An Explicit Loop Closing Technique for 6D SLAM

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

***Abstract*— Simultaneous Localization and Mapping (SLAM) is the problem of building a map of an unknown environment by a mobile robot while at the same time navigating the environment, using the unfinished map. For SLAM, two tasks have to be solved: First reliable feature extraction and data association, second the optimal estimation of poses and features. These two parts are often referred to as SLAM frontend and backend. Algorithms that solve SLAM by using laser scans commonly rely on matching closest points in the frontend part. Then the SLAM front- and backend have to be iterated to ensure that the map converges.**

This paper presents a novel approach for solving SLAM using 3D laser range scans. We aim at avoiding the iteration between the SLAM front- and backend and propose a novel explicit loop closing heuristic (ELCH). It *dissociates* the last scan of a sequence of acquired scans, reassociates it to the map, built so far by scan registration, and distributes the difference in the pose error over the SLAM graph. We describe ELCH in the context of SLAM with 3D scans considering 6 DoF. The performance is evaluated using ground truth data of an urban environment.

I. Introduction

Robots in recent research tend to leave the small laboratories and operate in large scale outdoor environments. This imposes two new challenges for mapping algorithms: First, they have to cope with non-flat surroundings, making 3D environment mapping necessary. Second, the size of the areas increases. In the past, automatic 3D mapping approaches in unstructured environments have been presented and successfully evaluated [18, 28] in competitions such as Robocup Rescue [26], the European Land Robotics Trial ELROB [7] or the DARPA Grand Challenge [5]. However, most of the approaches aim at mapping small environments or at constructing only local maps used for navigation tasks, in order to cope with the immense amount of data.

This paper presents a novel approach to large-scale 3D mapping. We aim at reducing the run time of our mapping system such that it performs fast in large environments using 3D laser scans. In particular, this paper presents an algorithm for efficient loop closing and consistent scan alignment that avoids iterative scan matching over *all* scans.

II. Related Work

A globally consistent representation of a robot's environment is crucial for many robotic applications. Equipped with a 3D depth-perceiving sensor, many mobile systems gather spatial information about their local 3D environments. Recent progress in environment sensing in robotics has led from initially custom made 3D scanners, as in [13, 25, 30], to sophisticated highly accurate 3D scanning systems, e.g., Riegl, Leica or Zoller+Fröhlich scanners, very fast scanning systems, like the Velodyne 3D scanner, and the emerging technology of

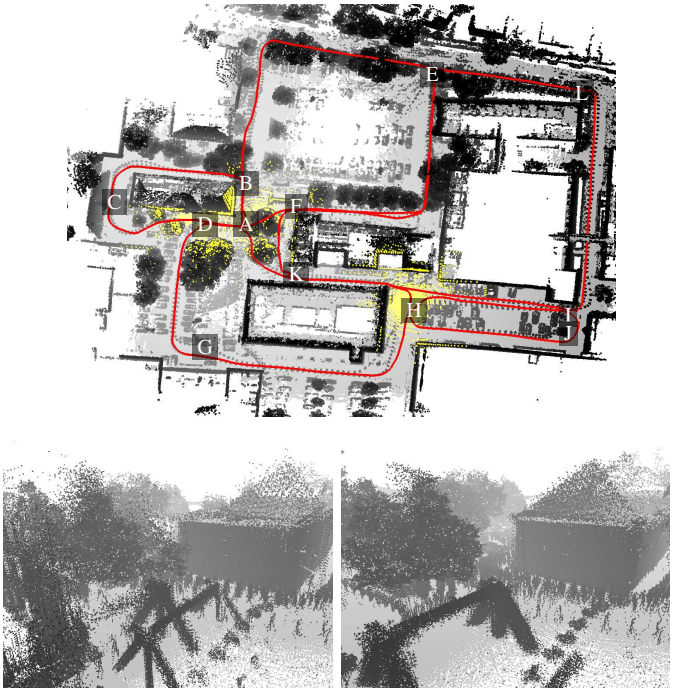


Fig. 1. Example trajectory (top view) that was automatically acquired by a mobile robot [29]. The overall time needed to process this data was reduced from 80.5 minutes to 6.4 minutes, which is faster than acquisition time. The 3D scans marked in yellow are used for evaluation. The scans have been taken according to the sequence: A-B-C-D-A-B-E-F-A-D-G-H-I-J-H-K-F-E-L-I-K-A. The bottom part of the figure shows two rendered 3D views. Left: Before loop closing. Right: Corrected scene.

