c = \begin{bmatrix} 0 & -x_0 \\ -1 & -y_0 \\ 0 & -1 \end{bmatrix} \cdot \dot{\mathbf{r}}, \quad (5)$$

where the camera ego-motion $\dot{\mathbf{r}}_c$ is computed according to Section III. And according to (5), we can determine the robot velocity vector $\dot{\mathbf{r}}$ using least squares method. Then the incremental measurements $(x, y, \theta)_{i-1, i}^T$ between frame $i-1$ and frame i (see Fig. 5) are computed as follows:

$$\begin{pmatrix} x \\ y \\ \theta \end{pmatrix}_{i-1, i} = \begin{pmatrix} \cos \theta_{i-1} & 0 & 0 \\ 0 & \sin \theta_{i-1} & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} T_{rx} \\ T_{ry} \\ \omega_{rx} \end{pmatrix}_{i-1} \cdot t_{i-1, i}, \quad (6)$$

with $t_{i-1, i}$ the time interval between frame $i-1$ and i .

Finally the robot position and orientation at step n are computed from the incremental measurements as:

$$x = \sum_{i=1}^n x_{i-1, i}; y = \sum_{i=1}^n y_{i-1, i}; \theta = \sum_{i=1}^n \theta_{i-1, i}. \quad (7)$$

B. System Calibration

To eliminate the systematic errors, the system should be calibrated. For instance, in our model the camera coordinate system is assumed parallel to the robot coordinate system. But when the camera is being mounted on ACE, it is very difficult to achieve that. There is an error angle ϕ as shown in Fig. 6. Even if it is a very small angle, it will influence our results strongly, since we must integrate the results in each time interval to obtain the end position and orientation. The measurements x_0 and y_0 in Fig. 6 can also not be very accurate.

Two scenarios are designed to estimate ϕ , x_0 and y_0 . The data for system calibration experiments is obtained when ACE is moving in an indoor environment with proper brightness, high-contrast texture and flat surface. In the first scenario, ACE runs about 6.7 m in a straight line with constant speed, which is taken as pure translation and used to

estimate the angle error ϕ . If ACE moves in a straight line, there is no velocity in Y_r -direction. That means as shown in Fig. 6:

$$\omega_{rz} x_0 = T_{cx} \cos \phi + T_{cy} \sin \phi = 0 \quad (8)$$

According to (8):

$$\tan \phi = -\frac{T_{cx}}{T_{cy}}, \quad (9)$$

where the optimal values of T_{cx} and T_{cy} are obtained from Section III-B.

After knowing ϕ , x_0 and y_0 are estimated in a pure rotation test. ACE only rotates at the starting point and passes about 460° , through which x_0 and y_0 can be estimated. In this test, the velocity T_{rx} is supposed to be zero, so

$$\tan \beta = \frac{x_0}{y_0} = \frac{T_{cy} \cos \phi - T_{cx} \sin \phi}{T_{cx} \cos \phi + T_{cy} \sin \phi}, \quad (10)$$

and according to (4),

$$T_{cx} = \omega_{cz} \cdot x_0. \quad (11)$$

C. Data Fusion

After completing the task of robot motion estimation using visual information, we fuse it with the conventional odometry data to reduce errors and increase accuracy and reliability, accomplished by applying a Kalman filter [16]. There are two angular encoders on the wheels of the platform (*BlueBotics SA*). The position x , y and orientation θ can be directly transferred from the platform to image processing computer through ethernet. The odometry information from two different sources, camera and encoders, should be combined to localize the ACE robot.

Let $\dot{x}^e, \dot{y}^e, \omega^e$ represent velocities from conventional odometry, and $\dot{x}^v, \dot{y}^v, \omega^v$ from visual odometry. In this Kalman filtering process the state vector is $\mathbf{x}_k = [x_k, \dot{x}_k, y_k, \dot{y}_k, \theta_k, \omega_k]^T$, and the measurement vector is $\mathbf{z}_k = [\dot{x}_k^e, \dot{y}_k^e, \omega_k^e, \dot{x}_k^v, \dot{y}_k^v, \omega_k^v]^T$. There is also a control input $\mathbf{u}_k = [u_{v_k}, u_{\omega_k}]^T$ given by the system to command the acceleration of the ACE robot, so the time update equation should be:

$$\mathbf{x}_k = \mathbf{A} \cdot \mathbf{x}_{k-1} + \mathbf{B} \cdot \mathbf{u}_{k-1} + \mathbf{w}_{k-1}, \quad (12)$$

where

$$A = \begin{bmatrix} 1 & \Delta t & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & \Delta t & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, B = \begin{bmatrix} \frac{1}{2} \cos \theta_{k-1} \Delta t^2 & 0 \\ \cos \theta_{k-1} \Delta t & 0 \\ \frac{1}{2} \sin \theta_{k-1} \Delta t^2 & 0 \\ \sin \theta_{k-1} \Delta t & 0 \\ 0 & \Delta t^2 \\ 0 & 1 \end{bmatrix},$$

where Δt is the inter-frame time interval and \mathbf{w}_k is process noise. The relationship between the state vector and the measurement vector is:

$$\mathbf{z}_k = H \cdot \mathbf{x}_k + \mathbf{v}_k, \quad (13)$$

where $H = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$, and \mathbf{v}_k is measurement

noise. The measurement noise covariance and process noise covariance in the Kalman-filter are assumed to be constant and determined empirically.

