ABOUT THE CONTROL OF HIGH SPEED MOBILE INDOOR ROBOTS

Kai Lingemann, Andreas Nüchter, Joachim Hertzberg

University of Osnabrück
Institute for Computer Science
Albrechtstraße 28
D-49069 Osnabrück, Germany

Hartmut Surmann

Fraunhofer Institute for
Autonomous Intelligent Systems (AIS)
Schloss Birlinghoven
D-53754 Sankt Augustin, Germany

ABSTRACT

This paper describes the control algorithms of the high speed mobile robot Kurt3D. Kurt3D drives up to 4 m/s autonomously and reliably in an unknown office environment. We present the reliable hardware, fast control cycle algorithms and a novel set value computation scheme for achieving these velocities. In addition we sketch a real-time capable laser based position tracking method that is well suited for driving with these velocities.

1. INTRODUCTION

Many car companies use their experience gathered during car races to improve their products. In analogy we have equipped the KURT2 robot platform [4] with more powerful motors and designed high speed control and localization algorithms to build more reliable robots. The developed control cycle including set value computation is fast (100 Hz), real time capable and easy to maintain. The motor controller consists of a feed forward PI controller combined with a lookup table for mapping speed to PWM values and linearizating the motor signal. It combines open and closed loop control concepts and ensures a fast response time with only a small overshoot. In the control cycle, the set values for the controllers are computed by one of three methods: Joysticks commands for manual control, fuzzy rules for autonomous driving, or a coordinate based control for goal directed motion. The control loop is round off with the precise planar pose tracking algorithm HAYAI that was first presented in [6]. Features are extracted from two consecutive laser scans and matched in order to track the robot's movement. The resulting control architecture shows many benefits even for slower robots.

The rest of the paper is organized as follows: Next we present related work and the robot Kurt3D itself. Then we discuss the high speed control, followed by the pose tracking algorithm. Since the spirit of autonomously driving high speed robots cannot really be presented in a paper, we advise the reader in Section 4 to view the WWW resources in order to gain an impression of the experiments and results. Section 5 provides a summary and outlook.

1.1. Related Work – High Speed Robotics

Current robots that drive indoor with "high speed" are reported achieving maximum velocities of 0.4 m/s to 1.6 m/s [5, 9, 11]. Santiso et al. present an algorithm for localization within a given map while driving with 2 m/s [8], without actually performing real world experiments at that speed, though. Methods for dealing with special aspects of driving and controlling a robot with high velocity, e.g., obstacle avoidance that takes into account the robot's speed, or reactively adapting paths for optimizing the driven trajectory, can be found in [1, 3, 9].

In applications with restricted, predefined environments, i.e., RoboCup, robots of a size similar to Kurt3D and with maximum velocities in the range of several meters per second can be found. However, their maximal velocities cannot be controlled autonomously nor can the pose be tracked in general environments like an office building [12]. In RoboCup scenarios, a high acceleration is mandatory to be successful, but is only maintained during a very short period of time due to space restrictions of the play field [13]. In the rest of the paper, we consider as *high-speed* for an indoor robot all speeds > 2 m/s, i.e., speeds considerably faster than walking speed.

1.2. The Mobile Robot Kurt3D

Kurt3D (Fig. 1, left) is a mobile robot platform with a size of 45 cm (length) $\times$ 33 cm (width) $\times$ 47 cm (height) and a weight of 22.6 kg. Its theoretical maximal velocity (idle motion) is 5.2 m/s; actually driving under autonomous control, up to 4 m/s has been achieved. It is equipped with a 3D laser range finder. Two 90 W (short-time 200 W) motors are used to power the 6 wheels, whereas the front and rear wheels have no tread pattern to enhance rotating. The core of the robot is a Pentium-III-600 MHz with 384 MB RAM.