3D cameras. The software development has to keep up with this progress in sensing hardware. This implies the need of new algorithms and data structures for handling the data. Fig. 1 presents a 3D map in bird's eye view that contains 15,338,164 data points from 924 automatically acquired 3D scans.

The local data contained in single 3D scans have to be registered to build a global map. A well established method for incremental registration of 3D point clouds is the iterative closest points (ICP) algorithm [2]. However, any incremental application of such matching algorithms leads to inconsistencies due to sensing errors and due to accumulating registration tolerances. To avoid these problems, global matching algorithms are needed, taking global correspondences between scans into account.

As [27] states, "virtually all state-of-the-art robotic mapping algorithms are probabilistic". All sensor readings are noisy, so they can patently be modeled by probability distributions. If one chooses to model measurements by Gaussians, i.e.,

using a mean and a standard deviation, solving SLAM reduces to solving a system of linear equations [9]. Closed loops, i.e., a second encounter of a previously visited area of the environment, play a special role in SLAM algorithms. Once detected, they enable the algorithms to bound the global error by deforming the already mapped area to make the model locally consistent. However, there is no guarantee for the model to be correct.

Global relaxation techniques can be divided into two major categories. First, direct methods establish correspondences between features, i.e., they address the data association problem and minimize the overall error in the SLAM graph. EKF based methods like [6, 8, 17] are examples. These methods are computationally expensive, since large linear equation systems have to be solved. The number of unknown variables depends on the number of poses and on the number of features to be estimated. Furthermore, feature detection and association need to be reliable, and difficulties occur due to linearization [9]. The second category is based on iterative methods, which overcome the feature extraction and data association problem. The input usually consists of unprocessed scan data, and correspondences between poses are computed based on closest data points. An example is the method by Lu and Milios [16] for 2D scans and its extension to 3D [4]. These algorithms solve systems of linear equations, too, to yield pose estimations. They iterate two steps, namely, scan matching and pose estimation, to compute a consistent global map. Loop closing is performed by adding additional edges, iff the robot encounters a position close to another where it had been before [12, 15, 23].

Since SLAM implies solving a system of linear equations when updating a single map hypothesis, the computational requirements are high, due to increasing matrix sizes during exploration and mapping. [15] presents a divide and conquer algorithm to handle large matrices, but it still suffers from the iterative approach of Lu and Milios style SLAM.

To build fast SLAM backends, Olson and Grisetti have proposed methods for distributing the error during loop closing over the SLAM graph [11, 20, 21]. The result corresponds to a fast solution of the linear system of equations, which is based on exploiting the graph structure. Similarly, [14] presents a reordering of the equations to compute the solution faster. [3] consequently exploits the sparseness of the equation system, obtaining similar results. [22] presents a divide and conquer method for the EKF SLAM approach. The tree map algorithm of Frese uses a partition of the map as well, and yields an approximative solution to SLAM [10]. Graph simplification is used in [8], aiming at reducing the number of vertices in the SLAM graph, thus reducing the number of equations. Note: All these SLAM backend approaches have to be combined with a SLAM frontend, i.e., with data association or scan matching. In the scan matching case, both methods have to be iterated. The assumption is that the point correspondences are correct in the last iteration. Grisetti et al. close a loop by using a spanning tree [11] to distribute the error.

In contrast to the previously mentioned algorithms, rather simple methods to distribute the error in a single closed loop have been proposed. They distribute the error uniformly in the loop, or weighted by the path length [18]. However, this technique is incapable of handling multiple, intertwined loops.

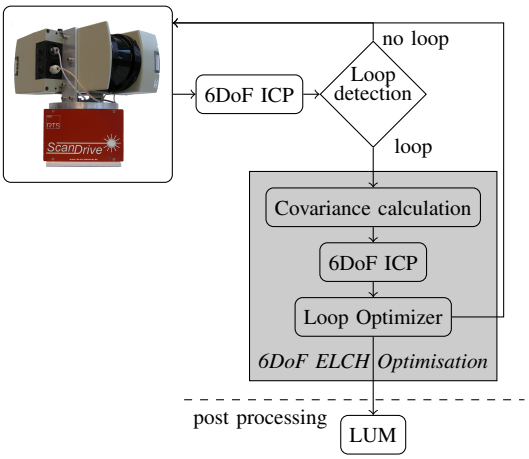


Fig. 2. Schematic overview of the complete algorithm, namely the interaction of ICP, ELCH and LUM. The dashed line separates the global optimization part that is executed as a post processing step.

