Mapping of Rescue Environments with Kurt3D

A. Nüchter, K. Lingemann, J. Hertzberg
University of Osnabrück
Institute for Computer Science
Knowledge-Based Systems Research Group
Albrechtstraße 28
D-49069 Osnabrück, Germany
nuechter@informatik.uni-osnabrueck.de

H. Surmann, K. Pervölz, M. Hennig,
K. R. Tiruchinapalli, R. Worst, T. Christaller
Fraunhofer Institute for
Autonomous Intelligent Systems (AIS)
Schloss Birlinghoven
D-53754 Sankt Augustin, Germany
hartmut.surmann@ais.fraunhofer.de

***Abstract*— Deploying rescue workers in an urban setting is often a perilous, time-, power-, and force-consuming job, and systems to assist in this effort are needed. A fundamental task for rescue is to localize injured persons. To this end, robotic systems are used for mapping a site and for remote inspection of suspicious objects. The mobile robot Kurt3D is the first rescue 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.**

I. Introduction

For protecting humans, it is nowadays important to build robots that are able to operate in earthquake, fire, explosive and chemical disaster areas. The community of Urban Search and Rescue Robotics (USAR) grows very fast. Many robots are manufactured, both from research institutes and from industry. However, until now, there have been no systems that can reliably map their environment. The mobile robot Kurt3D was presented at RoboCup Rescue 2004 in Lisbon (Fig. 1). The robot is capable of mapping its environment in 3D and self localize in six degrees of freedom, i.e., considering its x, y and z positions and the roll, yaw and pitch angles (6D SLAM).

Kurt3D's mapping system consists of three major parts. First is the precise planar pose tracking algorithm HAYAI. Using 2D laser scans, features that correspond to natural landmarks are extracted and paired with features of previous scans. Second, 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. The third module is an interactive, semi-automatic control program that enables the operator to interfere with the mapping process for corrections. This paper focusses on Kurt3D's mapping and localization modules and their interaction in the rescue context, devising the similarities between HAYAI and 6D SLAM.

A. State of the Art

1) Rescue Robotics Systems: Current rescue robots are mainly designed for searching for victims and paths through rubble that would be quicker to excavate, for structural inspection and for detection of hazardous material [7]. The robots are designed to go a bit deeper than traditional search equipment, i.e, cameras mounted on poles [7]. The actual operating range of current rescue robots is $5-20$ m. The robots, e.g., the microtracs "micro-VGTV" and "Solem" [8], are small tanks that are connected with the operator by wire for transmitting a video signal. In fact, cameras are the only sensors, thus mapping the environment is basically impossible. Other rescue robot systems are based on tank-like chassis, too. While mapping environments is a large research field in mobile robotics, only a little work has been done in the *automatic* mapping of rescue environments, as presented in [4].

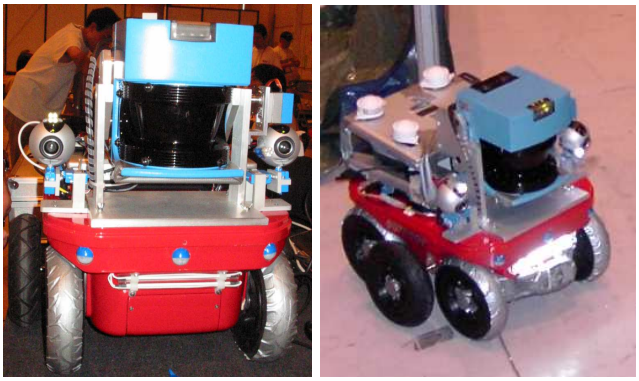

Fig. 1. The mobile robot Kurt3D equipped with the 3D laser range finder as presented at RoboCup 2004. The scanners technical basis is a SICK 2D laser range finder (LMS-200).

