Automatika

Journal for Control, Measurement, Electronics, Computing and Communications

ISSN: 0005-1144 (Print) 1848-3380 (Online) Journal homepage: http://www.tandfonline.com/loi/taut20

A Heuristic Loop Closing Technique for Large-Scale 6D SLAM

Jochen Sprickerhof, Prof. Andreas Nüchter, Kai Lingemann & Prof. Joachim Hertzberg

To cite this article: Jochen Sprickerhof, Prof. Andreas Nüchter, Kai Lingemann & Prof. Joachim Hertzberg (2011) A Heuristic Loop Closing Technique for Large-Scale 6D SLAM, Automatika, 52:3, 199-222

To link to this article: http://dx.doi.org/10.1080/00051144.2011.11828420

Online ISSN 1848-3380 , Print ISSN 0005-1144
ATKAFF 52(3), 199–222(2011)

Jochen Sprickerhof, Andreas Nüchter, Kai Lingemann, Joachim Hertzberg

A Heuristic Loop Closing Technique for Large-Scale 6D SLAM

UDK 004.92.023
IFAC 2.8; 1.1.9

Original scientific paper

This paper presents a novel heuristic for correcting scan pose estimations after loop closing in SLAM using 3D laser scans. Contrary to state of the art approaches, the built SLAM graph is sparse, and optimization is done without any iteration between the SLAM front and back end, yielding a highly efficient loop closing method.

Several experiments were carried out in an urban environment and evaluated against ground truth. The results are compared to other state of the art algorithms, proving the high quality, yet achieved faster by an order of magnitude.

Key words: 3D robotic mapping, Simultaneous localization and mapping, Loop closing

Heuristička metoda zatvaranja petlje za 6D SLAM velikih dimenzija. Članak prikazuje novu heuristiku čija je svrha korekcija estimacije pozicija očitanja 3D lasera nakon zatvaranja petlje u SLAM-u. U suprotnosti s postojećim pristupima, dobiveni je SLAM graf rijedak te se optimizacija provodi samo pri detekciji zatvaranja petlje, rezultirajući vrlo učinkovitom metodom.

Provedeno je nekoliko eksperimenata u urbanom okruženju i uspoređeno s točnim vrijednostima. Rezultati su također uspoređenim s postojećim algoritmima, čime je dokazana visoka kvaliteta algoritma uz red veličine veću brzinu izvođenja.

Ključne riječi: izgradnja karte u 3D mobilnim robotom, istovremena lokalizacija i izgradnja karte, zatvaranje petlje

1 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 [1, 2], in competitions such as Robocup Rescue [3], the European Land Robotics Trial ELROB [4] or the DARPA Grand Challenge [5]. However, most of the approaches aim at mapping small environments or at constructing 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[1]. We aim at reducing the run time of our mapping system such that it performs fast in large environments using 3D laser scans. The main achievement is a new algorithm for efficient loop closing and consistent scan alignment that avoids iterative scan matching over

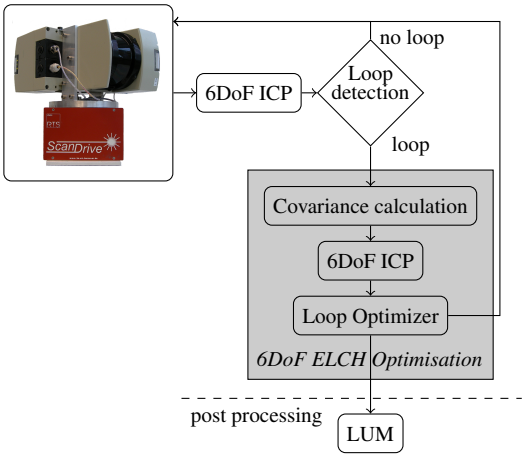


Fig. 1. Schematic overview of the complete algorithm, namely the interaction of ICP, ELCH and LUM. The dashed line separates the global optimization part that is executed one single time as a post processing step.

[1]This paper is an extended version of [6]

all scans, thereby outperforming comparable algorithms used in SLAM previously. Figure 1 depicts a schematic overview of the complete algorithm. The algorithm operates with 6D poses, i.e., is able to handle robot motion with six effective degrees of freedom (translation and rotation).

2 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 robotics in environment sensing has led from initially custom made 3D scanners, as in [7–9], 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 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. Figure 2 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 [12]. However, any incremental application of such matching algorithms leads to inconsistencies due to inaccurate sensor data and due to accumulating registration tolerances. To avoid these problems, global matching algorithms are needed, taking global correspondences between all scans simultaneously into account [13].

Sensor noise is modeled by probability distributions over measurements in probabilistic mapping approaches [14]. All sensor readings are noisy, and can 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 [15]. 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 [16–18] 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 [15]. The second category is based on iterative methods, which overcome the feature extraction. 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 [19] for 2D scans and its extension to 3D [13]. These algorithms also solve systems of linear equations, 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 [20–22]. Thus edges represent matchable 3D scans.

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. Konolige presents a divide and conquer algorithm to handle large matrices [21], but it still suffers from the iterative approach of Lu and Milios style SLAM.

To build fast SLAM back ends, Olson and Grisetti have proposed methods for distributing the error during loop closing over the SLAM graph [23–25]. The result corresponds to a fast solution of the linear system of equations, which is based on exploiting the graph structure. Similarly, Kaess et al. presents a re-ordering of the equations to compute the solution faster [26]. Borrmann et al. consequently exploits the sparseness of the equation system, obtaining similar results [27]. Paz et al. presents a divide and conquer method for the EKF SLAM approach [28]. The tree map algorithm of Frese uses a partition of the map as well, and yields an approximative solution to SLAM [29]. Graph simplification is used in [17], aiming at reducing the number of vertices in the SLAM graph, thus reducing the number of equations. Note: All these SLAM back end approaches have to be combined with a SLAM front end, 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 to iteratively distribute the error [23].

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 [1]. However, this technique is incapable of handling multiple, nested loops.

In contrast to almost all previous mentioned approaches, we propose the ELCH strategy, which does not iterate, and uses a sparse pose graph representation. Both ideas make