III. Explicit Loop Closing

A. Loop Detection and Graph Construction

Our previous work [4] to consistent mapping has applied global Lu/Milios style relaxation when a closed loop is detected (LUM). The procedure has been shown to perform well in many applications, but it suffers from its high computational complexity. When mapping large-scale outdoor environments, the scenes may contain hundreds of 3D scans. The global relaxation has to iterate the SLAM front- and backend. We have to compute closest point correspondences for all links in the SLAM graph in every iteration.

Explicit loop closing is proposed here to overcome this problem. When detecting a closed loop, scan matching is applied to transform the last acquired scan. This transformation *dissociates* the last vertex from the current SLAM graph and yields a transformation vector $\Delta\mathbf{X}$ that consists of a rotation $\mathbf{R}$ and translation $\mathbf{t}$. An additional effect of scan matching is that the last vertex is moved to a position with minimal error with respect to the first vertex of the loop. Afterwards, the transformation vector $\Delta\mathbf{X}$ has to be distributed over the SLAM graph, i.e., over the previously encountered poses. In our current system, global LUM style optimization is used as a post processing step to improve global consistency of the map.

The following subsections describe the algorithm in detail. The algorithm operates on a graph $G = (V, E)$, where the set of vertices V is given by the scan poses $\mathbf{X}$ and the set of edges E contain pairs of vertices that were already matched with ICP. Edges are labeled with the covariances $\mathbf{C}_{l,k}$ that approximate the uncertainty of the connected poses v_l and v_k.

B. Loop Closing using ICP

Using the robot trajectory estimated by means of the local ICP algorithm, we detect loops in the path using the Euclidean

distance between the current and all previous poses (distance threshold of 5 meters), or using GPS data if available. A threshold of minimal number of intermediate scans (e.g., 20) is used to circumvent continuous loop closing within consecutive scans. Given the first and last scan of a detected loop, we build two small metascans consisting of only few scans (here: two) around the first and last scan, respectively, and match those metascans using ICP. The difference in the pose of the last scan before and after application of ICP yields a transformation error $\Delta\mathbf{X}$ that has to be distributed between all poses on the loop, preserving the topology of the map. For example, laser scans that are near to another scan of the loop should still be close to the same scan after applying the deformation.

After the error distribution, only *one* edge is added to the graph, connecting the first and the last scan of the loop. Fig. 4 (right) emphasizes the difference of our SLAM graphs and graphs used for solving SLAM in [4, 11, 20, 21], where any two vertices that are close enough are connected.

C. Loop Optimization in SLAM Graphs

To motivate our graph optimization algorithm, consider the following two examples. Fig. 3 (left) presents a graph of a simple loop, where vertex E closes the loop to vertex A. The edges are labeled with the relative error between the connected vertices. We aim at calculating weights for the vertices that specify the fraction of the vector $\Delta\mathbf{X}$ by which the pose need to be changed to achieve a consistent map. It is obvious that vertex E has to be transformed by $\Delta\mathbf{X}$ while vertex A does not need to be transformed at all. The remaining vertices, except F, G, and H are adjusted by a fraction $w_i \in [0, 1]$ of the vector $\Delta\mathbf{X}$. The weight w_i of the vertex v_i is computed as follows:

$$w_i = \frac{d(v_s, v_i)}{d(v_s, v_e)},$$

where v_s is the first vertex in the loop and v_e last one. $d(v_l, v_k)$ specifies the summed uncertainties between the vertices v_l and v_k using the sum of the edge weights $c_{i,j}$ on the way, i.e.,

$$d(v_l, v_k) := \sum_{\substack{\text{edge}\{i,j\} \in \text{Path} \\ \text{from } v_l \text{ to } v_k}} c_{i,j}. \tag{1}$$

The attached vertices F, G, and H are adjusted in the same way as the vertex of the loop is transformed. The table on the right specifies the values.

Note: The example presents a graph with vertices specified as scalars, thus we use a scalar $c_{i,j}$ as edge weight. In case of k-dimensional input, we *dissociate* every single coordinate and decompose the problem into k subproblems. Hence, only the diagonal of the covariance matrix, i.e., the variances, is used in the k-dimensional case. In case of using SLERP we have $k = 4$ [11, 24], i.e., three for the position and one for the quaternion describing the rotation. Covariances are computed as described in [19].

As a second example we use the graph in Fig. 3 (right) with two alternate pathways. Since we want to distribute the $\Delta\mathbf{X}$ with the smallest possible error, we find the *shortest* path between the two loop closing vertices. After the correction is distributed over the shortest path, we adjust the remaining pathways to achieve consistency. To obtain the updates, we recursively exploit the same algorithm as for the main loop, but with the already computed weights of the start and end of the alternate path, instead of the default weighting of 0 and 1.