2) 3D Mapping: Instead of using 3D scanners which yield consistent 3D scans in the first place, some groups have attempted to build 3D volumetric representations of environments with 2D laser range finders. E.g., Thrun et al. [13] use two 2D laser range finders for acquiring 3D data. One laser scanner is mounted horizontally, the other vertically. The latter one grabs a vertical scan line which is transformed into 3D points based on the current robot pose. The horizontal scanner is used to compute the robot pose. The precision of 3D data points crucially depends on that pose and on the precision of the scanner. The same argument applies to the work of Wulf et al. who let the scanner rotate around the vertical axis [14].

A few other groups use high accurate, expensive 3D laser scanners [5], [10]. The RESOLV project aimed at modeling interiors for virtual reality and tele-presence [10]. They used a RIEGL laser range finder on robots and the ICP algorithm for scan matching [3]. The AVENUE project develops a robot for modeling urban environments [5],

using a CYRAX laser scanner. Nevertheless, in their recent work they do not use data of the laser scanner in the robot control architecture for localization [5]. The research group of M. Hebert has reconstructed environments using the Zoller+Fröhlich laser scanner and aims to build 3D models without initial position estimates, i.e., without odometry information [6].

II. The Exploration Robot Kurt3D

Kurt3D[1] (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, both 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, where the middle ones are 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) charge. 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 [11]. 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, 4.5 V with batteries of 4500 mAh) on one battery pack. The area of $180°(\text{h}) \times 90°(\text{max. } 120°)(\text{v})$ 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. Scanning the environment with a mobile robot is done in a stop-scan-go fashion.

Kurt3D is equipped with 2×4 super bright LEDs and two fluorescent tubes to illuminate the surroundings. The LEDs are attached to the two pan-and-tilt Logitech QuickCam 4000 cameras. Additional 8 NiMH cells are used to power the light.

III. Pose Tracking with HAYAI

This section describes the newly developed algorithm HAYAI (*Highspeed And Yet Accurate Indoor/outdoor-tracking*). The matching algorithm 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 $\Delta p = (\Delta x, \Delta y, \Delta\theta)^T$ as the optimal transformation for mapping $\check{D}$ onto $\check{M}$.
4) Update the robot's pose $p_n \xrightarrow[\Delta p]{} p_{n+1}$ according to formula (1).
5) Save the current scan as new reference scan $\mathcal{R} \leftarrow \mathcal{S}$.

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

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

A. Data Filtering

Scanning is noisy and small errors may occur, namely Gaussian noise and salt and pepper noise. The latter one arises for example at edges where the laser beam of the scanner 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).

B. Extraction and Matching of Features

As described above, the scan matching algorithm computes a transformation Δp 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

[1] Videos of explorations with Kurt3D can be found at: http://www.ais.fraunhofer.de/ARC/kurt3D/index.html

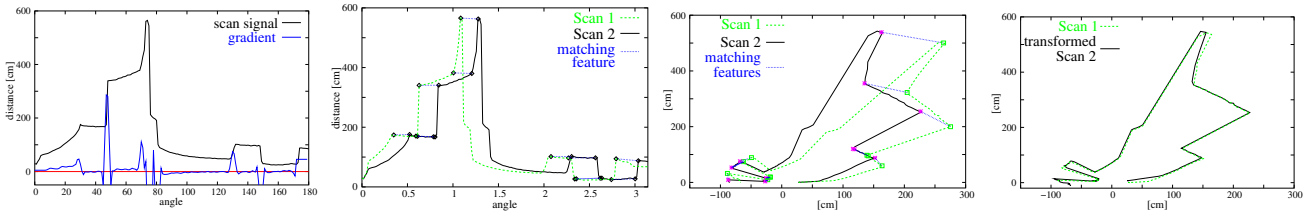

Fig. 2. From left to right: (1) Application of the feature detection filters. (2) and (3) Pairing of corresponding features. (φ, r) representation (2) vs. the euclidian one (3). (4) Transformed scan.