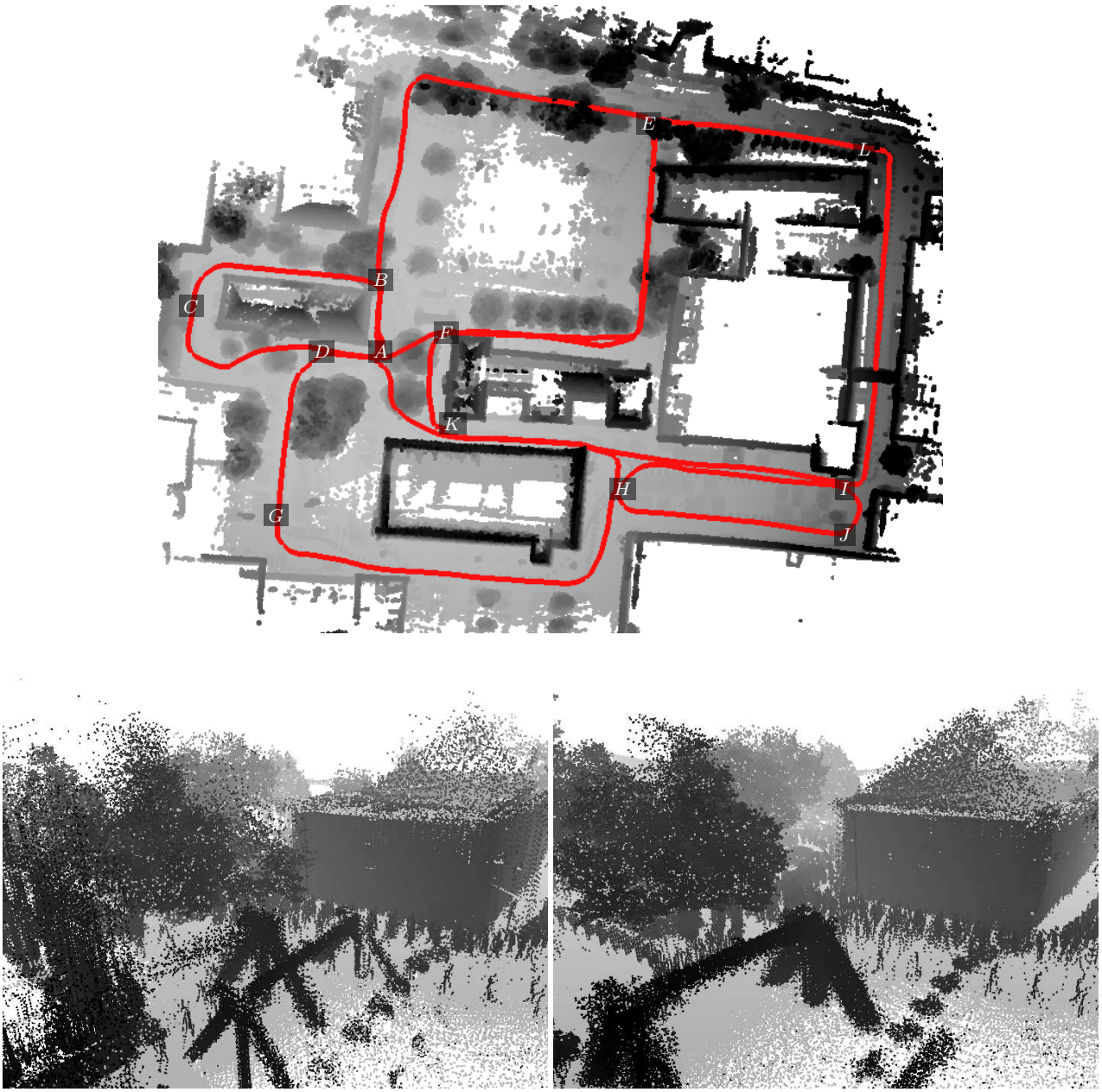


Fig. 2. Example trajectory (top view) that was automatically acquired by a mobile robot [10]. The overall time needed to process this data with our `slam6d` *software [11] was reduced from 18.2 minutes to 2.9 minutes, which is faster than acquisition time. The scans have been taken according to the sequence: 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.*

the proposed solution unique and faster than all competitive methods.

3 EXPLICIT LOOP CLOSING

Our previous work about consistent mapping [13] has applied global Lu/Milios style relaxation when a closed loop is detected (cf. Figure 3, LUM). The procedure proved to perform well in many applications, but suffers from its high computational complexity. When mapping large-scale outdoor environments, the scenes may contain hundreds of 3D scans, and the global relaxation has to iterate the SLAM front and back end. When applying global relaxation, closest point correspondences have to be computed anew in every iteration. This is inherently time-consuming.

3.1 Loop Detection and Graph Construction

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 first *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 $\mathbf{t}$. Dissociation here means to disregard for the moment all constraints between the loop closing scan pose and the previous scan poses represented as vertices in the SLAM graph; after the loop-closing scans are matched locally, the offset determined in the matching is propagated through the SLAM graph, using the previous constraints. 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 one single time for the otherwise completed map to improve global consistency of the map.

The following subsections describe the algorithm in detail. It 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 variances, that approximate the uncertainty of the connected poses v_l and v_k.

3.2 Loop Closing based on initial ICP registration

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 usually 15 meters), or using GPS data if available. A threshold of minimal number of intermediate scans (e.g., 20) is used to circumvent vacuous loop closing within consecutive scans. This simple heuristic was sufficient to produce the results shown in Section 4. More complex approaches can be used as well, if available. At this stage, we simply assume that the software has *some* method of recognizing loops, which is not the focus of this paper; we will concentrate on how to employ these closed loops efficiently instead.

Given the first and last scan of a detected loop, we build small metascans at its two ends consisting of only few scans (here: two) around the first and last scan, respectively, and match those metascans using ICP. This is done by ignoring all previous constraints, called dissociating of the scan. 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. Figure 4 emphasizes the difference of our SLAM graphs and graphs used for solving SLAM in [13, 23–25], where any two vertices that are close enough are connected. Having an order of magnitude less of edges in the SLAM graph, we do not have to compute the influences during the relaxation phase. This saves memory as well as computing time.

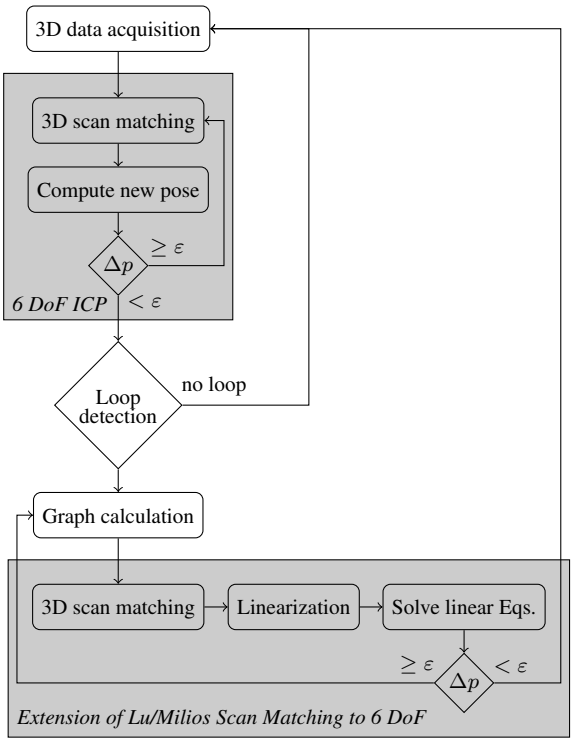
3.3 Loop Optimization in SLAM Graphs

To motivate our graph optimization algorithm, consider the following three examples; the algorithm is described for graphs in general in Section 3.5. Figure 5 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, i.e. the pose variances as explained above. 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 weight w_i of the vertex v_i is computed as follows:

$$w_i = \frac{d(v_s, v_i)}{d(v_s, v_e)}, \tag{1}$$

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

$$d(v_l, v_k) := \sum_{\substack{\text{edge}\{i,j\} \in \text{Path} \\ \text{from } v_l \text{ to } v_k}} c_{i,j}. \tag{2}$$



$\mathcal{O}(N \log N)$	(I)
$\mathcal{O}(N)$	(II)
$\mathcal{O}(1)$	(III)
$\mathcal{O}(n)$	(IV)
$\mathcal{O}(n^2)$	(V)
$\mathcal{O}((l+n)N \log N), \mathcal{O}(n^2), \mathcal{O}(n^3)$	(VI)
$\mathcal{O}(1)$	(VII)

Fig. 3. The interaction of ICP and LUM in 6D SLAM. See [13] for details. The threshold ε, used to determine if a scan changed its pose, is the same in both parts of the algorithm. The right part of the figure denotes the run times. N is the number of points in a single 3D scan, n denotes the number of poses, and l is the number of loops.

