6D SLAM with Cached kd-tree Search

Andreas Nüchter, Kai Lingemann, Joachim Hertzberg

University of Osnabrück, Institute of Computer Science
Knowledge Based Systems Research Group
Albrechtstr. 28, D-49069 Osnabrück, Germany
{nuechter|lingemann|hertzberg}@informatik.uni-osnabrueck.de

Abstract. 6D SLAM (Simultaneous Localization and Mapping) or 6D Concurrent Localization and Mapping of mobile robots considers six degrees of freedom for the robot pose, namely, the x, y and z coordinates and the roll, yaw and pitch angles. In previous work we presented our scan matching based 6D SLAM approach [10–12, 16], where scan matching is based on the well known iterative closest point (ICP) algorithm [3]. Efficient implementations of this algorithm are a result of a fast computation of closest points. The usual approach, i.e., using kd-trees is extended in this paper. We describe a novel search stategy, that leads to significant speed-ups. Our mapping system is real-time capable, i.e., 3D maps are computed using the resources of the used Kurt3D robotic hardware.

1 Introduction

Reliable environment mapping is one of the most fundamental robotic research issues. One on hand, it supersedes tedious manual environment mapping, on the other hand, it is the basis for many robotic taks, since it enables the robot to navigate in unknown terrain. Many applications benefit from solving this problem, e.g., rescue and inspection robotics, architecture modelling, or industrial automation.

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 and therefore the structure of the local maps, e.g., of single scans, needs to be used to create a precise global map. Hence, the simultaneous localization and mapping (SLAM) problem has a chicken-and-egg nature: Mapping solves the localization and localization solves the mapping problem. Finally, robot poses in natural outdoor environments necessarily involve yaw, pitch, roll angles and elevation, turning pose estimation as well as scan registration into a problem with six mathematical dimensions, i.e. 6D SLAM.

Scan matching plays an important role in mapping systems. It is used to correlate and align scans takes from different poses. Matching of scans is transformed to the problem of matching of points, e.g., closest points [3], followed by a computation of an optimal rotation and translation. Often this process is iterated, since the correct point matches are not known in advance and a heuristic is used instead. An initial matching is corrected in every iteration.

Since matching is done by matching closest points, a fast algorithm for finding a closest point for any given one is necessery. In earlier work we presented, we showed that instead of computing an exact closest point, the computation of approximate nearest neigbors does not significantly deteriorate the results of the scan matching, but results in a significant speed up [10]. Nevertheless, one would always prefer exact algorithms. This paper presents an excact and fast method for computing closest points. It is based on caching, i.e., memorization, of intermediate results and on traversing the kd-tree starting from a leaf.

The paper is organized as follows: First, we present briefly the state of the art in robotic mapping followed by introducing our robot Kurt3D. Then, we sketch out 6D SLAM algorithm and decribe in chapter 5 the performance issues of the scan matching in detail. Here, we introduce the cached kd-tree search. Chapeter 6 presents the result and conclusions.

2 State of the Art

2.1 Related Work

SLAM in well-defined, planar indoor environments is considered solved, since reliable probabilistic methods exists. Here, the robot has probabilistic motion models and uncertain (landmark) perception models. Through integration of these two distributions with a Bayes filter, e.g., Kalman or particle filter, it is possible to localize the robot precisely. Using multi hypothesis tracking these approaches are very stable. Please refer to [17], where Thrun reviews the existing SLAM methods.

6D SLAM still proposes a challenge, since several strategies for planar environments become infeasible, e.g., with 6 degrees of freedom the matrices in Kalman or information filter SLAM grow more rapidly and a multi hypothesis approach would certainly require too many particles. Therefore, 3D mapping systems [5,8,14,15] often rely on scan matching approaches. Hereby, ICP is used as in [5,14] as well as in our previous work [11,12,16].

2.2 Kurt3D

In our experiments the mobile robot Kurt3D is used. The robot is equipped with a 3D laser scanner.