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. 2 (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 m_i, d_j, with $m_i = (m_i^x, m_i^y)^T$ in Cartesian, or $(m_i^\varphi, m_i^r)^T$ in polar coordinates, resp. $(d_j$ analogously):

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

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:

$$\Theta(m_i, d_j) = \begin{cases} 0 & \Gamma(m_i) = \Gamma(d_j) \\ \infty & \text{else} \end{cases}$$

with a classification function $\Gamma\colon (M \cup D) \mapsto \{\text{max.}, \text{min.}, \text{inflection point}\}$. The resulting matrix $w_{i,j}$ denoting feature correspondences is simplified until the match is non-ambiguous. Fig. 2 shows the match of two scans.

C. Pose Calculation

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

$$E(R,t) = \sum_{i=1}^{N_m} \sum_{j=1}^{N_d} w_{i,j} \big|\big| m_i - (Rd_j + t) \big|\big|^2 \tag{3}$$

$$\propto \frac{1}{N} \sum_{i=1}^{N} ||m_i - (Rd_i + t)||^2, \tag{4}$$

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

$$c_m = \frac{1}{N}\sum_{i=1}^{N} m_i, \qquad c_d = \frac{1}{N}\sum_{i=1}^{N} d_i \tag{5}$$

and

$$M' = \{m_i' = m_i - c_m\}_{1,\ldots,N}, \tag{6}$$

$$D' = \{d_i' \ = d_i - c_d\}_{1,\ldots,N}. \tag{7}$$

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

$$\begin{aligned} E(R,t) &\propto \frac{1}{N} \sum_{i=1}^{N} ||m_i' - Rd_i' - \underbrace{(t - c_m + Rc_d)}_{=\tilde{t}}||^2 \\ &= \frac{1}{N} \sum_{i=1}^{N} \big|\big| m_i' - Rd_i' \big|\big|^2 + \frac{1}{N} \sum_{i=1}^{N} ||\tilde{t}||^2 \qquad (8a) \\ &\quad - \frac{2}{N} \tilde{t} \cdot \sum_{i=1}^{N} \big(m_i' - Rd_i'\big) \qquad (8b) \end{aligned}$$

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 (8a) has its minimum for $\tilde{t} = \mathbf{0}$ or $t = c_m - Rc_d$. Therefore the algorithm has to minimize only the first term, and the corresponding error function is:

$$E(R) \propto \sum_{i=1}^{N} \big|\big| m_i' - Rd_i' \big|\big|^2. \tag{9}$$

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

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

With given rotation, the translation is calculated as follows:

$$\underbrace{\begin{pmatrix}\Delta x\\ \Delta y\end{pmatrix}}_{=\Delta t} = c_m - \underbrace{\begin{pmatrix}\cos\Delta\theta & \sin\Delta\theta\\ -\sin\Delta\theta & \cos\Delta\theta\end{pmatrix}}_{=R_{\Delta\theta}} \cdot c_d. \tag{11}$$

IV. 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.

A. 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 to find the transformation consisting of a rotation R and a translation t which minimizes the cost function (3). Note: This time the vectors are in 3D space and 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 (R, t) that minimize $E(R,t)$ 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 (R,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 (R, 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 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 algorithm computes the optimal rotation by $R = VU^T$. Hereby the matrices V and U are derived by the singular value decomposition $H = U\Lambda V^T$ of a correlation matrix H. This (3×3) matrix H is given by

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

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., $||Rd_i'||^2 = ||d_i'||^2$, the error function (9) is expanded to

$$E(R,t) = \sum_{i=1}^{N} \left|\left|m_i'\right|\right|^2 - 2\sum_{i=1}^{N} m_i' \cdot Rd_i' + \sum_{i=1}^{N} \left|\left|d_i'\right|\right|^2.$$

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

$$\sum_{i=1}^{N} m_i' \cdot Rd_i' = \sum_{i=1}^{N} m_i'^T Rd_i'. \tag{13}$$

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

$$\mathrm{tr}\left(\sum_{i=1}^{N} Rd_i' m_i'^T\right) = \mathrm{tr}(RH).$$

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

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

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

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

for any orthonormal matrix B. Therefore the matrix 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)): $t = c_m - Rc_d$.

