Parallelization of Scan Matching for Robotic 3D Mapping

Andreas Nüchter
Institute of Computer Science, University of Osnabrück, Osnabrück, Germany
nuechter@informatik.uni-osnabrueck.de

***Abstract*— Robotic 3D Mapping of environments is computationally expensive, since 3D scanners sample the environment with many data points. In addition, the solution space grows exponentially with the additional degrees of freedom needed to represent the robot pose. Mapping environments in 3D must regard six degrees of freedom to characterize the robot pose. This paper extends our solution to the 3D mapping problem by parallelization. The availability of multi-core processors as well as efficient programming schemes as OpenMP permit the parallel execution of robotics task with on-board means.**

***Index Terms*— 3D scan matching, Iterative Closest Point Algorithm, GraphSLAM, Parallel Algorithms, Simultaneous Localization and Mapping, OpenMP.**

I. Introduction

Methods for solving the simultaneous localization and mapping (SLAM) problem are a key scientific issue in mobile robotics research. SLAM solutions are important in providing mobile systems with the ability to operate with real autonomy. Many mobile robots are nowadays equipped with a 3D laser scanner to gather information about the environment. Multiple 3D scans are necessary to digitalize environments without occlusions. To create a correct and consistent model, the scans have to be merged in one coordinate system. This initial registration is usually done with the well-known Iterative Closest Points (ICP) algorithm [4]. In all sequential strategies, where each scan is matched to some previous one, small errors add up to global inconsistencies. These errors are due to imprecise measurements and small registration errors and can never be avoided. SLAM algorithms that use information about closed loops help diminish these effects. So, Lu and Milios proposed a probabilistic scan matching algorithm for solving the simultaneous localization and mapping (LUM) [15]. In recent work, these algorithms are applied to 3D laser scan mapping [5], [21]

3D scan matching approaches to SLAM tend to be computationally expensive due to several reasons: First, the amount of data: A 3D laser range finder scans the environment with a large number of samples. Second, the additional three DoF result in an exponentially larger solution space. The solution is computationally more complex. New hardware developments help reduce these computational costs. In a popular formulation of Moore's law, one could state that the number of transistors on integrated circuits doubles every 18 months. This is emphasized by the observation of the appearance of dual-core and quad-core CPUs in the consumer market. These chips multiply the number of computing units whereas the increase of the clock rate of the chips seams to stop due to thermal issues.

This paper presents a variation of 3D scan matching algorithms, namely the parallel Iterative Closest Points (pICP) algorithm and the parallel Lu / Milios SLAM approach (pLUM). The algorithms have been optimized for execution on a shared memory machine with p processors, i.e., for quad and dual-core processors. The application to robotics is obvious: Several mobile robots are already controlled by dual core notebooks. OpenMP is used to program the dual-core processors in a multi threaded fashion. To this end, this paper focuses on the implementation details of scan matching for our SLAM front-end.

The paper is structured as follows: After a brief description of the state of the art, we describe our 3D scan matching algorithm and our SLAM approach (section II and III). The parallelization of these two algorithms are introduced in section IV and V. In Section VI the experiments and results on various data sets are presented starting with comments on the load balancing issue. Section VII concludes.

A. Current Trends in Processor Technology

In April of 2005, Intel announced the Intel Pentium processor Extreme Edition, featuring an Intel dual-core processor. An Intel dual-core processor-based PC enables a higher throughput and simultaneous computing using a multi-core architecture. Intel dual-core CPUs supporting Hyper-Threading Technology can process four software threads simultaneously by using more efficiently resources that otherwise may sit idle [18]. Since multi-core processors represent a major evolution in computing technology, Intel competitors have dual and quad core processors, too.

B. 3D Metric Robotic Mapping – State of the Art

Metrical maps represent explicit distances of the environment. These maps are either 2D, usually an upright projection, or 3D, i.e., a volumetric environment map. State of the art for 2D metric maps are probabilistic methods, where the robot has a probabilistic motion and perception model. SLAM in well-defined, planar indoor environments is considered solved, a survey of these techniques is presented by Thrun in [22]. Furthermore, SLAM approaches can be classified by the number of DoF 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. This emerging research topic is called 6D SLAM, In previous work, we presented the mobile robot Kurt3D that uses a tiltable 3D laser range finder [20] in a stop-scan-match-go-process to create a 3D map of the environment by merging several 3D scans into one coordinate system [17], [21]. Here online map generation, i.e., the map is available right after the robot run without extra map computing time, was possible, through using pairwise scan matching with an ICP algorithm. The speed-ups have been realized using data reduction and approximate *k*-d tree search. Similar experiments have been made by Newman et al. [16]. A recent trend in laser based 6D SLAM is to overcome stop-and go fashion of scan acquisition by rotating or pitching the scanner while moving [6], [24], [27].