D. The Loop Optimizer Algorithm

To compute the weights for arbitrary graphs, we propose the Loop Optimizer Algorithm (LOA) as listed in Algorithm 1. Its input is a connected, undirected graph $G = (V, E)$ with two special vertices v_f and v_l, specifying the first and the last vertex of a closed loop. The weights associated with v_f and v_l are set to 0 and 1, respectively, and both are added to a set Ω that holds vertices for later processing. The first part of the algorithm searches for all loops in the graph, using the Dijkstra algorithm. To this end, it iterates over the set Ω until all loops are processed. Dijkstra's algorithm is started for all elements of Ω to compute a path from Ω into Ω and the overall shortest one is used (starting at v_s and ending at v_e). On its first iteration, these vertices will be v_f and v_l, as these are the only vertices in Ω. A collateral outcome of the Dijkstra algorithm is the path cost to reach a vertex v_i from the start vertex v_s, which is equal to $d(v_s, v_i)$, as defined in (1). Based on these costs, we update the weights (w_i) of the vertices (v_i) on this path according to

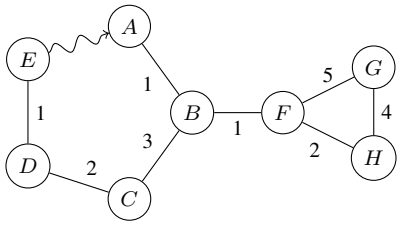
$$w_i = w_s + \frac{d(v_s, v_i)}{d(v_s, v_e)}(w_e - w_s).$$

By updating the vertices on the shortest path, we detect junctions, i.e., vertices whose degree is greater than 2, and add these vertices to the set Ω for later processing. Afterwards, we remove the processed path, i.e., the edges from the graph G, such that these edges are not used again, and remove the first and last vertex v_s and v_e, iff their degrees are reduced to zero. The algorithm is then iterated over the remaining set Ω, thus we process all loops connected to the loop closed by the vertices v_f and v_l. After repeating this algorithm for every path doubly connected to the main loop, vertices of a path that has only one connection to the main loop remain in Ω. These are finally processed by simply distributing their connecting weight to all vertices on such a path.

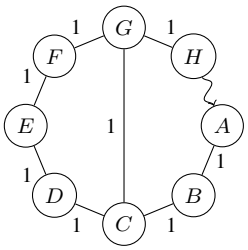
E. 6D SLAM with an Explicit Loop Closing Heuristic

Assume that in the process of acquiring and matching scans consecutively using ICP, we detect a closed loop. The Explicit Loop Closing Heuristics (ELCH) starts with covariance calculation of adjacent poses, after matching the 3D scans that form the closed loop, yielding a 4-dimensional translation vector $\Delta\mathbf{X}$. The LOA algorithm is executed separately for every dimension. Computing a fraction of a possibly large rotation cannot be performed by using Euler angles, since these consist of three angles that depend on each other. SLERP does not have this property and is therefore usually used for interpolation tasks.

In a post processing step we still iterate the scan matching and update the poses as presented in [4] to slightly improve overall consistency. This step is necessary since the difference



Vertex	Weight
A	0
B	$1/7$
C	$4/7$
D	$6/7$
E	1
F	$1/7$
G	$1/7$
H	$1/7$



Vertex	Weight
A	0
B	$1/4$
C	$2/4$
D	$9/16$
E	$10/16$
F	$11/16$
G	$3/4$
H	1

Fig. 3. Left: Graph with an extra branch connected to only one vertex. The table shows the computed weights of the vertices. Right: Two alternate circles and the resulting weights. The shortest path from A to H is A-B-C-G-H.

between the first and last scan of a loop and its distribution over the SLAM graph alone does not yield a map with an overall minimal error. Using LUM takes significantly more iterations to minimize the error and to close loops, because in every iteration the loops are closed in small steps. This strategy always considers all edges in the SLAM graph, while we gain performance by initially *dissociating* one link and adding it back afterwards.

So, the time-consuming iteration over all scans during acquisition is avoided, allowing a much larger number of scans to be handled in reasonable time. Experiments described in the next section confirm that the reduction is in fact significant (Table I). Since the number N of data points in a single scan (in the order of 30,000) is typically still larger than the number of scans n (in the order of 1,000), the run time is dominated by computing the scan matching, which is in $\mathcal{O}(N \log N)$. The required Dijkstra algorithm is implemented in $\mathcal{O}(n \log n)$ time for each loop.