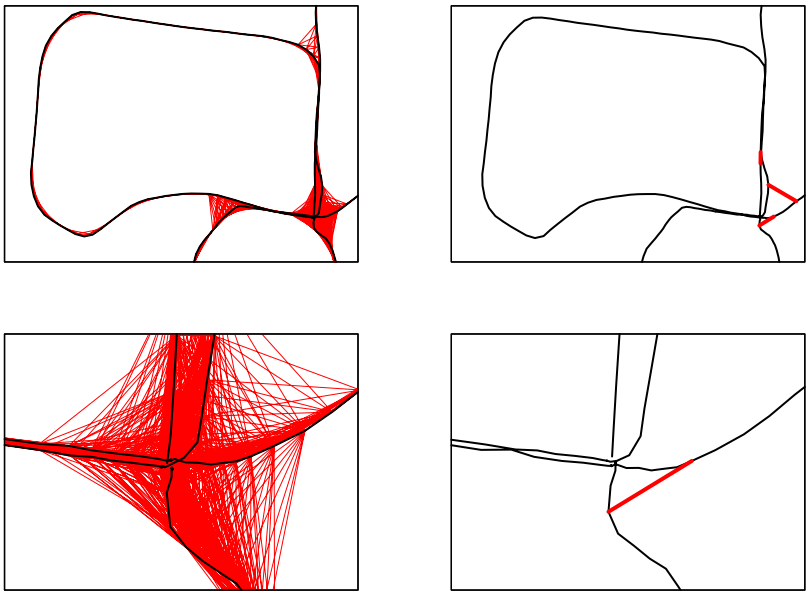
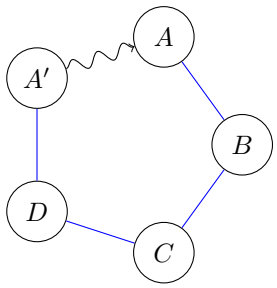


Fig. 4. Comparison of connectivity of different SLAM graphs. Left: State of the art graph SLAM approaches. Right: Method of this paper. The bottom figures are zoomed views of the plots above.



Vertex	Weight
A	0
B	$1/7$
C	$4/7$
D	$6/7$
A'	1

Fig. 5. Simple loop, starting at vertex A and ending at vertex E and the resulting weights for every vertex, as computed by the Loop Optimizer Algorithm. All edges are trail edges (blue), cf. Section 3.4.

In the first example (Figure 5), we specified the resulting weighting of the vertices in the table on the right.

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 separate every dimension 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, because the off diagonal elements are usually small. Using SLERP [30] we have $k = 4$, i.e., three for the position and one for the quaternion describing the rotation. Covariances are computed as described in Section A.2.

The second example is shown in Figure 6. It extends our first example by a path, attached to a single vertex. In this case, the attached vertices are adjusted in the same way as the vertex of the loop is transformed. The table on the right specifies the values.

As a third example we use the graph in Figure 7 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.

The described strategy could cope with arbitrary graphs, as shown in the next Section.

3.4 Arbitrary Graphs

The proposed ELCH strategy was primarily developed for SLAM graphs, but it is applicable to arbitrary graphs, as we will show with the two following definitions. In these, a trail is defined as a path where every vertex is only visited once (cf. for example Bondy [31]).

Definition 1 (Trail edge) *Given two vertices v_f and v_l in an undirected graph $G = (V, E)$, we name an edge e* trail edge *with respect to v_f and v_l, if there is a trail from v_f to v_l containing the edge e.*

Definition 2 (Branch edge) *Given two vertices v_f and v_l in an undirected graph $G = (V, E)$, we name an edge e* branch edge *with respect to v_f and v_l, if there is* no *trail from v_f to v_l containing the edge e.*

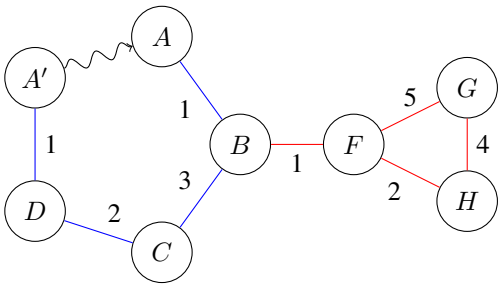
Following this definitions every edge in an arbitrary graph is either a trail edge or a branch edge.

3.5 The Loop Optimizer Algorithm

To compute the weights for arbitrary graphs, we propose the Loop Optimizer Algorithm (LOA) as listed in Algorithm 1, which is built on the previous two definitions. Its input is an arbitrarily 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 Equation (2). Based on these costs, we update the weights (w_i cf. Equation 1) of the vertices (v_i) on this path according to

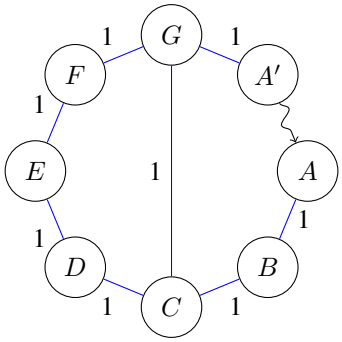
$$w_i = w_s + \frac{d(v_s, v_i)}{d(v_s, v_e)}(w_e - w_s).$$

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.



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

Fig. 6. Graph with an extra branch connected to only one vertex. The computed weights of the main loop are the same as in Figure 5, whereas the weight of vertex B is distributed to the extra vertices. Trail edges are marked blue, branch edges red, cf. Section 3.4.



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

Fig. 7. Two alternate circles and the resulting weights. The shortest path from A to H is A-B-C-G-A'. This graph contains only trail edges (marked blue), cf. Section 3.4.

3.6 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 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. The exact computation for the error distribution is outlined in APPENDIX B.

In a post processing step we iterate one more time the scan matching and update the poses as presented in [13] to improve the overall map quality. Note that these changes to the map are rather small. This step is necessary since the difference 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 2). 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. Figure 8 summarizes the overall control flow as well as the runtime of the different steps.

4 EXPERIMENTS AND RESULTS

4.1 Evaluation with Respect to Ground Truth

Experiments have been made with the software framework `slam6d` [11] and with various data sets. To demonstrate the results, we will use the publicly available data set HANNOVER2 here, as provided by O. Wulf, Leibniz

