Kurt3D
A Mobile Robot for 3D Mapping of Environments

Andreas Nüchter, Kai Lingemann, Joachim Hertzberg
Fachbereich Mathematik/Informatik
Institut für Informatik
Knowledge-Based Systems Research Group
Universität Osnabrück
http://www.informatik.uni-osnabrueck.de/kbs/

January 24, 2006

Abstract

The mobile robot Kurt3D is the first robot that is capable of mapping its environment in 3D and self localize in all six degrees of freedom, i.e., considering its x, y and z positions and the roll, yaw and pitch angles. Robot motion on natural surfaces has to cope with yaw, pitch and roll angles, turning pose estimation into a problem in six mathematical dimensions. Kurt3D creates a consistent volumetric scene in a common coordinate frame from multiple 3D laser scans. To create 3D volumetric maps it is necessary to gage several 3D scans and register them into one consistent 3D model. A fast variant of the Iterative Closest Points (ICP) algorithm is used for registration and relocalization.

1 Introduction

The mobile robot Kurt3D is developed at the University of Osnabrück. The development began at the Fraunhofer Institute for Autonomous Intelligent Systems (AIS) and is continued in Osnabrück. The key innovation of this highly integrated, compact system lies in the capability for autonomous or operator-assisted 6D SLAM (Simultaneous Localization and Mapping) and 3D map generation of natural scenes. 6D SLAM with mobile robots considers six dimensions for the robot pose, namely the x, y and z coordinates and the roll, yaw and pitch angles. Robot motion and localization on natural surfaces, e.g., driving with a mobile robot outdoor, must necessarily regard these degrees of freedom.

The precise planar pose tracking algorithm HAYAI is used to localize the robot while driving. Using 2D laser scans, features that correspond to natural landmarks are extracted and paired with features of previous scans. A fast and reliable 3D scan matching procedure, employing a sophisticated point reduction and approximate nearest neighbor search, is used to generate 3D maps and relocalize the robot. The basis of the 3D scan matching is the well know iterative closest points (ICP) algorithm [3]. An interactive, semi-automatic control program enables the operator to interfere with the mapping process for corrections, if necessary.

2 The Exploration Robot Kurt3D

Kurt3D (Fig. 1) is a mobile robot platform with a size of 45 cm (length) × 33 cm (width) × 26 cm (height) and a weight of 15.6 kg. Indoor as well as outdoor models exist. Equipped with the 3D laser range finder, the height increases to 47 cm and the weight increases to 22.6 kg. Two 90 W motors (short-term 200 W) are used to power the 6 wheels. Compared to the original Kurt3D robot platform, the outdoor version has larger wheels, whith the

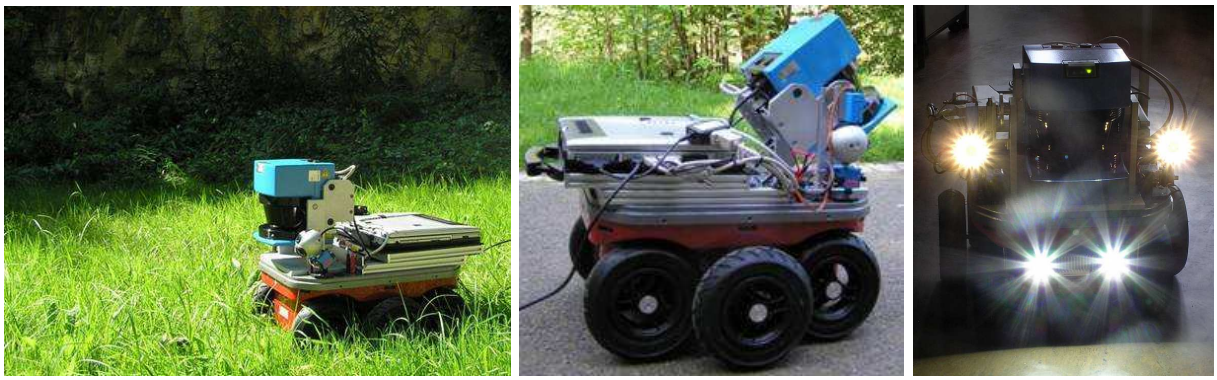

Figure 1: The mobile robot Kurt3D equipped with the 3D laser range finder. The 3D scanner's technical basis is a SICK 2D laser range finder (LMS-200). The robot is completely independent from external light.

middle ones shifted outwards. Front and rear wheels have no tread pattern to enhance rotating. Kurt3D operates for about 4 hours with one battery charge (28 NiMH cells, capacity: 4500 mAh). The core of the robot is an Intel-Centrino-1400 MHz with 768 MB RAM and a Linux operating system. An embedded 16-Bit CMOS microcontroller is used to process commands to the motor. A CAN interface connects the laptop with the microcontroller.

The 3D laser range finder (Fig. 1) is built on basis of a 2D range finder by extension of a mount and a standard servo motor [7]. The 2D laser range finder is attached to the mount in the center of rotation for achieving a controlled pitch motion. The servo is connected on the left side (Fig. 1). The 3D laser scanner operates for up to 5h (Scanner: 17 W, 20 NiMH cells with a capacity of 4500 mAh, Servo: 0.85 W, 4.8 V with batteries of 4500 mAh) on one battery pack. The area of 180°(h) × 90°(max. 120°)(v) is scanned with different horizontal (181, 361, 721) and vertical (128, 176, 256, 400, 500) resolutions. A plane with 181 data points is scanned in 13 ms by the 2D laser range finder (rotating mirror device). Planes with more data points, e.g., 361, 721, duplicate or quadruplicate this time. Thus a scan with 361 × 176 data points needs 4.5 seconds. In addition to the distance measurement the 3D laser range finder is capable of quantifying the amount of light returning to the scanner, resulting in a black/white reflectance image of the scene. 3D Scanning the environment with a mobile robot is done in a stop-scan-go fashion.