B. 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 and a closed loop will be inconsistent. Our 6D SLAM algorithm detects a closing loop by registering the last acquired 3D scan with earlier acquired scans. 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 [9], [12].

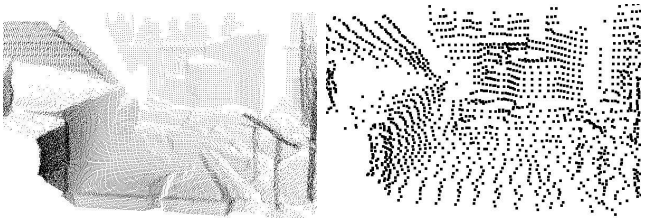

Fig. 3. Left: A view of a 3D scene (66785 3D data points). Right: Subsampled version (points have been enlarged, 6700 data points).

C. ICP Speedups

The computational requirements 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 *k*d-trees for the ICP algorithm.

Data reduction for the ICP algorithm is done using the proposed filters of subsection III-A. 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. 3).

*k*D-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 *k*d-trees is done recursively. For a given 3D point 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. 4, left).

Arya et al. introduce the following notion for approximating the nearest neighbor [2]: Given an $\varepsilon > 0$, then the point $p \in D$ is the $(1+\varepsilon)$-approximate nearest neighbor of the point p_q iff $||p-q|| \leq (1+\varepsilon)||p^*-q||$, whereas p^* denote the true nearest neighbor, i.e., p is within a relative error of ε of the true nearest neighbor. In every step the algorithm records the closest point p; the search finishes if the distance to the unanalyzed leafs is larger than $\left|\left|p_q - p\right|\right| / (1+\varepsilon)$. Fig. 4 (right) shows an example where the gray cell doesn't have to be analyzed, since the point p satisfies the approximation criterion. Fig. 5 shows the computation time for matching two 3D scans using *k*d-trees (left) and approximate *k*d-trees (right) with $\varepsilon = 50$ for

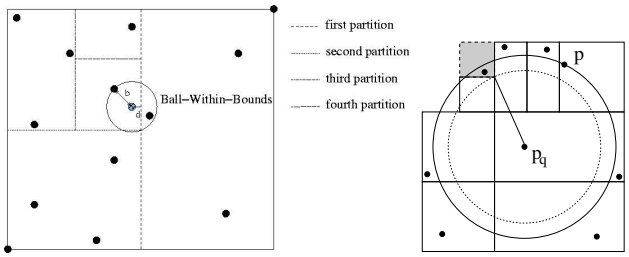

Fig. 4. Left: Construction of a *k*d-tree. Right: The $(1+\varepsilon)$-approximate nearest neighbor. The search algorithm doesn't have to analyze the gray cell, since the point p satisfies the approximation criterion.

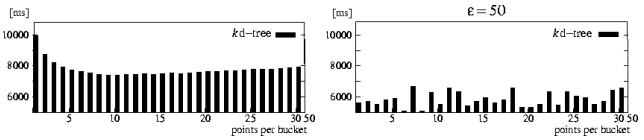

Fig. 5. Left: Run time of the ICP algorithm using *k*d-trees with different bucket sizes. The minimal time is reached for 10 points per bucket. Right: Computing time for Approximate *k*d-tree search.

different bucket sizes. In [9] 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.

V. The Software Architecture

Fig. 6 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 are processed and merged with a Kalman filter. The processing time of HAYAI is around 3% of the CPU load. Thus there is enough time to compress the camera images to jpeg and to transmit the data.

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 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 tried 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.

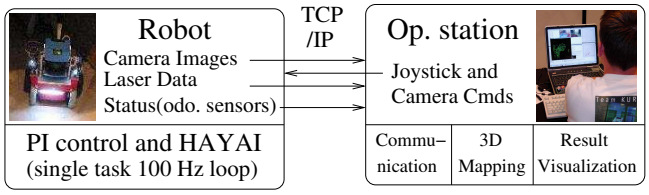

Fig. 6. System Overview: The robot runs a 100 Hz loop for motor control, scan processing, image encoding and transmission. The Kurt3D server executes three threads that are responsible for communicating, 3D mapping and result visualization.