V. EXPERIMENTAL RESULTS

In order to evaluate the robustness of our algorithm under different conditions, experiments have been conducted under five different ground conditions: marble, cement sidewalk, cobblestone sidewalk, asphalt road, and sands. Based on large amounts of measurements and experimental results, the average error rate of the conventional odometry in indoor environments is around 2.5%, while in outdoor environments it lies between 3% and 5%. In some extreme terrain, the estimation of the rotated angle of ACE is too erroneous, causing localization to fail. In this case, visual odometry provides a more accurate result.

A. Pure Translation and Pure Rotation Results

Pure translation and pure rotation tests are carried out indoor on marble surface. It is also necessary for the system calibration. Fig. 7 indicates the translation and rotation results in the pure translation test after using the calibration parameters, while the pure rotation results are illustrated in Fig. 8. In the pure translation test the error rate is reduced to only 1% after system calibration compared with ground truth which is measured using conventional measuring tools.

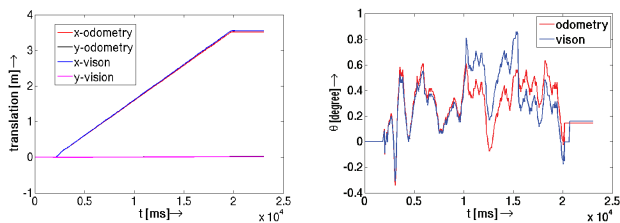


Fig. 7: Translation (left) and rotation (right) results in the pure translation test

In the pure rotation test ACE has passed about 375° around its vertical axis and the translation error is no more than 3 mm after system calibration. Our system calibration has performed well and eliminated the systematic errors successfully.

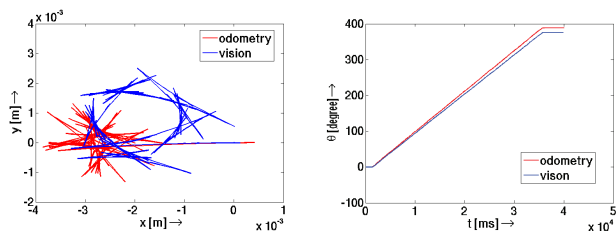


Fig. 8: Translation (left) and rotation (right) results in the pure rotation test.

B. Square Trajectory Test

Since there is no accurate ground truth in our experiment, a square trajectory test is designed to prove the validity of the algorithm, because the trajectory is closed. The distance between the real end point and the end point estimated by the wheel encoders and the vision data indicates the accuracy of the performance. This experiment is conducted on a cobblestone sidewalk. There is only a small distance between the start point (black cross) and end point (black point) which can be easily measured by traditional methods. From Fig. 9 it can be seen that the end point computed from visual odometry is closer to the true value compared with that from conventional odometry. Comparing the relative distances between the endpoints using conventional odometry and visual odometry, an approximate improvement of 50% using the latter one is achieved. Our visual odometry system has a very good performance under poor ground conditions.

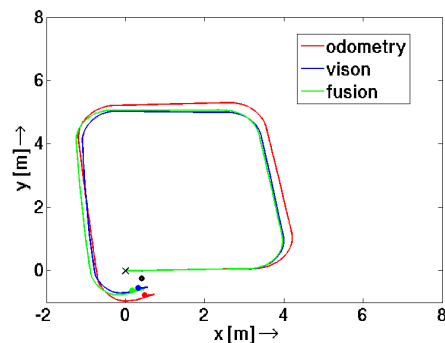


Fig. 9: Trajectory results in a square trajectory test, blue, red and green points represent the end points computed from visual odometry, angular encoder odometry and the fusion results respectively

C. Circle Trajectory Test

A circle trajectory test is used to verify whether the algorithm is valid when ACE has translation and rotation at the same time. Considering the different textures in varying terrains, this experiment is conducted on asphalt road surface to test the result quality in a relatively low-contrast environment. ACE moved in two circles over a total length of 25 m long. Fig. 10 shows the trajectory results of this scenario. The visual odometry result is also much better on

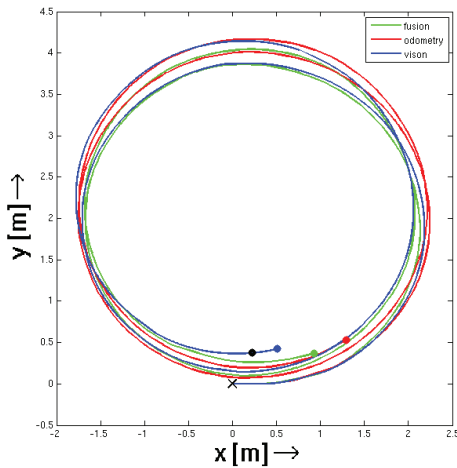


Fig. 10: Trajectory results in a circle trajectory test

asphalt road surface. Therefore, the fusion corrects the result of the conventional odometry.