Kurt3D is equipped with 2 × 10 super bright LEDs and two diffuse luminescent LEDs to illuminate the surroundings. The LEDs are attached to the two pan-and-tilt Logitech QuickCam 4000 cameras.

3 The Software Architecture

Fig. 2 sketches the software architecture. It is a client-server architecture where client and server are connected by wireless LAN. On the robot's side a 100 Hz control loop is running, adjusting the motor velocities. In this loop, odometry and HAYAI (cf. subsection 4.1) are processed and merged with a Kalman filter. The processing time of HAYAI is around 3% of the CPU load. Thus there is enough compute power left to compress the camera images to jpeg and transmit the data.

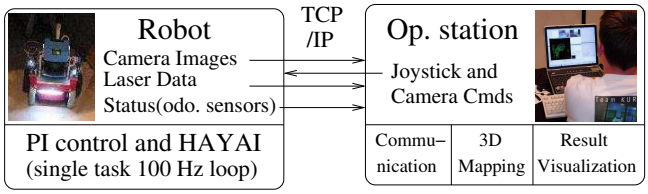

Figure 2: System Overview. The robot runs a 100 Hz loop for motor control, scan processing, image encoding and transmission. The Kurt3D server executes three threads that are responsible for communicating, 3D mapping and result visualization.

At the operator station the robot is teleoperated. Three processes are executed in parallel: The communication between remote joystick control and robot, the 3D mapping, and the interactive, OpenGL-based map viewer. The latter enables the operator to intervene in

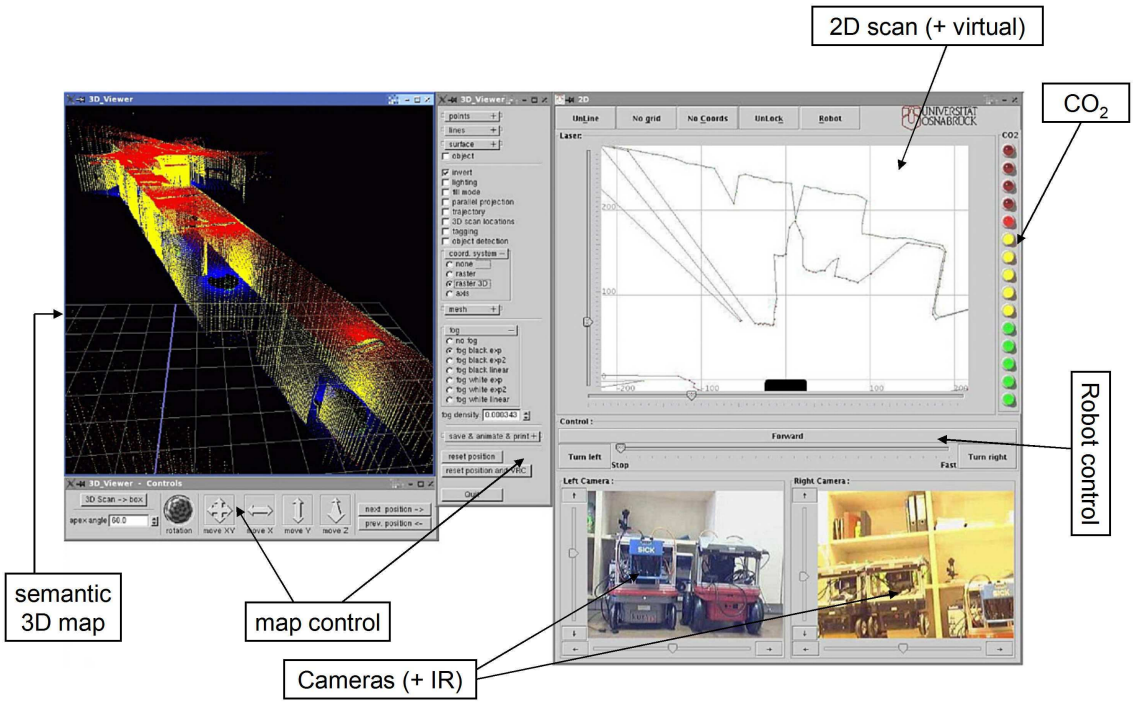

Figure 3: User interface for robot control. Left: 3D data viewer. Right: Robot control.

the map generation process, e.g., manually correcting the initial 6D pose of an acquired 3D scan and restarting the matching algorithm. During competition, the operator tries to minimize the number of acquired 3D scans to save time for victim detection. The virtual camera pose in the 3D mapping window is freely adjustable, though a view from top is used by default, since it provides the best situation awareness. Fig. 3 shows the user interface for Kurt3D.

4 Sensing and Data Processing

Kurt3D's main sensor is the 3D laser range finder that is used in a stop-scan-go fashion. However, while driving, the scanner is fixed in horizontal position and used for pose tracking.

4.1 Pose Tracking with HAYAI

This subsection describes the algorithm HAYAI (*Highspeed And Yet Accurate Indoor/outdoor-tracking*). It is based on the following scheme:

1. Detect features within scan $\mathcal{R}$, yielding feature set M (*model set*). Likewise compute 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 $\boldsymbol{p}_n \underset{\Delta\boldsymbol{p}}{\longrightarrow} \boldsymbol{p}_{n+1}$ according to formula (1).

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 $\boldsymbol{p}_n \underset{\Delta\boldsymbol{p}}{\longrightarrow} \boldsymbol{p}_{n+1}$ is calculated as follows:

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

4.1.1 Data Filtering

Scanning is noisy and small errors may occur, namely Gaussian noise and salt and pepper noise. The latter would arise, for example, at edges where the scanner's laser beam hits two surfaces, resulting in a mean and erroneous data value. Furthermore reflections, e.g., at glass surfaces, lead to suspicious data. We propose two fast filtering methods to modify the data in order to enhance the quality of each scan, typically containing 181 data points.

