16th IEEE International Conference on Control Applications
Part of IEEE Multi-conference on Systems and Control
Singapore, 1-3 October 2007

From Vision to Realtime Motion Control for the RoboCup Domain

Dennis Bruijnen, Wouter Aangenent, Jeroen van Helvoort, and René van de Molengraft

Abstract— **In this paper, an overview is given of how the path from vision to motion has been developed in the TechUnited team. The vision module includes: (i) color calibration using a union of convex hulls to select an area in the 3D-colorspace, (ii) automatic calibration of the mapping from the camera image to the field via a genetic algorithm, (iii) self localization based on field lines. The output of the vision module is used by the motion module which includes: (i) vision and encoder sensor fusion by monitoring the drift caused by odometry, (ii) generating a motion path complying with the robot's limitations to prevent wheel slippage, (iii) collocated motion control. In contrast to closing the loop on vision, our approach uses wheel encoders as the basis for motion control, which has several advantages such as less delay due to a higher sampling frequency. Vision is only used to compensate for slow drift caused by slip in the wheel-surface contact.**

I. INTRODUCTION

RoboCup [13] is an international joint project to stimulate research in the field of artificial intelligence, robotics and related fields. A robot soccer game is used as the central topic where a wide range of technologies can be developed and integrated. To establish intelligent motion for autonomous robots, sensing, signal processing, decision making and actuation have to work properly in an uncertain changing environment. Common sensors for the RoboCup domain are (omni-vision) cameras, encoders measuring the wheel rotations, lasers, and ultrasonic sensors. Using the sensor data, the robot's position and the location of the surrounding objects can be estimated. From this incomplete and distorted information, a desired path for the robot to track has to be generated. The path generation is rendered into actual robot displacement via the actuators.

Self localization is a vital part in intelligent robot motion where a lot of research has been focussed at in the recent years [1], [6], [7], [11], [16]. Especially self localization using omni-vision data has been considered, the performance of which heavily depends on calibration. Recent research in this area focusses on color segmentation [2], [18], auto-calibration [12], and adaptivity [11]. Results about path generation and tracking using this data can be found in [3], [5], [14], [19].

Encoders are mostly considered to be too inaccurate for self localization due to drift caused by e.g. wheel slippage [11] and numerical integration. The cause of wheel slippage can be divided into two parts: (i) slow drift due to finite stiffness of the wheel contact surface with the ground, (ii) fast drift by applying a too high torque regarding the grip and the wheel load. Fast drift can be largely prevented by limiting the motor torques satisfying the robot's limitations. Only if the robot is colliding with or pushing against other objects, fast drift can occur.

In this paper, we give an overview of how the path from vision to motion has been developed in the TechUnited team [15]. The main innovative contributions are the following: (i) color calibration via a set of 3D convex hull's added or subtracted in the YCbCr-colorspace, (ii) fully automated mirror calibration via a genetic algorithm [9] using line information extracted from an omni-vision picture compensating for manufacturing tolerances and mirror geometry, (iii) self localization by using extracted line points and optimization similar to [7]. Collocated control is applied using encoders on the motor shafts and the vision information is only used to update the robot's pose if the drift exceeds a specified limit. Such encoder centralized approach has the advantage that the motion control can be done at a much higher sampling frequency, typically 1 kHz, than if the loop is closed on vision, typically 30 Hz. A higher sampling frequency has the advantage that feedback and feedforward control performance is better due to e.g. less time delay caused by time discretization [4].

First, the application area is described. After that, the design of the TechUnited robot is briefly presented. Then, calibration and self localization are presented. Furthermore, motion signal processing is elaborated including the drift correction algorithm, path generation and motion control. Finally, we end with conclusions.

II. ROBOCUP MIDDLE SIZE LEAGUE

From the existing leagues of RoboCup [13], team TechUnited [15] participates in the middle size league. In Fig. 1, an impression is given of a soccer game in this league.