D. Discussion

From more than 30 test runs on different ground surfaces, some representative results are shown. The fusion is a compromise between vision data and wheel encoders. The former one performs better, if environmental conditions such as lighting conditions are appropriate, while the latter one works well under suitable ground condition. Since our navigation scenario contains different environments and various ground conditions, the fusion is robust against sensor failure and ensures a general improvement of the whole performance. An elaborate sensor selection model is being considered. Furthermore, the effect of the fusion algorithm is sometimes not prominent, because of the simplification under assumption that the system is a linear system using a normal Kalman-filter. It is being improved by utilizing an extended Kalman-filter.

In some situations, this algorithm is still not robust enough since the localization result accuracy is influenced by changing environment brightness and moving objects in the field of view. For example, sometimes fallen leaves lead to errors in optical flow computation. Attention should be paid to the enhancement of robustness against noises caused by changing environments.

VI. CONCLUSIONS

The visual odometry system proposed in this paper consists of optical flow computation based on PLK, camera motion estimation using image Jacobian matrix and least square method, a novel experimental system calibration procedure, robot motion reconstruction and fusion of encoders and vision data using Kalman-filter. Each part is designed and implemented according to the system requirement for an accurate and safe navigation in outdoor urban environments. The visual odometry system has shown a very good complement and is a good correction for the conventional

odometry. This algorithm runs at 30 fps. The error rate has been reduced from 5% or more to 1% – 2%. Two odometry systems complement each other in some extreme terrains where the result from one of them is not acceptable.

VII. ACKNOWLEDGMENTS

This work is supported in part within the DFG excellence initiative research cluster *Cognition for Technical Systems – CoTeSys*, see also www.cotesys.org and the BMBF funded Bernstein Center for Computational Neuroscience Munich, see also www.bccn-munich.de. We would like to thank the other ACE-Team members (G. Lidoris, K. Klasing, A. Bauer, T. Xu, Q. Muehlbauer, S. Sosnowski, F. Rohrmueller and D. Wollherr) for their excellent work designing and implementing the ACE platform.

REFERENCES

- [1] G. Lidoris, K. Klasing, A. Bauer, T. Xu, Q. Muehlbauer, T. Zhang, S. Sosnowski, F. Rohrmueller, K. Kuehnlenz, D. Wollherr, and M. Buss, *ACE - The Autonomous City Explorer Project*, www.ace-robot.de, 2008.
- [2] J. Campbell, R. Sukthankar, I. Nourbakhsh, and A. Pahwa, “A robust visual odometry and precipice detection system using consumer-grade monocular vision,” in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2005.
- [3] H. Wang, K. Yuan, W. Zou, and Q. Zhou, “Visual odometry based on locally planar ground assumption,” in *Proceedings of the IEEE International Conference on Information Acquisition*, 2005.
- [4] D. Nister, O. Narodisky, and J. Bergen, “Visual odometry,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, 2004.
- [5] C. Dornhege and A. Kleiner, “Visual odometry for tracked vehicles,” in *Proceedings of the IEEE International Workshop on Safety, Security and Rescue Robotics (SSRR)*, 2006.
- [6] J. Borenstein and L. Feng, “Gyrodometry: A new method for combining data from gyros and odometry in mobile robots,” in *Proceedings of the IEEE International Conference on Robotics and Automation*, 1996.
- [7] D. Fernandez and A. Price, “Visual odometry for an outdoor mobile robot,” in *Proceedings of the IEEE Conference on Robotics, Automation and Mechatronics*, 2004.
- [8] J. Bouguet, *Camera Calibration Toolbox for Matlab*, http://www.vision.caltech.edu/bouguetj/calib_doc, 1999.
- [9] J. Barron, D. Fleet, and S. Beauchemin, “Performance of optical flow techniques,” *International Journal of Computer Vision*, vol. 12:1, pp. 43–77, 1994.
- [10] *AMD64 Architecture Programmer’s Manual Volume 4: 128-Bit Media Instructions*.
- [11] B. Lucas and T. Kanade, “An iterative image registration technique with an application to stereo vision,” in *Proceedings of the DARPA Image Understanding Workshop*, 1981.
- [12] J. Bouguet, “Pyramidal implementation of the lucas kanade feature tracker description of the algorithm,” *OpenCV Documentation, Intel Corporation, Microprocessor Research Labs*, 1999.
- [13] S. Hutchinson, G. Hager, and P. Corke, “A tutorial on visual servo control,” *IEEE Transactions on Robotics and Automation*, vol. 12:5, pp. 651–670, 1996.
- [14] S. R. Buss, “Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods,” Department of Mathematics, University of California, San Diego, Tech. Rep., 2004.
- [15] C. Zhang, D. Arnold, N. Ghods, and A. S. M. Krstic, “Source seeking with non-holonomic unicycle without position measurement and with tuning of forward velocity,” in *Proceedings of the IEEE Conference on Decision and Control*, 2006.
- [16] G. Welch and G. Bishop, “An introduction to the kalman filter,” Department of Computer Science, University of North Carolina, Tech. Rep., 2006.