The data reduction, used for reducing Gaussian noise, works as follows: The scanner emits the laser beams in a spherical way, such that the data points close to the source are more dense. Multiple data points located close together are joined into one point. The number of these so-called *reduced points* is one order of magnitude smaller than the original one.

For eliminating salt and pepper noise, a median filter removes the outliers by replacing a data point with the median value of the n surrounding points (here: $n = 7$). The neighbor points are determined according to their index within the scan, since the laser scanner provides the data sorted in a counter-clockwise direction. The median value is calculated with regard to the Euclidian distance of the data points to the point of origin. In order to remove noisy data but leave the remaining scan points untouched, the filtering algorithm replaces a data point with the corresponding median value if and only if the Euclidian distance between both is larger than a fixed threshold (e.g., 200 cm).

4.1.2 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

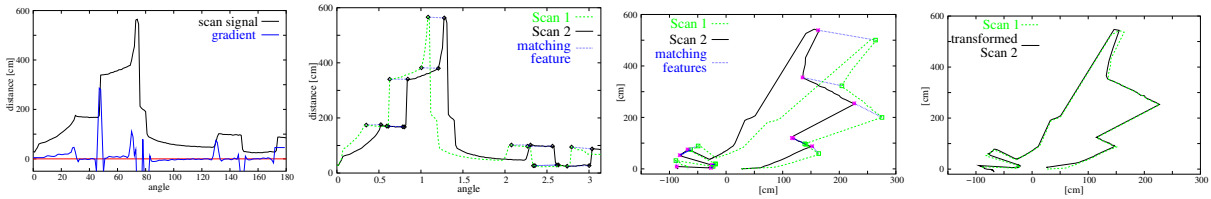

Figure 4: From left to right: (1) Application of the feature detection filters. (2) and (3) Pairing of corresponding features. (φ, r) representation (2) vs. the euclidian one (3). (4) Transformed scan.

set of the second scan. 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 $\Psi = \big[\,\psi_{-1},\ \psi_0,\ \psi_{+1}\,\big]$, 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 as follows:

1. Sharpen the data in order to emphasize the significant parts of the scan, i.e., the extrema, without modifying the residual scan, by applying a sharpen filter of the form $\Psi_1 = \big[\,-1,\ 4,\ -1\,\big]$.
2. Compute of the derivation signal by using a gradient filter $\Psi_2 = \big[\,-\frac{1}{2},\ 0,\ \frac{1}{2}\,\big]$.
3. Smooth the gradient signal to simplify the detection of zero crossings with a soften filter $\Psi_3 = \big[\,1,\ 1,\ 1\,\big]$.

Fig. 4 (left) illustrates the effects of the used filters.

After generating the sets of features M, D from both scans, a matching 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 $\boldsymbol{m}_i, \boldsymbol{d}_j$, 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):

$$\begin{aligned}\operatorname{dist}\big(\boldsymbol{m}_i, \boldsymbol{d}_j\big) = &\sqrt{\big(\omega_1 \cdot (m_i^{\varphi} - d_j^{\varphi})\big)^2 + \omega_2 (m_i^r - d_j^r)^2} \\ &+ \omega_3 \cdot \sqrt{(m_i^x - d_j^x)^2 + (m_i^y - d_j^y)^2} \\ &+ \Theta\big(\boldsymbol{m}_i, \boldsymbol{d}_j\big) \end{aligned} \tag{2}$$

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:

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

with a classification function $\Gamma\colon (M \cup D) \mapsto \{\mathsf{max.}, \mathsf{min.}, \mathsf{inflection\ point}\}$. The resulting matrix $w_{i,j}$ denoting feature correspondences is simplified until the match is non-ambiguous. Fig. 4 shows the match of two scans.

4.1.3 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 $(w_{i,j})$ is an optimization problem of the error function:

$$E(\boldsymbol{R}, \boldsymbol{t}) = \sum_{i=1}^{N_m} \sum_{j=1}^{N_d} w_{i,j} \left|\left|\boldsymbol{m}_i - (\boldsymbol{R}\boldsymbol{d}_j + \boldsymbol{t})\right|\right|^2 \tag{3}$$

$$\propto \frac{1}{N} \sum_{i=1}^{N} \left|\left|\boldsymbol{m}_i - (\boldsymbol{R}\boldsymbol{d}_i + \boldsymbol{t})\right|\right|^2, \tag{4}$$

since the matching is non-ambiguous. The first step of the computation is to decouple the calculation of the rotation $\boldsymbol{R}$ from the translation $\boldsymbol{t}$ using the centroids of the points belonging to the matching.

$$\boldsymbol{c}_m = \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{m}_i, \qquad \boldsymbol{c}_d = \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{d}_i \tag{5}$$

and

$$M' = \{\boldsymbol{m}_i' = \boldsymbol{m}_i - \boldsymbol{c}_m\}_{1,\ldots,N}, \tag{6}$$

$$D' = \{\boldsymbol{d}_i' \ = \boldsymbol{d}_i \ - \boldsymbol{c}_d\}_{1,\ldots,N}. \tag{7}$$

After replacing (5), (6) and (7) in the error function (4), $E(\boldsymbol{R}, \boldsymbol{t})$ becomes:

$$E(\boldsymbol{R}, \boldsymbol{t}) \propto \frac{1}{N} \sum_{i=1}^{N} ||\boldsymbol{m}'_i - \boldsymbol{R}\boldsymbol{d}'_i - \underbrace{(\boldsymbol{t} - \boldsymbol{c}_m + \boldsymbol{R}\boldsymbol{c}_d)}_{=\tilde{\boldsymbol{t}}}||^2$$

$$= \frac{1}{N} \sum_{i=1}^{N} ||\boldsymbol{m}'_i - \boldsymbol{R}\boldsymbol{d}'_i||^2 + \frac{1}{N} \sum_{i=1}^{N} \left|\left|\tilde{\boldsymbol{t}}\right|\right|^2 \tag{8a}$$

$$- \frac{2}{N} \tilde{\boldsymbol{t}} \cdot \sum_{i=1}^{N} (\boldsymbol{m}'_i - \boldsymbol{R}\boldsymbol{d}'_i) \tag{8b}$$