The main sensor of Kurt is a tiltable laser range finder. While driving, the scanner is used in fixed horizontal position only. 181 distance values are measured in 13 ms. This high frequency is the basis of the safe and reliable control and localization.
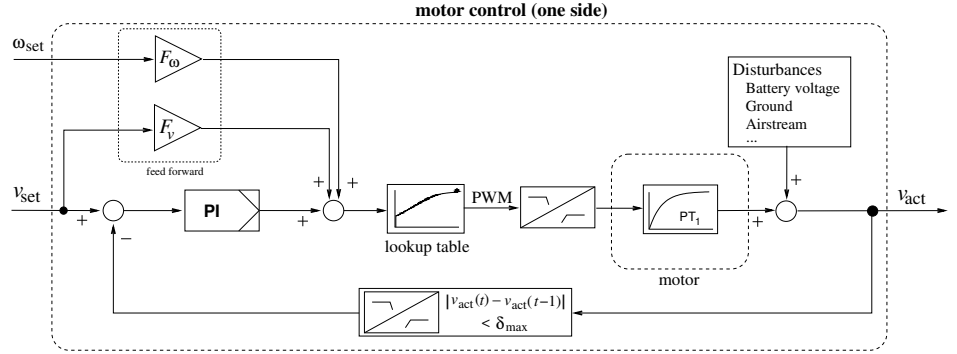
Figure 1: Left: The mobile robot Kurt3D, equipped with a 3D laser scanner based on a SICK LMS-200. Right: Control unit for one motor, based on a feed forward PI controller, thus combines open and closed loop control concepts.
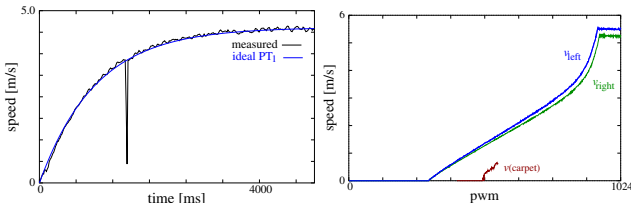

Figure 2: Left: Step response to maximum velocity (measured on carpet), approximating PT_1 The measurement error at $t = 1200$ is due to odometry: We measure the ticks per second at the motor, not the wheels; errors occur if the transmission belt slips. Right: Speed of left and right wheels as a function of incoming PWM signals, measured with no-load. In comparison, the robot's speed when driving on carpet differs approximately by a constant offset.

2. HIGH SPEED ROBOT CONTROL

Localizing a robot that drives with high speed (e.g., 4 m/s) does not only call for a very fast tracking algorithm to keep being localized. It also requires a vehicle that is able to drive safely with such velocities in an indoor environment, including the ability to react in real time to sudden changes of the surrounding, like opening doors or people walking around. The robot control implemented on Kurt3D consists of two parts, namely a fast motor controller and a set of behaviors for set value computation.

2.1. Motor Controller

Designing a control program for autonomous mobile robots consists of implementing a controller or set of behaviors including a set value computation for the controller. Hereby, the goal is to adjust the input signal to external disturbances, e.g., fluctuations of battery voltage and friction. Appropriate set values, e.g., the robot speed or turning velocity, are mapped by a controller or set of behaviors to the actors, i.e., its motors. The control cycle of Kurt3D's motors runs with 100 Hz. In order to model the physical characteristics, different step responses and the speed of the motors with no-load and on carpet have been measured. Fig. 2 (left) shows

the step response to maximum velocity, following the characteristics of a PT_1 element. The right plot presents the speed of the two motors when operating with no-load and on carpet moving straight forward. Every discrete PWM value from 0 up to 1023 is given to the system for one second, and the reaction, i.e., the speed at the end of the second, is measured. The system shows non-linearities in the upper speed range. The no-load measurement and the one on carpet differ only by a constant offset. The controller works under the assumption that this offset stays constant throughout the complete velocity interval, since it was not possible to measure the whole speed range on carpet due to physical limitations of the testing environment (the longest corridor was too short). The inverse of the lookup table T (Fig. 2, right) together with the offset build the base for the *open loop* part of the controller. A given speed and steering angle are scaled by two feed forward terms F_v and F_ω (see Fig. 1, right) [7] and mapped to a PWM value according to T. F_v is set to 1 according to the design of the lookup table and offset. An additional feed forward term is needed for the angular velocity ($F_\omega = \pi^{-1}$ cm) according to the significant friction during turning.