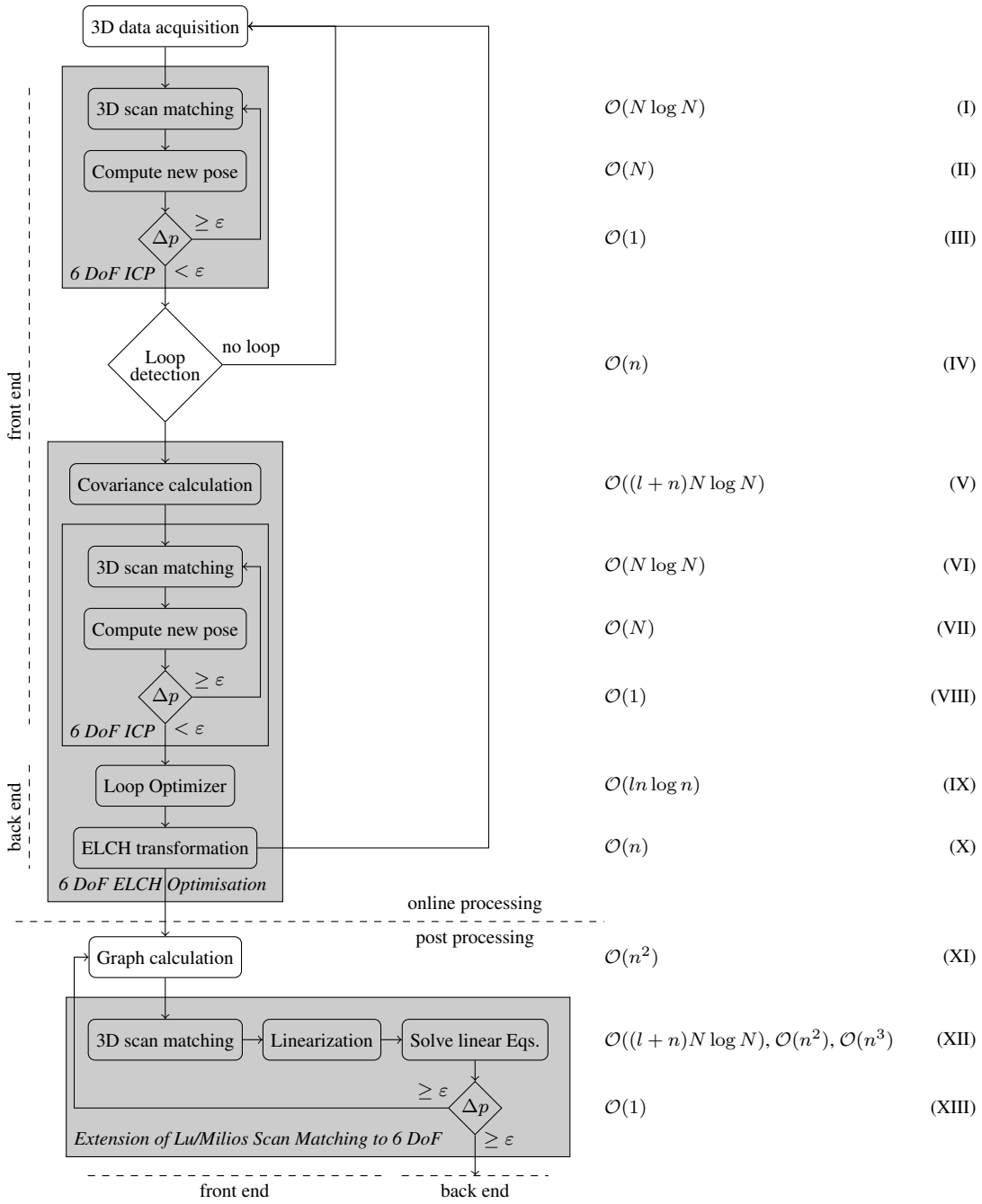


Fig. 8. Schematic overview of the complete algorithm, namely the interaction of ICP, ELCH and LUM (as extension of Figure 3). The dashed line separates the global optimization part that is executed a single time in the and as a post processing step. The threshold ε used to determine if a scan has changed its pose is the same in all three sub-algorithms. N is the number of points in a single 3D scan, n denotes the number of poses, and l is the number of loops.

University Hannover, Germany, to evaluate our algorithm. The data set, part of the Robotic 3D Scan Repository [32], has been acquired in an urban area and consists of 924 3D scans, each containing up to 35 000 3D data points (cf. Figure 2). A mobile robot uses a continuously rotating 3D scanner to deliver the data [10]. In [10] a benchmark for this data set has been presented using 6D SLAM with ICP and LUM.

Algorithm 1 Loop Optimizer Algorithm (LOA)

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 $w_i : V \to [0, 1]$

1. $w_f \leftarrow 0$
2. $w_l \leftarrow 1$
3. $\Omega \leftarrow \{v_f, v_l\}$ /* Loop Closing */
   /* Dijkstra returns a path $p := (v_s, v_1, v_2, \ldots, v_n, v_e)$ */
   /* and minimal costs $d(v_s, v_s), d(v_s, v_1), \ldots, d(v_s, v_e)$ */
4. **while** find shortest paths between any two vertices $\{v_s, v_e\} \in \Omega$ with Dijkstra **do**
5. **for all** vertices v_i on p **do** /* trail edge */
6. $w_i \leftarrow w_s + \frac{d(v_s,v_i)}{d(v_s,v_e)}(w_e - w_s)$
7. **if** $deg(v_i) > 2$ **then**
8. $\Omega = \Omega \cup \{v_i\}$
9. **end if**
10. **end for**
11. remove edges of path p in G
12. **if** $deg(v_s) = 0$ **then**
13. $\Omega = \Omega \setminus \{v_s\}$
14. **end if**
15. **if** $deg(v_e) = 0$ **then**
16. $\Omega = \Omega \setminus \{v_e\}$
17. **end if**
18. **end while**
19. **while** $\Omega \neq \emptyset$ **do** /* only branch edges left */
20. select $v_i \in \Omega$
21. **for all** neighbors v_n of v_i **do**
22. $w_n \leftarrow w_i$
23. delete edge $\{v_i, v_n\}$
24. **if** $deg(v_n) > 0$ **then**
25. $\Omega = \Omega \cup \{v_n\}$
26. **end if**
27. **end for**
28. $\Omega = \Omega \setminus \{v_i\}$
29. **end while**

To evaluate the map computed by our algorithm, some kind of ground truth is necessary. In [10] 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 9): The 2D map is extrapolated to 3D

Fig. 9. 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).

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". Figure 10 shows the trajectories of LUM and ELCH compared to ground truth and Figure 11 shows the result of applying both algorithms and the generated map, overlayed over a sattelite image for visual comparison. To benchmark our strategy, we used the publicly available back ends TORO and HOG-Man by Grisetti et al. [23, 33] and incorporated them into our SLAM software. Details can be found in Section 4.2 and the resulting trajectories for using them as a back end to ELCH are LUM in Figure 12.