Another trend in SLAM research is to apply probabilistic methods to 3D mapping. Katz et al. use a probabilistic notion of ICP scan matching [12]. Weingarten et al. [26] and Cole et al. [6] use extended Kalman filter to the mapping problem. We extend this state of the art by a GraphSLAM method. A similar approach was used in [23]. However, their algorithm is not practical due to the reported computational requirements. Furthermore Frese presented an extension of his treemap SLAM algorithm to six degrees of freedom, which however covers the least-square estimation core and no actual scan-data processing [7].

II. The ICP Algorithm

The ICP Algorithm was developed by Besl and McKay [4] and is usually used to register two given point sets in a common coordinate system. The algorithm calculates iteratively the registration. In each iteration step, the algorithm selects the closest points as correspondences and calculates the transformation, i.e., rotation and translation $(\mathbf{R}, \mathbf{t})$, for minimizing the equation

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

where N_m and N_d, are the number of points in the model set M and data set D, respectively, and w_{ji} are the weights for a point match. The weights are assigned as follows: $w_{ji} = 1$, if $\mathbf{m}_i$ is the closest point to $\mathbf{d}_j$, $w_{ji} = 0$ otherwise. Eq. (1) is reduced to

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

with $N = \sum_{i=1}^{N_m} \sum_{j=1}^{N_d} w_{i,j}$, since the correspondence matrix can be represented by a vector $\mathbf{v}$ containing the point pairs, i.e., $\mathbf{v} = ((\mathbf{d}_1, \mathbf{m}_{f(\mathbf{d}_1)}), (\mathbf{d}_2, \mathbf{m}_{f(\mathbf{d}_2)}), \ldots, (\mathbf{d}_{N_d}, \mathbf{m}_{f(\mathbf{d}_{N_d})}))$, with $f(\mathbf{x})$ the search function returning the closest point. The assumption is that in the last iteration step the point correspondences, thus the vector of point pairs, are correct.

Besl and McKay show that the iteration terminates in a minimum [4]. Note: Normally, implementations of ICP would use a maximal distance for closest points to handle partially overlapping point sets. In this case the proof in [4] does no longer hold, since the number of points as well as the value of $E(\mathbf{R}, \mathbf{t})$ might increase after applying a transformation.

The time complexity of the algorithm described above is dominated by the time for determining the closest points (brute force search $\mathcal{O}(n^2)$ for 3D scans of n points). Several enhancements have been proposed, like the commonly used *k*-d trees [3], [4].

Four methods are available to calculate the transformation in each ICP iteration: A SVD based method of Arun et al. [2], a quaternion method of Horn [10], an algorithm using orthonormal matrices of Horn et al. [11] and a calculation based on dual quaternions of Walker et al. [25]. These algorithms show similar performance and stability concerning noisy data [14]. The difficulty of the minimization problem is to enforce the orthonormality of matrix $\mathbf{R}$.

In the following, we give a brief overview of the SVD-based algorithm. 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., for all points in vector $\mathbf{v}$:

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

and

$$M' = \{\mathbf{m}'_i = \mathbf{m}_i - \mathbf{c}_m\}_{1,\ldots,N}, \qquad (4)$$

$$D' = \{\mathbf{d}'_i \ = \mathbf{d}_i \ - \mathbf{c}_d\}_{1,\ldots,N}. \qquad (5)$$

After replacing (3), (4) and (5) in the error function, $E(\mathbf{R}, \mathbf{t})$ eq. (2) becomes:

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

$$= \frac{1}{N} \sum_{i=1}^{N} ||\mathbf{m}'_i - \mathbf{R}\mathbf{d}'_i||^2 \qquad (6a)$$

$$- \frac{2}{N} \tilde{\mathbf{t}} \cdot \sum_{i=1}^{N} (\mathbf{m}'_i - \mathbf{R}\mathbf{d}'_i) \qquad (6b)$$

$$+ \frac{1}{N} \sum_{i=1}^{N} ||\tilde{\mathbf{t}}||^2. \qquad (6c)$$

In order to minimize the sum above, all terms have to be minimized. The second sum (6b) is zero, since all values refer to centroid. The third part (6c) has its minimum for $\tilde{\mathbf{t}} = \mathbf{0}$ or

$$\mathbf{t} = \mathbf{c}_m - \mathbf{R}\mathbf{c}_d. \qquad (7)$$

Therefore the algorithm has to minimize only the first term, and the error function is expressed in terms of the rotation only:

$$E(\mathbf{R}, \mathbf{t}) \propto \sum_{i=1}^{N} ||\mathbf{m}'_i - \mathbf{R}\mathbf{d}'_i||^2. \qquad (8)$$

Theorem: The optimal rotation is calculated by $\mathbf{R} = \mathbf{V}\mathbf{U}^T$. Herby 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

$$\begin{aligned}\mathbf{H} &= \sum_{i=1}^{N} \mathbf{m}_i'^T \mathbf{d}_i' = \sum_{i=1}^{N} (\mathbf{m}_i - \mathbf{c}_m)(\mathbf{d}_i - \mathbf{c}_d)^T \qquad (9)\\ &= \begin{pmatrix} S_{xx} & S_{xy} & S_{xz} \\ S_{yx} & S_{yy} & S_{yz} \\ S_{zx} & S_{zy} & S_{zz} \end{pmatrix},\end{aligned}$$

with $S_{xx} = \sum_{i=1}^{N} m'_{ix} d'_{ix}$, $S_{xy} = \sum_{i=1}^{N} m'_{ix} d'_{iy}$, ... The analogous algorithm is derived directly from this theorem.

III. Lu Milios Style GraphSLAM

To solve SLAM, a 6D graph optimization algorithm for global relaxation based on the method of Lu and Milios [15] is employed, namely Lu and Milios style SLAM (LUM). Details of the 6D optimization, i.e., how the matrices have to be filled, can be found in [5]:

Given a network with $n+1$ nodes $X_0, ..., X_n$ representing the poses $V_0, ..., V_n$, and the directed edges $D_{i,j}$, we aim to estimate all poses optimally to build a consistent map of the environment. For simplicity, we make the approximation, that the measurement equation is linear, i.e.,

$$D_{i,j} = X_i - X_j$$

An error function is formed such that minimization results in improved pose estimations:

$$\mathbf{W} = \sum_{(i,j)} (D_{i,j} - \bar{D}_{i,j})^T C_{i,j}^{-1} (D_{i,j} - \bar{D}_{i,j}). \qquad (10)$$

where $\bar{D}_{i,j} = D_{i,j} + \Delta D_{i,j}$ models random Gaussian noise added to the unknown exact pose $D_{i,j}$. The covariance matrices $C_{i,j}$ describing the pose relations in the network are computed, based on the paired points of the ICP algorithm. The error function Eq. (10) has a quadratic form and is therefore solved in closed form by Cholesky decomposition in the order of $\mathcal{O}(n^3)$ for n poses ($n \ll N$). The algorithm optimizes Eq. (10) gradually by iterating the following five steps [5]:

1) Compute the point correspondences (n closest points) for any link (i, j) in the given graph.
2) Calculate the measurement vector $\bar{D}_{ij}$ and its covariance C_{ij}.
3) From all $\bar{D}_{ij}$ and C_{ij} form a linear system $\mathbf{GX} = \mathbf{B}$.
4) Solve for $\mathbf{X}$
5) Update the poses and their covariances.

IV. The Parallel ICP Algorithm

The basic work for parallelization of the ICP algorithm (pICP) was done by by Langis et al. [13]. pICP showed compelling results on a cluster with p processors for registration of large data sets. The basic idea is to divide the data set D into p parts and to send this part together with the whole model set M to the child processes that compute concurrently the expensive closest point queries. Afterwards, these points correspondences are transmitted back to the parent, that uses it for computing the transformation by minimizing Eq. (2). Then this transformation is sent to the childs, which transform the data set D. The process is iterated until the minimum of Eq. (2) is reached.