The 3D laser range finder. The 3D laser range finder (Fig. 1) [15] is built on the basis of a SICK 2D range finder by extension with a mount and a small servomotor. The 2D laser range finder is attached in the center of rotation to the mount for achieving a controlled pitch motion with a standard servo.

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

The mobile robot. Kurt3D (Fig. 1) 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.

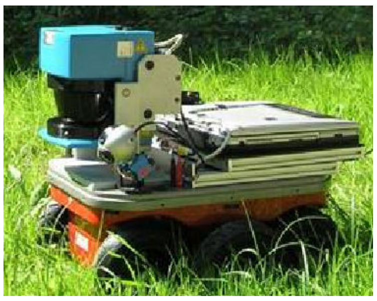


Fig. 1: Kurt3D.

3 6D SLAM

To create a correct and consistent environment map, 3D scans have to be merged into one coordinate system. This process is called registration. If the robot carrying the 3D scanner were localized precisely, 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.

Kurt3D's SLAM algorithm consists of four steps, that are explained in the following subsections.

3.1 Odometry extrapolation

The odometry is extrapolated to 6 degrees of freedom using previous registration matrices, i.e., the change of the robot pose $\Delta\mathbf{P}$ given the odometry information $(x_n, z_n, \theta_{y,n})$, $(x_{n+1}, z_{n+1}, \theta_{y,n+1})$ and the registration matrix $\mathbf{R}(\theta_{x,n}, \theta_{y,n}, \theta_{z,n})$ is calculated by solving:

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

3.2 Calculating Heuristic Initial Estimations for ICP Scan Matching

For the given two sets M and D of 3D scan points stemming from the 3D scans, our heuristic computes two octrees based on these point clouds. The octree's rigid transformations are applied to the second octree, until the number of overlapping cubes has reached its maximum. The transformations are computed in nested loops. However, the computational complexity is reduced due to the fact that we limit the search space relative to the octree cube size. Details can be found in [11].

3.3 Scan Registration

We use the well-known Iterative Closest Points (ICP) algorithm to calculate a rough approximation of the transformation while the robot is acquiring the 3D scans [3]. Given two independently acquired sets of 3D points, M and D, which correspond to a single shape, we aim at finding the transformation consisting of a rotation $\mathbf{R}$ and a translation $\mathbf{t}$ which minimizes the following cost function:

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

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 ($\mathbf{R}$, $\mathbf{t}$) that minimizes $E(\mathbf{R},\mathbf{t})$ 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 $(\mathbf{R},\mathbf{t})$ for minimizing equation (1). In the last iteration step, the point correspondences are assumed to be correct. Besl et al. prove that the method terminates in a minimum [3]. 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.

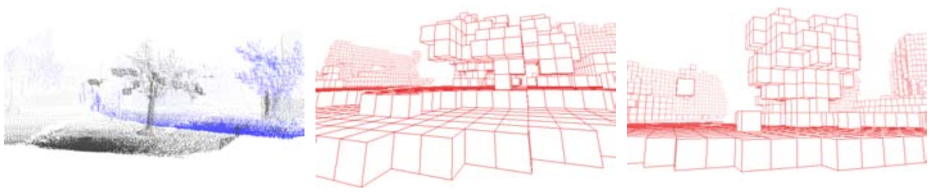


Fig. 2. Left: Two 3D point clouds. Middle: Octree corresponding to the black point cloud. Right: Octree based on the blue points.

In every iteration, the optimal transformation ($\mathbf{R}$, $\mathbf{t}$) has to be computed. Eq. (1) can be reduced to

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

with $N = \sum_{i=1}^{|M|}\sum_{j=1}^{|D|} w_{i,j}$, since the correspondence matrix can be represented by a vector $\mathbf{v}$ containing the point pairs.