In order to minimize the sum above, all terms have to be minimized. The third sum (8b) is zero, since all values refer to centroid. The second part in (8a) has its minimum for $\tilde{\boldsymbol{t}} = \boldsymbol{0}$ or $\boldsymbol{t} = \boldsymbol{c}_m - \boldsymbol{R}\boldsymbol{c}_d$. Therefore the algorithm has to minimize only the first term, and the corresponding error function is:

$$E(\boldsymbol{R}) \propto \sum_{i=1}^{N} ||\boldsymbol{m}'_i - \boldsymbol{R}\boldsymbol{d}'_i||^2 . \tag{9}$$

By solving the equation $\frac{\partial}{\partial \Delta\theta} E(\boldsymbol{R}_{\Delta\theta}) = 0$ for a 2D rotation $\boldsymbol{R}_{\Delta\theta} = \boldsymbol{R}$, the optimal rotation is calculated as

$$\Delta\theta = \arctan \left(\frac{\sum_{i=1}^{N} \left(m'^{y}_i d'^{x}_i - m'^{x}_i d'^{y}_i \right)}{\sum_{i=1}^{N} \left(m'^{x}_i d'^{x}_i + m'^{y}_i d'^{y}_i \right)} \right). \tag{10}$$

With given rotation, the translation is calculated from $\boldsymbol{t} - \boldsymbol{c}_m + \boldsymbol{R}\boldsymbol{c}_d$:

$$\underbrace{\begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix}}_{=\Delta \boldsymbol{t}} = \boldsymbol{c}_m - \underbrace{\begin{pmatrix} \cos\Delta\theta & \sin\Delta\theta \\ -\sin\Delta\theta & \cos\Delta\theta \end{pmatrix}}_{=\boldsymbol{R}_{\Delta\theta}} \cdot \boldsymbol{c}_d. \tag{11}$$

4.2 3D Mapping and 6D Robot Relocalization

Multiple 3D scans are necessary to digitalize environments without occlusions. To create a correct and consistent model, the scans have to be merged into one coordinate system. This process is called registration. If the localization of the robot with the scanner were precise, the registration could be done directly based on the robot pose. However, relative self localization is erroneous, even with HAYAI, so the geometric structure of overlapping 3D scans has to be considered for registration.

4.2.1 6D Registration of 3D Scans

The following method for registration of point sets is part of many publications, so only a brief summary is given here. The complete algorithm was invented in 1992 and can be

found, e.g., in [3]. The method is called *Iterative Closest Points (ICP) algorithm.* The procedure considers all six degrees of freedom, i.e., the roll, yaw and pitch orientation and the x, y, and z position of the robot

Given two independently acquired sets of 3D points, M ($|M| = N_m$) and D ($|D| = N_d$), which correspond to a single shape, we aim of finding the transformation consisting of a rotation $\boldsymbol{R}$ and a translation $\boldsymbol{t}$ which minimizes the cost function (3). Note: This time the vectors are in 3D space and $\boldsymbol{R}$ has to be an orthonormal 3×3 matrix. Now, $w_{i,j}$ is assigned 1 if the i-th point of M describes the same point in space as the j-th point of D. Otherwise $w_{i,j}$ is 0. Two things have to be calculated: First, the corresponding points, and second, the transformation ($\boldsymbol{R}$, $\boldsymbol{t}$) that minimize $E(\boldsymbol{R}, \boldsymbol{t})$ on the base of the corresponding points.

The ICP algorithm calculates iteratively the point correspondences. In each iteration step, the algorithm selects the closest points as correspondences and calculates the transformation $(\boldsymbol{R}, \boldsymbol{t})$ for minimizing equation (3). The assumption is that in the last iteration step the point correspondences are correct. In every iteration the optimal transformation ($\boldsymbol{R}$, $\boldsymbol{t}$) has to be computed. Like before, Eq. (3) can be reduced to Eq. (4). The difficulty of the minimization problem is to enforce the orthonormality of matrix $\boldsymbol{R}$. The following method, first published by Arun et al, is based on singular value decomposition (SVD) [1]. It is robust and easy to implement, thus we give a brief overview here:

The conversion of (3) to (4) holds in 3D space, too. The SVD algorithm computes the optimal rotation by $\boldsymbol{R} = \boldsymbol{V}\boldsymbol{U}^T$. Hereby the matrices $\boldsymbol{V}$ and $\boldsymbol{U}$ are derived by the singular value decomposition $\boldsymbol{H} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{V}^T$ of a correlation matrix $\boldsymbol{H}$. This (3×3) matrix $\boldsymbol{H}$ is given by

$$\boldsymbol{H} = \sum_{i=1}^{N} \boldsymbol{m}_i'^T \boldsymbol{d}_i' = \begin{pmatrix} S_{xx} & S_{xy} & S_{xz} \\ S_{yx} & S_{yy} & S_{yz} \\ S_{zx} & S_{zy} & S_{zz} \end{pmatrix}, \tag{12}$$

with $S_{xx} = \sum_{i=1}^{N} m_{ix}' d_{ix}'$, $S_{xy} = \sum_{i=1}^{N} m_{ix}' d_{iy}'$, ...