IV. Experiments and Results

In this paper, we use the publicly available dataset Hannover2, provided by Oliver Wulf, Leibniz University Hannover, to evaluate our algorithm. The data set, part of the Robotic 3D Scan Repository [1], has been acquired in an urban area and consists of 924 3D scans, each containing up to 35,000 3D data points (cf. Fig. 1). The mobile robot Erika [29] uses a continuously rotating 3D scanner to deliver the data. In [29] a benchmark for this data set has been presented using 6D SLAM with ICP and LUM. Although parts of the path are traversed repeatedly, as described in the caption of Fig. 1, only two distinctive loop closing events are triggered, marked as a and b in Fig. 4.

To evaluate the map computed by our algorithm, some kind of ground truth is necessary. In [29] we presented a method to compute planar reference poses and a reference orientation about the vertical axis. For the evaluation in this paper we extend our results: A 2D ground truth map of the area is provided by the German land registry office (*Katasteramt*). It contains the buildings with a precision of 1 cm. In addition, we obtained airborne based 3D data. Based on this data, so-called reference data is generated as follows (see Figure 5): The 2D map is extrapolated to 3D by vertical 3D points and fused with the 3D data from the airplane. The result is a precise 3D reference map. Using this 3D reference map, we generate ground truth poses for all 924 3D laser scans by matching the scans with the reference map. We will refer to these poses as "ground truth".

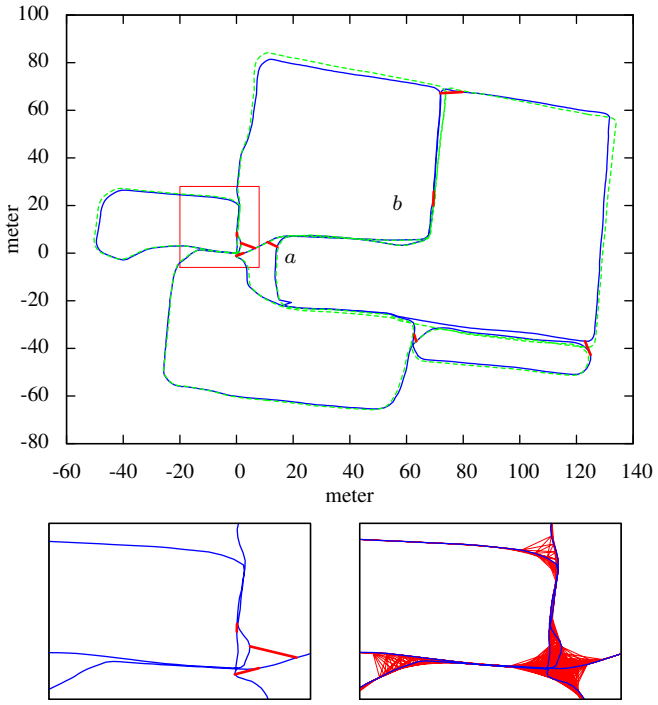


Fig. 4. Left: Ground truth (- -) and ELCH corrected (—) trajectory with closed loops marked as — (please refer to the pdf file for a color versions). Right: Zoom into the box drawn on the left shows the loops closing edges (—) of our ELCH algorithm (top) and of reference strategies (bottom) [4, 11, 20, 21]. At a the robot starts to drive the same track, so a loop closing event is triggered. At b the trajectories diverged so far that a new loop is closed.

For analyzing the optimization of different SLAM strategies, we plot the error in the transformation of three exemplary scans. To visualize the translational error of every iteration, we use the Euclidean distance of the scan to the ground truth position as

$$e_{\text{translation}} = \left|\left|\hat{\mathbf{X}}_i - \hat{\mathbf{X}}_{i,\text{ref}}\right|\right|,$$

where a scan pose $\mathbf{X}$ is defined as $(\hat{\mathbf{X}}, \tilde{\mathbf{X}})^T$, with $\hat{\mathbf{X}} = (x, y, z)^T$. To describe the rotational error, we convert the rotation part $\tilde{\mathbf{X}}$ of their poses $\mathbf{X}$ into the quaternion representation, i.e., $\tilde{\mathbf{X}} = (p, q, r, s)^T$. The inner product of it yields the angle between the two 4 dimensional vectors.

$$e_{\text{rotation}} = \arccos \left|\tilde{\mathbf{X}}_i \cdot \tilde{\mathbf{X}}_{i,\text{ref}}\right|.$$

Fig. 6 presents the rotational and translational errors for two scans while executing our new strategy ELCH in comparison