Robots are equipped with laptops and play soccer autonomously. The regulations consist of the FIFA rules with a list of modifications such that it is suited for state-of-the-art robotics. Each year, some modifications are removed, eventually converging to the actual FIFA rules which is the long term goal. As an example, objects can be distinguished by its color: the two goals are blue and yellow respectively, field lines are white, the field itself is green, the ball is orange and all robots are supposed to be black. Another example is the field size, which increases by the years.

III. TECHUNITED ROBOT

Development of the TechUnited robot, the so-called TURTLE platform (TechUnited Robot Team: Limited Edition), was started in 2005. A picture of the robot is shown in

All authors are with the Faculty of Mechanical Engineering, Control Systems Technology, Technische Universiteit Eindhoven, The Netherlands d.j.h.bruijnen@tue.nl

1-4244-0443-6/07/$20.00 ©2007 IEEE.

Fig. 1. A picture taken during a soccer game at the RoboCup World Championship in Bremen 2006, TechUnited vs. Cops Stuttgart

Fig. 2. A key aspect of the robot is its modular design in software and hardware which enables rapid development. Each robot is equipped with a notebook running RTAI-Linux [8], a realtime operating system, in combination with Matlab/Simulink/RTW [10] which provides a huge amount of toolboxes and provides a modular framework for development. For data acquisition and motion control, the robot is equipped with two TUEDACS devices [17] which are connected via a realtime USB 2.0 connection.

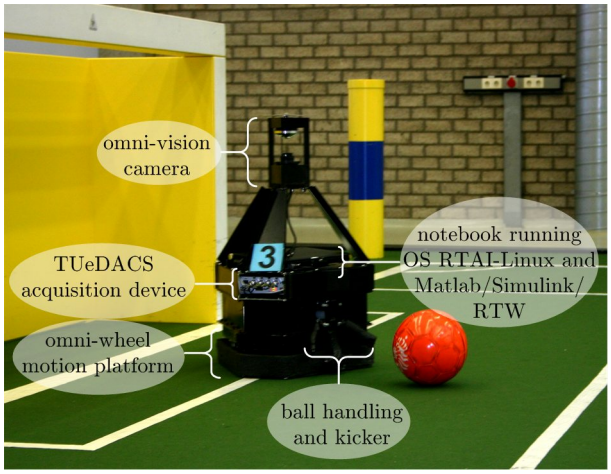

Fig. 2. The TechUnited robot.

Also the hardware is modular. The robot consists of three layers which can be interchanged with other robots, (1) omni-vision camera, data-acquisition devices and the notebook, (2) ball handling and kicker, (3) driven wheels and batteries. Software and hardware are connected via a single USB hub connection and a single firewire-connection for the omni-vision camera. As shown in Fig. 3, three omni-wheels are used for motion, which is an obvious choice because a holonomic system is obtained which makes motion tasks easier. Omni-wheels are able to apply a torque in one direction whereas the omni-wheel can roll freely in the orthogonal direction via small rollers around the circumference. Suspension stiffness of the omni-wheels with respect to each other is important to prevent parasitic vibrations due to a varying point of support for each omni-wheel which is inherent to omni-wheels with two rows of small rollers around the circumference as visible in Fig. 3.

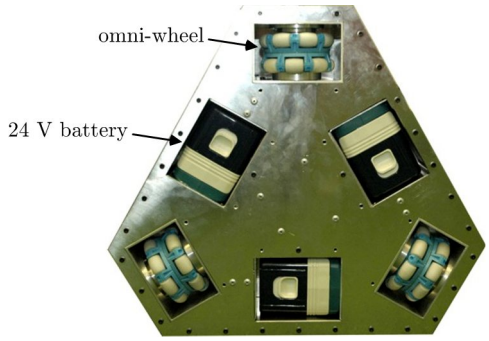

Fig. 3. The bottom view of the TechUnited robot.

IV. VISION SIGNAL PROCESSING

The omni-vision camera is a sensor to determine the absolute position of the robot and surrounding objects. This information is used by the behavior module to obtain a desirable behavior of the robot. The goal of the vision module is to deliver as complete and accurate as possible information about the surroundings, such that the behavior module will make the correct decisions. The performance of the vision module heavily depends on calibration. Regarding this calibration, color segmentation and auto mirror calibration will be discussed. After that, self localization and obstacle localization will be presented.

