6D SLAM – 3D Mapping Outdoor Environments

Andreas Nüchter*, **Kai Lingemann, Joachim Hertzberg**
Institute of Computer Science
University of Osnabrück
D-49069 Osnabrück, Germany
{nuechter|lingemann|hertzberg}@informatik.uni-osnabrueck.de

Hartmut Surmann
Fraunhofer Institute IAIS
Schloss Birlinghoven
D-53754 Sankt Augustin, Germany
hartmut.surmann@iais.fraunhofer.de

Abstract

6D SLAM (Simultaneous Localization and Mapping) or 6D Concurrent Localization and Mapping of 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 outdoor with a mobile robot, must regard these degrees of freedom. This paper presents a robotic mapping method based on locally consistent 3D laser range scans. *Iterative Closest Point* (ICP) scan matching, combined with a heuristic for closed loop detection and a global relaxation method, results in a highly precise mapping system. A new strategy for fast data association, cached kd-tree search, leads to feasible computing times. With no ground-truth data available for outdoor environments, point relations in maps are compared to numerical relations in uncalibrated aerial images in order to assess the metric validity of the resulting 3D maps.

1 Introduction

Automatic environment sensing and modeling is a fundamental scientific issue in robotics, since the availability of maps is essential for many robot tasks. Manual mapping of environments is a hard and tedious job: Thrun et al. report a time of about one week of hard work for creating a map of the museum in Bonn for the robot RHINO (Thrun, 1998). In particular mobile systems with 3D laser scanners that automatically perform multiple steps such as scanning, gaging and autonomous driving have the potential to greatly improve mapping.

*WWW http://www.informatik.uni-osnabrueck.de/nuechter/

Reliable 3D robotic mapping has an evident impact on safety, security, and rescue robotics. It enables rescue robots to build maps, which can be used by rescue workers to recover victims and precisely locate potential threats. As another example application, we showed in earlier work the mapping of abandoned underground mines. Serious threats emanate from abandoned mines, such as structural shifts, which can cause the surface above to collapse, and ground water contamination (Nüchter et al., 2004). Other applications also benefit from automatic and precise 3D environment modeling, e.g., industrial automation, architecture, agriculture, the construction or maintenance of tunnels, and factory design, facility management, urban and regional planning.

The robotic mapping problem is that of acquiring a spatial model of a robot's environment. If the robot poses were known, the local sensor inputs of the robot, i.e., local maps, could be registered into a common coordinate system to create a map. Unfortunately, any mobile robot's self localization suffers from imprecision. Therefore, the structure of the local maps, e.g., of single scans, needs to be used to create a precise global map. Finally, robot poses in natural outdoor environments involve yaw, pitch, roll angles and elevation, turning pose estimation as well as scan registration into a problem in six mathematical dimensions.

This paper proposes algorithms that allow us to digitize large environments and solve the 6D SLAM problem. In previous works we already presented our core 6D SLAM algorithm with global relaxation (Surmann et al., 2003; Surmann et al., 2004) and loop closing (Surmann et al., 2004). This paper's contribution is threefold: First, we present an octree-based matching heuristic that allows us to match scans with rudimentary starting guesses and detect closed loops. Second, we present a novel search procedure, namely cached kd-trees, exploiting iterative behavior of the ICP algorithm. It results in a significant speed-up. Finally, the paper summarizes our previous work on 3D mapping, and gives a complete view of our algorithms.

The paper is organized as follows: Sec. 2 describes related work. Then we introduce our solution to the 6D SLAM problem. Sec. 4 describes our strategies to render the algorithms computationally feasible. Sec. 5 presents a brief description of the used hardware and experiment and results. Finally, Sec. 7 concludes.

2 Related Work

One way to categorize mapping algorithms is by the map type. In general, the map can either be topological or metrical. Metrical maps represent explicit distances of the environment. These maps can either be 2D, usually an upright projection, or 3D, i.e., a volumetric environment map. Furthermore, SLAM approaches can be classified by the number of degrees of freedom of the robot pose. A 3D pose estimate contains the (x, y)-coordinate and a rotation θ, whereas a 6D pose estimate considers all degrees of freedom a rigid mobile robot can have, i.e., the (x, y, z)-coordinate and the roll, yaw and pitch angles.

3D maps can be generated by three different techniques: First, a planar localization method combined with a 3D sensor, second a precise 6D pose estimate combined with a 2D sensor and third, a 3D sensor with a 6D localization method. Tab. 1 summarizes these mapping

techniques (black) in comparison with planar 2D mapping (grey). In this paper we focus on 3D data and 6D localization, hence on 6D SLAM.

Table 1: Overview of the dimensionality of SLAM approaches. Grey: 2D maps. Black: 3D maps.

		Dimensionality of pose representation	
		3D	6D
Sensor data	2D	**Planar 2D mapping** 2D mapping of planar sonar and laser scans. See (Thrun, 2002) for an overview.	**Slice-wise 6D SLAM** 3D mapping using a precise localization, considering the x,y,z-position and the roll yaw and pitch angle.
	3D	**Planar 3D mapping** 3D mapping using a planar localization method and, e.g., an upward looking laser scanner or 3D scanner.	**Full 6D SLAM** 3D mapping using 3D laser scanners or (stereo) cameras with pose estimates calculated from the sensor data.

2.1 Planar 2D Mapping

The state of the art for planar metric maps are probabilistic methods, where the robot has probabilistic motion models and uncertain perception models. By integrating of these two distributions with, e.g., a Kalman or particle filter, it is possible to localize the robot (Smith et al., 1986; Leonard and Durrant-Whyte, 1991). Mapping is often an extension to this estimation problem. Beside the robot pose, positions of landmarks are estimated. Closed loops, i.e., a second encounter of a previously visited area of the environment, play a special role here. Once detected, they enable the algorithms to bound the error by deforming the already mapped area such that a topologically consistent model is created. However, there is no guarantee for a correct model. Several strategies exist for solving SLAM. Thrun reviews in (Thrun, 2002) existing techniques, i.e., maximum likelihood estimation (Frese and Hirzinger, 2001; Folkesson and Christensen, 2003), expectation maximization (Thrun et al., 1997), extended Kalman filter (Dissanayake et al., 2001) or (sparse extended) information filter (Thrun et al., 2004). In addition to these methods, FastSLAM (Thrun et al., 2000), that approximates the posterior probabilities, i.e., robot poses, by particles, and the method of Lu/Milios on the basis of IDC scan matching (Lu and Milios, 1997) play an important role in 2D.

In principle, the probabilistic methods from planar 2D mapping are extendable to 3D mapping with 6D pose estimates (Weingarten and Siegwart, 2005). However, no reliable feature extraction nor a strategy for reducing the computational costs of multi hypothesis tracking, e.g., FastSLAM, that grows exponentially with the degrees of freedom, has been published to our knowledge. The qualitative shift in the complexity is due to the necessity to draw samples in each dimension.

2.2 3D Mapping

Planar 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. Thrun et al. (Thrun et al., 2000), Früh et al. (Früh and Zakhor, 2001) and Zhao et al. (Zhao and Shibasaki, 2001) use two 2D laser scanners for acquiring 3D data. One 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 3D robot pose. Since the vertical scanner is not able to scan sides of objects, Zhao et al. use two additional, vertically mounted 2D scanners, shifted by 45° to reduce occlusions (Zhao and Shibasaki, 2001). The horizontal scanner is used to compute the 3D robot pose. The precision of 3D data points depends, besides on the precision of the scanner, critically on that pose estimation.

Recently, different groups employ rotating SICK scanners for acquiring 3D data (Kohlhepp et al., 2003; Wulf et al., 2004). Wulf et al. let the scanner rotate around the vertical axis. They acquire 3D data while moving, thus the quality of the resulting map crucially depends on the pose estimate that is given by inertial sensors, i.e., gyros (Wulf et al., 2004). In addition, their SLAM algorithms do not consider all six degrees of freedom.

Slice-wise 6D SLAM. Local 3D maps built by 2D laser scanners and 6D pose estimates are often used for mobile robot navigation. A well-known example is the grand challenge, where the Stanford racing team used this technique for high speed terrain classification (Thrun et al., 2006).

Similar to the planar 3D mapping case, the accuracy of the resulting 3D map depends on the robot's pose estimate. This cannot be accomplished with inexpensive sensors.

Full 6D SLAM. A few other groups use highly accurate, expensive 3D laser scanners (Allen et al., 2001; Georgiev and Allen, 2004; Sequeira et al., 1999). The RESOLV project aimed at modeling interiors for virtual reality and tele-presence (Sequeira et al., 1999). They used a RIEGL laser range finder on robots and the ICP algorithm for scan matching (Besl and McKay, 1992). The AVENUE project develops a robot for modeling urban environments (Allen et al., 2001), using a CYRAX scanner and a feature-based scan matching approach for registering the 3D scans. However, in their recent work they do not use data of the laser scanner in the robot control architecture for localization (Georgiev and Allen, 2004). M. Hebert's group has reconstructed environments using the Zoller+Fröhlich laser scanner and aims to build 3D models without initial position estimates, i.e., without odometry information (Hebert et al., 2001). Recently, Magnusson and Duckett proposed a 3D scan alignment method that – in contrast to the previously mentioned research groups – does not use the ICP algorithm, but the normal distribution transform instead (Magnusson and Ducket, 2005).

Other Approaches. Other approaches use information from CCD-cameras that provide a view of the robot's environment (Biber et al., 2004; Se et al., 2001). Nevertheless, cameras

are difficult to use in natural environments with changing light conditions. Camera-based approaches to 3D robot vision, e.g., stereo cameras and structure from motion, have difficulties providing reliable navigation and mapping information for a mobile robot in real-time. Thus some groups try to solve 3D modeling by using planar scanner based SLAM methods and cameras, e.g., in (Biber et al., 2004).

3 Range Image Registration and Robot Relocalization

Multiple 3D scans taken from different poses are necessary to digitalize environments without occlusions. To create a correct and consistent model, the scans have to registered in one common coordinate system. If the robot carrying the 3D scanner were precisely localized, the registration could be done directly based on the robot pose. However, due to the imprecise robot sensors, self localization is erroneous, so the geometric structure of overlapping 3D scans has to be considered for registration. As a by-product, successful registration of 3D scans relocalizes the robot in 6D, by providing the transformation to be applied to the robot pose estimation at the recent scan point. In this manner, localization and scan registration, i.e. mapping, are intertwined.

Our solution to the 6D SLAM problem is based on scan registration. We consider a mobile robot recording its odometry and scanning the environment in a stop-scan-go fashion. The alignment of two scans is done by the ICP algorithm (Besl and McKay, 1992). However, ICP alone is not sufficient for solving the 6D SLAM problem. The following extensions with regards to flexibility and speed have been made to this end:

1. Extrapolate the odometry to all 6 degrees of freedom.
2. Calculate heuristic initial estimations for ICP scan matching based on this extrapolation.
3. Register the 3D scans into a common coordinate system using ICP.
4. If applicable close the loop and distribute the error.
5. After all scans are taken, refine the model by global relaxation.

These extensions will be handled in in the following subsections. Our algorithm maintains a single 6D robot pose estimate. The extensions provide the basis for reliable mapping, as shown in the result section. Combining 3D ICP scan matching and 6D poses in a multi hypotheses approach is not computationally feasible. The approach presented in this paper concentrates on single loops.