Four direct methods are known to minimize eq. (2) [7]. In earlier work [15] we used a quaternion based method [3], 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 [1]. 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 $\mathbf{t}$ using the centroids of the points belonging to the matching, i.e.,

$$\mathbf{c}_m = \frac{1}{N}\sum_{i=1}^{N}\mathbf{m}_i, \qquad \mathbf{c}_d = \frac{1}{N}\sum_{i=1}^{N}\mathbf{d}_j \tag{3}$$

and

$$M' = \{\mathbf{m}'_i = \mathbf{m}_i - \mathbf{c}_m\}_{1,\ldots,N}, \qquad D' = \{\mathbf{d}'_i = \mathbf{d}_i - \mathbf{c}_d\}_{1,\ldots,N}. \tag{4}$$

After substituting (3) and (4) into the error function, Eq. (2) becomes:

$$E(\mathbf{R},\mathbf{t}) \propto \sum_{i=1}^{N} ||\mathbf{m}'_i - \mathbf{R}\mathbf{d}'_i||^2 \quad \text{with} \quad \mathbf{t} = \mathbf{c}_m - \mathbf{R}\mathbf{c}_d. \tag{5}$$

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

$$\mathbf{H} = \sum_{i=1}^{N}\mathbf{d}'_i\mathbf{m}'^T_i = \begin{pmatrix} S_{xx} & S_{xy} & S_{xz} \\ S_{yx} & S_{yy} & S_{yz} \\ S_{zx} & S_{zy} & S_{zz} \end{pmatrix}, \tag{6}$$

with $S_{xx} = \sum_{i=1}^{N} m'_{ix}d'_{ix}$, $S_{xy} = \sum_{i=1}^{N} m'_{ix}d'_{iy}$, $\ldots$ [1].

We proposed and evaluated algorithms to accelerate ICP, namely point reduction and approximate kd-trees [12, 15, 16], which are used here, too. They will be addressed in Detail in Sec. 4.

3.4 Loop Closing

After matching multiple 3D scans, errors have accumulated and loops would normally not be closed. 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 section 3.2 to revise the hypothesis. Finally, if a registration is possible, the computed error, i.e., the transformation ($\mathbf{R}$, $\mathbf{t}$) is distributed over all 3D scans.

3.5 Model Refinement

Based on the idea of Pulli we designed the relaxation method *simultaneous matching* [15]. The first scan is the masterscan and determines the coordinate system. It is fixed. The following three steps register all scans and minimize the global error, after a queue is initialized with the first scan of the closed loop:

1. Pop the first 3D scan from the queue as the current one.
2. If the current scan is not the master scan, then 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. 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, 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 [13]. 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 [16].

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 [10, 12].

Searching kd-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 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 [2, 4, 6].

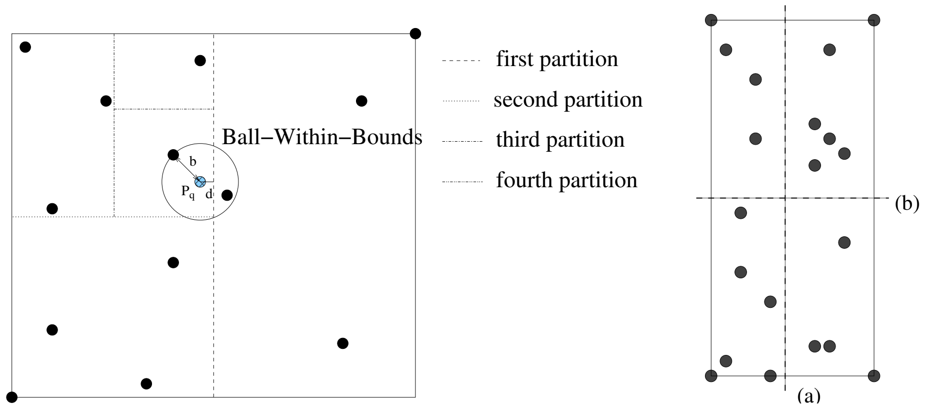


Fig. 3. Left: Recursive construction of a kd-tree. If the query consists of point $\mathbf{p}_q$, kd-tree search has to backtrack to the tree root to find the closest point. Right: Partitioning of a point cloud. Using the cut (b) rather than (a) results in a more compact partition and a smaller probability of backtracking [4].