The fast reaction time of the overall system and the reliable *closed loop* motor control is formed by adding a PI-term. By design, PI controllers generate a correction term iff a control deviation exists. The P-term decreases the reaction time until reaching a modified set value. The I-term avoids steady state errors but leads to a small overshoot. However, the combination of feed forward terms with the lookup table shifts the motor signals near the working point so that the PI controller only controls small disturbances, e.g., different battery states and robot loads. Therefore, a D-term in the controller part is neither necessary nor desirable, since derivative terms do have difficulties with noise in the measurement, compare Fig. 2 (left).

Finally, the motors receive bandpass filtered PWM signals as inputs. Fig. 3 right shows the overall system, containing the motor controllers for the left and right wheels as well as the state transformation formulas.
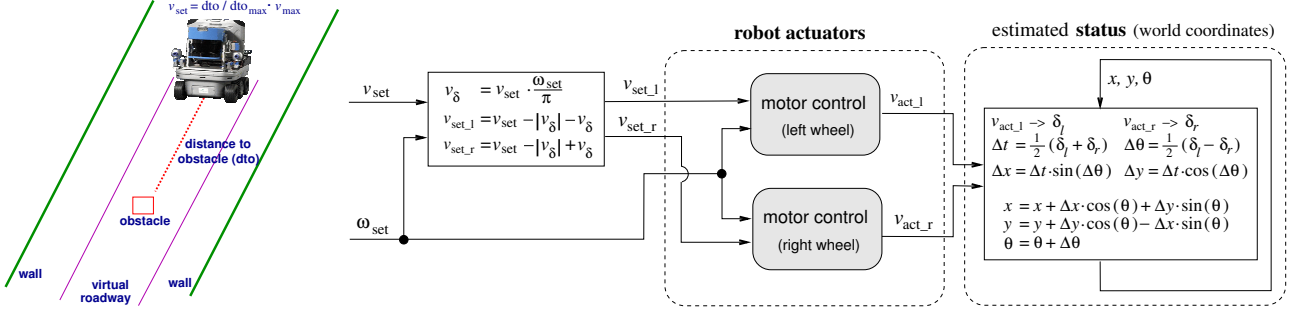
Figure 3: Left: For safe navigation, the reference velocity v_{set} is computed as a function of the distance to an obstacle lying within a virtual roadway. Right: Schematic overview of the robot's control system, containing two motor controllers (see Fig. 1) for the left and right wheel.

2.2. Set Value Computation

We have implemented three alternatives for computing the set values for Kurt3D. Depending on the application, teleoperated control by a human operator, fuzzy control for autonomous high-speed driving or globally stable robot control for driving precisely to specified coordinates is used.

2.2.1. Teleoperation

For teleoperation, a joystick is used. The joystick signals are directly mapped to the reference velocity v_{set} and the reference turning velocity ω_{set}. To our experience, it is difficult for human operators to control Kurt3D manually beyond a speed of 1 m/s.

2.2.2. Fuzzy Control

To overcome the problems with a human operator and to enable the robot to drive full speed, a fuzzy controller is implemented that steers the robot autonomously into free space, yielding an autonomous "wander around" behaviour with obstacle avoidance. In addition, the driving direction is relatively stable and the robot's trajectory does not oscillate. 181 instances of a fuzzy rule with the following structure (Eq. (1)) are used to calculate the driving direction. These fuzzy rules operate directly on the 2D laser range data of the scanner: The i-th rule is applied to the i-th measurement, i.e., there is exactly one rule per measured scan value:

```
IF (angle_i is in driving direction) AND
   (distance_i is large)                        (1)
THEN drive in this direction.
```

The fuzzy AND is implemented as multiplication, the steering angle α results from the addition of all i computed direction vectors. In detail, given a set of measurements $\{(\varphi_i, r_i)\}_{i=1,\dots,181}$, the angle α is calculated by

$$\alpha = \mathrm{atan2} \left(\begin{array}{c} \sum\limits_{i=1}^{181} \sin(\varphi_i) \cdot f_1(\varphi_i) \cdot f_2(r_i) \, , \\ \sum\limits_{i=1}^{181} \cos(\varphi_i) \cdot f_1(\varphi_i) \cdot f_2(r_i) \end{array} \right). \quad (2)$$
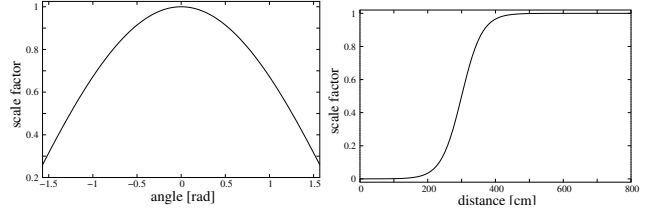