Since rotations are length preserving, i.e., $||\boldsymbol{R}\boldsymbol{d}_i'||^2 = ||\boldsymbol{d}_i'||^2$, the error function (9) is expanded to

$$E(\boldsymbol{R}, \boldsymbol{t}) = \sum_{i=1}^{N} ||\boldsymbol{m}_i'||^2 - 2 \sum_{i=1}^{N} \boldsymbol{m}_i' \cdot \boldsymbol{R}\boldsymbol{d}_i' + \sum_{i=1}^{N} ||\boldsymbol{d}_i'||^2 .$$

The rotation affects only the middle term, thus it is sufficient to maximize

$$\sum_{i=1}^{N} \boldsymbol{m}_i' \cdot \boldsymbol{R}\boldsymbol{d}_i' = \sum_{i=1}^{N} \boldsymbol{m}_i'^T \boldsymbol{R}\boldsymbol{d}_i'. \tag{13}$$

Using the trace of a matrix, (13) can be rewritten to obtain

$$\mathrm{tr}\left(\sum_{i=1}^{N} \boldsymbol{R}\boldsymbol{d}_i' \boldsymbol{m}_i'^T \right) = \mathrm{tr}\left(\boldsymbol{R}\boldsymbol{H}\right).$$

Hereby, matrix $\boldsymbol{H}$ has to be defined as in (12). Now we have to find the matrix $\boldsymbol{R}$ that maximizes $\mathrm{tr}(\boldsymbol{RH})$. Assume that the singular value decomposition of $\boldsymbol{H}$ is $\boldsymbol{H} = \boldsymbol{U\Lambda V}^T$, with $\boldsymbol{U}$ and $\boldsymbol{V}$ orthonormal 3×3 matrices and $\boldsymbol{\Lambda}$ a 3×3 diagonal matrix without negative elements. Suppose $\boldsymbol{R} = \boldsymbol{VU}^T$. Then, $\boldsymbol{R}$ is orthonormal and

$$\begin{aligned} \boldsymbol{RH} &= \boldsymbol{VU}^T\boldsymbol{U\Lambda V}^T \\ &= \boldsymbol{V\Lambda V}^T \end{aligned}$$

is a symmetric, positive definite matrix. Arun, Huang and Blostein provide a lemma to show that

$$\mathrm{tr}(\boldsymbol{RH}) \geq \mathrm{tr}(\boldsymbol{BRH}),$$

for any orthonormal matrix $\boldsymbol{B}$. Therefore the matrix $\boldsymbol{R}$ is optimal. Proving the lemma is straightforward, using the inequation of Chauchy-Schwarz [1]. The optimal translation is calculated as (cf. (8b) and (11)): $\boldsymbol{t} = \boldsymbol{c}_m - \boldsymbol{Rc}_d$.

4.2.2 ICP-based 6D SLAM

To digitalize environments, multiple 3D scans have to be registered. After registration, the scene has to be globally consistent. A straightforward method for aligning several 3D scans is *pairwise matching*, i.e., the new scan is registered against the scan with the largest overlapping areas. The latter one is determined in a preprocessing step. Alternatively, *incremental matching* could be used, i.e., the new scan is registered against a so-called *metascan*, which is the union of the previously acquired and registered scans. Each scan matching has a limited precision. Both methods accumulate the registration errors such that the registration of a large number of 3D scans leads to inconsistent scenes and to problems with the robot localization.

After matching multiple 3D scans, errors have accumulated normally. In consequence, any closed loops (the robot comes back to a position mapped earlier) will be inconsistent. Our 6D SLAM algorithm detects a closing loop by registering the last acquired 3D scan with earlier acquired scans [6]. If a registration is possible, the computed error is distributed over all 3D scans. A second step minimizes the global error. The registration of one scan is followed by registration of all neighboring scans, such that the error is minimized. In an iterative fashion a consistent model is produced. Details of the full algorithm can be found in [5, 8].

4.2.3 ICP Speedups

The computational requirements of ICP are reduced by two methods: First we reduce the 3D data, i.e., we compute point clouds that approximate the scanned 3D surface and contain only a small fraction of the 3D point cloud. Second is the fast approximation of the closest point with kD-trees for the ICP algorithm.

Data reduction for the ICP algorithm is done using the proposed filters of subsection 4.1.1. Without filtering, a few outliers may lead to multiple wrong point pairs during the 3D matching phase and results in an incorrect 3D scan alignment. Reduction and filtering are done in every single 2D scan slice while scanning, they are implemented as online algorithms and run in parallel to the 3D scan acquisition. In the end, the data for the scan matching are collected from every third scan slice. This fast vertical reduction yields a good surface description (cf. Fig. 5).

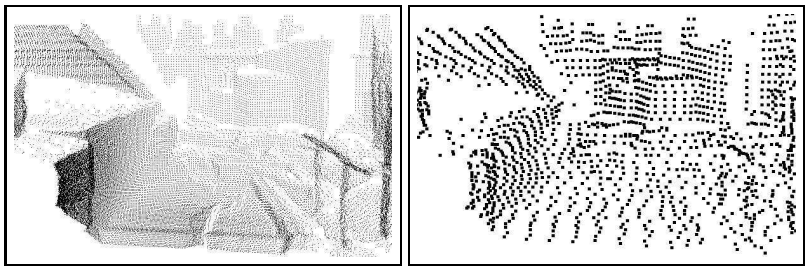

Figure 5: Left: A view of a 3D scene (66785 3D data points). Right: Subsampled version (points have been enlarged, 6700 data points).

kD-trees are a generalization of binary search trees. Every node represents a partition of a point set to the two successor nodes. The root represents the whole point cloud and the leafs form a disjunct partition of the set. These leafs are called buckets. Furthermore, every node contains the limits of the represented point set. Searching in kD-trees is done recursively. For a given 3D point $\boldsymbol{p}_q$, a comparison with the separating plane has to be performed in order to decide on which side the search must continue. This procedure is executed until the leafs are reached. There, the algorithm has to evaluate all bucket points. However, the closest point may be in a different bucket, iff the distance to the limits is smaller than the one to the closest point in the bucket. In this case backtracking has to be performed (Fig. 6, left).