4.1 Cached kd-trees

The cached kd-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-

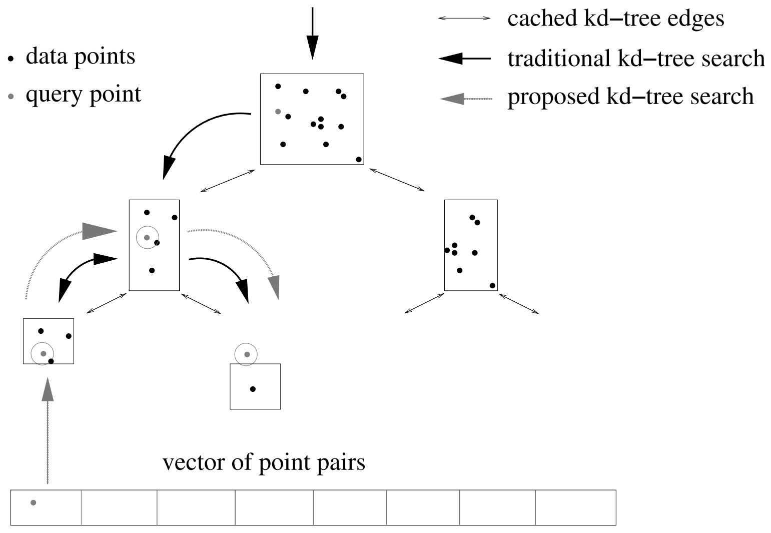


Fig. 4. Schematic description of the proposed search method: Instead of closest point searching from the root of the tree to the leafs that contain the data points, a pointer to the leafs is cached. In the second and following ICP iteration, the tree is searched backwards.

within-bounds test 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. 4). Algorithm 1 summarizes the ICP with cached kd-tree search.

Algorithm 1 ICP with cached kd-tree search

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

Performance of cached kd-tree search. The proposed ICP variant uses exact closest point search. In contrast to the previously discussed approximate kd-tree search for ICP algorithms [6, 10], registration inaccuracies or errors due to approximation cannot occur.

Friedman et al. prove that searching for closest points using kd-trees needs logarithmic time [4], 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 $\mathcal{O}((I + \log|M|)\ |D|)$ time in the best case. This performance is reached if constant time is needed for backtracking. Obviously the backtracking time depends on the computed ICP transformation $(\mathbf{R}, \mathbf{t})$. For small transformations the time is nearly constant. Cached kd-tree search needs $\mathcal{O}(|D|)$ extra memory for the vector $\mathbf{v}$, i.e., for storing the pointers to the tree leafs. Furthermore, additional $\mathcal{O}(\log|M|)$ memory is needed for storing the backwards pointers in the kd-tree. Fig. 5 shows the overall speedup dependend on the bucket size of the kd-tree (top) and the speed-up per ICP iteration (bottom). Note that in the first iteration traditional kd-tree search and cached kd-tree search have the same search time, since the cache needs to be initialized. A detailed analysis of cached kd-tree search can be found in [9].

5 Results and Conclusions

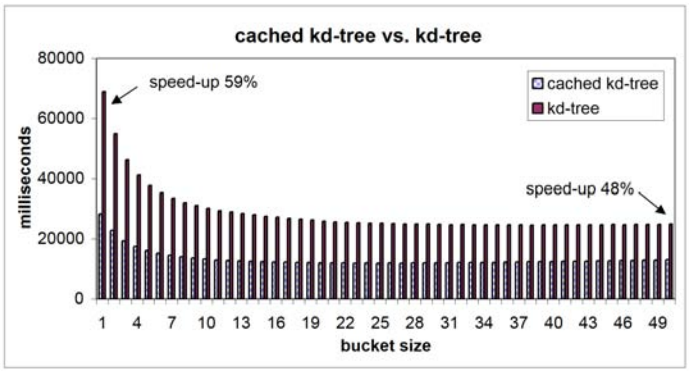
This paper extends our scan matching based 6D SLAM approach. An efficient implementation of the ICP algorithm are a result of a fast computation of closest points. The usual approach, i.e., using kd-trees is slightly modified in this paper. The novel search stategy, is based on traversing the kd-tree backwards, starting from memorized, i.e., chaced intermediate data. We end up with significant speed-up.