In our SLAM framework robot poses are represented in different contexts in one of three different ways, namely, first, by an OpenGL-style 4×4 Matrix, with the robot position $\boldsymbol{t} = (x, y, z)$ and its orientations given as as the orthonormal matrix $\mathbf{R} \in \mathbb{R}^3$

$$\mathbf{P} = \left(\begin{array}{c|c} \mathbf{R} & \mathbf{0} \\ \hline \boldsymbol{t} & 1 \end{array}\right)$$

$$= \left(\begin{array}{ccc|c} \cos\theta_y \cos\theta_z & \sin\theta_x \sin\theta_y \cos\theta_z + \cos\theta_x \sin\theta_z & -\cos\theta_x \sin\theta_y \cos\theta_z + \sin\theta_x sin\theta_z & \\ -\cos\theta_y \sin\theta_z & -\sin\theta_x \sin\theta_y \sin\theta_z + \cos\theta_x \cos\theta_z & \cos\theta_x \sin\theta_y \sin\theta_z + \sin\theta_x \cos\theta_z & \mathbf{0} \\ \sin\theta_y & -\sin\theta_x \cos\theta_y & \cos\theta_x \cos\theta_y & \\ \hline x & y & z & 1 \end{array}\right),$$

or, second, the corresponding 6-vector, consisting of the position and of the Euler angles

$$\boldsymbol{P} = (x, y, z, \theta_x, \theta_y, \theta_z),$$

or, third, as position and quaternion (see Sec. 3.4 for details)

$$\boldsymbol{P} = (\boldsymbol{t}, \boldsymbol{q}) \qquad \text{with } \boldsymbol{q} = (q_0, q_x, q_y, q_z)$$

Note the bold-italic ($\boldsymbol{P}$, vectors) and bold ($\mathbf{P}$, matrices) notation. The conversion between the representations, i.e., Euler angles, matrix representations and quaternions is done by standard algorithms (Matrix FAQ, 1997).

The reason for using different, theoretically equivalent representations, is that they have different numerical problems, such as gimbal locks, in different situations. In the context of odometry processing we use Euler angles, for scan alignment and rendering issues orthonormal matrices, and for interpolation tasks quaternions. Converting the representation is more efficient than coping with the problems in one single representation.

3.1 Odometry Extrapolation

Since nearly all mobile robots have an odometer to measure traveled distances, our algorithm uses these measurements to calculate a first pose estimation. The odometry is extrapolated to 6 degrees of freedom using previous registration matrices. We are using a left-handed coordinate system, i.e., the y coordinate represents elevation. Then, the change of the robot pose $\Delta\boldsymbol{P}$ given the odometry information $(x_n^{\text{odo}}, z_n^{\text{odo}}, \theta_{y,n}^{\text{odo}})$, $(x_{n+1}^{\text{odo}}, z_{n+1}^{\text{odo}}, \theta_{y,n+1}^{\text{odo}})$ and the registration matrix $\mathbf{R}(\theta_{x,n}, \theta_{y,n}, \theta_{z,n})$, is calculated by solving:

$$\begin{pmatrix} x_{n+1}^{\text{odo}} \\ 0 \\ z_{n+1}^{\text{odo}} \\ 0 \\ \theta_{y,n+1}^{\text{odo}} \\ 0 \end{pmatrix} = \begin{pmatrix} x_n^{\text{odo}} \\ 0 \\ z_n^{\text{odo}} \\ 0 \\ \theta_{y,n}^{\text{odo}} \\ 0 \end{pmatrix} + \left(\begin{array}{ccc|ccc} & & & & & \\ & \mathbf{R}(\theta_{x,n}, \theta_{y,n}, \theta_{z,n}) & & & \mathbf{0} & \\ & & & & & \\ \hline & & & 1 & 0 & 0 \\ & \mathbf{0} & & 0 & 1 & 0 \\ & & & 0 & 0 & 1 \end{array}\right) \cdot \underbrace{\begin{pmatrix} \Delta x_{n+1} \\ \Delta y_{n+1} \\ \Delta z_{n+1} \\ \Delta\theta_{x,n+1} \\ \Delta\theta_{y,n+1} \\ \Delta\theta_{z,n+1} \end{pmatrix}}_{\Delta\boldsymbol{P}}.$$

Therefore, calculating $\Delta \boldsymbol{P} = (\Delta x_{n+1}, \Delta y_{n+1}, \Delta z_{n+1}, \Delta\theta_{x,n+1}, \Delta\theta_{y,n+1}, \Delta\theta_{z,n+1})$ requires a matrix inversion. If $n = 1$, $\mathbf{R}(\theta_{x,n}, \theta_{y,n})$ is set to the identity matrix. Finally, the 6D pose $\mathbf{P}_{n+1}$ is calculated by

$$\mathbf{P}_{n+1} = \Delta\mathbf{P} \cdot \mathbf{P}_n \tag{1}$$

using the poses' matrix representations. Thus, the planar odometry is extrapolated in 6D to the (x, z)-plane defined by the last robot pose. Note that the values for y_{n+1}, $\theta_{x,n+1}$, and $\theta_{z,n+1}$ are usually not equal 0, due to the matrix inversion.

3.2 Calculating Heuristic Initial Estimations for ICP Scan Matching

We use the following heuristic to compute an initial estimation for the following ICP scan matching. It allows us to match scans with rudimentary starting guesses, as given by odometry. The heuristic computes octree representations from the last acquired scan D (data set) and the previously acquired scans M (model set) and aligns them. More precisely, the following steps are executed:

1. Generate an octree $\mathfrak{O}_M$ for the n-th 3D scan (model set M)
2. Generate an octree $\mathfrak{O}_D$ for the $(n+1)$-th 3D scan (data set D).
3. For each search depth $d \in [o(s_{\text{Start}}), \ldots, o(s_{\text{End}})]$ in the octrees, based on a function $o(x)$ returning the depth within the octree that corresponds to a given cube size x, estimate a transformation $\Delta\boldsymbol{P}_{\text{best}} = (\boldsymbol{t}, \boldsymbol{R})$ as follows, using the $\mathbf{0}$-vector as initial value of $\Delta\boldsymbol{P}_{\text{best}}$:
   (a) Calculate a maximal displacement and rotation $\Delta\boldsymbol{P}_{\max}$ depending on the search depth d and currently best transformation $\Delta\boldsymbol{P}_{\text{best}}$:
   $$\Delta\boldsymbol{P}_{\max} = (d_{\max} - d + 1)\boldsymbol{c} + \Delta\boldsymbol{P}_{\text{best}},$$
   for some constant displacement vector $\boldsymbol{c}$.
   (b) For all discrete 6-tuples $\Delta\boldsymbol{P}_i \in [-\Delta\boldsymbol{P}_{\max}, \Delta\boldsymbol{P}_{\max}]$ in the domain $\Delta\boldsymbol{P} = (x, y, z, \theta_x, \theta_y, \theta_z)$, displace the octree $\mathfrak{O}_D$ by $\Delta\mathbf{P}_i \cdot \Delta\mathbf{P} \cdot \mathbf{P}_n$. Evaluate the matching of the two octrees by counting the number of overlapping cubes and save the best transformation as $\Delta\boldsymbol{P}_{\text{best}}$.

Note: Step 3b requires 6 nested loops, but the computational requirements are bounded by the coarse-to-fine strategy inherited from the octree processing. The size of the octree cubes decreases exponentially with increasing d. We start the algorithm with a cube size of 75 cm^3 and stop when the cube size falls below 10 cm^3. Using these values, $d \in \{1, \ldots, 3\}$ in our experiments. Fig. 1 shows two 3D scans and the corresponding octrees. Furthermore, note that the heuristic works best outdoors. Due to the diversity of the environment the match of octree cubes will show a significant maximum, while indoor environments with their many geometry symmetries and similarities, e.g., in a corridor, are in danger of producing many plausible matches.

To summarize, we used the following constants: $s_{\text{Start}} = 75$ cm, $s_{\text{End}} = 10$ cm, and $\boldsymbol{c} = (10, 10, 10, 5, 10, 5)$ for quantifying the search windows, such that $\Delta\boldsymbol{P}_{\max}$ decreases from

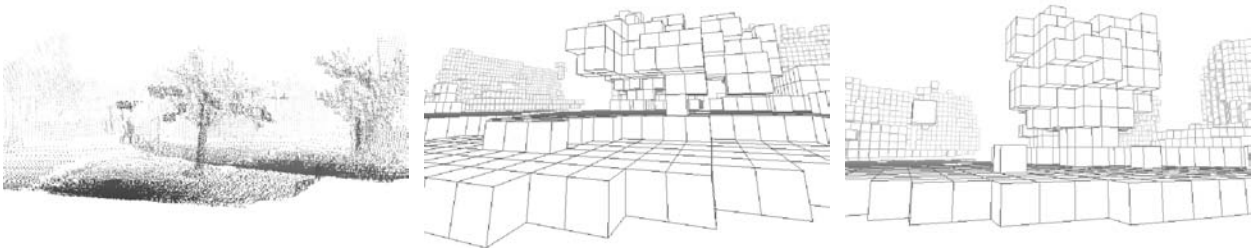

Figure 1: Left: Two 3D point clouds. Middle: Octree corresponding to the black/front point cloud. Right: Octree based on the grey/back points.

iteration to iteration in each dimension by the corresponding entry of $\boldsymbol{c}$. Finally, $d_{\max}$ is the maximal depth of the tree. Informally spoken, the algorithm generates cube representations of each 3D scan (cf. Fig. 1). Then it searches for the optimal displacement that results in the maximum overlap of these cubes. In the next iteration the cube size is reduced, i.e., cube representations given by a deeper level of the octree are utilized. Additionally, the size of the search interval is reduced with ongoing search depth/iteration.

After an initial starting guess is found, the range image registration, as described in the following section, proceeds with the robot pose estimation given as:

$$\mathbf{P}_{n+1} = \underbrace{\Delta\mathbf{P}_{\text{best}}}_{\text{octree heuristic}} \cdot \underbrace{\Delta\mathbf{P}}_{\text{odometry extrapolation}} \cdot \underbrace{\mathbf{P}_n}_{\text{previous robot pose}} \tag{2}$$

3.3 Scan Registration

The following method registers point sets into a common coordinate system. It is called ICP algorithm (Besl and McKay, 1992). Given two independently acquired sets of 3D points, $\hat{M}$ and $\hat{D}$, which correspond to a single shape, we aim to find the transformation consisting of a rotation $\mathbf{R}$ and a translation $\boldsymbol{t}$ which minimizes the following cost function:

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

The weights $w_{i,j}$ are 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 ($\mathbf{R}$, $\boldsymbol{t}$) that minimizes $E(\mathbf{R}, \boldsymbol{t})$ 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 $(\mathbf{R}, \boldsymbol{t})$ for minimizing Eq. (3). In the last iteration step, the point correspondences are assumed to be correct. Fig. 11 shows three frames of the alignment process. Besl et al. prove that the method terminates in a minimum (Besl and McKay, 1992). However, this theorem does not hold in our case, since we use a maximum tolerable distance $d_{\max}$ for associating the scan data. Such a threshold is required though, given that 3D scans overlap

only partially. Since the correct overlap is not known, using a threshold $d_{\max}$ resembles an estimation. Thus, the number of matched points in the iterations is not constant and Eq. (3) does not decrease monotonically.

In every iteration, the optimal transformation $(\mathbf{R}, \boldsymbol{t})$ has to be computed. Eq. (3) is reduced to

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

with $N = |D| = \sum_{i=1}^{|M|} \sum_{j=1}^{|D|} w_{i,j}$ and $D \subseteq \hat{D}, M \subseteq \hat{M}$ such that $\boldsymbol{m}_i$ corresponds to $\boldsymbol{d}_{f(\boldsymbol{m}_i)}$. The correspondences are stored in a vector $\boldsymbol{v} = \left((m_i, d_{f(\boldsymbol{m}_i)})\right)_i$ containing the point pairs, using a search function $f(\boldsymbol{m}_i)$ that returns the index to a point in D with minimal distance to $\boldsymbol{m}_i$.