A. Color Segmentation

The RoboCup domain typically uses color segmentation based on rectangular thresholds because of its simplicity and computational cheap implementation [2]. However, tuning is difficult and not robust to light variations. The bounds are conservative and have to be tuned while making compromises. Adaptive color segmentation methods have been proposed to overcome this problem [18], however they are less computational efficient.

Our approach is based on creating tight bounds in a 3D colorspace (e.g. Y-Cb-Cr) by using 3D convex hull's. The calibration data is saved in a 3D lookup table. For a set of images made at several locations on the field, a region of pixels is selected using blob growing or a user-defined polygon. For this selection, a 3D convex hull is computed. For the points of the 3D lookup table which lie in this 3D convex hull, a color label can either be assigned or removed. By iterating for the set of images and colors, the color label regions are tightly shaped in the 3D lookup table without compromising for rectangular thresholds. The result is a fast color segmentation using a 3D lookup table which is robust

against light variations over the field. In Fig. 4, an example is shown of such a 3D lookup table.

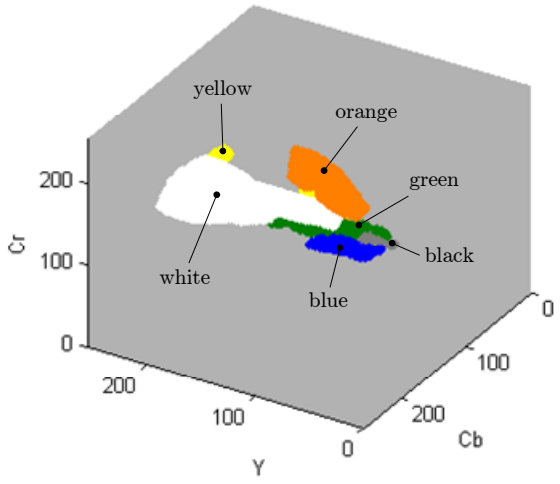

Fig. 4. 3D lookup table used for color segmentation.

During the World Championship 2006 in Bremen and the European Championship 2007 in Hannover, it appeared to be robust against the light variations during games. Although the lookup table is fixed during a soccer game, robustness against light variations is obtained by adding a sufficient amount of images to the set of calibration images over the field.

B. Genetic Algorithm based Automatic Mirror Calibration

For localization purposes, the relative position of each pixel on the field to the robot has to be determined. Because of deformations and manufacturing tolerances, the mirror and camera position with respect to the field are different for each robot and hence calibration is required to determine the mapping from image to field coordinates. Our goal is an automatic calibration method where the robot can be placed on the field after which calibration is done automatically. To achieve this, we have extended the method described in [7] with a genetic algorithm [9]. Line-points are detected in radial directions after which a parameterized mapping is optimized using a genetic algorithm. We use a genetic algorithm because we want to obtain the global optimum of a non-convex optimization problem which is described below.

In Fig. 5, an overview of the mapping from the image frame I to the field frame F is shown. First, we manually select the robot center $\underline{c}^I = \begin{bmatrix} c_x \\ c_y \end{bmatrix}$ in the image frame I. The distance in pixels from point $\underline{p}_i^I = \begin{bmatrix} p_{x,i} \\ p_{y,i} \end{bmatrix}$ to this center, $||\underline{c}^I - \underline{p}_i^I||_2$, is mapped to a distance to the robot $p_{d,i}$ in meters:

$$p_{d,i} = a_1 \tan(a_2 ||\underline{c}^I - \underline{p}_i^I||_2) \tag{1}$$

and the angle $p_{\varphi,i}$ becomes:

$$p_{\varphi,i} = \angle \left(\underline{c}^I - \underline{p}_i^I \right). \tag{2}$$