Figure 4: Set value generation with fuzzy rules, see Eq. (1), (2). Left: Function f_1 weighs the orientation of the data ("is in driving direction"). Right: f_2 implements a weighting on the measured distance ("is large").

Fig. 4 shows the used functions f_1, f_2, implementing a weighting on orientation resp. distance according to the fuzzy rules (1). The turning velocity ω_{set} is directly proportional to the heading α.

In order to drive safely with high speed, we apply the following algorithm to set Kurt3D's velocity: A virtual roadway is defined according to the robot's width (Fig. 3, left). If there is no obstacle on this roadway in front of the robot within a certain distance dto_{max}, the reference velocity v_{set} is set to a maximal velocity v_{max}. Otherwise, the speed is scaled down to $v_{set} = dto/dto_{max} \cdot v_{max}$, with the measured distance to the obstacle dto. In this way, the fuzzy control navigates safely around obstacles. Nevertheless, if dto falls below a fixed value dto_{min}, a different behavior is evoked, i.e., v_{set} is set to 0, ω_{set} to a constant. This results in a rotation of the vehicle until the virtual roadway is free again. For the set value computation we use the following constants: $dto_{min} = 50$ cm, $dto_{max} = 600$ cm and v_{max} of 4 m/s.

2.2.3. Coordinate Based Robot Control

The coordinate based robot control is used for approaching discrete coordinates, e.g., computed by a next best view planner that is utilized for the autonomous generation of consistent 3D environment models [10]. In this mode, the non holonomic robot Kurt3D is controlled by a closed loop,

time invariant and globally stable motor controller, developed by G. Indiveri [2]. The target configuration is always approached on a straight line and the vehicle is requested to move in only one specified forward direction, thus avoiding cusps in the paths and satisfying a major requirement for the implementation of such strategy on many real systems. Following the notation in [10], let (x_G, y_G, φ) be the robot pose in the target centered coordinate system. The controller is based on a Cartesian kinematic model described by:

$$\dot{x}_G = v_{\text{set}} \cdot \cos \varphi \quad \dot{y}_G = v_{\text{set}} \cdot \sin \varphi \quad \dot{\varphi} = \omega_{\text{set}} = v_{\text{set}} \cdot c.$$

Thereby v_{set} is the robot's linear velocity, ω_{set} the angular velocity and c the (bounded) curvature. $(0, 0, 0)$ is the final position. The transformation of the Cartesian coordinates into polar like coordinates results in

$$
\begin{aligned}
e &= \sqrt{x_G{}^2 + y_G{}^2} & \dot{e} &= -v_{\text{set}} \cdot \cos \alpha \\
\theta &= \text{atan2}(-y_G, -x_G) \quad \Rightarrow \quad & \dot{\alpha} &= v_{\text{set}} \left(c - \frac{\sin \alpha}{e} \right) \\
\alpha &= \theta - \varphi & \dot{\varphi} &= v_{\text{set}} \cdot \frac{\sin \alpha}{e}.
\end{aligned}
$$

G. Indiveri uses for the robot speed the equation $v_{\text{set}} = \gamma \cdot e$ with $\gamma > 0$ and a Lyapunov-like based control law synthesis to derive the following formula for the curvature:

$$c = \frac{\sin \alpha}{e} + h \frac{\theta}{e} \cdot \frac{\sin \alpha}{\alpha} + \beta \frac{\alpha}{e},$$

with $h > 1$ and $2 < \beta < h + 1$ [2]. These two formulas for the velocity and curvature (or angular velocity) form the closed loop, time-invariant and globally stable motor controller with constant time complexity.