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

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

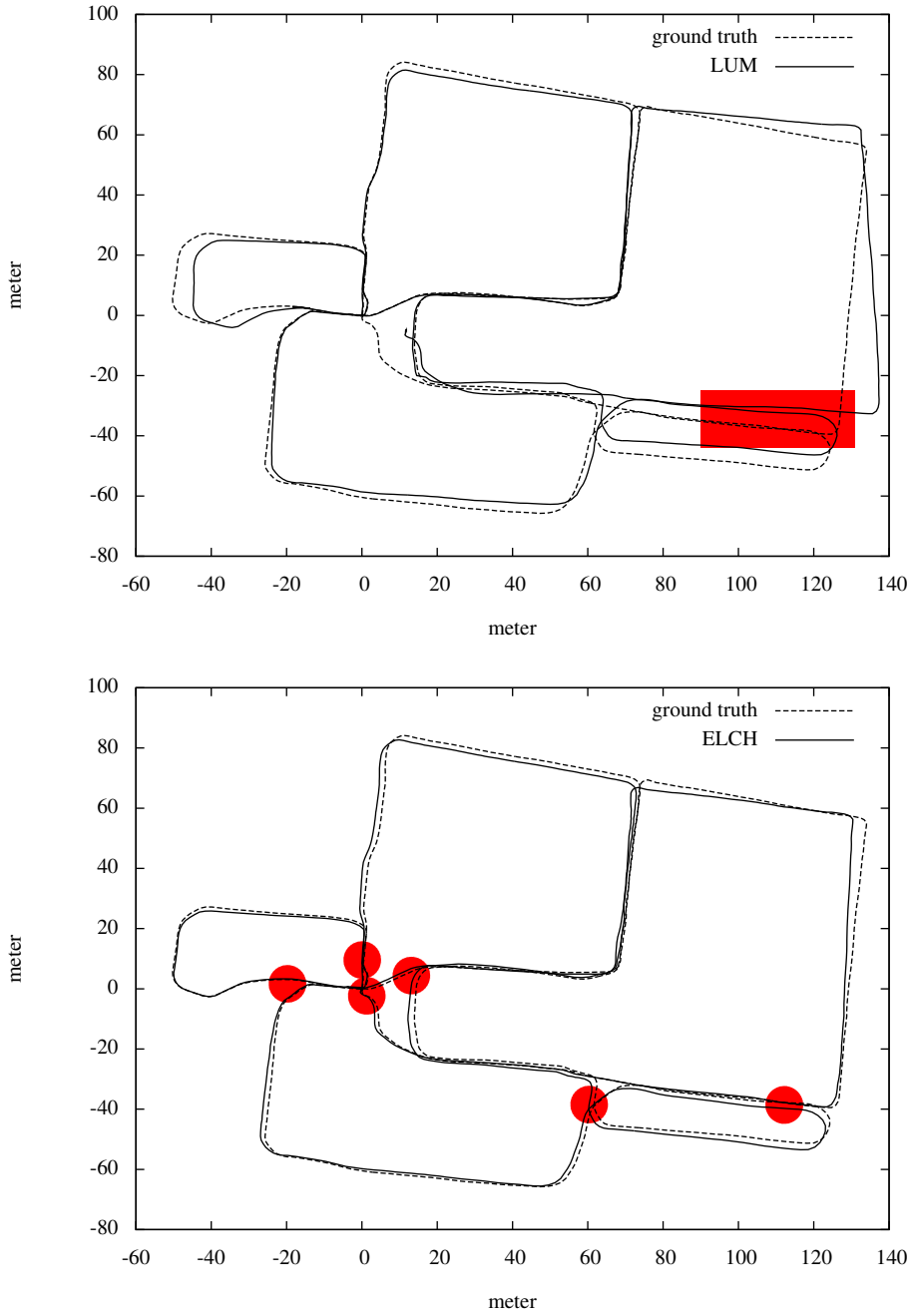


Fig. 10. Comparison of globally optimized graph SLAM (LUM, top) and the proposed ELCH strategy (bottom) against ground truth. The dots on the bottom represent the places where loops were closed, whereas LUM did not find the loop in the rectangle on the top.

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.

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

The Figures 13 and 14 present graphs the residual error for every scan, using these equations and Table 1 compares the means and standard deviations. Our ELCH heuristics returns the best results, which we count as a fortunate coincidence, since the ground truth poses are *unknown* to the

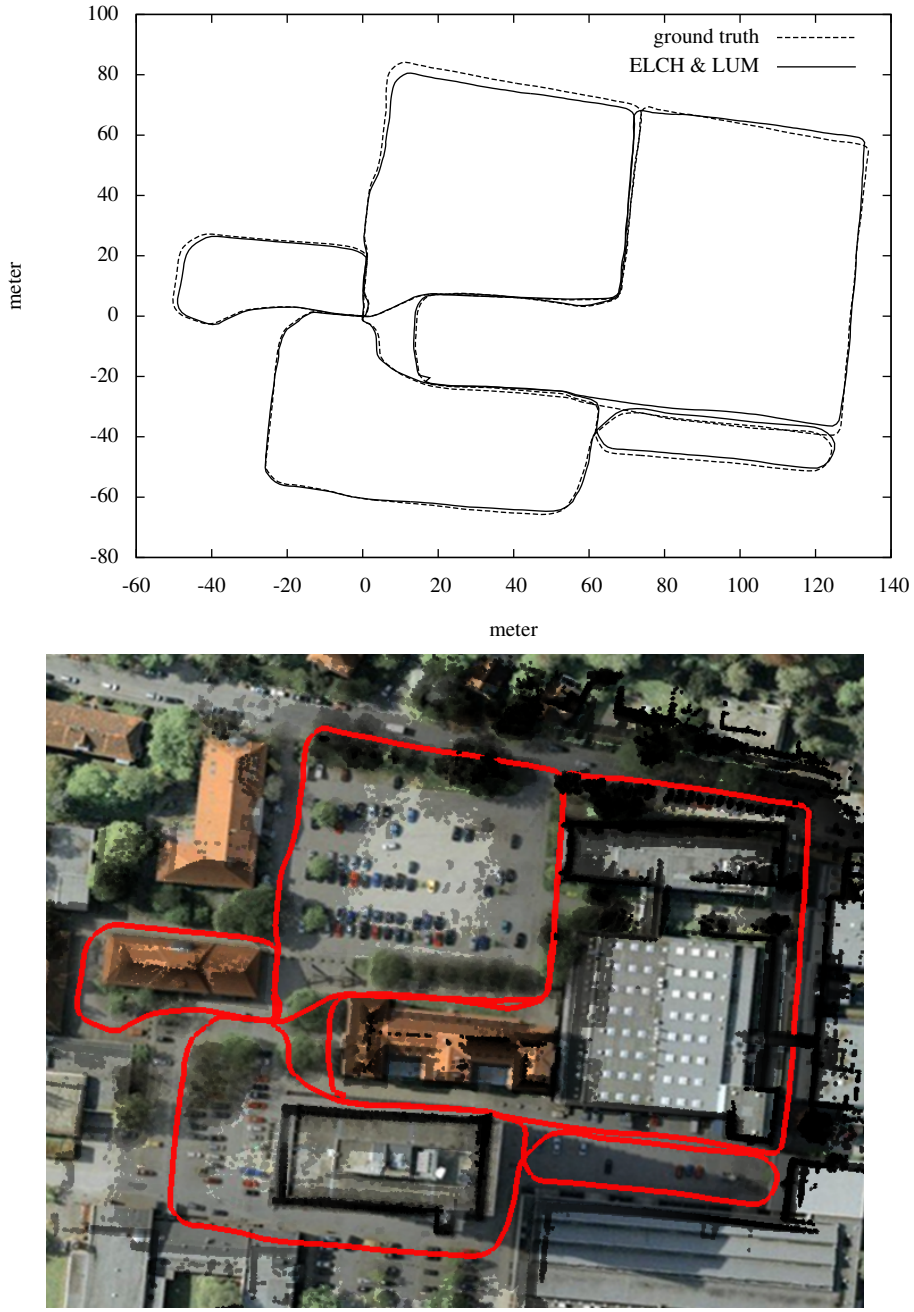


Fig. 11. Top: Comparison of the final combined result, applying LUM offline after processing the scans online with ELCH, against ground truth. Bottom: Image from Google Maps overlayed with the generated map (cf. Figure 2).

optimization processes. The scans are moved to gain a better mutual fit, but this might relocate them away from ground truth. Judging the results visually, as it is commonly done, all registrations seem to be good and valid. However, a consistent map is not necessarily a correct one.