Four direct methods are known to minimize Eq. (4) (Lorusso et al., 1995). In earlier work (Surmann et al., 2003) we used a quaternion based method (Besl and McKay, 1992), but the following one, based on singular value decomposition (SVD), is robust and easy to implement, thus we give a brief overview of the SVD-based algorithm. It was first published by Arun, Huang and Blostein (Arun et al., 1987). The difficulty of this minimization problem is to enforce the orthonormality of the matrix $\mathbf{R}$. The first step of the computation is to decouple the calculation of the rotation $\mathbf{R}$ from the translation $\boldsymbol{t}$ using the centroids of the points belonging to the matching, i.e.,

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

and

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

After substituting (5) and (6) into the error function, Eq. (4) becomes:

$$E(\mathbf{R}, \boldsymbol{t}) \propto \sum_{i=1}^{N} ||\boldsymbol{m}'_i - \mathbf{R}\boldsymbol{d}'_i||^2 \quad \text{with} \quad \boldsymbol{t} = \boldsymbol{c}_m - \mathbf{R}\boldsymbol{c}_d. \tag{7}$$

The registration calculates the optimal rotation by $\mathbf{R} = \mathbf{V}\mathbf{U}^T$. Hereby, the matrices $\mathbf{V}$ and $\mathbf{U}$ are derived by the singular value decomposition $\mathbf{H} = \mathbf{U}\mathbf{\Lambda}\mathbf{V}^T$ of a correlation matrix $\mathbf{H}$. This 3×3 matrix $\mathbf{H}$ is given by

$$\mathbf{H} = \sum_{i=1}^{N} \boldsymbol{d}'_i \boldsymbol{m}'^T_i = \begin{pmatrix} S_{xx} & S_{xy} & S_{xz} \\ S_{yx} & S_{yy} & S_{yz} \\ S_{zx} & S_{zy} & S_{zz} \end{pmatrix}, \tag{8}$$

with $S_{xx} = \sum_{i=1}^{N} m'_{ix} d'_{ix}$, $S_{xy} = \sum_{i=1}^{N} m'_{ix} d'_{iy}$, ... (Arun et al., 1987).

We proposed and evaluated algorithms to accelerate ICP, namely point reduction and approximate kd-trees (Nüchter et al., 2004; Surmann et al., 2003; Surmann et al., 2004), which are used here, too. They will be addressed in Detail in Sec. 4.

3.4 Loop Closing

After registration, the scene has to be correct and globally consistent. The method just described for aligning several 3D scans is called *pairwise matching*, i.e., the new scan is registered against a previous one. Alternatively, an *incremental matching* method is used, where 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. Closing loop detection and error diffusing avoid these problems and compute consistent scenes. They are described next.

A loop is closed in SLAM if the robot returns to a pose close to one where a previous scan was taken. If the 3D scans where perfect and pairwise or incremental matching would produce no errors, then there would be no need for matching the last against the first scan of a loop. In practice matching errors accumulate leading to inconsistent maps.

Our algorithm automatically detects a to-be-closed loop by registering the last acquired 3D scan with earlier acquired scans. Hereby we first create a hypothesis based on the maximum laser range and on the robot pose, so that the algorithm does not need to process all previous scans. Then we use the octree based method presented in Sec. 3.2 to revise the hypothesis. Finally, a loop is detected if registration is possible, i.e., the number of closest points exceeds a certain threshold. The computed registration error, i.e., the transformation ($\mathbf{R}$, $\boldsymbol{t}$), is distributed over all 3D scans in between. The respective part is weighted by the distance covered between the scans, i.e.,

$$c_i = \frac{\text{length of the path from start of the loop to scan pose } i}{\text{overall length of the loop}}$$

1. The translational part is calculated as $\boldsymbol{t}_i = c_i \boldsymbol{t}$.
2. Of the three possibilities of representing rotations, namely, orthonormal matrices, quaternions and Euler angles, quaternions are best suited for our interpolation task. The problem with matrices is to enforce orthonormality and Euler angles show Gimbal locks (Matrix FAQ, 1997). A quaternion as used in computer graphics is the 4 vector $\boldsymbol{q}$. Given a rotation as matrix $\mathbf{R}$, the corresponding quaternion $\boldsymbol{q}$ is calculated as follows:

$$\boldsymbol{q} = \begin{pmatrix} q_0 \\ q_x \\ q_y \\ q_z \end{pmatrix} = \begin{pmatrix} \frac{1}{2}\sqrt{\text{trace}\,(\mathbf{R})} \\ \frac{1}{2}\frac{r_{3,3}-r_{3,2}}{\sqrt{\text{trace}(\mathbf{R})}} \\ \frac{1}{2}\frac{r_{2,1}-r_{2,3}}{\sqrt{\text{trace}(\mathbf{R})}} \\ \frac{1}{2}\frac{r_{1,2}-r_{1,1}}{\sqrt{\text{trace}(\mathbf{R})}} \end{pmatrix}, \quad \text{with the elements } r_{i,j} \text{ of } \mathbf{R}.^1 \tag{9}$$

   The quaternion describes a rotation by an axis $\boldsymbol{a} \in \mathbb{R}^3$ and an angle θ that are computed by

[1] If trace $(\mathbf{R})$ (sum of the diagonal terms) is zero, the above calculation has to be altered: Iff $r_{1,1} > r_{2,2}$ and

$$\boldsymbol{a} = \begin{pmatrix} \frac{q_x}{\sqrt{1-q_0^2}} \\ \frac{q_y}{\sqrt{1-q_0^2}} \\ \frac{q_z}{\sqrt{1-q_0^2}} \end{pmatrix} \qquad \text{and} \qquad \theta = 2 \arccos q_o.$$

The angle θ is distributed over all scans using the factor c_i and the resulting matrix is derived as (Matrix FAQ, 1997):

$$\mathbf{R}_i = \begin{pmatrix} \cos(c_i\theta) + a_x^2(1-\cos(c_i\theta)) & a_z \sin(c_i\theta) + a_x a_y(1-\cos(c_i\theta)) & -a_y \sin(c_i\theta) + a_x a_z(1-\cos(c_i\theta)) \\ -a_z \sin(c_i\theta) + a_x a_y(1-\cos(c_i\theta)) & \cos(c_i\theta) + a_y^2(1-\cos(c_i\theta)) & -a_x \sin(c_i\theta) + a_y a_z(1-\cos(c_i\theta)) \\ a_y \sin(c_i\theta) + a_x a_z(1-\cos(c_i\theta) & -a_x \sin(c_i\theta) + a_y a_z(1-\cos(c_i\theta)) & \cos(c_i\theta) + a_z^2(1-\cos(c_i\theta)) \end{pmatrix}. \tag{10}$$

3.5 Model Refinement

Pulli presents a semi-automatic registration method that minimizes the global error and avoids inconsistent scenes (Pulli, 1999). The registration of one scan is followed by registering all neighboring scans such that the global error is distributed. Other matching approaches with global error minimization have been published, e.g., (Benjemaa and Schmitt, 1997; Eggert et al., 1998). Benjemaa and Schmitt establish point-to-point correspondences first and then use randomized iterative registration on a set of surfaces. Eggert et al. compute motion updates, i.e., a transformation $(\mathbf{R}, \boldsymbol{t})$, using force-based optimization, with data sets considered as connected by groups of springs.

Based on the idea of Pulli we designed the relaxation method *simultaneous matching* (Surmann et al., 2003). The first scan is the master scan S_0 and determines the coordinate

---

$r_{1,1} > r_{3,3}$ then,

$$\boldsymbol{q} = \begin{pmatrix} \frac{1}{2}\frac{r_{2,3}-r_{3,2}}{\sqrt{1+r_{1,1}-r_{2,2}-r_{3,3}}} \\ \frac{1}{2}\sqrt{1+r_{1,1}-r_{2,2}-r_{3,3}} \\ \frac{1}{2}\frac{r_{1,2}+r_{2,1}}{\sqrt{1+r_{1,1}-r_{2,2}-r_{3,3}}} \\ \frac{1}{2}\frac{r_{3,1}+r_{1,3}}{\sqrt{1+r_{1,1}-r_{2,2}-r_{3,3}}} \end{pmatrix}, \text{if } r_{2,2} > r_{3,3}\ \boldsymbol{q} = \begin{pmatrix} \frac{1}{2}\frac{r_{3,1}-r_{1,3}}{\sqrt{1-r_{1,1}+r_{2,2}-r_{3,3}}} \\ \frac{1}{2}\frac{r_{1,2}+r_{2,1}}{\sqrt{1-r_{1,1}+r_{2,2}-r_{3,3}}} \\ \frac{1}{2}\sqrt{1-r_{1,1}+r_{2,2}-r_{3,3}} \\ \frac{1}{2}\frac{r_{2,3}+r_{3,2}}{\sqrt{1-r_{1,1}+r_{2,2}-r_{3,3}}} \end{pmatrix},$$

otherwise the quaternion $\boldsymbol{q}$ is calculated as

$$\boldsymbol{q} = \begin{pmatrix} \frac{1}{2}\frac{r_{1,2}-r_{2,1}}{\sqrt{1-r_{1,1}-r_{2,2}+r_{3,3}}} \\ \frac{1}{2}\frac{r_{3,1}+r_{1,3}}{\sqrt{1+r_{1,1}-r_{2,2}-r_{3,3}}} \\ \frac{1}{2}\frac{r_{2,3}+r_{3,2}}{\sqrt{1-r_{1,1}-r_{2,2}+r_{3,3}}} \\ \frac{1}{2}\sqrt{1-r_{1,1}-r_{2,2}+r_{3,3}} \end{pmatrix}.$$

system. It is fixed. The following three steps refine the model by minimizing the global scan matching error, after a queue is initialized with the first scan of the closed loop (cf. Algorithm 1):

1. Pop the first 3D scan from the queue as the current one.

2. A set of neighbors (set of all scans that overlap with the current scan) is calculated. This set of neighbors forms one point set M. The current scan forms the data point set D and is aligned with the ICP algorithms, if the current scan is not the master scan S_0. One scan overlaps with another iff more than p corresponding point pairs exist. In our implementation, $p = 250$.

3. If the current scan changes its location by applying the transformation (translation or rotation) in step 2, e.g., the displacement is larger than 5 cm, then each single scan of the set of neighbors that is not in the queue is added to the end of the queue. If the queue is empty, terminate; else continue at step 1.

In contrast to Pulli's approach, our method is totally automatic and no interactive pairwise alignment has to be done. Furthermore the point pairs are not fixed (Pulli, 1999). Our algorithm, the function *align_scans()* (line 11) recomputes the point correspondences, whereas Pullis algorithm uses correspondences established in an initialization step. The accumulated alignment error is spread over the whole set of acquired 3D scans. This diffuses the alignment error equally over the set of 3D scans (Surmann et al., 2004).

Algorithm 1 Model Refinement

```
 1: scan_queue.push(S_closed_loop)                    /* First scan of the closed loop */
 2: while scan_queue ≠ ∅ do
 3:    s_current = scan_queue.pop()
 4:    meta_scan = ∅
 5:    for i = 0 to max_number_of_scans do
 6:       p = number_of_closest_points(S_current, S_i)
 7:       if p ≥ 250 then
 8:          meta_scan.push(S_i)
 9:       end if
10:    end for
11:    (ΔP, transformation) = align_scans(s_current, meta_scan)
12:    if S_current ≠ S_0 then
13:       apply transformation on S_current
14:    end if
15:    if ΔP > ε then
16:       scan_queue = scan_queue ∪ meta_scan
17:    end if
18: end while
```

4 Performance Issues

The five steps in our SLAM algorithms have different computational costs. In our experiments, we acquire usually 3D scans with 20,000 up to 300,000 3D data points. While the first step (odometry extrapolation) is computed instantaneously, the octree based heuristic, applied naively, would need up to 2 seconds for calculating the two octrees and the rough alignment of the scans. Since computing octrees is done in logarithmic time, the influence of larger data sets is negligible. The loop closing step (step four) has similar computational costs, since we use the octree heuristic again.

Most computational time is needed in the scan matching step (step three) and in the model refinement (step five). While the model refinement can easily be done offline, i.e., after the robot has finished the data acquisition, the scan matching is an essential part of the mapping procedure. We have a number of methods available to reduce significantly the computational costs, namely point reduction, kd-trees, approximate kd-trees and cached kd-trees.