Arya et al. introduce the following notion for approximating the nearest neighbor [2]: Given an $\varepsilon > 0$, then the point $\boldsymbol{p} \in D$ is the $(1+\varepsilon)$-approximate nearest neighbor of the point $\boldsymbol{p}_q$ iff $||\boldsymbol{p} - \boldsymbol{q}|| \leq (1+\varepsilon)\,||\boldsymbol{p}^* - \boldsymbol{q}||$, whereas $\boldsymbol{p}^*$ denote the true nearest neighbor, i.e., $\boldsymbol{p}$ is within a relative error of ε of the true nearest neighbor. In every step the algorithm records the closest point $\boldsymbol{p}$; the search terminates if the distance to the unanalyzed leafs is larger than $\left|\left|\boldsymbol{p}_q - \boldsymbol{p}\right|\right| / (1+\varepsilon)$. Fig. 6 (right) shows an example where the gray cell need not be analyzed, since the point $\boldsymbol{p}$ satisfies the approximation criterion. Fig. 6 shows the computation time for matching two 3D scans using kD-trees (top) and approximate kD-trees (bottom) with $\varepsilon = 50$ for different bucket sizes. In [5] we show that the quality of the scan matching is not affected by the approximation, due to the large number of points and the iterative fashion of the 3D scan matching.

4.3 Operator Assisted SPLAM

To build a map, the following three tasks must run in a permanent cycle:

1. Plan and goto the next scan pose.
2. Localize yourself in the current environment model and

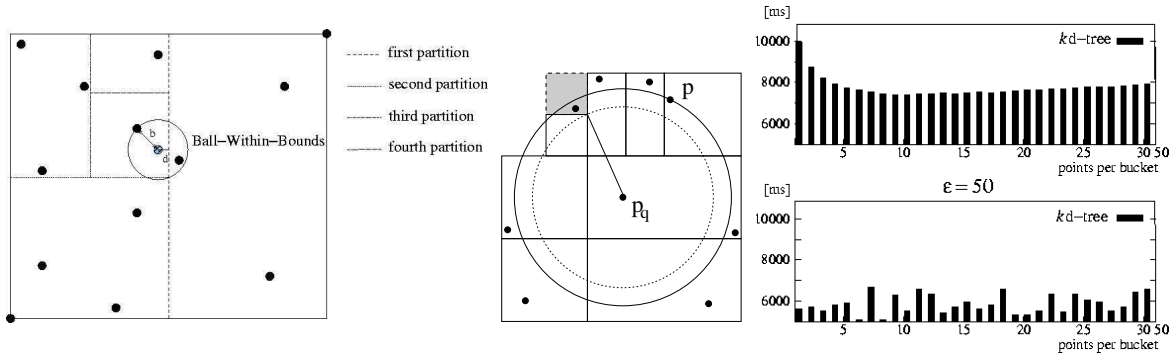

Figure 6: From left to right: (1) Construction of a kD-tree. (2) The $(1+\varepsilon)$-approximate nearest neighbor. The search algorithm doesn't have to analyze the gray cell, since the point $\boldsymbol{p}$ satisfies the approximation criterion. (3, top) Run time of the ICP algorithm using kD-trees with different bucket sizes. The minimal time is reached for 10 points per bucket. (3, bottom) Computing time for Approximate kD-tree search

3. Take a new scan and register it consistently with the previous model.

We call this overall process SPLAM (Simultaneous Planning, Localization and Mapping). The following simple procedure is based on the horizontal 2D laser scans, taken while the robot is driving. It is designed to help the operator plan the next 3D scan pose:

1. Transform the first 2D scan with the corresponding robot pose and create a polygon from the data by connecting neighboring points. Set it as reference scan.
2. Transform the next scan with the current robot pose and and build a polygon, too. Use a polygon clipping algorithm to to calculate the intersection of the polygons.
3. Calculate the area A_R of the reference polygon, the area A_C of current polygon and the area A_I of result of the intersection polygon. Suggest to the operator to acquire a new 3D scan, iff $\min(A_I/A_R, A_I/A_C)$ falls below a threshold (here 0.5).
4. If a new 3D scan is acquired, set the corresponding 2D scan as new reference scan.

Fig. 7 shows the polygon in a SPLAM process. Uncertainties of the robot pose are handled implicitly by the algorithms, i.e., as the robot pose gets inaccurate, the overlap of the scans shrinks and a 3D scan is acquired to correct the robot pose.

5 Vehicle Control

5.1 Extracting Drivable Surfaces

In addition to remote control, Kurt3D can drive autonomously. An autonomous 3D mapping robot for outdoor environments must be able to compute drivable surfaces. We

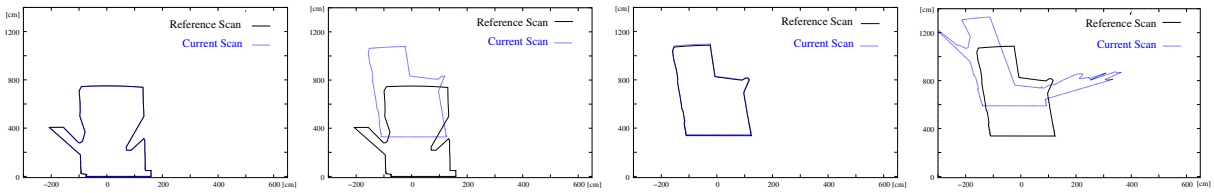

Figure 7: From left to right: (1) Reference 2D scan and current scan are identical. (2) When moving the robot the overlap of the scans reduce. (3) After acquiring an 3D scan the reference scan is updated. (4) The SPLAM process is restarted.

have adapted a classification system of Wulf et al. for calculating the ground [9]. They use a yawing scanning configuration; their basic idea of labeling 3D points as ground is to use the gradient between vertically neighboring points. A 3D point cloud that is scanned in a yawing scan configuration can be described as a set of points $\boldsymbol{p}_{i,j} = (\varphi_i, r_{i,j}, z_{i,j})^T$ given in a cylindrical coordinate system, with i the index of a vertical raw scan and j the point index within one vertical raw scan counting bottom up. The gradient $\alpha_{i,j}$ is calculated by the following equation:

$$\alpha_{i,j} = \arctan\left(\frac{z_{i,j} - z_{i,j-1}}{r_{i,j} - r_{i,j-1}}\right) \quad \text{with} \quad -\frac{1}{2}\pi \leq \alpha_{i,j} < \frac{3}{2}\pi.$$