2.3. The Robot Control Architecture

Fig. 5 shows an overview of the complete system, including the previously presented set value units and the localization algorithm HAYAI described in the next section. The motor control loop runs with 100 Hz on a Linux laptop. The generated PWM signals are set via an Infineon C167 microcontroller connected with a CAN bus and a Microcontrol PCMCIA CAN card. The set value computation runs inside this loop, iff new input signals, i.e., sensor data, are present. The fuzzy control generates new values with 75 Hz. The fast localization algorithm HAYAI is executed in the control loop, restricting the time for scan matching to 10 ms. The following section describes this fast computation.

3. POSE TRACKING WITH HAYAI

This section describes the newly developed algorithm HAYAI (*Highspeed And Yet Accurate Indoor/outdoor-tracking*), originally published in [6], whose main points we recapitulate here to make this paper self-sufficient. The matching algorithm is based on the following scheme:

1. Detect features within scan $\mathcal{R}$, yielding a feature set M (*model set*). Likewise compute a set D (*data set*) from a previous scan $\mathcal{S}$.
2. Search for pairwise corresponding features from both sets, resulting in two subsets $\check{M} \subseteq M$ and $\check{D} \subseteq D$.
3. Compute the pose shift $\Delta \boldsymbol{p} = (\Delta x, \Delta y, \Delta \theta)^T$ as the optimal transformation for mapping $\check{D}$ onto $\check{M}$.
4. Update the robot's pose $\boldsymbol{p}_n \xrightarrow[\Delta \boldsymbol{p}]{} \boldsymbol{p}_{n+1}$ according to formula (3).
5. Save the current scan as new reference scan $\mathcal{R} \leftarrow \mathcal{S}$.

Given a pose $\boldsymbol{p}_n = (x_n, y_n, \theta_n)$ and a transformation $\Delta \boldsymbol{p} = (\Delta x, \Delta y, \Delta \theta)$, the transition $\boldsymbol{p}_n \xrightarrow[\Delta \boldsymbol{p}]{} \boldsymbol{p}_{n+1}$ is calculated as follows:

$$
\begin{pmatrix} x_{n+1} \\ y_{n+1} \\ \theta_{n+1} \end{pmatrix} = \begin{pmatrix} x_n \\ y_n \\ \theta_n \end{pmatrix} + \begin{pmatrix} \cos \theta_n & \sin \theta_n & 0 \\ -\sin \theta_n & \cos \theta_n & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \Delta x \\ \Delta y \\ \Delta \theta \end{pmatrix} \quad (3)
$$

3.1. Extraction and Matching of Features

As described above, the scan matching algorithm computes a transformation $\Delta \boldsymbol{p}$ such that a *set of features*, extracted from the first scan, is mapped optimally to a feature set of the second scan. In order to be usable for a pose tracking algorithm, these features have to fulfill two requirements: First, they have to be *invariant* with respect to rotation and translation. Second, they have to be *efficiently* computable in order to satisfy real time constraints.

Using the inherent order of the scan data allows the application of linear filters for a fast and reliable feature detection. HAYAI chooses *extrema* in the polar representation of a scan as natural landmarks. These extrema correlate to corners and jump edges in Cartesian space. The usage of polar coordinates implicates a reduction by one dimension, since all operations deployed for feature extraction are fast linear filters, operating on the sequence of range values $(r_i)_{i \in \mathbb{N}}$ of a scan $\mathcal{S} = \big((\varphi_i, r_i)\big)_{i=1,\ldots,N}$.

Given a one dimensional filter $\Psi = [\psi_{-1}, \psi_0, \psi_{+1}]$, the filtered value r_i^Ψ of a scan point r_i ($i = 2, \ldots, N - 1$) is defined as $r_i^\Psi = \sum_{k=-1}^{1} \psi_k r_{i+k}$. For feature detection, the scan signal is filtered by a sharpen filter ($\Psi_1 = [-1, 4, -1]$), the gradient signal is computed ($\Psi_2 = [-\frac{1}{2}, 0, \frac{1}{2}]$) and softened softened ($\Psi_3 = [1, 1, 1]$). Details can be found in [6], Fig. 6 (left) illustrates the effects of these filters.