The proposed methods have been tested on various data sets, including test runs at RoboCup Rescue and ELROB. Fig. 6 show three closed loops. 3D animations of the bottom ones can be found at `http://kos.informatik.uni-osnabrueck.de/download/6Dpre/` and `http://kos.informatik.uni-osnabrueck.de/download/6Doutdoor/`. The loop in the top and bottom left part of Fig. 6 was closed manually, whereas the bottom right loop was detected automatically.

These large loops require an reliable robot control architecture for driving the robot and efficient 3D data handling and storage methods. In future work we will tackle the emerging topic of map management.

References

1. K. S. Arun, T. S. Huang, and S. D. Blostein. Least square fitting of two 3-d point sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(5):698 – 700, 1987.
2. J. L. Bentley. Multidimensional binary search trees used for associative searchin. *Communications of the ACM*, 18(9):509 – 517, September 1975.



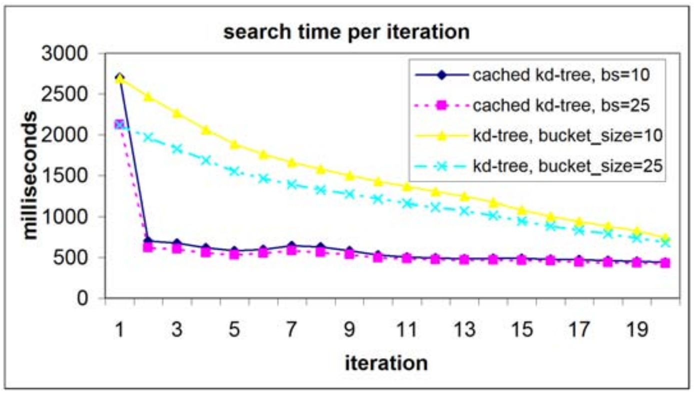


Fig. 5. Top: Achieved speedups of cached kd-tree search compared to traditional kd-tree search for an ICP based registration of two point sets. Bottom: Time consumption per ICP iteration.

3. P. Besl and N. McKay. A method for Registration of 3–D Shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239 – 256, February 1992.
4. J. H. Friedman, J. L. Bentley, and R. A. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transaction on Mathematical Software*, 3(3):209 – 226, September 1977.
5. A. Georgiev and P. K. Allen. Localization Methods for a Mobile Robot in Urban Environments. *IEEE Transaction on Robotics and Automation (TRO)*, 20(5):851 – 864, October 2004.
6. M. Greenspan and M. Yurick. 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, October 2003.
7. A. Lorusso, D. Eggert, and R. Fisher. A Comparison of Four Algorithms for Estimating 3-D Rigid Transformations. In *Proceedings of the 5th British Machine Vision Conference (BMVC '95)*, pages 237 – 246, Birmingham, England, September 1995.