4.1 Point Reduction

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.

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. For point reduction, multiple data points located close together (Euclidian distance) are replaced by their mean, using a standard reduction filter. 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 in each scan slice 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).

Data reduction for the ICP algorithm is done using the proposed filters. 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 is collected from every third scan slice. This fast vertical reduction yields a good surface description (cf. Fig. 2).

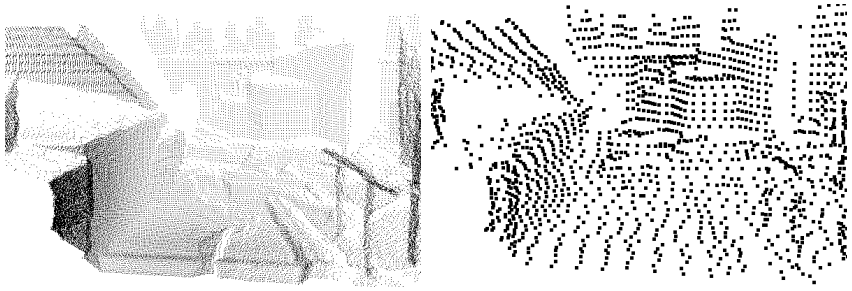

Figure 2: left: A view of a 3D scene (66785 3D data points). Right: Subsampled version (points have been enlarged, 6700 data points). To visualize the scanned 3D data, a viewer program has been implemented. The task of this program is to project a 3D scene to the image plane, i.e., the monitor, such that the data can be drawn and inspected from every perspective.

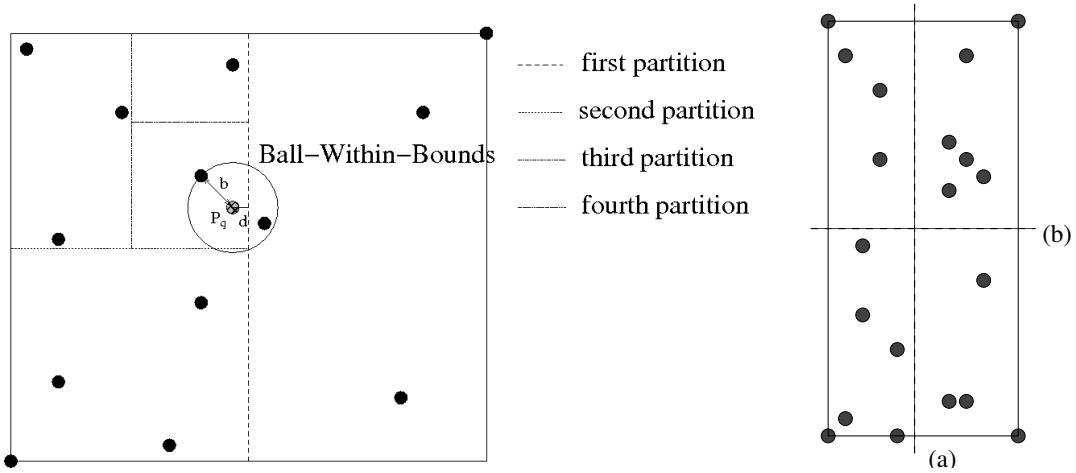

Figure 3: Left: Recursive construction of a kd-tree. If the query consists of point $\boldsymbol{p}_q$, kd-tree search has to backtrack to the tree root to find the closest point. Right: Partitioning of a point cloud. Using the cut (b) rather than (a) results in a more compact partition and a smaller probability of backtracking (Friedman et al., 1977).

4.2 *k*d-trees

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 leaves provide a complete disjunct partition of the points. These leaves are called buckets (cf. Fig. 3). Furthermore, every node contains the limits of the represented point set.

4.2.1 Searching *k*d-trees

A kd-tree is searched recursively for a closest point of a given query 3D point. The 3D point needs to be compared with the separating plane in order to decide on which side the search must continue. This procedure is executed until the leaves are reached. There, the algorithm

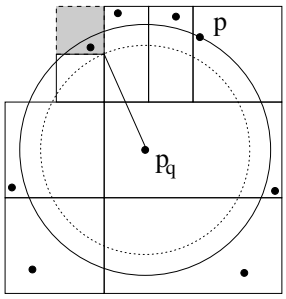

Figure 4: The $(1+\varepsilon)$-approximate nearest neighbor. The solid circle denotes the ε environment of $\boldsymbol{p}_q$. The search algorithm need not analyze the gray cell, since $\boldsymbol{p}$ satisfies the approximation criterion. (Fig. adapted from (Arya and Mount, 1993))

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. 3 shows a backtracking case, where the algorithm has to go back to the root. The test is known as ball-within-bounds test (Bentley, 1975; Friedman et al., 1977; Greenspan and Yurick, 2003).

4.2.2 The Optimized *k*d-tree

The objective of optimizing kd-trees is to reduce the expected number of visited leaves. Three parameters are adjustable, namely, the direction and position of the split axis as well as the maximal number of points in the buckets. Splitting the point set at the *median* ensures that every kd-tree entry has the same probability. The median can be found in linear time, thus computing the mean does not account for the time complexity of constructing the tree. Furthermore, the split axis should be oriented *perpendicular* to the longest axis to minimize the amount of backtracking (see Fig. 3). Besides these results on complexity, Friedman and colleagues prove that a bucket size of 1 is optimal (Friedman et al., 1977). Nevertheless, in practice it turned out that a slightly larger bucket size is faster.

4.3 Approximate *k*d-tree Search

S. Arya and D. Mount introduce the following notion for approximating the nearest neighbor in kd-trees (Arya and Mount, 1993): 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

$$||\boldsymbol{p} - \boldsymbol{q}|| \leq (1+\varepsilon)\,||\boldsymbol{p}^* - \boldsymbol{q}||\,,$$

where $\boldsymbol{p}^*$ denotes the true nearest neighbor, i.e., $\boldsymbol{p}$ has a maximal distance of ε to the true nearest neighbor (cf. Fig. 4). Using this notation, in every step the algorithm records the closest point $\boldsymbol{p}$. The search terminates if the distance to the unanalyzed leaves is larger than

$$\left|\left|\boldsymbol{p}_q - \boldsymbol{p}\right|\right| /(1+\varepsilon).$$

In the extreme case, the kd-tree does not implement any backtracking. To evaluate the quality of the scan matching, we acquired two 3D scans and measured the pose shift by a

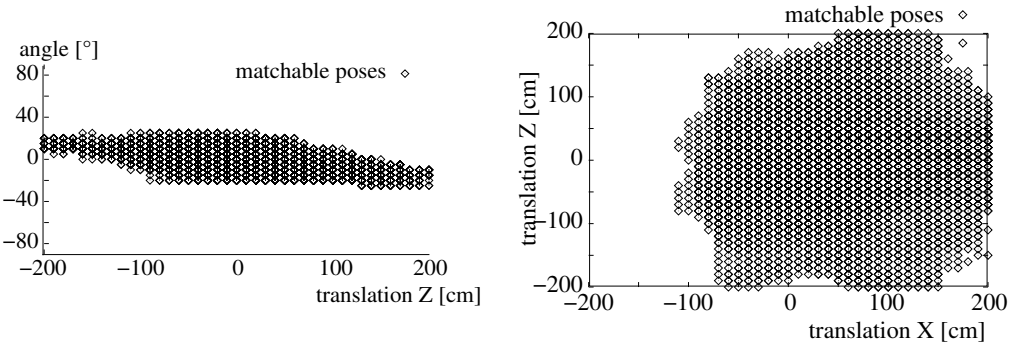

Figure 5: The poses (x, z, θ_y) from which a correct alignment of two 3D scans is possible. Similar results for "matchable" poses have been obtained for different values of ε (Nüchter et al., 2005).

reference system, i.e., a meter rule. Fig. 5 shows the starting poses from which a correct scan matching is possible. We conclude that the approximation does not influence the scan matching significantly, due to the large number of points used and the iterative nature of the algorithm. The running time of ICP scan matching decreases to roughly 75% in case of approximate kd-tree search. For a detailed evaluation see (Nüchter et al., 2005). However, the next method outperforms the approximate kd-tree search without the need of any approximation.

4.4 Cached *k*d-trees

4.4.1 The Cached *k*d-tree Search

kd-trees with caching contain, in addition to the limits of the represented point set and to the two child node pointers, one pointer to the predecessor node. The root node contains a null pointer. During the recursive construction of the tree, this information is available and no extra computations are required.

For the ICP algorithm, we distinguish between the first and the following iterations: In the first iteration, a normal kd-tree search is used to compute the closest points. However, the return function of the tree is altered, such that in addition to the closest point, the pointer to the leaf containing the closest point is returned and stored in the vector of point pairs. This supplementary information forms the cache for future look-ups.

In the following iterations, these stored pointers are used to start the search. If the query point is located in the bucket, the bucket is searched and the ball-within-bounds test (cf. Sec. 4.2.1) applied. Backtracking is started, iff the ball lies not completely within the bucket. If the query point is not located within the bucket, then backtracking is started, too. Since the search is started in the leaf node, explicit backtracking through the tree has to be implemented using the pointers to the predecessing nodes (see Fig. 6). Algorithm 2 summarizes the ICP with cached kd-tree search.

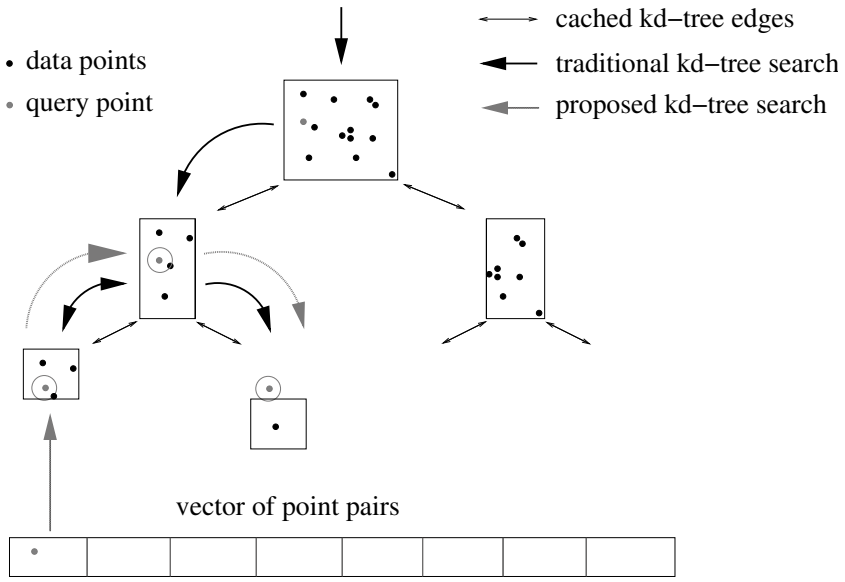

Figure 6: Schematic description of the proposed search method: Instead of closest point searching from the root of the tree to the leaves that contain the data points, a pointer to the leaves is cached. In the second and following ICP iteration, the tree is searched backwards. The vector of point pairs memorizes the starting point of the search and therefore serves as a cache.