Suppose that the pose of the robot is $\underline{r}^F = \begin{bmatrix} r_x \\ r_y \\ r_\varphi \end{bmatrix}$ with respect to the field frame F, then the position of point $\underline{p}_i^I$ in the image on the field is:

$$\underline{p}_i^F = \begin{bmatrix} r_x + p_{d,i} \cos(p_{\varphi,i} + r_\varphi) \\ r_y + p_{d,i} \sin(p_{\varphi,i} + r_\varphi) \end{bmatrix} \tag{3}$$

The parameters which have to be calibrated are: c_x, c_y, a_1, a_2, r_x, r_y and r_φ. The objective function to be minimized is chosen as:

$$J = \sum_{i=1}^{N_{linepoints}} D\left(\underline{p}_i^F\right) \tag{4}$$

where D is a mapping $\mathbb{R}^2 \rightarrow \mathbb{R}^+$ being monotonically increasing with the shortest distance from $\underline{p}_i^F$ to a field line. To reduce computational time, mapping D is represented by a lookup table. J is minimized using a genetic algorithm. Less than a minute is needed on a standard Pentium to obtain a satisfactory result. An example of such a result is shown in Fig. 5.

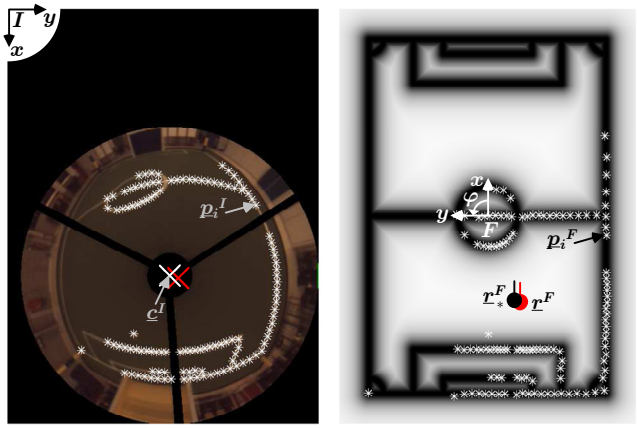

Fig. 5. Automatic mirror calibration via a genetic algorithm

Next, a mask is used to remove redundant image data as shown in the left figure. From the image center, line points are collected. By using (1), (2) and (3), the estimated locations of the line points are computed as shown in the right figure. This a good check to assess the quality of the auto mirror calibration. The red dot is the estimated position $\underline{r}^F$ which corresponds to the optimized mirror center (red cross in the left figure). A remarkable observation is that this point deviates slightly from the actual robot center (white cross in the left figure). This way, manufacturing tolerances are compensated. Fortunately, this offset is constant, hence the offset can be taken into account for the self localization. The pose corrected for this offset will be denoted as $\underline{r}_*^F$.

C. Self Localization using Line Detection and Optimization

The four calibrated mirror parameters, c_x, c_y, a_1 and a_2, and the offset correction in x and y are used by the vision module of the robot for self localization. The same objective function (4) is used to determine the remaining 3 parameters

being the pose of the robot $\underline{r}^F$. Because of the limited available computational time a global optimization routine such as a genetic algorithm is not suitable for real-time application. Instead, a Nelder-Mead simplex algorithm is used for local optimization without using gradient information. If the initial pose is close to the real pose (suppose that the robot knew where it was the previous sample) than the optimization method will easily find the real pose within 20-50 iterations (computational time $<$ 5 ms). If not enough line points are found or the objective function value does not drop below some threshold (which is a measure for the average distance of the line points to the field lines), then the robot's pose is lost and will not be used further. Random poses around the field will be used as an initial guess until the objective function value is smaller than the threshold again. Normally, if new images are of sufficient quality for self localization, this position is found again within 100 samples. For a vision sampling rate of 32 Hz, this is only about 3 s.

In the order of 100 line points are used to obtain an accurate estimate of the robot's pose due to averaging of those line points. The position estimation accuracy lies in the order of 0.1 m as observed during experiments, however, further investigation is required to quantify this.