Figures 15 and 16 present the rotational and translational errors for two scans while executing our new strategy ELCH in comparison to our previous strategy LUM. 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

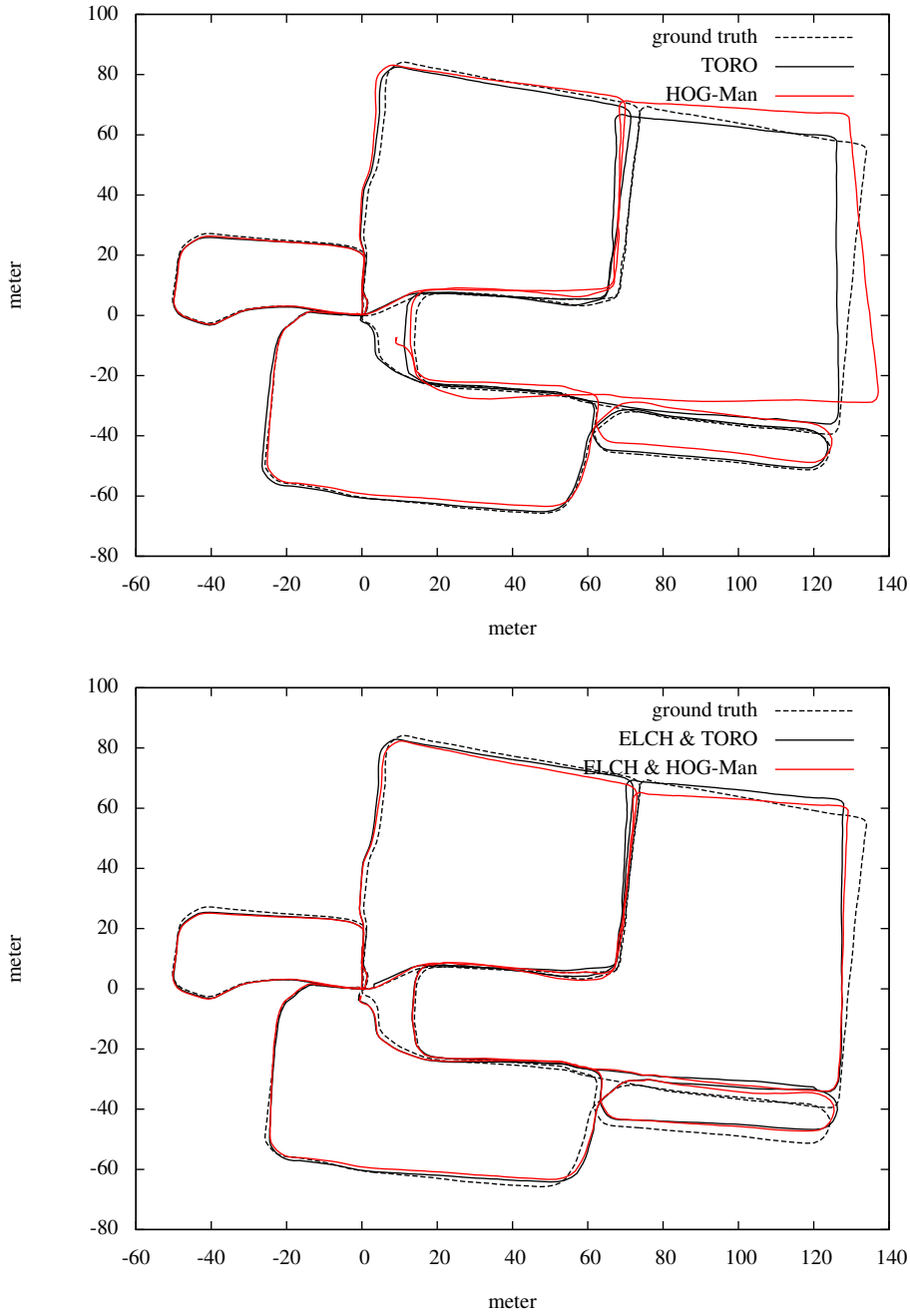


Fig. 12. Top: Comparison of trajectories, as computed by TORO and HOG-Man using the ELCH graph, against ground truth. Bottom: Computed trajectories with ELCH for online processing and TORO respectively HOG-Man for post processing.

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.

Figure 15 shows scan 344 which is in 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 than

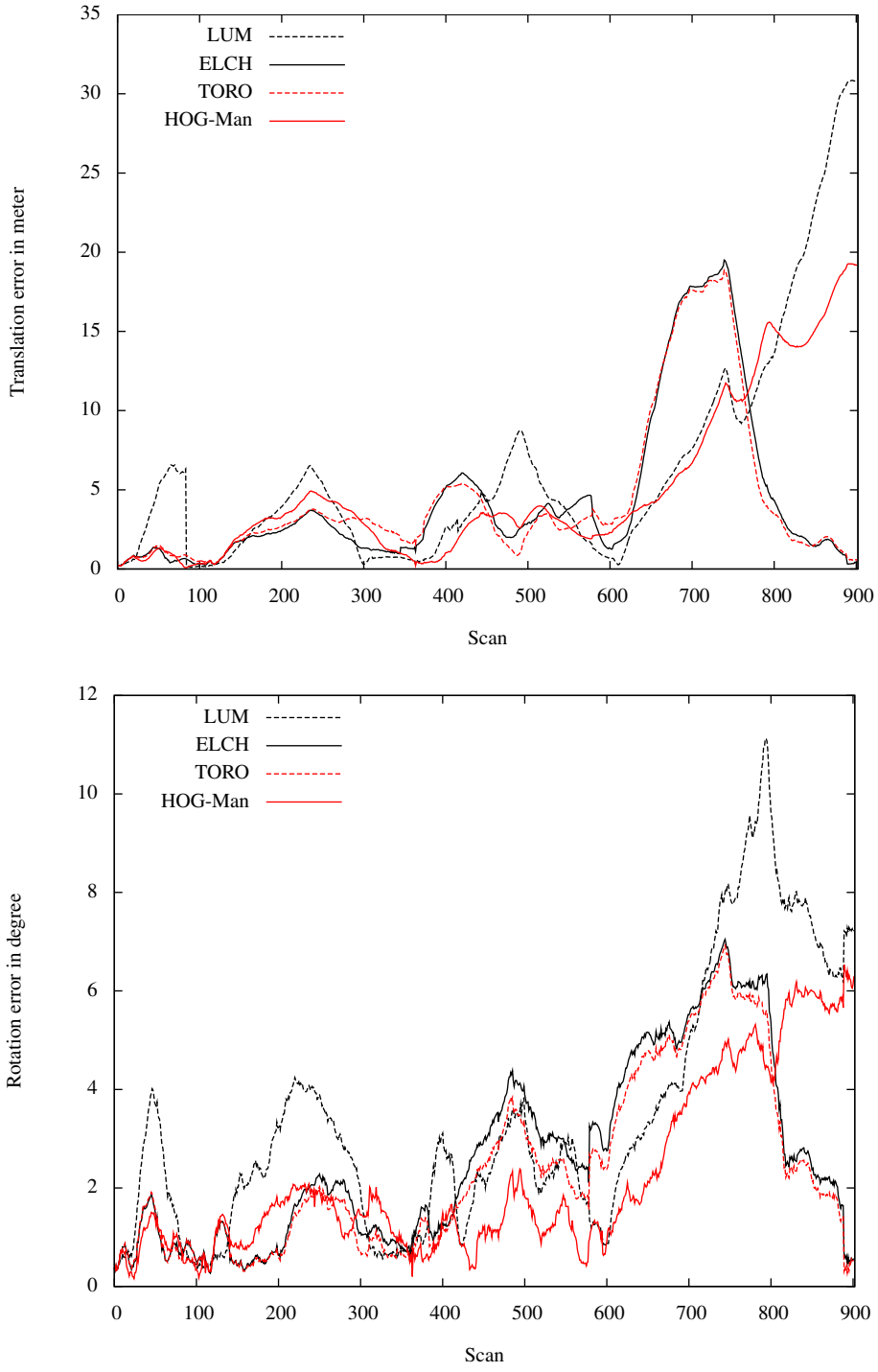


Fig. 13. Plot of the residual error after applying LUM, ELCH, TORO and HOG-Man as online processing. Comparing the translation error (top) and the rotation error (bottom) against ground truth.

by the loop closing constraint. The ELCH algorithm, on the other hand, has only the loop closing constraint, and applies the offset in one step.