The parallelization scheme for ICP uses a lot of bandwidth for transmitting corresponding points. Thus, Langis et al. [13] proposed several enhancements to the parallel method. One of these improvements is avoiding the transfer of the corresponding points. The computation of the correlation matrix $\mathbf{H}$ Eq. (9) is parallelized and partially executed by the child processes, i.e., Eq. (9) is transformed s follows

$$\begin{aligned}\mathbf{H} &= \sum_{i=1}^{N} (\mathbf{m}_i - \mathbf{c}_m)(\mathbf{d}_i - \mathbf{c}_d)^T \\ &= \sum_{i=1}^{p} \sum_{j=1}^{N_i} (\mathbf{m}_{i,j} - \mathbf{c}_m)(\mathbf{d}_{i,j} - \mathbf{c}_d)^T, \qquad (11)\end{aligned}$$

where N_i denotes the number of points of child i, $\mathbf{c}_{m_i}, \mathbf{c}_{d_i}$ the centroid of points of child i, and $(\mathbf{m}_{i,j}, \mathbf{d}_{i,j})$ the jth corresponding point of child i. Furthermore, for Eq. 11 holds

$$\begin{aligned}\mathbf{H} &= \sum_{i=1}^{p} \sum_{j=1}^{N_i} \Big((\mathbf{m}_{i,j} - \mathbf{c}_{m_i} + (\mathbf{c}_{m_i} - \mathbf{c}_m)) \\ &\qquad (\mathbf{d}_{i,j} - \mathbf{c}_{d_i} + (\mathbf{c}_{d_i} - \mathbf{c}_d))^T \Big) \\ &= \sum_{i=1}^{p} \left(\mathbf{H}_i + N_i (\mathbf{c}_{m_i} - \mathbf{c}_m)(\mathbf{c}_{d_i} - \mathbf{c}_d)^T \right) \qquad (12)\end{aligned}$$

$$\text{where, } \mathbf{H}_i = \sum_{j=1}^{N_i} (\mathbf{m}_{i,j} - \mathbf{c}_{m_i})(\mathbf{d}_{i,j} - \mathbf{c}_{d_i})^T \qquad (13)$$

Therefore, after the point correspondences are parallely computed Eq. 12 and 13 form the algorithm for computing the correlation matrix $\mathbf{H}$ in a parallel fashion. The centroids $\mathbf{c}_m$ and $\mathbf{c}_d$ (cf. Eq. 3) are also computed from these intermediate results, i.e.,

$$\mathbf{c}_m = \frac{1}{N} \sum_{i=1}^{p} N_i \mathbf{c}_{m_i}, \qquad \mathbf{c}_d = \frac{1}{N} \sum_{i=1}^{p} N_i \mathbf{c}_{d_j}$$

Finally, after the parent has collected the vectors $\mathbf{c}_{m_i}$, $\mathbf{c}_{d_i}$ and the matrix $\mathbf{H}_i$ from the childs, it is possible to compute the transformation $(\mathbf{R}, \mathbf{t})$ to align the point sets.

A. Parallelization of k-d tree search

This paper considers an OpenMP implementation for the pICP algorithm, that is a shared memory architecture with p processors or threads (symmetric multiprocessing). Therefore, there is no need to create p copies of the model set M. However the search is executed in parallel using parallel *k*-d tree search.

1) k-d trees: *k*-d trees are a generalization of binary search trees. Every node represents a partition of a point set to the two successor nodes. The root represents the whole point cloud and the leaves provide a complete disjunct partition of the points. These leaves are called buckets. Furthermore, every node contains the limits of the represented point set.

2) Parallel Construction of k-d trees: Since the *k*-d tree partitions the point set in two disjunct subsets, the construction is easily partitioned by executing the two recursive cases in parallel.

3) Searching k-d trees: *k*-d trees are searched recursively. A given 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. The test is known as ball-within-bounds test [3], [8], [9].

4) Parallel Searching k-d trees: In the pICP algorithm several search requests have to be handled by the *k*-d tree at the same time. Efficient implementations of *k*-d tree search algorithms use static, i.e., global variables to save the current closest point. This current point is first derived by depth first search and then updated in the backtracking case. When executing the search with parallelism p copies of this point must be created.

B. Implementation Details

Most high performance processors insert a cache buffer between slow memory and the high speed registers of the CPU. If several threads use unique data elements on the same cache line for read and write, then so-called false sharing might occur. If one of the threads writes to a cache line, the same cache line referenced by the second thread is invalidated of the cache. Any new references to data in this cache line by the second thread results in a cache miss and the data has to be loaded again from memory. To avoid false sharings, we pad each thread's data element to ensure that elements owned by different threads all lie on separate cache lines as follows:

```
class KDParams {
public:
  /** pointer to the closest point.
    *  size = 4 bytes of 32 bit machines
    */
  double *closest;
  /** distance to the closest point.
    * size = 8 bytes
    */
  double closest_d2;
  /** pointer to the point,
    * size = 4 bytes of 32 bit machines
    */
  double *p;
  /** expand to 128 bytes to avoid
    * false-sharing, 16 bytes from above
    * + 28*4 bytes = 128 bytes */
  int padding[28];
};
```