VI. Results and Conclusion

The proposed algorithms have been evaluated at RoboCup Rescue 2004 in Lisbon. Fig. 7 shows an online generated 3D map (top view). Two 3D views are given in Fig. 8. An offline-rendered animation of the acquired data can be found at `http://www.ais.fhg.de/ARC/kurt3D/rr.html`.

Based on the principles of 3D scan matching, we have designed a SLAM algorithm in six dimensions using loop closing and global error minimization [9]. This global error minimization is currently too slow to be deployed on a rescue system. The computing time of this relaxation algorithm is in the order of several minutes for a typical rescue arena, thus in the designated scenario we use pure 3D scan matching for mapping. Since 3D scans potentially provide a lot of information and the area is small, i.e., 6 m $\times$ 6 m, scan matching is sufficient for mapping.

This paper has presented a mobile robotic system for teleoperated 3D mapping of environments. The mapping is done by means of a fast pose tracking algorithm and 3D scan matching. We have demonstrated our 3D mapping capabilities at the RoboCup Rescue 2004 in Lisbon, were our team won the 2nd prize. The aim of future work is combining the mapping algorithms with mechatronic robotic systems, i.e., building a robot system that can actually go into the third dimension and can cope with the red arena in RoboCup Rescue. Furthermore, we concentrate on enhancing the system's autonomy: In addition to automatic mapping, autonomous driving and exploration are planned.

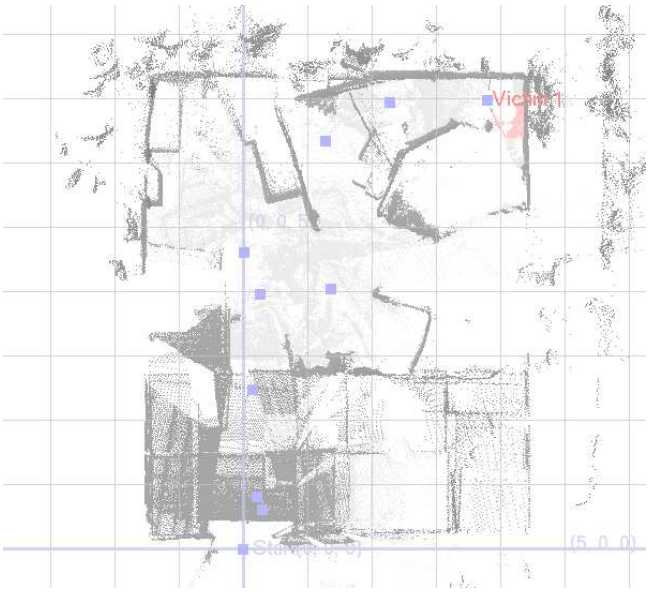

Fig. 7. A 3D map of the rescue arena (orange) during RoboCup 2004 as a point cloud (top view). The points on the ground have been colored in light grey. The 3D scan positions (blue) and a found victim (red) are marked.