After generating the sets of features M, D from both scans, a match between both sets has to be calculated. Instead of solving the hard optimization problem of searching for an optimal match, we use a heuristic approach, utilizing inherent knowledge about the problem of matching features, e.g., the fact that the features' topology cannot change fundamentally from one scan to the following. The basic aim is to build a matrix of possible matching pairs, based on an error function defining the distance between two points
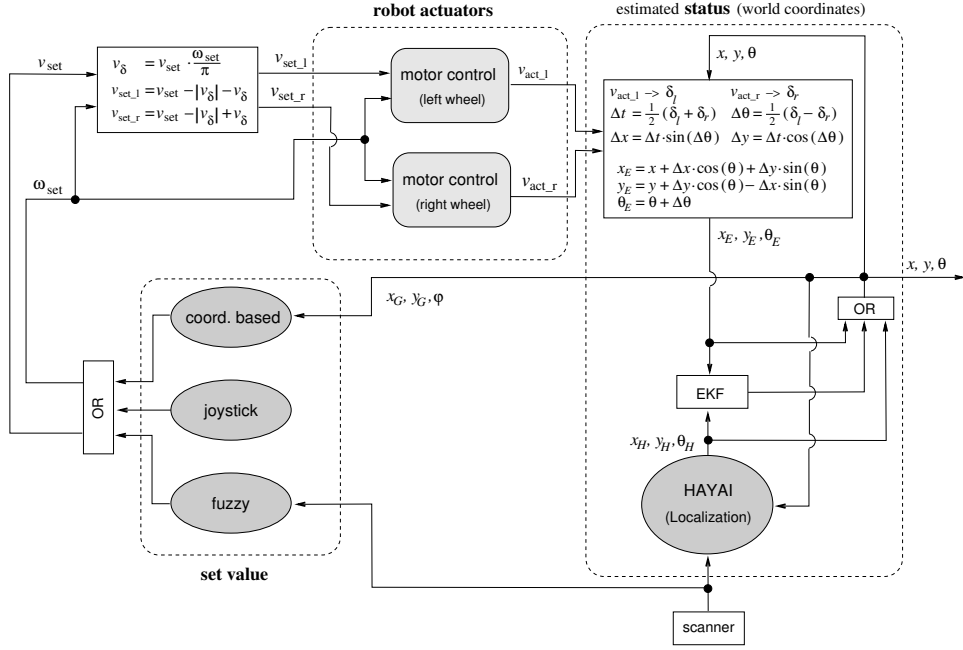
Figure 5: Schematic overview of the overall control system. Extension of Fig. 3 by different set value units and HAYAI as localization method for tracking the vehicle's pose, optionally fused with the robot's dead reckoning data with an EKF [6].
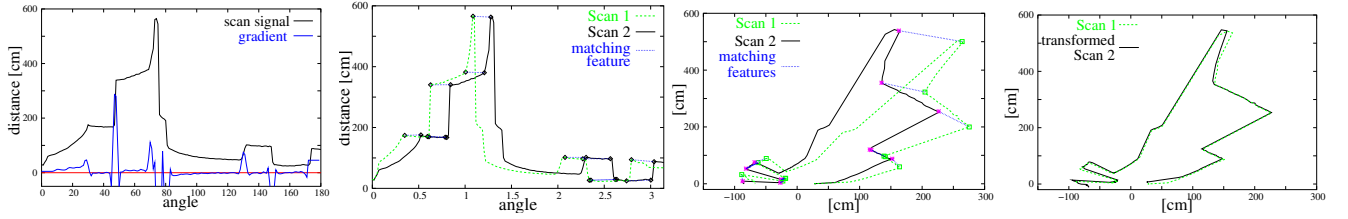

Figure 6: From left to right: (1) Application of the feature detection filters. (2) and (3) Pairing of corresponding features. (φ, r) representation (2) vs. the euclidian one (3). (4) Transformed scan.