The ground classification of point $\boldsymbol{p}_{i,j}$ is directly derived from the gradient $\alpha_{i,j}$: $\alpha_{i,j} < \tau$ (here: $\tau = 20°$). The algorithm has been adapted to our scanner configuration. Around a 3D point $\boldsymbol{p}_{i,j}$ a search window is used to extract the neighboring yawing 3D point by converting the $\boldsymbol{p}_{i,j}$ from $\mathbb{R}^3$ into cylindrical coordinates followed by an examination of the angles φ_i. A result of the ground segmentation is displayed in fig. 9. The classification of a scan point as "ground" based on its neighborhood instead of performing a simple height comparison is essential due to potential inaccuracy in the scanner calibration and unknown starting pose, namely the pitch angle of the robot, as well as the significant unevenness of the terrain, making this more sophisticated method necessary.

Figure 8: A rendered view from within Osnabrück's Botanical Garden.

5.2 Robot Control

Fig. 10 (top) shows the control architecture of the robot. It consist of a PI controller with feed forward term. A lookup table is used for linearization and conversion of velocities to PWM signals. Finally, the motors receive bandpass filtered PWM signals as inputs. Fig. 10 (bottom) right shows the overall system, containing the motor controllers for the left and right wheels as well as the state transformation formulas. (See [4] for a more detailed discussion.)

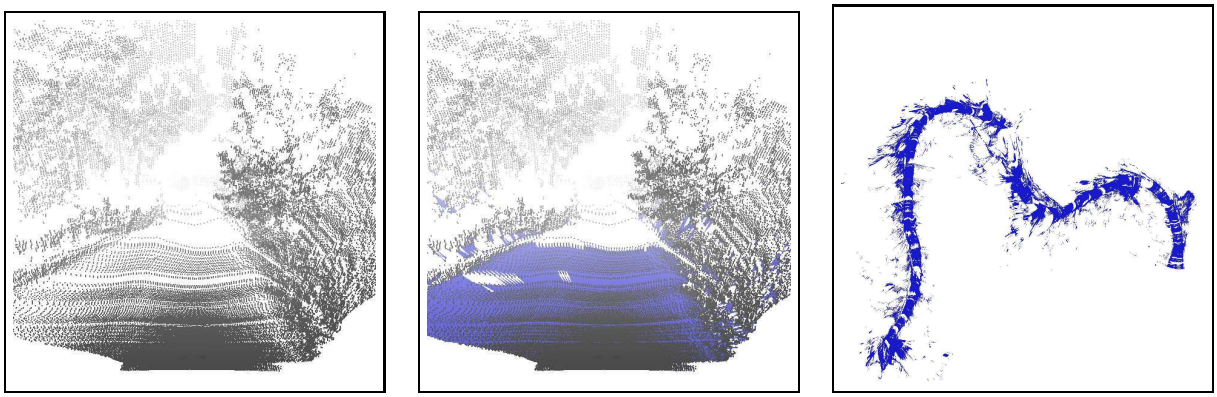

Figure 9: Left: A single outdoor 3D scan that corresponds to the first 3D scan of Fig. 8. Middle: Drivable surfaces are extracted and marked with blue color. Right: The road (gravel path) and parts of the lawn are marked as drivable. The partial discontinuity is due to the highly uneven ground.

5.2.1 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.

5.2.2 Autonomous Exploration

To overcome the problems with a human operator, a fuzzy rule is implemented that steers the robot autonomously into free drivable space, yielding an autonomous "wander around" behaviour with obstacle avoidance. For each extracted drivable ground point, an instance of a fuzzy rule with the following structure (Eq. (14)) is 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)                    (14)
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 $\{(\varphi_i, r_i)\}_{i=1,\ldots,181}$, the angle α is calculated by

$$\alpha = \text{atan2}\left(\begin{array}{l} \sum_{i=1}^{181} \sin(\varphi_i) \cdot f_1(\varphi_i) \cdot f_2(r_i) \ , \\ \\ \sum_{i=1}^{181} \cos(\varphi_i) \cdot f_1(\varphi_i) \cdot f_2(r_i) \end{array} \right). \tag{15}$$

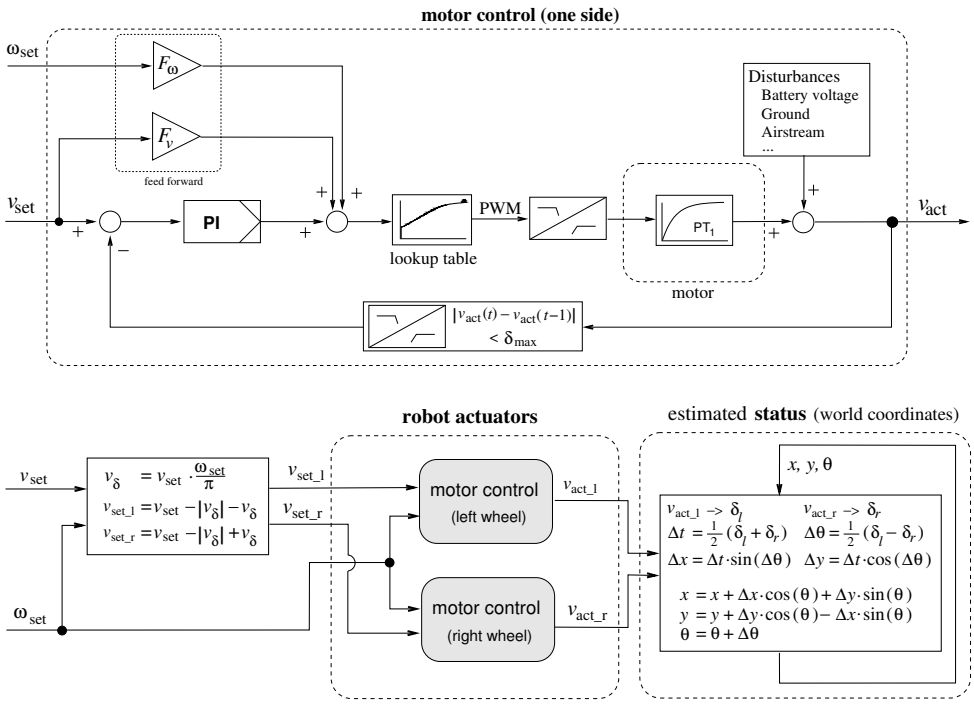

Figure 10: Top: Control unit for one motor, based on a feed forward PI controller, thus combines open and closed loop control concepts. Bottom: Schematic overview of the robot's control system, containing two motor controllers.