Algorithm 1 Loop Optimizer

Input: Graph $G = (V, E)$
first vertex v_f
last vertex v_l
edge costs $c_{l,k} : E \to \mathbb{R}_+$
Output: vertex weights $w_i : V \to [0, 1]$

$w_f \leftarrow 0$
$w_l \leftarrow 1$
$\Omega \leftarrow \{v_f, v_l\}$ /* Loop Closing */
/* Dijkstra returns a path $p := (v_s, v_1, v_2, \ldots, v_n, v_e)$ */
/* and minimal costs $d(v_s, v_s), d(v_s, v_1), \ldots, d(v_s, v_e)$ */
while find shortest path between any two vertices $\{v_s, v_e\} \in \Omega$ with Dijkstra **do**
  for all vertices v_i on the path p **do**
    $w_i \leftarrow w_s + \frac{d(v_s,v_i)}{d(v_s,v_e)}(w_e - w_s)$
    if $deg(v_i) > 2$ **then** /* i.e. junction */
      $\Omega = \Omega \cup \{v_i\}$
    end if
  end for
  remove edges of path p in G
  if $deg(v_s) = 0$ **then**
    $\Omega = \Omega \setminus \{v_s\}$
  end if
  if $deg(v_e) = 0$ **then**
    $\Omega = \Omega \setminus \{v_e\}$
  end if
end while
while $\Omega \neq \emptyset$ **do** /* Error Propagation */
  select $v_i \in \Omega$
  for all neighbors v_n of v_i **do**
    $w_n \leftarrow w_i$
    delete edge $\{v_i, v_n\}$
    if $deg(v_n) > 0$ **then**
      $\Omega = \Omega \cup \{v_n\}$
    end if
  end for
  $\Omega = \Omega \setminus \{v_i\}$
end while

to our previous strategy LUM. The corresponding scans have been colored yellow in Fig. 1. To give greater detail, we zoom into the curves of the ELCH algorithm on the right side, showing the individual steps. In the first iteration (white background, very small), the initial pose correction is applied. Then (shown in very light gray), the ICP algorithm registers the scan, relative to the previous scan. The third step (in light gray) depicts the error during the ICP loop closing iterations. Scans not at the end of a closed loop do not have this step. Fourth (gray) are the corrections of LOA, and last (dark gray) the iterations of the final relaxation algorithm can be seen.

The left part shows scan 344 which closes the second loop, thus the two compared strategies display different initial errors. We notice that LUM starts to move the scan slowly into the right position, getting faster when it is almost done. The reason is that the movement is forced by the other scans of the loop at this stage of the procedure, rather by the loop closing constraint itself. The ELCH algorithm, on the other hand, has only the loop closing constraint, such that the ICP algorithm starts converging later, yet in fewer iterations.

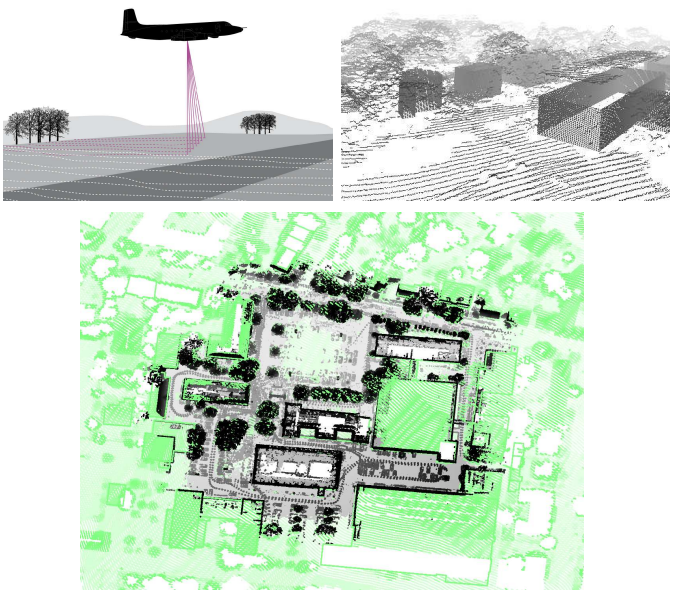


Fig. 5. Top left: Schema of the airborne based acquisition of reference data. Top right: 3D map consisting of aerial laser data and extrapolated 2D reference data. Bottom: Airborne and 3D map (green) with superimposed 3D scans (black).

TABLE I
RUNTIME COMPARISON ON AN INTEL CORE 2 QUAD AT 2.66 GHZ WITH 4 GB RAM PROCESSING THE COMPLETE DATASET HANNOVER2.