In our search tree these padded parameters are included using memory alignment:

```
class KDtree {
  // [snip]
public:
   __declspec (align(16)) \
      static KDParams params[MAX_OPENMP_NUM_THREADS];
};
```

V. Parallel Lu Milios based GraphSLAM

There are several steps of the Lu Milios based GraphSLAM algorithm that might be executed in a parallel fashion, resulting in the pLUM algorithm. The point correspondences for any link (i, j) in the given graph are computed in parallel (step 1) as well as the computation of the measurement vector $\bar{D}_{ij}$ (step 2) and the formation of the linear system (step 3). In principle also the last step, i.e., solving the linear system of equations, can be executed in a parallel fashion by the parallel Cholesky decomposition [19]. However, this algorithm is mainly used in solving very large problems in the supercomputing research field.

Two possible strategies exist for parallelizing step 1: First, for all links the computation of the correspondence search proceed as the unimproved pICP case in section IV. The drawback of this strategy is that the closest point pairs have to be transmitted back to the parent thread. The more advanced strategy is to compute for the n given links, the correspondences on the p processors. In this strategy several *k*-d trees of different data sets have to be constructed, maintained and searched in a parallel fashion. However, it turns out that the parallelization scheme in section IV-A and IV-B also work in this pLUM variant. Since the global variables of the parallel *k*-d tree depend on the thread number, several data sets are processed at the same time. The search function of every *k*-d tree that is associated with a data set stores its local values separately.

VI. Experiments and Results

A. Load Balancing

The computations carried out in a parallel fashion have to be scheduled to the p processors. The goal of balancing the load for the processors is to maximize the gained speed-up aiming to reach the optimal runtime of t_s/p, with t_s the time for executing scan matching on a single processors.

The scheduling of parallel point searching for pICP is done by OpenMP altering the for statement. Every thread determines its thread number and uses this information for the parallel *k*-d tree search. Prior splitting of the data set avoids the determination of the thread number for every point. Splitting the data set has to be done in a randomized fashion to ensure load balancing. It turned out that this requires more time than the determination of the thread number in every iteration. Similar arguments apply for pLUM.

B. Matching Laser Scans Acquired with Kurt3D

In this experiment we use the exploration robot Kurt3D. Fig. 1 shows the robot that is equipped with a tiltable SICK laser range finder in a natural outdoor environment. The 3D laser range finder [20] is built on the basis of a SICK 2D range finder by extension with a mount and a small servomotor. Resolution can be adjusted at expense of scanning time; scanning is done in a stop-scan-go fashion. All computations have been carried out be Kurt3D that is equipped with a Panasonic Toughbook CF-74 with an Intel Core Duo-T2400, 1.83 GHz.

We made two experiments with Kurt3D. First we drove a closed loop in our office environment and acquired 11 3D scan with 85000 data points each. Fig. 2 shows the office scene. Table I shows the results. The overall running time

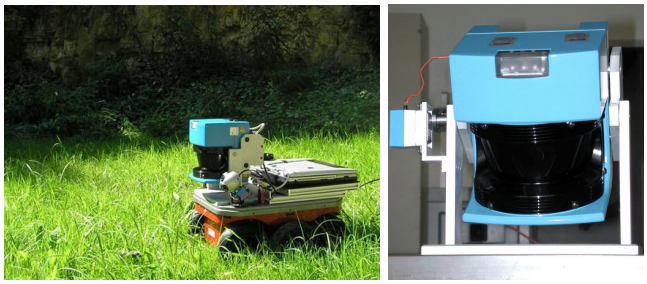


Fig. 1. Left: Kurt3D. Right: The 3D laser scanner is built of 2D laser scanner and a servo motor step-rotating the scanner.

TABLE I

RUNNING TIMES IN MILLISECONDS FOR SINGLE THREAD COMPUTATION VS. MULTI-THREAD COMPUTATION. THE FIRST ROWS REPRESENT THE SPEED-UP FOR PURE ICP AND LUM COMPUTATIONS ON AN INTEL CORE DUO-T2400.

algorithm	one thread	two threads	four threads
ICP	12688	9821	9750
LUM	703	610	453
total	13391	10431	10203

was reduced 76.2%. Using four instead of two threads yields a slight improvement, probably due to improved scheduling issues. Using more than four threads did not lead to an additional advances.