The fuzzy predicates `is in driving direction` and `is large` are defined by membership functions f_1, f_2, resp., which are given in Fig. 11.

Kurt3D drives a certain distance into this direction and acquires 3D scans to build automatically a 3D maps. Furthermore, the SPLAM module is used in autonomous mode to ensure enough overlap between two consecutive 3D scans.

6 System Tests

The Kurt3D robot has been tested in the RoboCup Rescue Competition 2004 in Lisbon (Team Kurt3D) and 2005 in Osaka (Team Deutschland1, a cooperation between University of Osnabrück, RTS University of Hannover and Fraunhofer AIS). Additional tests have been carried out in Outdoor environments [6].

Aknowledgements

We thank Mathias Hennig, Kai Pervölz and Hartmut Surmann for preceeding joint research. The robot platform KURT2 is commercially available from KTO (`www.kto.de`) and the 3D laser range finder from Fraunhofer AIS (`www.ais.fraunhofer.de/ARC/3D`).

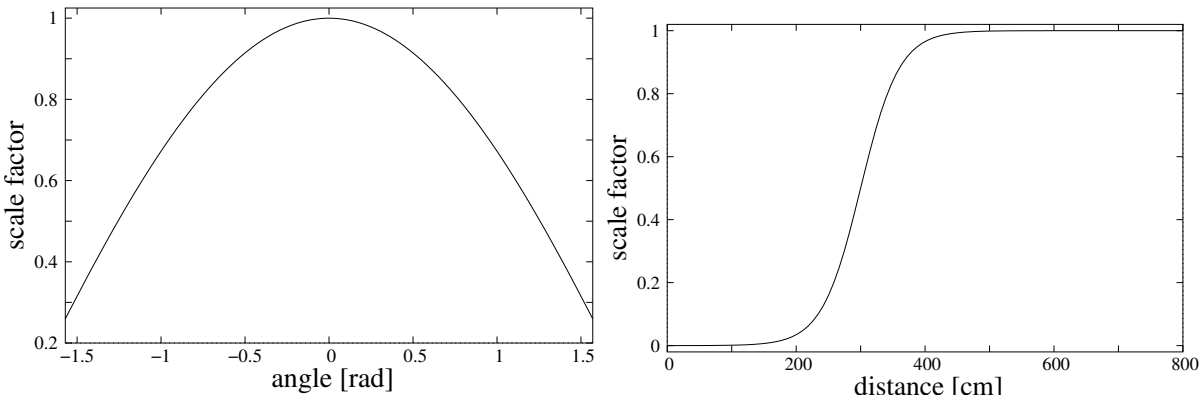

Figure 11: Set value generation with fuzzy rules, see Eq. (14), (15). Left: Function f_1 weighs the orientation of the data ("`is in driving direction`"). Right: f_2 implements a weighting on the measured distance ("`is large`"). Left: $\cos(\varphi/1.2)$. Right: $1/(1 + \exp((300 - r)/30))$.

References

[1] K. S. Arun, T. S. Huang, and S. D. Blostein. Least square fitting of two 3-d point sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(5):698 – 700, 1987.

[2] S. Arya and D. M. Mount. Approximate nearest neigbor queries in fixed dimensions. In *Proceedings of the 4th ACM-SIAM Symposium on Discrete Algorithms*, pages 271 – 280, 1993.

[3] P. Besl and N. McKay. A method for Registration of 3–D Shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239 – 256, February 1992.

[4] K. Lingemann, A. Nüchter, J. Hertzberg, and H. Surmann. About the Control of High Speed Mobile Indoor Robots. In *Proceedings of the Second European Conference on Mobile Robotics (ECMR 2005)*, pages 218 – 223, Ancona, Italy, September 2005.

[5] A. Nüchter, K. Lingemann, J. Hertzberg, and H. Surmann. 6D SLAM with Approximate Data Associoation. In *Proceedings of the 12th International Conference on Advanced Robotics (ICAR '05)*, pages 242 – 249, July 2005.

[6] A. Nüchter, K. Lingemann, J. Hertzberg, and H. Surmann. Heuristic-Based Laser Scan Matching for Outdoor 6D SLAM. In *KI 2005: Advances in Artificial Intelligence. 28th Annual German Conference on AI, Proceedings Springer LNAI vol. 3698*, pages 304 – 319, Koblenz, Germany, September 2005.

[7] H. Surmann, K. Lingemann, A. Nüchter, and J. Hertzberg. A 3D laser range finder for autonomous mobile robots. In *Proceedings of the of the 32nd International Symposium on Robotics (ISR '01)*, pages 153 – 158, Seoul, Korea, April 2001.

[8] H. Surmann, A. Nüchter, K. Lingemann, and J. Hertzberg. 6D SLAM A Preliminary Report on Closing the Loop in Six Dimensions. In *Proceedings of the 5th IFAC Symposium on Intelligent Autonomous Vehicles (IAV '04)*, Lisabon, Portugal, July 2004

[9] O. Wulf, K. O. Arras, H. I. Christensen, and B. A. Wagner. 2D Mapping of Cluttered Indoor Environments by Means of 3D Perception. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '04)*, pages 4204 – 4209, New Orleans, USA, April 2004.