Algorithm 2 ICP with cached kd-tree search

1: **for** $i = 0$ to *maxIterations* **do**
2: **if** $i = 0$ **then**
3: **for all** $\boldsymbol{d}_j \in D$ **do**
4: search kd-tree of set M top down for point $\boldsymbol{d}_j$
5: $\boldsymbol{v}_i = \left(\boldsymbol{d}_j, \boldsymbol{m}_{f(\boldsymbol{d}_j)}, \text{ptr\_to\_bucket}(\boldsymbol{m}_{f(\boldsymbol{d}_j)})\right)$
6: **end for**
7: **else**
8: **for all** $\boldsymbol{d}_j \in D$ **do**
9: search kd-tree of set M, bottom up, for point $\boldsymbol{d}_j$ using ptr_to_bucket($\boldsymbol{m}_{f(\boldsymbol{d}_j)}$)
10: $\boldsymbol{v}_i = (\boldsymbol{d}_j, \boldsymbol{m}_{f(\boldsymbol{d}_j)}, \text{ptr\_to\_bucket } \boldsymbol{m}_{f(\boldsymbol{d}_j)})$
11: **end for**
12: **end if**
13: calculate transformation $(\mathbf{R}, \boldsymbol{t})$ that minimizes the error function Eq. (4)
14: apply transformation on data set D
15: **end for**

4.4.2 Performance of Cached *k*d-tree Search

The proposed ICP variant uses exact closest point search. In contrast to the previously discussed approximate kd-tree search for ICP algorithms (Greenspan and Yurick, 2003; Nüchter et al., 2005), registration inaccuracies or errors due to approximation cannot occur.

Friedman et al. prove that searching for closest points using kd-trees needs logarithmic time

(Friedman et al., 1977), i.e., the amount of backtracking is independent of the number of stored points in the tree. Since the ICP algorithm iterates the closest point search, the performance derives to $\mathcal{O}(I\ |D|\ \log|M|)$, with I the number of iterations. Note: Brute-force ICP algorithms have a performance of $\mathcal{O}(I\ |D|\ |M|)$.

The proposed cached kd-tree search needs $\mathcal{O}((I + |M|)\ |D|)$ time in the best case. This performance is reached if constant time is needed for backtracking, resulting in $|D| \log |M|$ time for constructing the tree, and $I \cdot |D|$ for searching in case no backtracking is necessary. Obviously the backtracking time depends on the computed ICP transformation $(\mathbf{R}, \boldsymbol{t})$. For small transformations the time is nearly constant.

Cached kd-tree search needs $\mathcal{O}(|D|)$ extra memory for the vector $\boldsymbol{v}$, i.e., for storing the pointers to the tree leaves. Furthermore, additional $\mathcal{O}(|M|)$ memory is needed for storing the backwards pointers in the kd-tree.

Fig. 7 shows the following results of a detailed evaluation of the proposed cached kd-tree search algorithm:

1. The performance of the cached kd-tree search depending on a change of the bucket size was tested: For small bucket sizes, the speed-up is larger (Fig. 7, top left). This behavior originates from the increasing time needed to search larger buckets.
2. The search time per iteration was recorded during the experiments (Fig. 7, top right). For the first iteration the search times are equal, since cached kd-tree search uses conventional kd-tree search to create the cache. In the following iterations, the search time drops significantly and remains nearly constant. The conventional kd-tree search increases in speed, too. Here, the amount of backtracking is reduced due to the fact that the calculated transformations $(\mathbf{R}, \boldsymbol{t})$ are getting smaller.
3. The number of points to register influences the search time. With increasing number of points, the positive effect of caching algorithms becomes more and more significant (Fig. 7, bottom left).
4. The overall performance of the ICP algorithm depends both on the search time and on the construction time of the tree. However, the construction time of the trees seems to be negligible. In addition, a comparison with a reference implementation shows the effective implementation. As reference implementation the software from the papers (Arya and Mount, 1993; Arya et al., 1998) was used (Fig. 7, bottom right).

5 Experiments and Results

5.1 Hardware Used in our Experiments

The 3D laser range finder is built on the basis of a SICK 2D range finder by extension with a mount and a small servomotor (Fig. 8) (Surmann et al., 2003). The 2D laser range finder is attached at the center of rotation to the mount for achieving a controlled pitch motion with a standard servo.

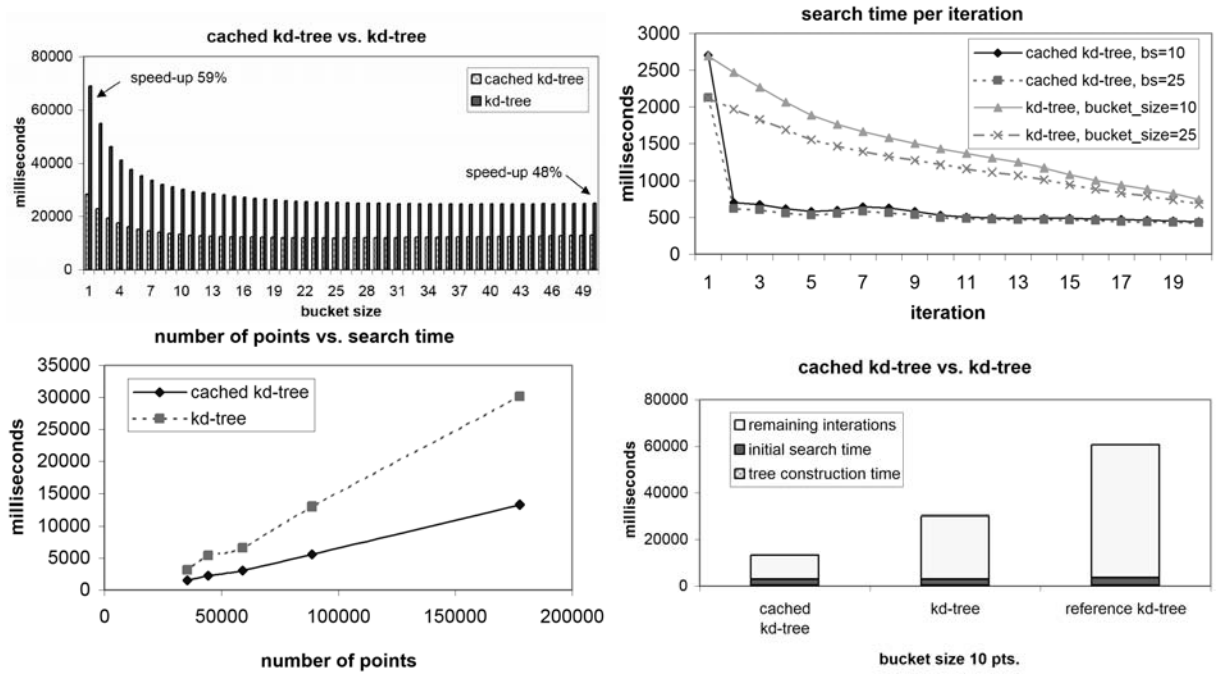

Figure 7: Top Left: Achieved speedups of cached kd-tree search compared to traditional kd-tree search for an ICP based registration of two point sets. Top Right: Search time per iteration for bucket sizes 10 and 25. Bottom Left: Time consumption per ICP iteration. Bottom right: Overall comparison of the algorithms and a reference kd-tree implementation (Arya and Mount, 1993).

Figure 8: Kurt3D in a natural environment. Left to right: Lawn, forest track, pavement.

The area of up to 180°(h) × 120°(v) is scanned with different horizontal (181, 361, 721) and vertical (128, 225, 420, 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 181 × 256 data points needs 3.4 seconds. Scanning the environment with a mobile robot is done in a stop-scan-go fashion.

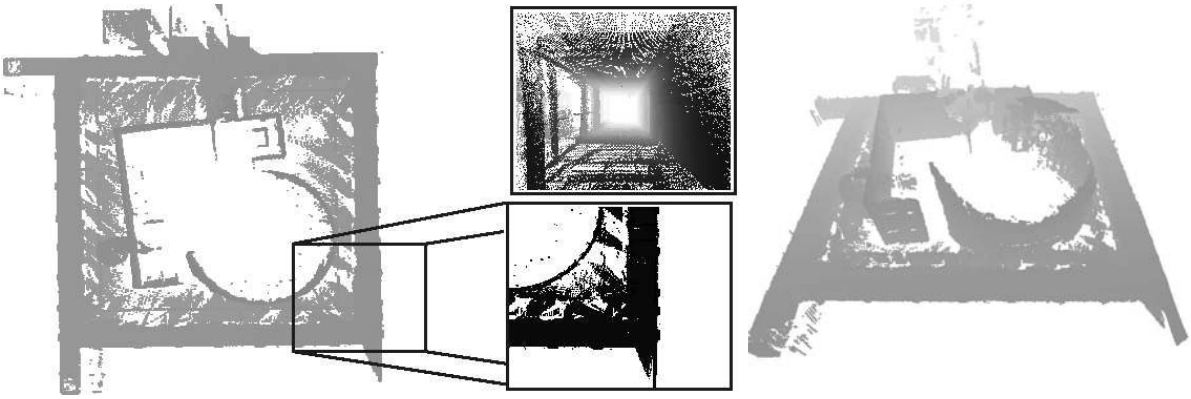

Figure 9: 3D digitalization of the International Conference and Research Center Schloss Dagstuhl. Left: 3D point cloud (top view). Right: 3D view.

The mobile robot Kurt3D in its outdoor version (Fig. 8) is a mobile robot with a size of 45 cm (length) × 33 cm (width) × 29 cm (height) and a weight of 22.6 kg. Two 90 W motors are used to power the 6 skid-steered wheels, whereas the front and rear wheels have no tread pattern to enhance rotating. The core of the robot is a Pentium-Centrino-1400 with 768 MB RAM and Linux. An embedded 16-Bit CMOS microcontroller is used to control the motor.

5.2 Full 6D SLAM in a planar environment

We did apply the algorithms to a data set acquired at Schloss Dagstuhl. It contains 84 3D scans, each with 81225 (361 × 225), 3D data points, in a large, i.e., ∼240 m, closed loop. The average distance between consecutive 3D scans was 2.5 m. Fig. 9 shows the built 3D map.

We have also tested the algorithms on a 2D data set, computed from horizontal scan slices. This allows us to compare full 6D SLAM with planar 2D mapping (cf. Tab. 1). Fig. 10 shows the map. There are noticeable errors in the 2D alignment. The 2D scans do not provide enough structure for correct alignment. Many approaches bypass this problem by scanning the environment more often. However, the 3D data is much richer in information, therefore 3D scan taken at sparse discrete locations are matched correctly (cf. Fig. 9).

5.3 Full 6D SLAM in an Indoor/Outdoor Environment

The proposed algorithms have been applied to a data set acquired at the robotic lab in Birlinghoven. 32 3D scans, each containing 302820 (721 × 420) range data points, were taken. The robot had to cope with a height difference between two buildings of 1.05 meter, covered, on the one hand, by a sloped driveway in open outdoor terrain, and, on the other hand, by a ramp of 12° inside the building. The 3D model was computed after acquiring all 3D scans.

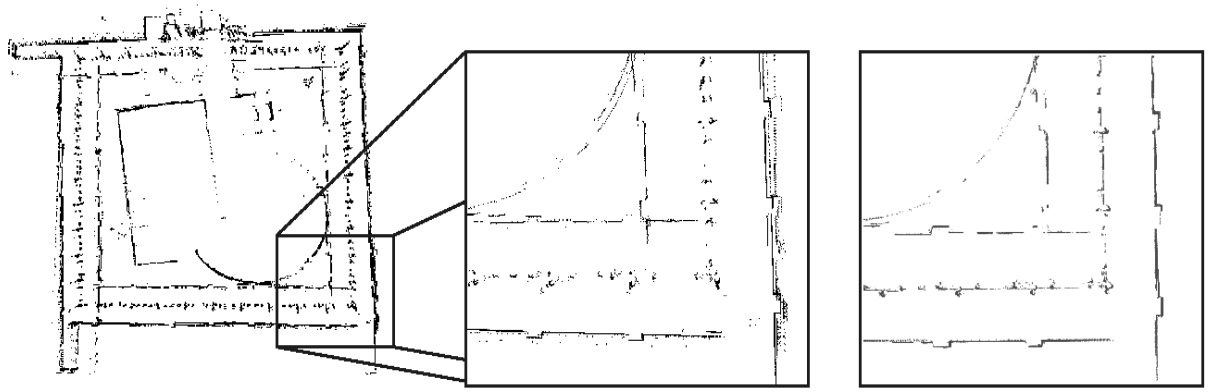

Figure 10: 2D digitalization of the environment with alignment problems at the wall on the right. Right: For comparison, the same closeup area of a horizontal slice from the generated 3D map (Fig. 9).