V. MOTION SIGNAL PROCESSING

A. Drift Correction Algorithm

The pose estimation from the vision has some drawbacks to use it for feedback control: (i) the sampling rate is too low for fast motion control, (ii) the data can be quite noisy due to light variations and robot vibrations, (iii) the pose is lost occasionally due to the presence of obstacles that block the robot's view. From a motion control point of view, it would be much more desirable to use the encoders to determine the change in pose. The signal is less noisy, less sensitive to influences from the environment (such as light conditions and obstructed view) and a much higher motion sampling rate is possible. To overcome the drawbacks of both sensing methods, their information is merged.

The information of the encoders is used to estimate the pose of the robot and to thightly control the motors of the robot. The vision information is used to occasionally update the estimation of the pose of the robot, provided that some conditions are fulfilled. These conditions are:

C1: The pose estimation from the vision is to be trusted.
C2: The pose estimate from the vision significantly differs from the pose estimate from the encoders.
C3: The pose estimates differs for a prolonged period.

The threshold on the objective function value of the vision pose estimation is equal to condition C1. If the objective function value is too high, the vision pose estimation is not to be trusted and the pose information of the encoders is used instead.

Condition C2 prevents abundant updates, by imposing a minimum difference between vision and encoder estimates. Since the vision estimation is only accurate to within a certain margin, the difference between the two estimates should at least indicate that the encoder estimate is not within this margin.

To suppress the effects of noise in the image, condition C3 is added. Conditions C1 and C2 are required to be fulfilled for a number of images, before the pose estimate is updated. Of course, postponing the update causes some delay, however, if for instance a period of 3 images is required, the delay is only approximately (3-1)/32 = 0.06 s which is considered small at the level of path generation / task assignment.

In Fig. 7, a schematic representation of the drift update procedure is shown. Here, C1, C2 and C3 are the three conditions that determine whether the vision pose estimate $\underline{r}_*^F(t)$ is used or the encoder pose estimate $\underline{r}_{e*}^F(t)$. $d\underline{r}^O(t)$ represents the change in position since the previous sample determined by odometry. The resulting estimated robot pose is denoted by $\underline{r}_e^F(t)$.

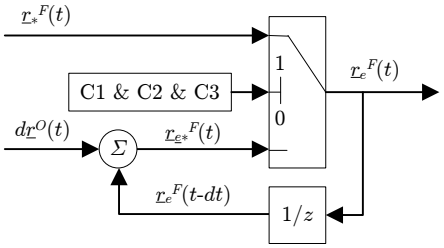

Fig. 7. Schematic representation of the pose update procedure

B. Motion Path Generation

In [3], a computationally cheap algorithm is presented which generates a motion path complying with the robot's physical limitations such as velocity, acceleration and jerk limitations in all directions. A simulation example is shown in Fig. 8.

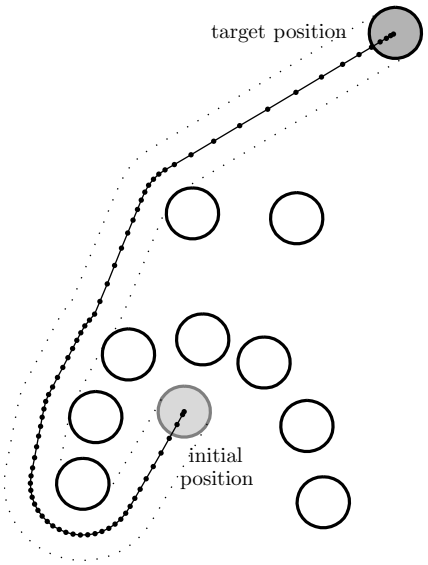

Fig. 8. Simulation of the motion path generation algorithm.