The second experiment was performed in Dagstuhl castle, where we acquired a 3D model of interior of the new building wing (cf. Fig. 3) containing 83 3D scans with the same resolution as in the previous experiment. The overall time spent by the 3D mapping algorithms, i.e., ICP and LUM was 112141 ms using single thread execution, 78141 ms using two threads and 70218 ms using four threads.

C. Matching of Continously Acquired 3D Scans

Wulf et al. presented a rotating 3D scanner in [27]. In the following experiment we processed 468 3D scans containing approximately 20000 3D points per scans. These scan where

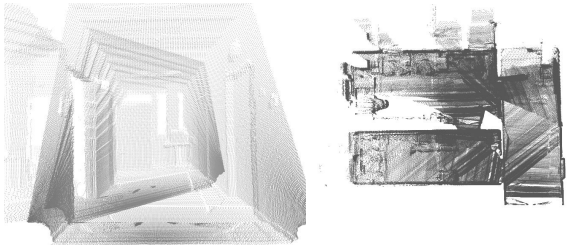


Fig. 2. Left: Kurt3D. Right: 3D scans on an office environment. Right: Top view of the map representing one small closed loop.

Fig. 3. Left: 3D view of the new wing of Dagstuhl castle. Right: Top view.

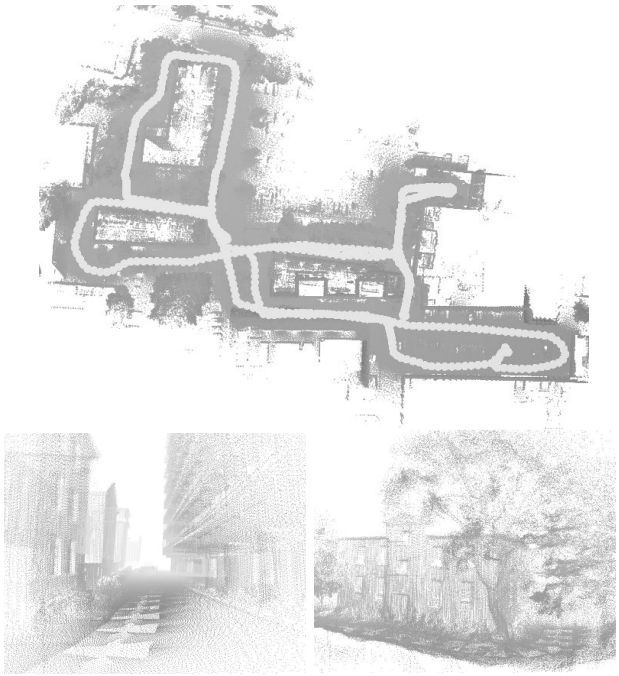


Fig. 4. Top: Trajectory and map of the campus of Leibniz Universität Hannover. Bottom: Two detailed 3D views. Bottom left: modelled skyscraper. Bottom right: building with tree in front.

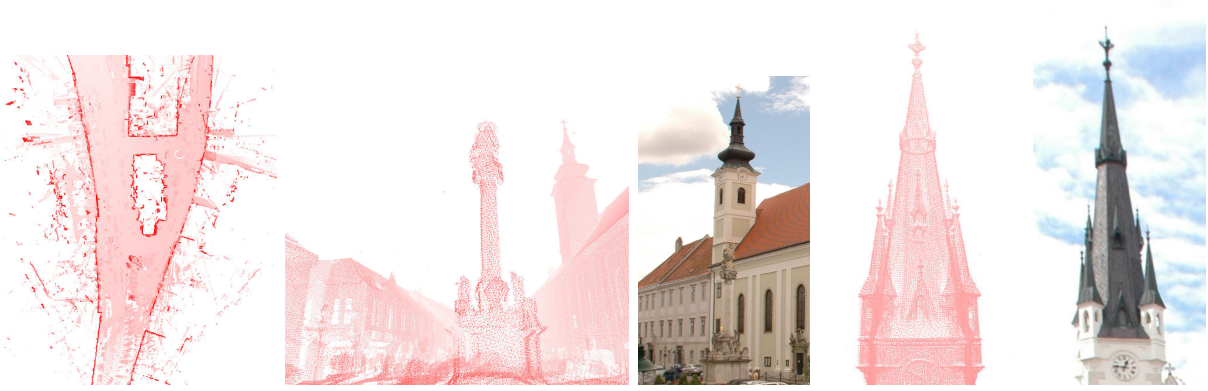
matched with ICP and since there were several loops we applied LUM after each loop detection. Loop detection is done using a simple distance criterion: If two estimated robot poses V_i and V_j are close enough, i.e., their distance falls below a threshold (here: 5 meters) then we assume these scans overlap and are matchable. The overall time spent by the 3D mapping algorithms, was reduced from 4526 s to 3420 s using four threads. Fig. 4 shows the final map (top view) and two detailed views.