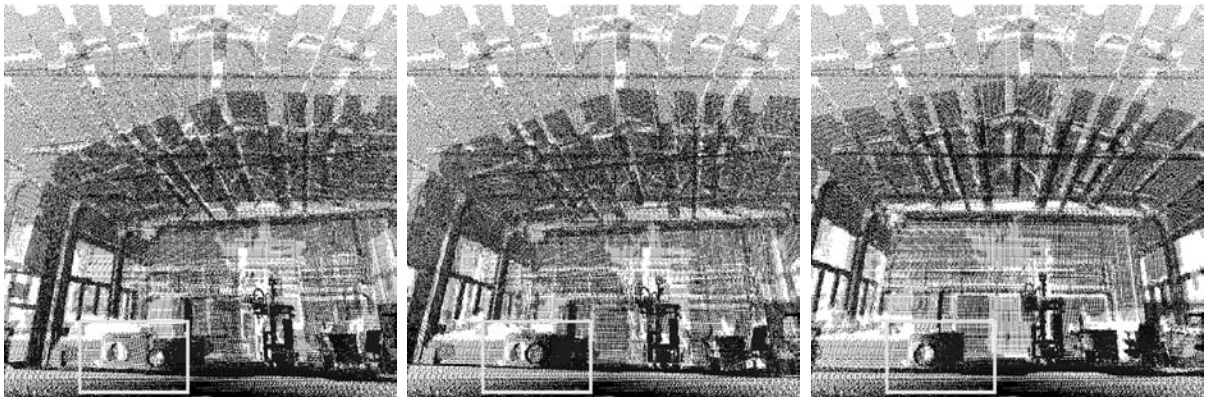

Figure 11: Scan matching of the IAIS robotic lab. Left: Initial pose of two 3D scans. Middle: Pose after five ICP iterations. Right: Final alignment.

Fig. 11, 12 and 13 show rendered 3D scans. The latter figure presents the final model with the closed loop. Please refer to the website `http://kos.informatik.uni-osnabrueck.de/download/6Dpre/` for a computed animation and video through the scanned 3D scene.

In this data set, we analyzed the performance of the ICP scan matching. Details about this analysis can be found in (Surmann et al., 2004) and (Nüchter, 2006). For ICP registration, the error tolerance for the initial estimation, i.e., the robot's self localization, is about 1 meter in (x, y, z) position and about 15° in the orientation. These conditions can be easily met using a cheap odometer and the presented heuristic for initial estimates for the ICP algorithm.

5.4 Full 6D SLAM in an Outdoor Environment

The following experiment has been made at the campus of Schloss Birlinghoven with Kurt3D. Fig. 14 (left) shows the scan point model of the first scans in top view, based on odometry

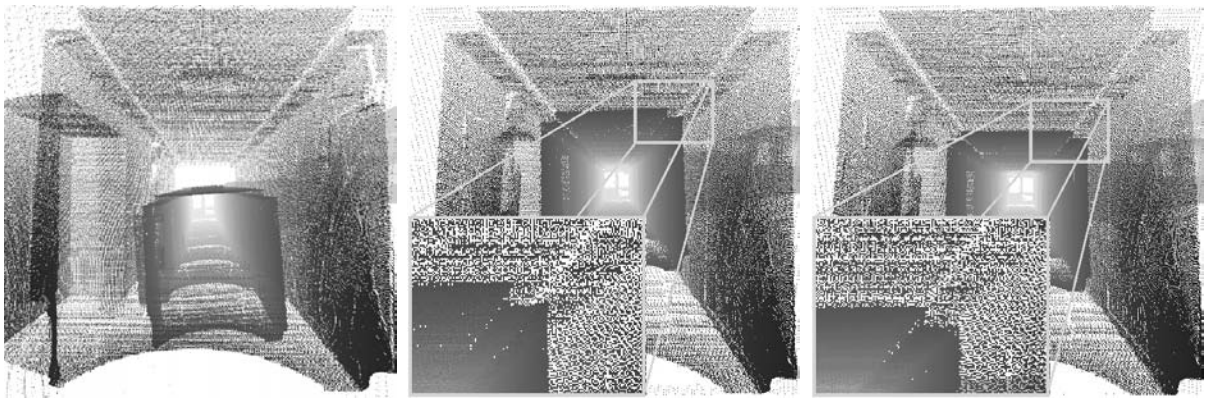

Figure 12: Left: Initial poses of 3D scans when closing the loop. Middle: Poses after detecting the loop and equally sharing the resulting error. Right: Final alignment after error diffusion with correct alignment of the edge structure at the ceiling.

Figure 13: The closed loop with a top viewing position and orthogonal projection. The distance d measured in the point cloud model is 2096 cm, measured by meter rule 2080 cm. The right part demonstrates the change in elevation. Top right: A ramp connecting two buildings is correctly modeled (height difference 1.05 m). The ramp connects the basement of the left building with the right building. Bottom right: Outdoor environment modeling of the downhill part.

only. The first part of the robot's run, i.e., driving on asphalt, contains a systematic drift error, but driving on lawn shows more stochastic characteristics. The right part shows the first 62 scans, covering a path length of about 240 m. The heuristic described in Sec. 3.2 has been applied and the scans have been matched. The open loop is marked with a red rectangle.

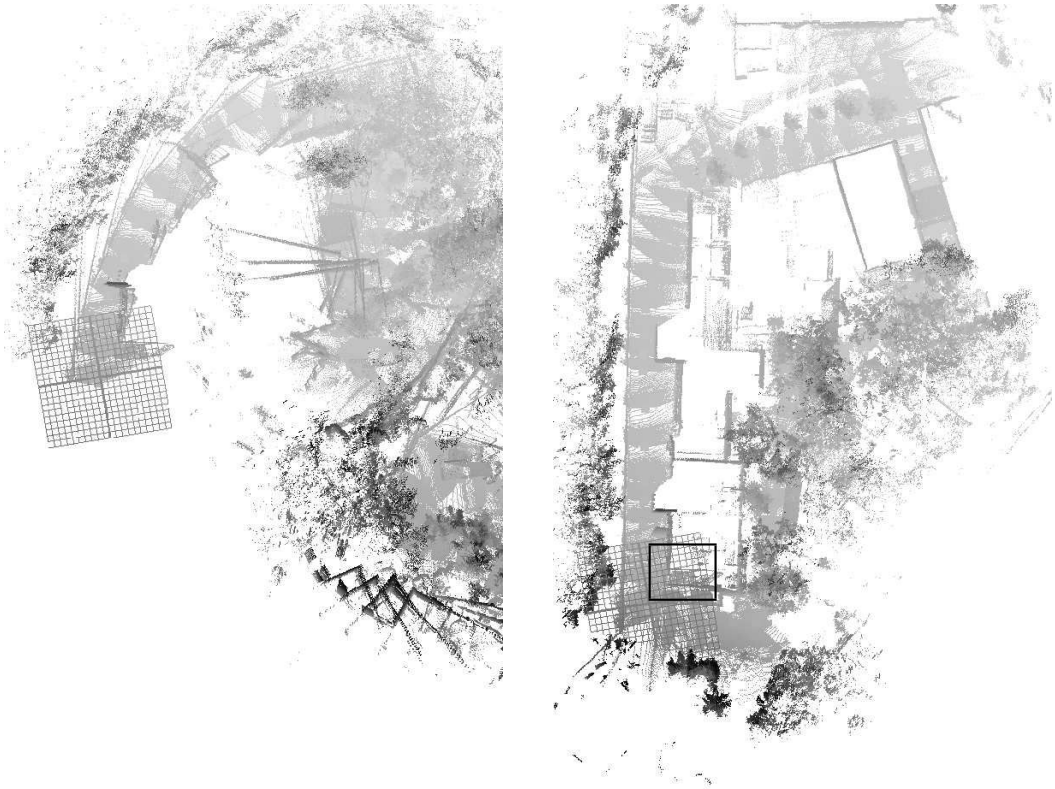

Figure 14: 3D model of an experiment to digitize part of the campus of Schloss Birlinghoven campus (top view). Left: Registration based on odometry only. Right: Model based on incremental matching right before closing the loop, containing 62 scans each with approx. 100000 3D points. The grid at the bottom denotes an area of 20×20 m^2 for scale comparison. The 3D scan poses are marked by grey points.

At that point, the loop is detected and closed. More 3D scans have then been acquired and added to the map. Fig. 15 (left and right) shows the model with and without global relaxation to visualize its effects. The relaxation is able to align the scans correctly even without explicitly closing the loop. The best visible difference is marked by a rectangle. The final map in Fig. 15 contains 77 3D scans, each consisting of approx. 100000 data points (361 × 275). Fig. 16 shows two detailed views, before and after loop closing. The bottom part of Fig. 15 displays an aerial view as ground truth for comparison. Table 2 compares distances measured in the photo and in the 3D scene, at corresponding points (taking roof overhangs into account). The lines in the photo have been measured in pixels, whereas real distances, i.e., the (x, z)-values of the points, have been used in the point model. Considering that pixel distances in mid-resolution non-calibrated aerial image induce some error in ground truth, the correspondence show that the point model at least approximates reality quite well.

Mapping would fail without first calculating the heuristic initial estimations for ICP scan matching, since ICP would likely converge to an incorrect minimum. The resulting 3D map would be some mixture of Fig. 14 (left) and Fig. 15 (right).

Table 2: Length ratio comparison of measured distances in the aerial photographs with distances in the point model as shown in Fig. 15.

1st line	2nd line	ratio in aerial views	ratio in point model	deviation
AB	BC	0.683	0.662	3.1%
AB	BD	0.645	0.670	3.8%
AC	CD	1.131	1.141	0.9%
CD	BD	1.088	1.082	0.5%