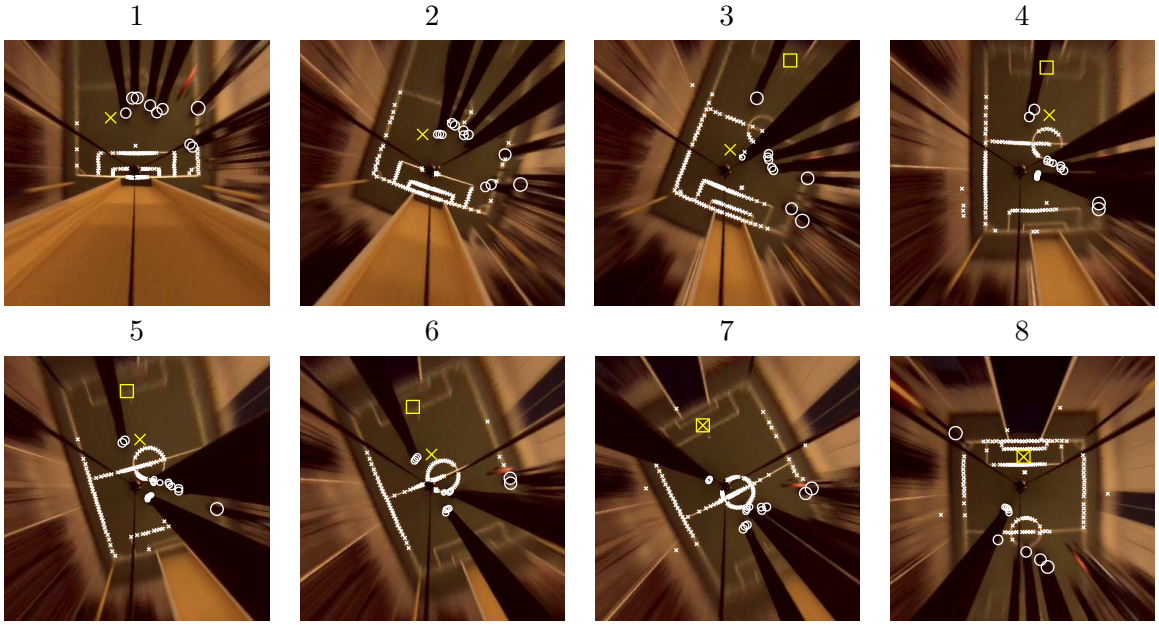

Fig. 6. Experiment where the robot has to move across the field while avoiding the obstacles. Images on the robot's path are captured and shown above. For each image, the following marks are also shown: target (square), subtarget (large cross), detected obstacles (circles), detected line points (small crosses).

This algorithm has been implemented on the robot. Experimental results are shown in Fig. 6. The omni-vision image is mapped to a top view and the detected objects are plotted in the images. This experiment shows that the algorithm works in practice and is robust for uncertainties regarding self and obstacle positions. The resulting (sub)target is denoted by $\underline{r}^F_{target}(t)$.

C. Motion Control

In Section V-A and V-B the robot's pose $\underline{r}^F_e(t)$ and its target position $\underline{r}^F_{\text{target}}(t)$ have been determined. The remaining issue is that of motion control: how do we move from $\underline{r}^F_e(t)$ to $\underline{r}^F_{\text{target}}(t)$? For this purpose, the schematic overview of Fig. 9 has been implemented in Matlab/Simulink/RTW, where RTW automatically generates real-time code from this implementation. The three encoders, denoted by E, measure the radial displacement $d\underline{r}^E(t)$ of the motor (M) axles in the encoder frame E. This radial displacement is then transformed by $\underline{T}^{OE}$ to the odometry frame O and integrated resulting in the robots pose based on odometry $\underline{r}^O(t)$.