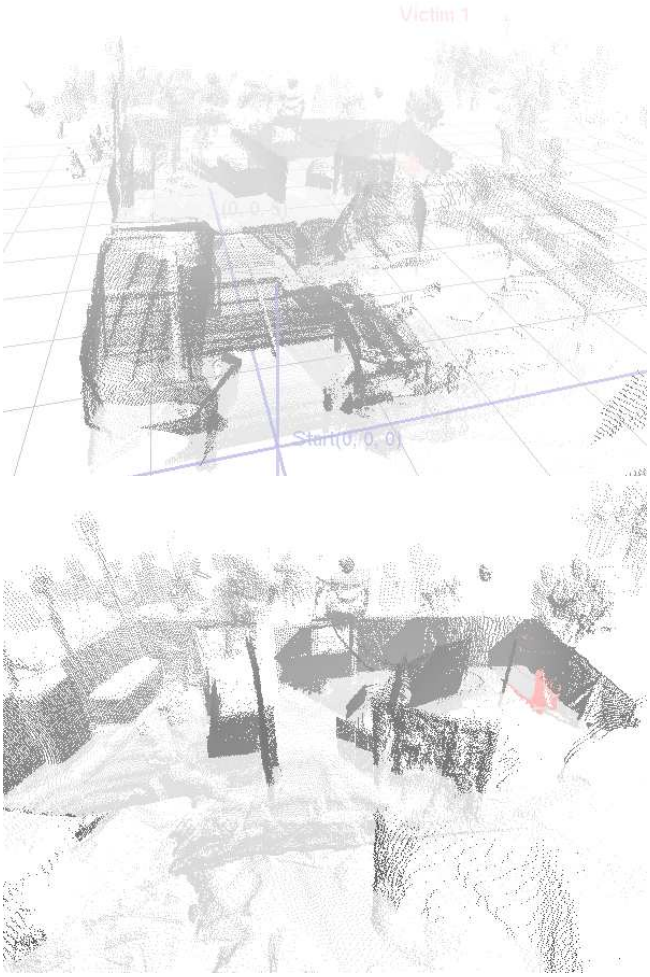

Fig. 8. A 3D view of the map of Fig. 7 rendered from a pose slightly above the arena. The 1 m^2 grid is superimposed in the first map.

References

[1] K. S. Arun, T. S. Huang, and S. D. Blostein. Least square fitting of two 3-d point sets. *IEEE PAMI*, 9(5), 1987.

[2] S. Arya et al. Approximate nearest neigbor queries in fixed dimensions. In *Proc. 4th ACM Symp. on Discrete Algorithms*, 1993.

[3] P. Besl and N. McKay. A method for Registration of 3–D Shapes. *IEEE PAMI*, 14(2), 1992.

[4] St. Carpin, H. Kenn, and A. Birk. Autonomous Mapping in the Real Robots Rescue League. In *Proc. RoboCup 2003*, 2004.

[5] A. Georgiev and P. K. Allen. Localization methods for a mobile robot in urban environments. *IEEE TRO*, 20(5), 2004.

[6] M. Hebert, M. Deans, D. Huber, B. Nabbe, and N. Vandapel. Progress in 3–D Mapping and Localization. In *Proc. SIRS*, 2001.

[7] R. R. Murphy. Activities of the rescue robots at the world trade center from 11-21 September 2001. *IEEE Robotics & Automation Magazine*, 11(3), 2004.

[8] R. R. Murphy. Rescue robotics for homeland security. *Com. of the ACM, Special Issue on Homeland Security*, 27(3), 2004.

[9] A. Nüchter, K. Lingemann, J. Hertzberg, and H. Surmann. 6D SLAM with Approximate Data Associoation. In *Submitteed to ICAR, 2005*.

[10] V. Sequeira et al. Automated 3D reconstruction of interiors with multiple scan–views. In *Proc. SPIE 99*, USA, 1999.

[11] H. Surmann et al. A 3D laser range finder for autonomous mobile robots. In *Proc. 32nd ISR*, 2001.

[12] H. Surmann, A. Nüchter, K. Lingemann, and J. Hertzberg. 6D SLAM A Preliminary Report on Closing the Loop in Six Dimensions. In *Proc. IFAC Symp. IAV*, Lisbon, Portugal, 2004.

[13] S. Thrun, D. Fox, and W. Burgard. A real-time algorithm for mobile robot mapping with application to multi robot and 3D mapping. In *Proc. IEEE ICRA*, San Francisco, USA., 2000.

[14] O. Wulf, K. O. Arras, H. I. Christensen, and B. A. Wagner. 2D Mapping of Cluttered Indoor Environments by Means of 3D Perception. In *Proc. IEEE ICRA*, New Orleans, USA, April 2004.