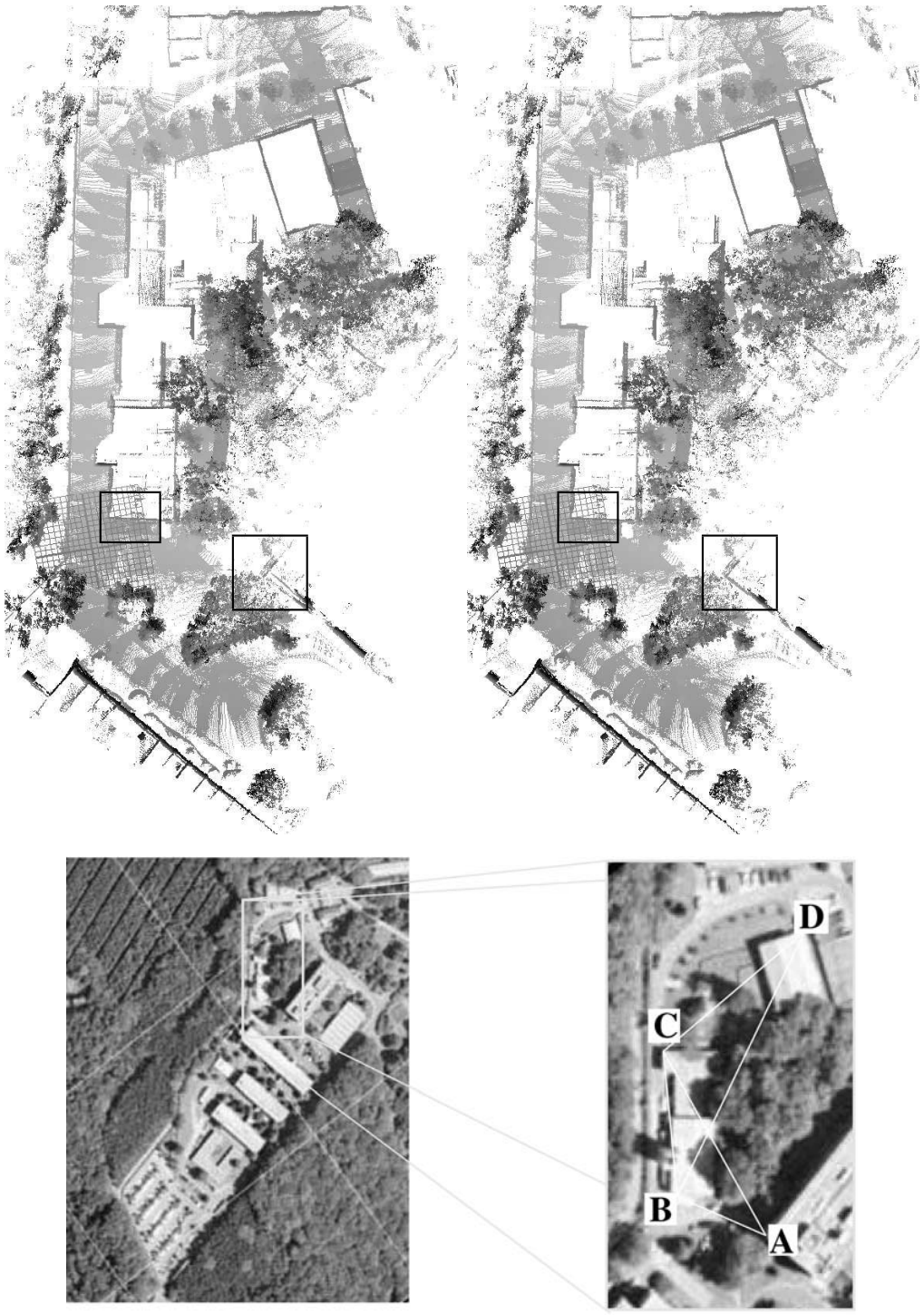

Figure 15: Top left: Model with loop closing, but without global relaxation. Differences to Fig. 14 right and to the right image are marked. Top right: Final model of 77 scans with loop closing and global relaxation. Bottom: Aerial view of the scene. The points A–D are used as reference points in the comparison in Table 2.

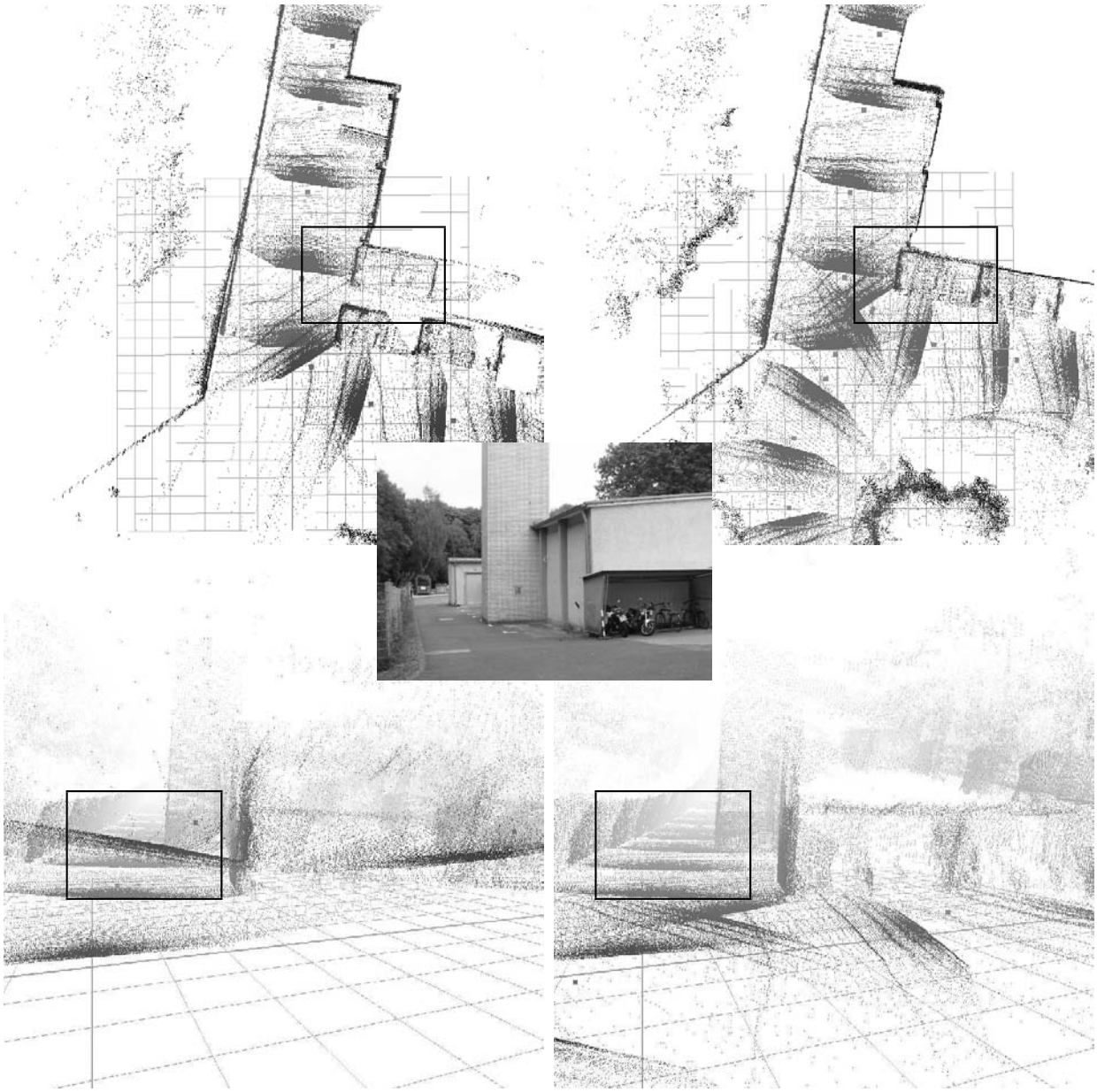

Figure 16: Closeup view of the 3D model of Fig. 15. Left: Model before loop closing. Right: After loop closing, global relaxation and adding further 3D scans. Top: Top view. Bottom: Front view.

Fig. 17 shows three views of the final model. These model views correspond to the locations of Kurt3D in Fig. 8. An updated robot trajectory has been plotted into the scene. Thereby, we assign every 3D scan that part of the trajectory which leads from the previous scan pose to the current one. Since scan matching did align the scans, the trajectory initially has gaps after the alignment (see Fig. 18).

Figure 18: The trajectory after mapping shows gaps, since the robot poses are corrected at 3D scan poses.

We calculate the transformation $(\mathbf{R}, \boldsymbol{t})$ that maps the last pose of such a trajectory patch to the starting pose of the next patch. This transformation is then used to correct the trajectory patch by distributing the transformation as described in Sec. 3.4. In this way the algorithm computes a continuous trajectory. An animation of the scanned area is available at `http://kos.informatik.uni-osnabrueck.de/download/6Doutdoor/`. The video shows the scene along the trajectory as viewed from about 1 m above Kurt3D's actual position.

The 3D scans were acquired within one hour by tele-operation of Kurt3D. Scan registration and closed loop detection took only about 10 minutes on a Pentium-IV-2800 MHz, while we did run the global relaxation for 2 hours. However, computing the flight-thru-animation took about 3 hours, rendering 9882 frames with OpenGL on consumer hardware.

5.5 Stress Tests – RoboCup Rescue

Our 3D mapping algorithms have been tested in various experiments. We participate in RoboCup Rescue competitions on a regular basis. RoboCup is an international joint project to promote AI, robotics and related fields. It is an attempt to foster AI and intelligent robotics research by providing standard problems where a wide range of technologies can be integrated and examined. Besides the well-known RoboCup soccer leagues, the Rescue

Figure 17: Detailed views of the resulting 3D model, corresponding to the robot locations of Fig. 8.

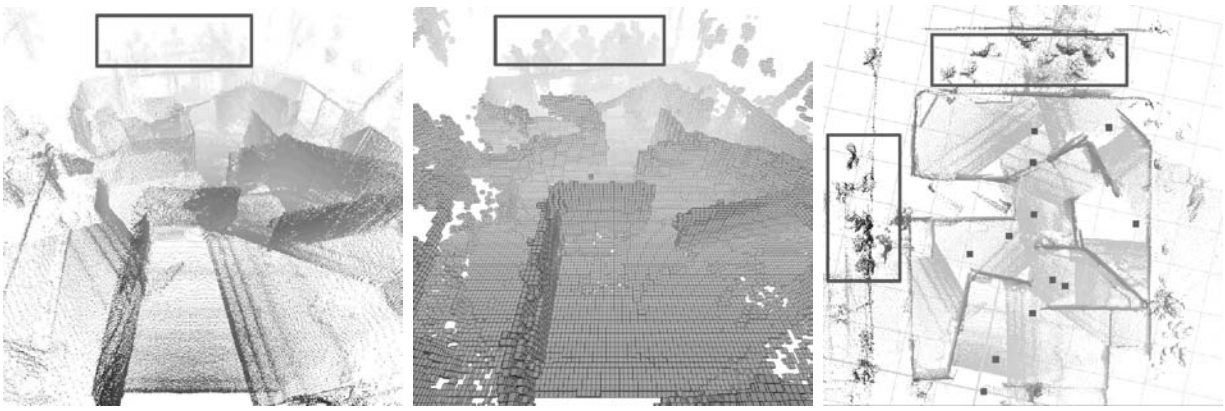

Figure 19: 3D maps of the yellow RoboCup arena. The 3D scans include spectators that are marked with a rectangle. Left: Mapped area as 3D point cloud. Middle: Voxel (volume pixel) representation of the 3D map. Right: Mapped area (top view). The points on the ground have been colored in light grey. The 3D scan positions are marked with squares. A 1 m^2 grid is superimposed. Following the ICP scan matching procedure, the first 3D scan defines the coordinate system and the grid is rotated.

league is getting increasing attention. Its real-life background is the idea of developing mobile robots that are able to operate in earthquake, fire, explosive and chemical disaster areas, helping human rescue workers to do their jobs. A fundamental task for rescue robots is to find and report injured persons. To this end, they need to explore and map the disaster site and inspect potential victims and suspicious objects. The RoboCup Rescue Contest aims at evaluating rescue robot technology to speed up the development of working rescue and exploration systems (NIST, 2007).

These kinds of competitions allow us to measure the level of system integration and the engineering skills of the teams to be evaluated. It makes high demands on the reliability of the algorithms, since one cannot redo the experiments. A total of 21 robot runs were performed by Kurt3D in the World Championships over the last three years. One major subgoal of such a rescue mission is to create a map of the unstructured environment during the mission time. The test field is a square with 6 meter long sides. Detailed maps of the environment have been presented to the referees. Fig. 19 shows one such map. With the superimposed grid in Fig. 19 the referees evaluated the maps facilely.

In addition to the RoboCup Rescue competitions, the proposed algorithms have also been tested at the European Land Robotics Trial, ELROB (FGAN, 2007). Please refer to `http://kos.informatik.uni-osnabrueck.de/download/Lisbon_RR/` and `http://kos.informatik.uni-osnabrueck.de/download/elrob2006/` for some results.

5.6 Benchmarking Mapping Results

Bechmarking experiments are used for measuring the objective performance of a dedicated algorithm. In the past, many researchers published their results in the Radish (The Robotics

Data Set Repository) repository (Howard and Roy, 2006). These data sets are accompanied by maps depicted as figures. Most researchers aimed at creating consistent maps. Recently, on the theoretical side of SLAM, Bailey et al. proves that EKF-SLAM fails in large environments (Bailey et al., 2006a) and FastSLAM is inconsistent as a statistical filter: It always underestimates its own error in the medium to long-term (Bailey et al., 2006b). Besides focusing on these consistency issues, little effort at correctness has been made in the SLAM community.