Algorithm	**runtime (sec)**
LUM	4831
ELCH (without post processing)	89
ELCH (with post processing LUM)	384

As a second example (Fig. 6, right) is the 556$^{\text{th}}$ scan (label H in Fig. 1). Again, it is a scan that closes a loop, so the ICP loop closing iterations are shown. Being a small loop with no big error, the correction of it is quite small. Note that this scan is far away from the origin and not connected to any other loops when it is first corrected, so LOA changes its position again once it gets connected to other loops. As in all three scans, the LUM algorithm at the end of the ELCH part converges much faster because of the previous corrections. This is visible in table I, too, where we compare the runtime of the different strategies. An animated comparison between ELCH and LUM can be seen at `http://kos.informatik.uni-osnabrueck.de/download/elch/` [1].

V. OUTLOOK AND CONCLUSION

This paper has introduced a novel approach to scan matching based GraphSLAM. The usual approach to loop closing

[1]The video compares our previous strategy LUM (on the left) with our new strategy ELCH. It shows the different steps during computation the elapsed computing time.

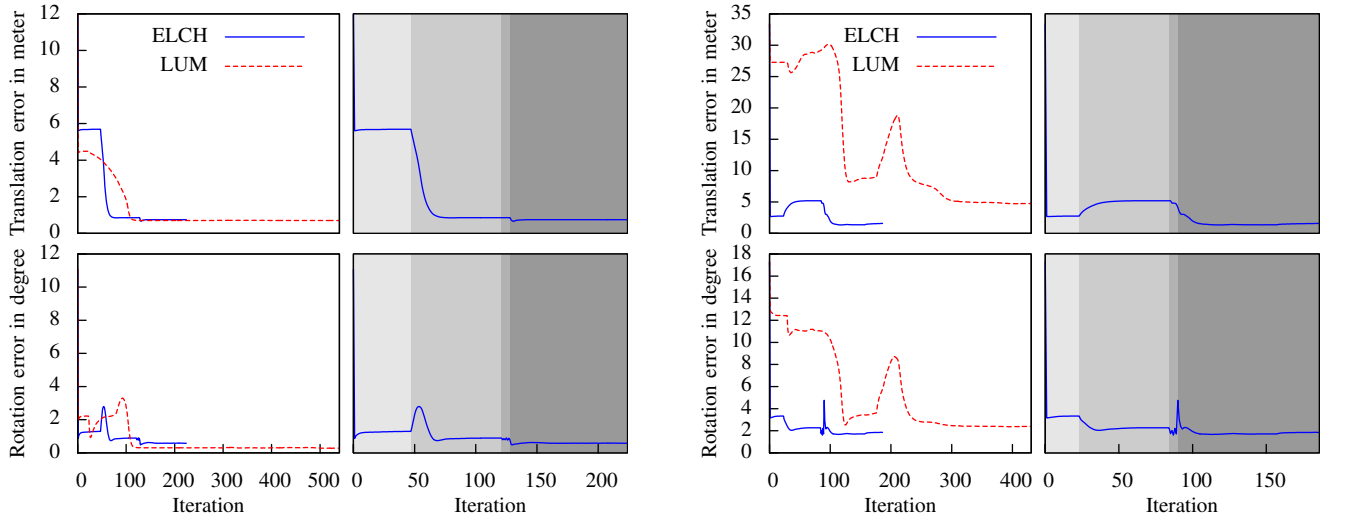


Fig. 6. Convergence of the 344th and the 556th scan. The right side shows the stages of ELCH, namely ICP, ELCH ICP, LOA and LUM

is to build a graph of poses and optimize it afterwards by iterating scan matching and graph optimization. Our approach *dissociates* the scan that closes the loop from its previous scan and registers it explicitly. The resulting offset is distributed over the SLAM graph such that a minimal error occurs, with respect to the uncertainties in correlated poses.

In addition, this paper has applied a new method for evaluating the accuracy of 3D scans acquired in an outdoor setting, extending our benchmarking attempts in [18]. We use independently acquired aerial 3D scans in combination with a 2D reference map from the land registry office as genuine truth. This enabled us to measure derivations with 6 degrees of freedom.

The ELCH algorithm proposed in this paper yields an improved structure of the scene that speeds up the global post processing step LUM, if not enabling a correct global optimization in the first place by closing loops smartly. The major part of the algorithm's run time, however, is consumed by this post processing step, which is still necessary to obtain a globally consistent optimization at a detailed level. Thus, further research will be invested on speeding up the LUM algorithm.