Due to drift, the odometry frame O moves with respect to the field frame F. As a simple solution, one could think of resetting the pose based on odometry to the vision-derive pose estimate ($\underline{r}^O(t)$=$\underline{r}^F_e(t)$) every time a vision update is available. However, since motion control is done in a collocated fashion, this would result in frequent discontinuities in the control loop signals, which is undesirable. Therefore, a much better solution from a motion control point of view is to transform the target $\underline{r}^F_{\text{target}}(t)$ to the odometry frame O. This is done with the transformation $\underline{T}^{OF}(\underline{r}^O(t))$. The summation of $d\underline{r}^O(t)$ and $\underline{r}^O(t)$ yields the desired target with respect to the odometry frame O. In order to account for physical limitations of the robot (e.g. maximum acceleration and speed), a desirable reference trajectory from $\underline{r}^O(t)$ to $\underline{r}^O_{target}(t)$ is computed by the smoothifier S, [3]. The difference between the reference trajectory $\underline{r}^O_{ref}(t)$ and $\underline{r}^O(t)$ is transformed back to the encoder frame E used for collocated feedback control (C). The desired acceleration and velocity profiles $\ddot{\underline{r}}^O_{ref}(t)$ and $\dot{\underline{r}}^O_{ref}(t)$, respectively, are used to compute the feedforward signal by F, which also includes the transformation $\underline{T}^{EO} = \left(\underline{T}^{OE}\right)^{-1}$, to improve tracking performance.

The motion control module is implemented at a sampling rate of 1 kHz. There is a substantial amount of backlash in the wheel modules, and since the dynamic response of the modules is subject to varying wheel pressure e.g. in case the robot accelerates or is lifted due to a collision, the frequency range of interest for the closed loop dynamics is 1 Hz to 50 Hz. Together with the tuned feedforward, this results in satisfactory performance. In Fig. 10, the frequency responses of the three wheel modules are depicted, measured while moving sideways as well as moving forward/backward. Since all wheel modules show similar dynamic behavior for both directions which is a mass-line with a highly damped resonance/anti-resonance peak, a single LTI controller is designed for all three loops.

VI. CONCLUSIONS

In this work, we presented how the TechUnited robot is controlled by using encoder information from the wheels and image data from the omni-vision camera. By generating

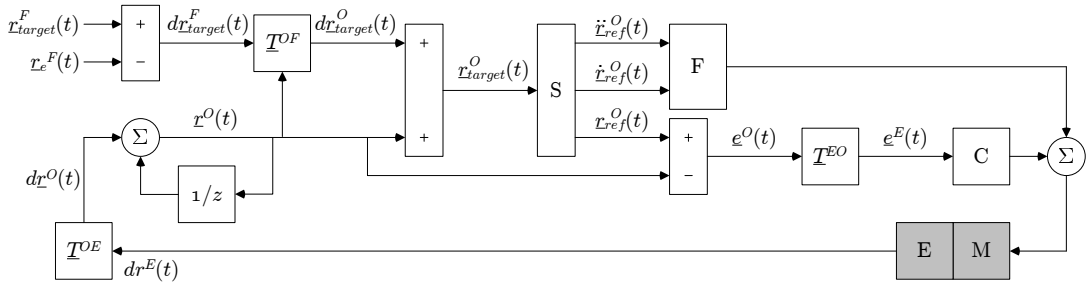

Fig. 9. Schematic representation of the motion control module

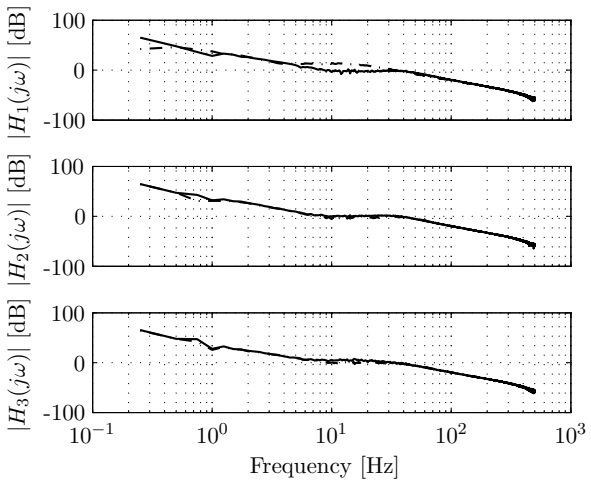

Fig. 10. Magnitudes of frequency responses for all three motors, measured while moving sideways (solid) and while moving forward/backward (dash-dotted)

motion paths, fast drift is largely prevented, hence, the encoders can be used as the basis for the self localization where the vision only compensates for the slow drift. This results in a robot position estimation which is robust to wheel slippage and temporarily failing pose estimates from the vision module.