Testing algorithms and heuristics for objective correctness includes providing ground truth data. In computer vision research, it is a common technique to provide hand-labeled ground truth images and algorithms that calculate performance metrics. Up to now, such a performance metric is missing for SLAM algorithms. In this paper, we choose a sketchy comparison with uncalibrated aerial images. In ongoing work, we provide a novel method for evaluating SLAM algorithms applied to large-scale problems based on given ground truth maps and a Monte Carlo localization in these maps (Wulf et al., 2007). A valuable source for state of the art performance are competitions (cf. Sec. 5.5) that, however, aim to evaluate whole systems under operational conditions and are not well suited for measuring the performance of one single algorithm.

6 Conclusions and Future Work

This paper has presented a new solution to the simultaneous localization and mapping (SLAM) problem with six degrees of freedom. The method is based on ICP scan matching, with odometry extrapolation, initial pose estimation using a coarse-to-fine strategy with an octree representation and closing loop detection. Furthermore, the paper investigates approximate data association using kd-trees and a novel exact data association called cached kd-tree search.

We see a number of basically independent ideas work together in our 6D SLAM approach. ICP is now among the standard algorithms for scan registration; our contribution with respect to using it is to make it efficient for 6D registration of 3D scans by using octree representation, point reduction and kd-trees, including approximation and caching.

Next, having ICP available in an on-line on-board version for 6D registration of 3D scans allows 6D scan registration to be used as a means for posterior pose correction, based on the rich amount of pose difference information that 3D scans yield. As a consequence, we can afford to use just one pose rather than a distribution of poses as in probabilistic approaches: our pose tracking, aided by 6D scan registration (i.e., the prior pose estimation as modified by the posterior correction gained from registration), is typically sufficiently accurate.

Third, loop closing is another common topic in SLAM; here we profit again from the relatively accurate and robust posterior 6D pose estimation.

A rich set of experiments, including competitions, has confirmed our approach. In fact, more experiment data sets than those presented previously, have been used and are available (see below).

The algorithms are implemented without using probabilistic concepts. Keeping track of multi hypotheses leads to enormous computational requirements, which cannot currently be made available on a mobile platform. This dependence on one hypothesis has led us to methods that improve incrementally the 6D pose estimate. However, in future work we plan to adapt concepts from probabilistic robotics, like explicit representations of uncertainties, i.e., computing covariance matrices from scan matching. Furthermore, we will focus on multi robot 3D mapping and on integrating vision sensors in the mapping system.

To foster research in *Quantitative Performance Evaluation of Robotic and Intelligent Systems* we will continue participating in the NIST evaluation, e.g., at RoboCup Rescue events. In addition we plan to start a public 3D scan repository, to make 3D scans available to the robotics community, like the Radish (The Robotics Data Set Repository) repository (Howard and Roy, 2006). Material can currently be accessed under: `http://kos.informatik.uni-osnabrueck.de/3Dscans/`

References

Allen, P., Stamos, I., Gueorguiev, A., Gold, E., and Blaer, P. (2001). AVENUE: Automated Site Modelling in Urban Environments. In *Proceedings of the Third International Conference on 3D Digital Imaging and Modeling (3DIM '01)*, Quebec City, Canada.

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

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

Arya, S., Mount, D. M., Netanyahu, N. S., Silverman, R., and Wu, A. Y. (1998). An Optimal Algorithms for Approximate Nearest Neighbor Searcching in Fixed Dimensions. *Journal of the ACM*, 45:891 – 923.

Bailey, T., Nieto, J., Guivant, J., Stevens, M., and Nebot, E. (2006a). Consistency of the EKF-SLAM Algorithm. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '06)*, Bejing, China.

Bailey, T., Nieto, J., and Nebot, E. (2006b). Consistency of the FastSLAM Algorithm. In *IEEE International Conference on Robotics and Automation (ICRA '06))*, Orlando, Florida, U.S.A.

Benjemaa, R. and Schmitt, F. (1997). Fast Global Registration of 3D Sampled Surfaces Using a Multi-Z-Buffer Technique. In *Proceedings IEEE International Conference on Recent Advances in 3D Digital Imaging and Modeling (3DIM '97)*, Ottawa, Canada.

Bentley, J. L. (1975). Multidimensional binary search trees used for associative searchin. *Communications of the ACM*, 18(9):509 – 517.

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

Biber, P., Andreasson, H., Duckett, T., and Schilling, A. (2004). 3D Modeling of Indoor Environments by a Mobile Robot with a Laser Scanner and Panoramic Camera. In *Pro-*

ceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '04), Sendai, Japan.

Dissanayake, M. W. M. G., Newman, P., Clark, S., Durrant-Whyte, H. F., and Csorba, M. (2001). A Solution to the Simultaneous Localization and Map Building (SLAM) Problem. *IEEE Transactions on Robotics and Automation*, 17(3):229 – 241.

Eggert, D., Fitzgibbon, A., and Fisher, R. (1998). Simultaneous Registration of Multiple Range Views Satisfying Global Consistency Constraints for Use In Reverse Engineering. *Computer Vision and Image Understanding*, 69:253 – 272.

FGAN (2007). `http://www.elrob2006.org/`.

Folkesson, J. and Christensen, H. (2003). Outdoor Exploration and SLAM using a Compressed Filter. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '03)*, pages 419–426, Taipei, Taiwan.

Frese, U. and Hirzinger, G. (2001). Simultaneous Localization and Mapping – A Discussion. In *Proceedings of the IJCAI Workshop on Reasoning with Uncertainty in Robotics*, pages 17 – 26, Seattle, USA.

Friedman, J. H., Bentley, J. L., and Finkel, R. A. (1977). An algorithm for finding best matches in logarithmic expected time. *ACM Transaction on Mathematical Software*, 3(3):209 – 226.

Früh, C. and Zakhor, A. (2001). 3D Model Generation for Cities Using Aerial Photographs and Ground Level Laser Scans. In *Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR '01)*, Kauai, Hawaii, USA.

Georgiev, A. and Allen, P. K. (2004). Localization Methods for a Mobile Robot in Urban Environments. *IEEE Transaction on Robotics and Automation (TRO)*, 20(5):851 – 864.

Greenspan, M. and Yurick, M. (2003). Approximate K-D Tree Search for Efficient ICP. In *Proceedings of the 4th IEEE International Conference on Recent Advances in 3D Digital Imaging and Modeling (3DIM '03)*, pages 442 – 448, Banff, Canada.

Hebert, M., Deans, M., Huber, D., Nabbe, B., and Vandapel, N. (2001). Progress in 3–D Mapping and Localization. In *Proceedings of the 9th International Symposium on Intelligent Robotic Systems, (SIRS '01)*, Toulouse, France.

Howard, A. and Roy, N. (2003 – 2006). Radish: The Robotics Data Set Repository, Standard data sets for the robotics community. `http://radish.sourceforge.net/`.

Kohlhepp, P., Walther, M., and Steinhaus, P. (2003). Schritthaltende 3D-Kartierung und Lokalisierung für mobile Inspektionsroboter. In Dillmann, R., Wörn, H., and Gockel, T., editors, *Proceedings of the Autonome Mobile Systeme 2003, 18. Fachgesprche.*

Leonard, J. J. and Durrant-Whyte, H. F. (1991). Mobile robot localization by tracking geometric beacons. *IEEE Transactions Robotics and Automation (TRA)*, 7(3):376 – 382.

Lorusso, A., Eggert, D., and Fisher, R. (1995). A Comparison of Four Algorithms for Estimating 3-D Rigid Transformations. In *Proceedings of the 4th British Machine Vision Conference (BMVC '95)*, pages 237 – 246, Birmingham, England.

Lu, F. and Milios, E. (1997). Globally Consistent Range Scan Alignment for Environment Mapping. *Autonomous Robots*, 4(4):333 – 349.

Magnusson, M. and Ducket, T. (2005). A Comparison of 3D Registration Algorithms for Autonomous Underground Mining Vehicles. In *Proceedings of the Second European Conference on Mobile Robotics (ECMR '05)*, pages 86 – 91, Ancona, Italy.

Matrix FAQ (1997). Version 2, `http://vamos.sourceforge.net/matrixfaq.htm`.

NIST (2007). National institute of standards and technology, intelligent systems division, `http://robotarenas.nist.gov/competitions.htm`.

Nüchter, A. (2006). *Semantische 3D-Karten für autonome mobile Roboter (in German)*. PhD thesis, University of Bonn.

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

Nüchter, A., Surmann, H., Lingemann, K., Hertzberg, J., and Thrun, S. (2004). 6D SLAM with an Application in autonomous mine mapping. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1998 – 2003, New Orleans, USA.

Pulli, K. (1999). Multiview Registration for Large Data Sets. In *Proceedings of the 2nd International Conference on 3D Digital Imaging and Modeling (3DIM '99)*, pages 160 – 168, Ottawa, Canada.

Se, S., Lowe, D., and Little, J. (2001). Local and Global Localization for Mobile Robots using Visual Landmarks. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '01)*, Hawaii, USA.

Sequeira, V., Ng, K., Wolfart, E., Goncalves, J., and Hogg, D. (1999). Automated 3D reconstruction of interiors with multiple scan–views. In *Proceedings of SPIE, Electronic Imaging '99, The Society for Imaging Science and Technology /SPIE's 11th Annual Symposium*, San Jose, CA, USA.

Smith, R., Self, M., and Cheeseman, P. (1986). Estimating uncertain spatial relationships in robotics. In *Proceedings of the 2nd Annual Conference on Uncertainty in Artificial Intelligence (UAI '86)*, pages 435–461.

Surmann, H., Nüchter, A., and Hertzberg, J. (2003). An autonomous mobile robot with a 3D laser range finder for 3D exploration and digitalization of indoor environments. *Journal Robotics and Autonomous Systems*, 45(3 – 4):181 – 198.

Surmann, H., Nüchter, A., Lingemann, K., and Hertzberg, J. (2004). 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)*, Lisbon, Portugal.

Thrun, S. (1998). Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, 99(1):21–71.

Thrun, S. (2002). Robotic mapping: A survey. In Lakemeyer, G. and Nebel, B., editors, *Exploring Artificial Intelligence in the New Millenium*. Morgan Kaufmann.

Thrun, S., Burgard, W., and Fox, D. (1997). A probabilistic approach to concurrent mapping and localization for mobile robots. *Machine Learning and Autonomous Robots*, 31(5):1 – 25.

Thrun, S., Fox, D., and Burgard, W. (2000). A real-time algorithm for mobile robot mapping with application to multi robot and 3D mapping. In *Proceedings of the IEEE*

International Conference on Robotics and Automation (ICRA '00), San Francisco, CA, USA.

Thrun, S., Liu, Y., Koller, D., Ng, A. Y., Ghahramani, Z., and Durrant-Whyte, H. F. (2004). Simultaneous localization and mapping with sparse extended information filters. *Machine Learning and Autonomous Robots*, 23(7 – 8):693 – 716.

Thrun, S., Montemerlo, M., and Aron, A. (2006). Probabilistic Terrain Analysis For High-Speed Desert Driving. In *Proceedings of Robotics: Science and Systems*, Cambridge, USA.

Weingarten, J. and Siegwart, R. (2005). EKF-based 3D SLAM for structured environment reconstruction. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '05)*, pages 2089 – 2094, Edmonton, Alberta Canada.

Wulf, O., Arras, K. O., Christensen, H. I., and Wagner, B. A. (2004). 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.

Wulf, O., Nüchter, A., Hertzberg, J., and Wagner, B. (2007). Ground Truth Evaluation of Large Urban 6D SLAM. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '07)*, San Diego.

Zhao, H. and Shibasaki, R. (2001). Reconstructing Textured CAD Model of Urban Environment Using Vehicle-Borne Laser Range Scanners and Line Cameras. In *Second International Workshop on Computer Vision System (ICVS '01)*, pages 284 – 295, Vancouver, Canada.