References

[1] 3D Scan Repository, 2008. http://kos.informatik.uni-osnabrueck.de/3Dscans/

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

[3] D. Borrmann, J. Elseberg, K.Lingemann, A. Nüchter, and J. Hertzberg. The Efficient Extension of Globally Consistent Scan Matching to 6 DoF. In *Proc. of the 4th Int. Symp. on 3D Data Processing, Visualization and Transmission (3DPVT '08)*, pages 29–36, Atlanta, GA, USA, June 2008.

[4] D. Borrmann, J. Elseberg, K. Lingemann, A. Nüchter, and J. Hertzberg. Globally Consistent 3D Mapping with Scan Matching. *J. Robotics and Autonomous Sytems*, 65(2):130–142, 2008.

[5] DARPA. http://www.darpa.mil/grandchallenge/, 2007.

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

[7] FGAN. http://www.elrob.org/, 2008.

[8] J. Folkesson and H. I. Christensen. Graphical SLAM for Outdoor Applications. *J. of Field Robotics (JFR)*, 24(1–2):51–70, February 2007.

[9] U. Frese. A Discussion of Simultaneous Localization and Mapping. *Autonomous Robots*, 20(1):25–42, 2006.

[10] U. Frese. Efficient 6-DOF SLAM with Treemap as a Generic Backend. In *Proc. of the IEEE ICRA*, Rome, Italy, April 2007.

[11] G. Grisetti, C. Stachniss, and W. Burgard. Non-linear constraint network optimization for efficient map learning. *IEEE Transaction on Intelligent Transportation Systems*, 2008.

[12] J.-S. Gutmann and K. Konolige. Incremental Mapping of Large Cyclic Environments. In *Proc. IEEE CIRA*, 2000.

[13] D. Hähnel and W. Burgard. Probabilistic Matching for 3D Scan Registration. In *Proce. 2nd German conference on robotics (ROBOTIK '02)*, Ludwigsburg, Germany, June 2002.

[14] M. Kaess, A. Ranganathan, and F. Dellaert. iSAM: Fast incremental smoothing and mapping with efficient data association. In *Proc. IEEE ICRA*, pages 1670–1677, Rome, Italy, 2007.

[15] K. Konolige. Large-scale map-making. In *Proc. AAAI*, 2004.

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

[17] P. Newman and Kin Ho. Slam-loop closing with visually salient features. In *Proc. IEEE ICRA*, pages 635–642, 2005.

[18] A. Nüchter, K. Lingemann, J. Hertzberg, and H. Surmann. 6D SLAM - 3D Mapping Outdoor Environments. *J. of Field Robotics, Special Issue on Quantitative Performance Evaluation of Robotic and Intelligent Systems*, 24(8/9):699–722, 2007.

[19] Andreas Nüchter. *3D Robotic Mapping – The Simultaneous Localization and Mapping Problem with Six Degrees of Freedom*, volume 52 of *STAR*. Springer, Heidelberg, 2009.

[20] E. Olson, J. Leonard, and S. Teller. Fast iterative alignment of pose graphs with poor initial estimates. In *Proc. IEEE ICRA*, 2006.

[21] Edwin Olson. *Robust and Efficient Robotic Mapping*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 2008.

[22] L.M. Paz, J.D. Tardos, and J. Neira. Divide and Conquer: EKF SLAM in $O(n)$. *IEEE Transactions on Robotics*, 24(5):1107–1120, 2008.

[23] P. Pfaff, R. Triebel, and W. Burgard. An efficient extension to elevation maps for outdoor terrain mapping and loop closing. *IJRR*, 26, 2007.

[24] K. Shoemake. Animating rotation with quaternion curves. *Computer Graphics*, 19(3):245 – 254, July 1985.

[25] H. Surmann, K. Lingemann, A. Nüchter, and J. Hertzberg. A 3D laser range finder for autonomous mobile robots. In *Proc. ISR*, April 2001.

[26] The RoboCup Federation. http://www.robocup.org/, 2008.

[27] S. Thrun. Robotic mapping: A survey. In G. Lakemeyer and B. Nebel, editors, *Exploring Artificial Intelligence in the New Millenium*. 2002.

[28] S. Thrun et al. Winning the darpa grand challenge. *J. of Field Robotics (JFR)*, 23(9):661 – 692, August 2006.

[29] O. Wulf, A. Nüchter, J. Hertzberg, and B. Wagner. Benchmarking Urban Six-Degree-of-Freedom Simultaneous Localization and Mapping. *J. of Field Robotics*, 25(3):148–163, 2008.

[30] O. Wulf and B. Wagner. Fast 3D-scanning methods for laser measurement systems. In *Int. Conf. Control Systems and Computer Science*, 2003.