$\boldsymbol{m}_i, \boldsymbol{d}_j$, with $\boldsymbol{m}_i = (m_i^x, m_i^y)^T$ in Cartesian, or $(m_i^\varphi, m_i^r)^T$ in polar coordinates, resp. ($\boldsymbol{d}_j$ analogously):

$$\text{dist}(\boldsymbol{m}_i, \boldsymbol{d}_j) = \sqrt{\left(\omega_1 \cdot (m_i^\varphi - d_j^\varphi)\right)^2 + \omega_2 (m_i^r - d_j^r)^2} \\ + \omega_3 \cdot \sqrt{(m_i^x - d_j^x)^2 + (m_i^y - d_j^y)^2} \\ + \Theta(\boldsymbol{m}_i, \boldsymbol{d}_j) \qquad (4)$$

with constants $(\omega_k)_{k \in \{1,2,3\}}$, implementing a weighting between the polar and Cartesian distances. The function Θ inhibits matchings between two features of different types:

$$\Theta(\boldsymbol{m}_i, \boldsymbol{d}_j) = \begin{cases} 0 & \Gamma(\boldsymbol{m}_i) = \Gamma(\boldsymbol{d}_j) \\ \infty & \text{else} \end{cases}$$

with a classification function $\Gamma: (M \cup D) \mapsto \{\text{max.,} \text{min., inflection point}\}$. The resulting matrix $w_{i,j}$, denoting feature correspondences, is simplified until the match is unique. Fig. 6 shows the match of two scans.

3.2. Pose Calculation

Given two sets of features $\check{M} = \{\boldsymbol{m}_i \mid \boldsymbol{m}_i \in \mathbb{R}^2, i = 1, \ldots, N_m\}$ and $\check{D} = \{\boldsymbol{d}_i \mid \boldsymbol{d}_i \in \mathbb{R}^2, i = 1, \ldots, N_d\}$, the calculation of the optimal transformation for mapping D onto M is an optimization problem of the error function:

$$E(\boldsymbol{R}, \boldsymbol{t}) = \sum_{i=1}^{N_m} \sum_{j=1}^{N_d} w_{i,j} \, ||\boldsymbol{m}_i - (\boldsymbol{R}\boldsymbol{d}_j + \boldsymbol{t})||^2,$$

$$\propto \frac{1}{N} \sum_{i=1}^{N} ||\boldsymbol{m}_i - (\boldsymbol{R}\boldsymbol{d}_i + \boldsymbol{t})||^2$$

since the match is unique. In [6] we showed that the optimal rotational angle $\Delta\theta$ and translation $\Delta\boldsymbol{t}$ are computed, given $\boldsymbol{m}'_i = \boldsymbol{m}_i - \underbrace{\frac{1}{N} \sum_{i=1}^{N} \boldsymbol{m}_i}_{\boldsymbol{c}_m}, \; \boldsymbol{d}'_i = \boldsymbol{d}_i - \underbrace{\frac{1}{N} \sum_{i=1}^{N} \boldsymbol{d}_i}_{\boldsymbol{c}_d}$, as

$$\Delta\theta = \arctan\left(\frac{\sum\limits_{i=1}^{N}(m'^x_i d'^x_i + m'^y_i d'^y_i)}{\sum\limits_{i=1}^{N}(m'^y_i d'^x_i - m'^x_i d'^y_i)}\right), \qquad (5)$$

$$\Delta t = c_m - \begin{pmatrix} \cos \Delta\theta & \sin \Delta\theta \\ -\sin \Delta\theta & \cos \Delta\theta \end{pmatrix} \cdot c_d . \qquad (6)$$

4. EXPERIMENTS AND RESULTS

Please refer to `http://kos.informatik.uos.de/download/highspeed/` for videos showing the experiments and results. Kurt3D driving with 1.5 m/s up to 4 m/s as well as localization results can be found. Furthermore, safety tests with suddenly changing environments are presented: When the robot is driving full speed and a dynamic obstacle suddenly appears within the virtual roadway, Kurt3D stops safely and, if necessary, bypasses it if the distance to the obstacle is at least 1 m.

A number of unavoidable problems occur when driving high-speed: Vibrations, for example, lead in particular to erroneous sensor readings – check out the URL mentioned above to see a drive of the robot, viewed from a camera mounted on top of the laser scanner, to get an impression of the scanner's movement. Moreover, the robot itself is bound to skid when driving and turning on a smooth surface.