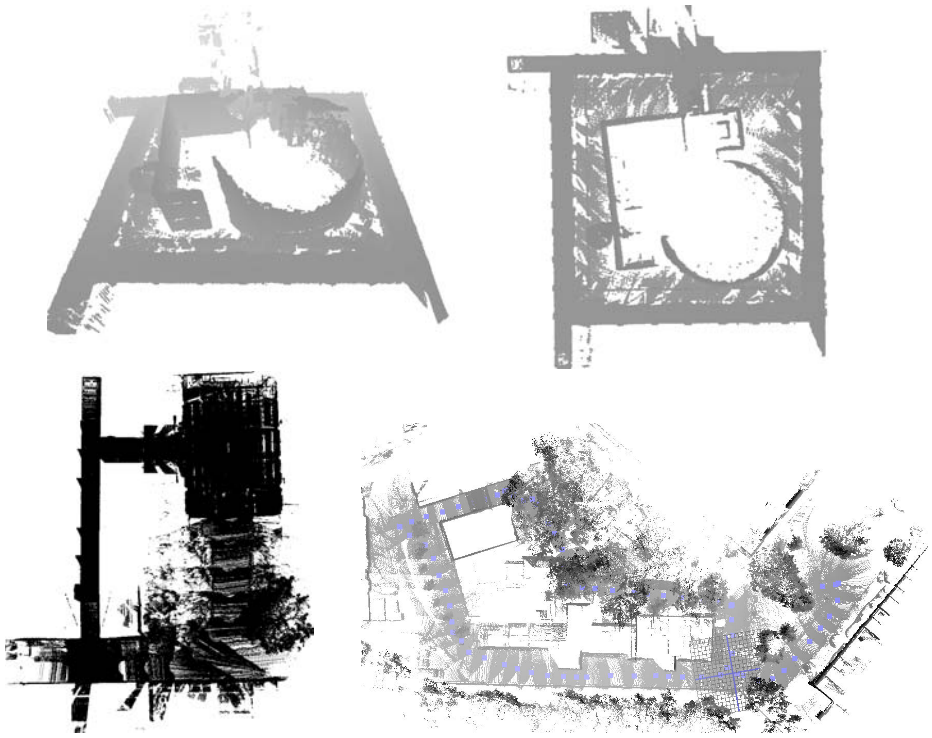


Fig. 6. Three 3D point clouds. The top row shows a closed loop of Dagstuhl castle as 3D view and in top view containing 84 3D scans, each with 6.8 million 3d points. Bottom left: Closed loop with 9 million 3D data points. Bottom right: Loop with 7 million points and a path length of over 250 m.

8. M. Magnusson and T. Duckett. A comparison of 3d registration algorithms for autonomous underground mining vehicles. In *Proceedings of the European Conference on Mobile Robotics (ECMR 2005)*, 2005.
9. A. Nüchter, K. Lingemann, and J. Hertzberg. Cached kd-tree search for ICP Algorithms. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, (submitted), 2006.
10. A. Nüchter, K. Lingemann, J. Hertzberg, and H. Surmann. 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., July 2005.
11. A. Nüchter, K. Lingemann, J. Hertzberg, and H. Surmann. Heuristic-Based Laser Scan Matching for Outdoor 6D SLAM. In *KI 2005: Advances in Artificial Intelligence. 28th Annual German Conference on AI, Proceedings Springer LNAI vol. 3698*, pages 304 – 319, Koblenz, Germany, September 2005.
12. A. Nüchter, H. Surmann, K. Lingemann, J. Hertzberg, and S. Thrun. 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, April 2004.

13. K. Pulli. 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, October 1999.
14. V. Sequeira, K. Ng, E. Wolfart, J. Goncalves, and D. Hogg. 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, January 1999.
15. H. Surmann, A. Nüchter, and J. Hertzberg. An autonomous mobile robot with a 3D laser range finder for 3D exploration and digitalization of indoor en vironments. *Journal Robotics and Autonomous Systems*, 45(3 – 4):181 – 198, December 2003.
16. H. Surmann, A. Nüchter, K. Lingemann, and J. Hertzberg. 6D SLAM A Preliminary Report on Closing the Loop in Six Dimensions. In *Proceedings of the 5th IFAC Symposium on Intelligent Autonomous Vehicles (IAV '04)*, Lisabon, Portugal, July 2004.
17. S. Thrun. Robotic mapping: A survey. In G. Lakemeyer and B. Nebel, editors, *Exploring Artificial Intelligence in the New Millenium*. Morgan Kaufmann, 2002.