The second example (Figure 16) is the 556th scan (label H in Figure 2). 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. 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

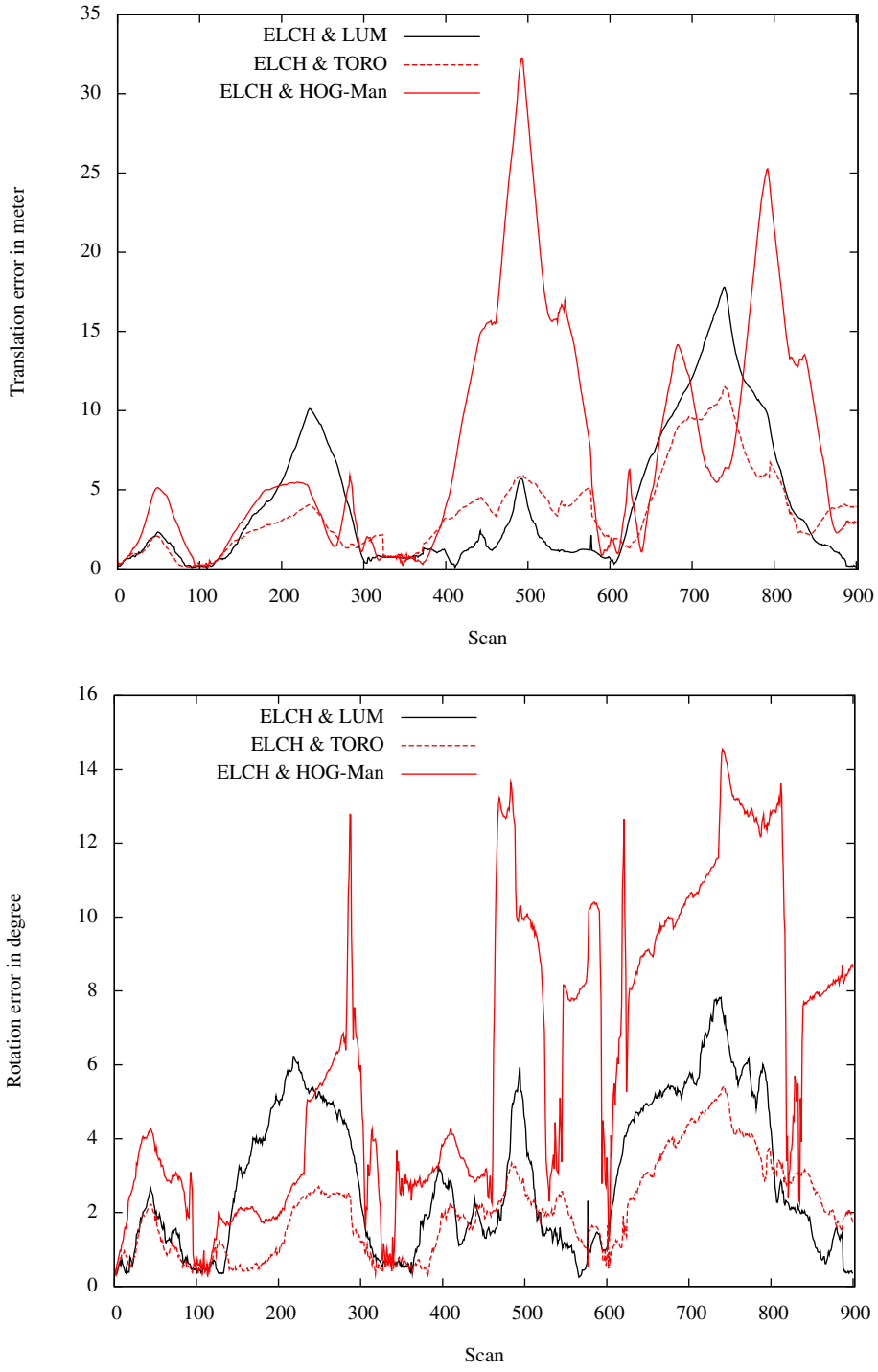


Fig. 14. Plot of the residual error after applying LUM, TORO and HOG-Man as post processing, with ELCH as online processing. Comparing the translation error (top) and the rotation error (bottom) against ground truth.

loops. In both cases, the LUM algorithm at the end of the ELCH part converges much faster because of the previous corrections. This is visible in Table 2, 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/` [2].

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

Table 1. Means and standard deviations of the residual error of the scans, after being processed with the denoted algorithms.

Algorithm	**Translation [m]**	**Translation [m] (only x and y)**	**Rotation [°]**
ICP	9,16 ± 6,70	8,35 ± 6,22	3,31 ± 2,64
LUM	6.24 ± 7.07	3.78 ± 4.66	3.44 ± 2.55
ELCH	4.35 ± 4.96	1.50 ± 2.95	2.61 ± 1.84
TORO	4.37 ± 4.74	2.33 ± 2.85	2.35 ± 1.76
HOG-Man	5.00 ± 5.05	4.34 ± 4.09	2.19 ± 1.73
ELCH & LUM	4.05 ± 4.33	1.37 ± 2.86	2.90 ± 2.03
ELCH & TORO	3.58 ± 2.61	2.30 ± 2.28	2.04 ± 1.25
ELCH & HOG-Man	7.60 ± 7.28	2.17 ± 5.63	6.10 ± 3.93

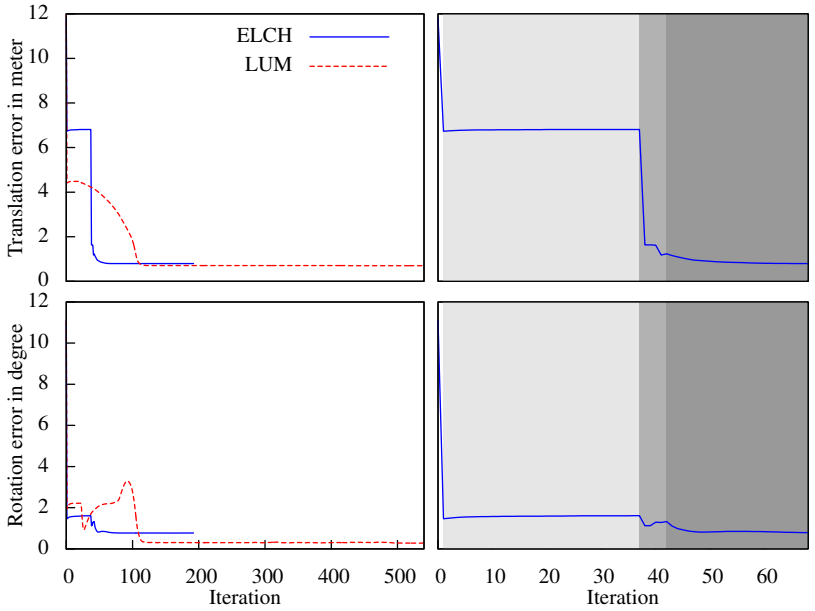


Fig. 15. Convergence of the 344th scan. The right side shows the stages of ELCH, namely ICP , LOA and LUM