Advantages of this approach are: (i) a much higher sampling frequency for the control loop can be used resulting in a better motion performance, (ii) self localization accuracy is still high because drift is compensated by vision, (iii) self localization is less dependent on environment conditions. If the vision module is not able to compute the robot's position for some period e.g. due to objects blocking the view, the robot can continue its actions because it can accurately drive further using the encoder information.

REFERENCES

[1] A. Bais and R. Sablatnig. Landmark based global self-localization of mobile soccer robots. In Narayanan P.J., Nayar S.K., and Shum H.-Y., editors, *Proc. of 7th Asian Conference on Computer Vision*, volume 2 of *Lecture Notes in Computer Science 3852*, pages 842–851, Hyderabad, India, 2006. Springer.

[2] J. Bruce, T. Balch, and M. Veloso. Fast and cheap color image segmentation for interactive robots. In *In Proc. of the 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2061–2066, 2000.

[3] D. Bruijnen, J. van Helvoort, and R. van de Molengraft. Realtime motion path generation using subtargets in a changing environment. In *Proceedings of the 2006 American Control Conference*, pages 4243–4248, 2006.

[4] G. F. Franklin, D. J. Powell, and A. Emami-Naeini. *Feedback Control of Dynamic Systems*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2001.

[5] R. Hafner and M. Riedmiller. Reinforcement learning on a omnidirectional mobile robot. In *IEEE/RSJ International Conference on Intelligent Robots and Systems for Human Security, Health, and Prosperity*, 2003.

[6] L. Iocchi and D. Nardi. Self-localization in the robocup environment. In *RoboCup-99: Robot Soccer World Cup III*, pages 318–330, London, UK, 2000. Springer-Verlag.

[7] M. Lauer, S. Lange, and M. Riedmiller. Calculating the perfect match: An efficient and accurate approach for robot self-localization. In *RoboCup*, pages 142–153, 2005.

[8] P. Mantegazza, E. L. Dozio, and S. Papacharalambous. RTAI: Real Time Application Interface. *Linux J.*, 2000(72es):10, 2000.

[9] The MathWorks. Genetic algorithm and direct search toolbox.

[10] The MathWorks. Real-Time Workshop.

[11] A. Merke, S. Welker, and M. Riedmiller. Line based robot localization under natural light conditions. In *European Conference on Arificial IntelligenceMachine Learning (ECAI)*, 2004.

[12] B. Mičušík and T. Pajdla. Para-catadioptric camera auto-calibration from epipolar geometry. In *Proc. of the Asian Conference on Computer Vision (ACCV)*, pages 748–753, Seoul, Korea South, 2004. Asian Federation of Computer Vision Societies.

[13] RoboCup. http://www.robocup.org.

[14] R. Rojas and A. G. Förster. Holonomic control of a robot with an omnidirectional drive. *KI - Knstliche Intelligenz*, 20(2):12–17, 2006.

[15] TechUnited. http://www.techunited.nl, 2006.

[16] H. Utz, A. Neubeck, G. Mayer, and G. K. Kraetzschmar. Improving vision-based self-localization. In Gal Kaminka, Pedro Lima, and Raul Rojas, editors, *RoboCup-2002: Robot Soccer World Cup VI*, volume 2752 of *Lecture Notes in Artificial Intelligence*, Berlin, Heidelberg, Germany, 2002. Springer-Verlag.

[17] M. J. G. v.d. Molengraft, M. Steinbuch, and A. de Kraker. Integrating experimentation into control courses. *IEEE Control Systems Magazine*, 25:40–44, 2005.

[18] Z. Wasik and A. Saffiotti. Robust color segmentation for the robocup domain. In *Proc. of the Int. Conf. on Pattern Recognition (ICPR)*, Quebec City, Quebec, CA, 2002.

[19] K. Watanabe, Y. Shiraishi, S. G. Tzafestas, J. Tang, and T. Fukuda. Feedback control of an omnidirectional autonomous platform for mobile service robots. *J. Intell. Robotics Syst.*, 22(3-4):315–330, 1998.