D. Experiments with a High Resolution Data Set

In the last experiment, we tested the proposed parallelization on high resolution 3D scans provided by RIEGL Laser Measurement Systems GmbH [1]. For mapping we use 11 3D scans containing over 300000 data point each, covering a very large area. Table II shows the performance on this data set, Fig. 5 shows the final map, two detailed views with photos of the scene. The most gain is achieved by parallelization of ICP, i.e., up to 48.2%. The total speed-up on this data set is 23.7%. Besides processing these scans on a dual-core T2400, we tried our algorithms on Osnabrück's compute server with 4 dual-core Itanium-II processors and achieved tremendous speed-ups, up to 70%. Measurements are not available since the server is operated in multi-user mode.

VII. SUMMARY AND OUTLOOK

This paper has presented an approach to parallelize two well know mapping algorithms, namely ICP and LUM. Since 2D mapping is a special case of 3D mapping, the presented algorithms can handle 2D laser range data as well, but processing 3D data imposes a greater need for efficiency. The focus of the parallelization was to keep up with current hardware developments given by the introduction of dual and multi-core



(a) (b) (c) (d) (e)

Fig. 5. The main square in Horn (Austria). (a) 3D map in top view. (b) Monument in the center of the main square. (c) Corresponding photo (right part). (d) Church spire. (e) Photo of the white-steepled of the St. Georg church. Data provided by courtesy of RIEGL LMS GmbH [1].

TABLE II

RUNNING TIMES IN MILLISECONDS FOR SINGLE THREAD COMPUTATION VS. MULTI-THREAD COMPUTATION. THE FIRST ROWS REPRESENT THE SPEED-UP FOR PURE ICP AND LUM COMPUTATIONS.

number of 3D scans	one thread	two threads	four threads
2 (ICP)	20750	10985	10938
3 (ICP)	41984	21750	21750
6 (ICP)	134031	77390	77047
11 (ICP)	369828	218125	210515
11 (LUM)	794110	690023	678111
total	1163938	908148	888626

CPUs. The achieved speed-ups vary between the algorithms, for ICP a maximum speed-up of 48.2% can be reported, which is close to optimal on a dual-core CPU. The average speed-up is lower (approximately 25%). This difference is due to algorithm parts that cannot be parallelized and due to data managing processes.

In future work, we will concentrate on parallelizing probabilistic robotic algorithms to exploit current computing hardware. Particle filters seem to be good candidates for parallelization. The overall goal is to combine the reliability of probabilistic algorithms, like particle filters with the precission of deterministic approaches like scan matching.

ACKNOWLEDGMENT

The author would like to acknowledge Nikolaus Studnicka (RIEGL Laser measurement Systems GmbH, Horn) and Oliver Wulf, Bernardo Wagner (Leibniz Universität Hannover) for providing the data sets. Further thanks to Dorit Borrmann, Jan Elseberg for implementing the Lu / Milios style GraphSLAM with 6 DoF [5], to Joachim Hertzberg and Kai Lingemann for supporting this work.