On the other hand, even driving slowly benefits from this high speed architecture, since more reliability, stability and availability are achieved. Driving with high speed made it necessary to implement a fast motor controller that reaches the set values with very little delay. As a result, algorithms that are depending on these values (i.e., the coordinate based control, section 2.2.3) can operate at a higher speed, too, resulting in a more exact execution of tasks like path following, independent of the actual speed of the robot.

We also used the presented control scheme for a slower, off-road version of Kurt3D in the RoboCup Rescue 2004 competition in Lisbon. Results and videos of the rescue robot are available at `http://kos.informatik.uos.de/download/Lisbon_RR/`. Following the racing car analogy mentioned in the introduction, extensive tests with the high speed robot have led to several improvements of the hardware, which are now transfered to the standard KURT2 platform.

Future work will concentrate on how the control of the robot has to change with higher speeds: while the dynamics of a slow moving vehicle can be neglected, the described scenario leaves no choice than to think about stopping distance, maximal velocity and minimal turning angle when driving a curved trajectory, etc.

5. SUMMARY AND OUTLOOK

This paper has presented the control architecture of the high speed mobile robot Kurt3D. The ability to drive reliably and safely with velocities considerably faster than walking speed, e.g. 4 m/s, in an unknown office environment has been presented using standard hardware. The robot is controlled with a feed forward PI controller combined with a lookup table for mapping speed to PWM values and for the linearization of the I/O behavior. Different forms of set value computations, e.g., a reactive fuzzy control and a coordinate based one, used for deliberative path planning, have been discussed. HAYAI, a fast robot localization, fits into this simple software architecture, since only a few calculations are necessary to match 2D laser scans reliably.

6. REFERENCES

[1] R. Alami, T. Simeon, and K. M. Krisha. On the influence of sensor capacities and environment dynamics onto collision-free motion plans. In *Proc. IEEE/RSJ IROS*, 2002.

[2] G. Indiveri. Kinematic Time-invariant Control of a 2D Nonholonomic Vehicle. In *Proc. of the 38th Conf. on Decision and Control*, USA, 1999.

[3] K. M. Krisha, R. Alami, and T. Simeon. Moving Safely but not Slowly – Reactively Adapting Paths for Better Trajectory Times. In *Proc. IEEE ICAR*, 2003.

[4] KTO / Fraunhofer AIS. The KURT2 robot platform, `http://www.ais.fraunhofer.de/kurt2`.

[5] A. Lankenau and T. Röfer. Mobile Robot Self-Localization in Large-Scale Environments. In *Proc. IEEE ICRA*, San Francisco, CA, USA, 2002.

[6] K. Lingemann, H. Surmann, A. Nüchter, and J. Hertzberg. Indoor and Outdoor Localization for Fast Mobile Robots. In *Proc. IEEE/RSJ IROS*, 2004.

[7] K. Ogata. *Modern Control Engineering (4th Edition)*. Prentice Hall, November 2001.

[8] E. Santiso, M. Mazo, J. Ureña, J. A. Jiménez, J. J. García, and M. C. Serra. Mobile Robot Positioning with Natural Landmark. In *Proc. IEEE ICAR*, 2003.

[9] D. Sekimori, T. Usui, Y. Masutani, and F. Miyazaki. High-speed Obstacle Avoidance and Self-localization for Mobile Robot Based on Omni-directional Imaging of Floor Region. *Proc. IPSJ Trans. CV and IM*, 2001.

[10] H. Surmann, A. Nüchter, and J. Hertzberg. An autonomous mobile robot with a 3D laser range finder for 3D exploration and digitalization of indoor environments. *J. Robotics and Auton. Syst.*, 2003.

[11] S. Thrun, D. Fox, W. Burgard, and F. Dellaert. Robust Monte Carlo Localization for Mobile Robots. *Artificial Intelligence*, 128(1–3):99–441, 2001.

[12] Volksbot. `http://www.ais.fraunhofer.de/BE/volksbot/detail.html`, 2005.

[13] T. Weigel, J.-S. Gutmann, M. Dietl, A. Kleiner, and B. Nebel. CS-Freiburg: Coordinating Robots for Successful Soccer Playing. *J. TRA*, 18(5):685–699, 2002.