REFERENCES

[1] Riegl Laserscanner, `http://www.riegl.co.at/`, 2007.
[2] K. S. Arun, T. S. Huang, and S. D. Blostein. Least square fitting of two 3-d point sets. *IEEE Trans. on PAMI*, 9(5):698 – 700, 1987.
[3] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509 – 517, Sept. 1975.
[4] P. Besl and N. McKay. A method for Registration of 3–D Shapes. *IEEE Trans. on PAMI*, 14(2):239 – 256, February 1992.
[5] D. Borrmann, J. Elseberg, K. Lingemann, A. Nüchter, and J. Hertzberg. The Extension of Lu and Milios Style SLAM to 6 Degrees of Freedom. In *IEEE/RSJ IROS 2007, (submitted)*.
[6] D. M. Cole and P. M. Newman. Using Laser Range Data for 3D SLAM in Outdoor Environments. In *Proc. IEEE ICRA*, U.S.A., May 2006.
[7] U. Frese. Efficient 6-DOF SLAM with Treemap as a Generic Backend. In *Proc. IEEE ICRA*, Rome, Italy, April 2007.
[8] 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.
[9] M. Greenspan and M. Yurick. Approximate K-D Tree Search for Efficient ICP. In *Proc. IEEE 3DIM*, Banff, Canada, October 2003.
[10] B. K. P. Horn. Closed–form solution of absolute orientation using unit quaternions. *J. of the Opt. Soc. of America A*, 4(4):629 – 642, 1987.
[11] B. K. P. Horn, H. M. Hilden, and Sh. Negahdaripour. Closed–form solution of absolute orientation using orthonormal matrices. *Journal of the Optical Society of America A*, 5(7):1127 – 1135, July 1988.
[12] R. Katz, N. Melkumyan, J. Guivant, T. Bailey, J. Nieto, and E. Nebot. Integrated Sensing Framework for 3D Mapping in Outdoor Navigation. In *Proc. IEEE/RSJ IROS*, Beijing, China, October 2006.
[13] C. Langis, M. Greenspan, and G. Godin. The parallel iterative closest point algorithm. In *Proc. IEEE 3DIM*, Quebec City, Canada, June 2001.
[14] A. Lorusso, D. Eggert, and R. Fisher. A Comparison of Four Algorithms for Estimating 3-D Rigid Transformations. In *Proceedings of the 4th British Machine Vision Conference (BMVC '95)*, Sept. 1995.
[15] F. Lu and E. Milios. Globally Consistent Range Scan Alignment for Environment Mapping. *Autonomous Robots*, 4(4):333 – 349, Oct. 1997.
[16] P. M. Newman, D. M. Cole, and K. Ho. Outdoor SLAM using Visual Appearance and Laser Ranging. In *Proc. IEEE ICRA*, Barcelona, 2005.
[17] A. Nüchter, K. Lingemann, J. Hertzberg, H. Surmann, K. Pervölz, M. Hennig, K. R. Tiruchinapalli, R. Worst, and T. Christaller. Mapping of Rescue Environments with Kurt3D. In *Proc. IEEE SSRR*, June 2005.
[18] Intel Dual-Core Processors. `http://www.intel.com/technology/computing/dual-core/`, 2007.
[19] E. E. Santos and P.-Y. Chu. Efficient and optimal parallel algorithms for cholesky decomposition. *Journal of Mathematical Modelling and Algorithms*, 2(3):217–234, September 2003.
[20] H. Surmann, K. Lingemann, A. Nüchter, and J. Hertzberg. A 3D laser range finder for autonomous mobile robots. In *Proc. 32nd Int. Symp. on Robotics (ISR '01)*, pages 153 – 158, Seoul, Korea, April 2001.
[21] H. Surmann, A. Nüchter, K. Lingemann, and J. Hertzberg. 6D SLAM A Preliminary Report on Closing the Loop in Six Dimensions. In *Proc. 5th IFAC Symp. on Int. Aut. Vehicles (IAV '04)*, Lisbon, July 2004.
[22] S. Thrun. Robotic mapping: A survey. In G. Lakemeyer and B. Nebel, editors, *Exploring Artificial Intelligence in the New Millenium*. Morgan Kaufmann, 2002.
[23] R. Triebel and W. Burgard. Improving Simultaneous Localization and Mapping in 3D Using Global Constraints. In *Proceedings of the National Conference on Artificial Intelligence (AAAI '05)*, 2005.
[24] R. Triebel, P. Pfaff, and W. Burgard. Multi-level surface maps for outdoor terrain mapping and loop closing. In *Proc. IEEE/RSJ IROS*, Beijing, China, October 2006.
[25] M. W. Walker, L. Shao, and R. A. Volz. Estimating 3-d location parameters using dual number quaternions. *CVGIP: Image Understanding*, 54:358 – 367, November 1991.
[26] J. Weingarten and R. Siegwart. EKF-based 3D SLAM for structured environment reconstruction. In *Proc. IEEE/RSJ IROS*, August 2005.
[27] O. Wulf, K. O. Arras, H. I. Christensen, and B. A. Wagner. 2D Mapping of Cluttered Indoor Environments by Means of 3D Perception. In *Proc. IEEE ICRA*, USA, April 2004.