Table 2. Runtime (in µs) comparison on an Intel Core i7 at 2.66 GHz with 8 GB RAM processing the complete data set HANNOVER2. *Total runtime is is the runtime of the complete strategy, including utility functions. A barchart of the most important results (in bold face) can be found in Figure 17.*

strategy		**ICP**	**online processing**		**post processing**		**total**
online pro.	**post pro.**		**front end**	**back end**	**front end**	**back end**	**runtime**
LUM		29 348 000	911 942 000	71 495 000			1 091 669 000
ELCH		**25 664 000**	**14 082 000**	**7 142**			40 725 000
TORO		**25 730 000**	**14 835 000**	**7 217 000**			49 764 000
HOG-Man		**25 360 000**	**8 122 000**	**10 409 000**			45 733 000
ELCH	LUM	25 697 000	13 987 000	7 182	**106 136 000**	**11 855 000**	172 435 000
ELCH	TORO	25 237 000	14 083 000	7 065	**159 431 000**	**352 000**	201 092 000
ELCH	HOG-Man	25 969 000	13 769 000	7 080	**156 931 000**	**3 830 000**	201 928 000

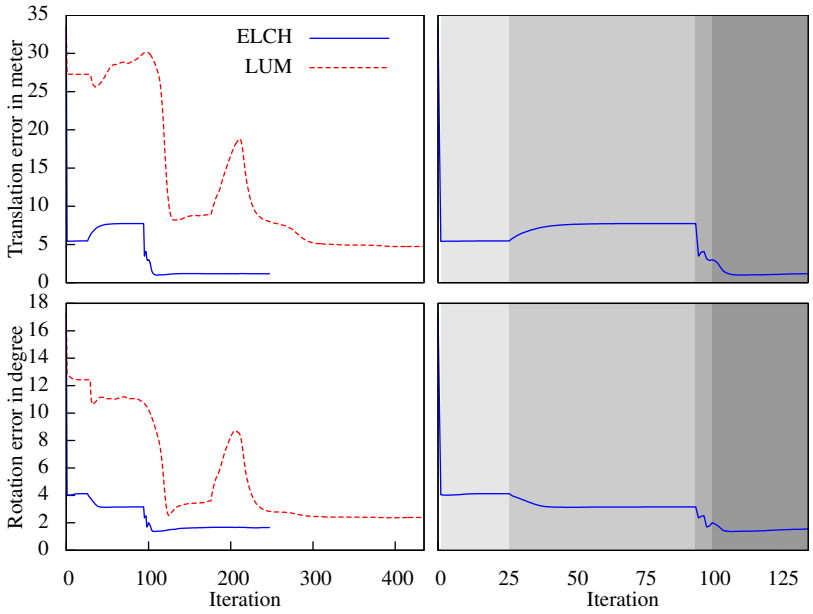


Fig. 16. Convergence of the 556th scan. The right side shows the stages of ELCH, namely ICP, ELCH ICP, LOA and LUM

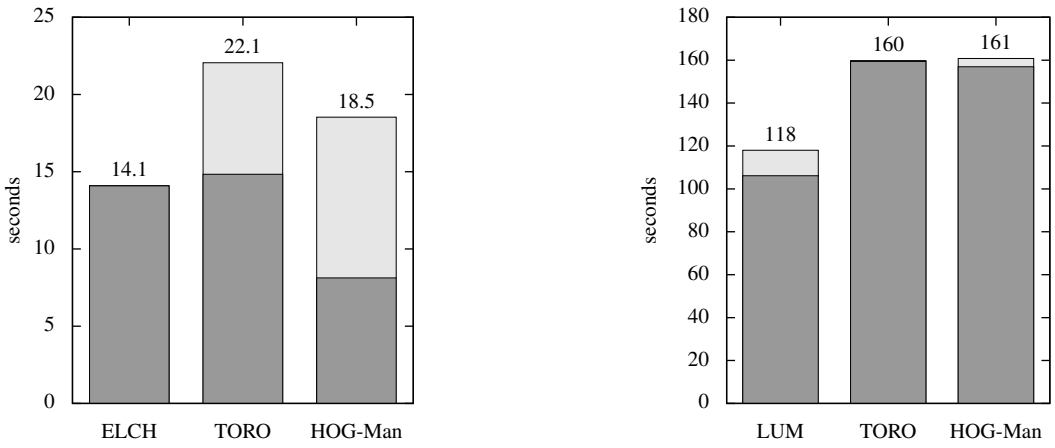


Fig. 17. Left: runtime of the online SLAM without ICP, graph generation and the different optimization methods. Right: runtime of the post processing with graph generation and optimization. While the absolute differences in the online part are minor, LUM outperforms the other approaches in the post processing phase. Please note that the runtimes for the online part are cumulative and previous corrections influences future loop detection, this explains the different durations of the front end.

We evaluated the ELCH algorithm on other data sets, as well. As a second example we used the data set HANNOVER1 which is also available under [32] and was used before in [13]. The boost in runtime and map quality is comparable to the other experiment, so we present only the resulting map (Figure 18) and the trajectories, compared to ground truth (Figure 19).

4.2 Comparison with Related Work

To show the different design ideas of our algorithms compared to other state of the art approaches, we will describe the data flow first (cf. Figure 20). The approach by Grisetti et al. [33] strictly separates a SLAM front end that builds the SLAM graph, and a SLAM back end that optimizes the graph. The SLAM front end consists of 4 parts: First, for every new laser scan, a vertex is added to the graph and connected to the previous one via scan

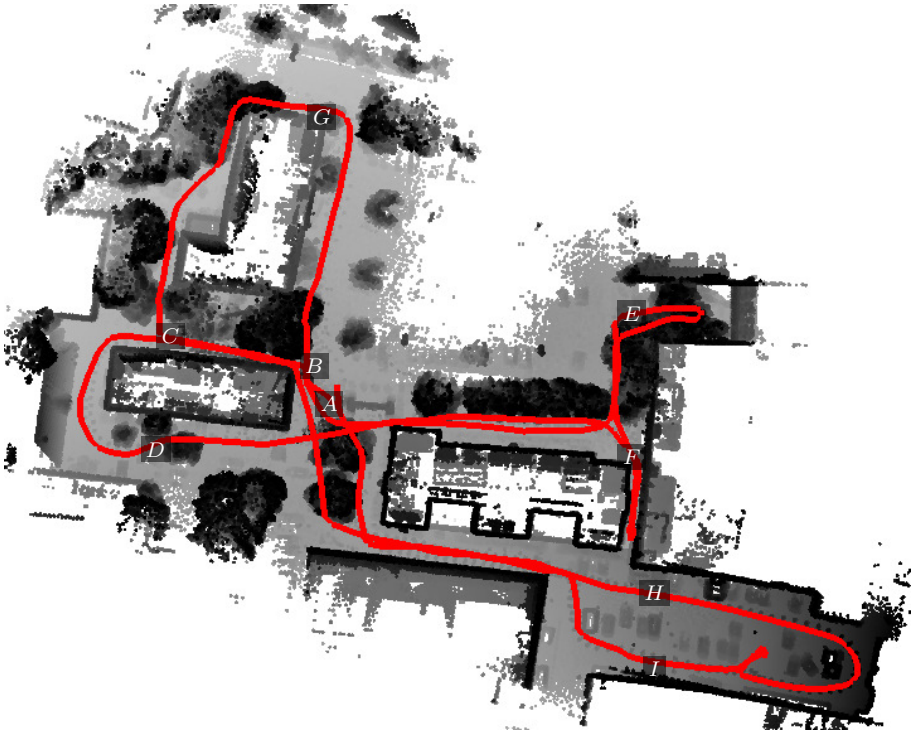


Fig. 18. View of the final map of the data set HANNOVER 1*, processed with ELCH and LUM. The robot explored the area along the trajectory, following the route A-B-C-D-A-E-F-A-B-G-C-B-A-H-I-A.*

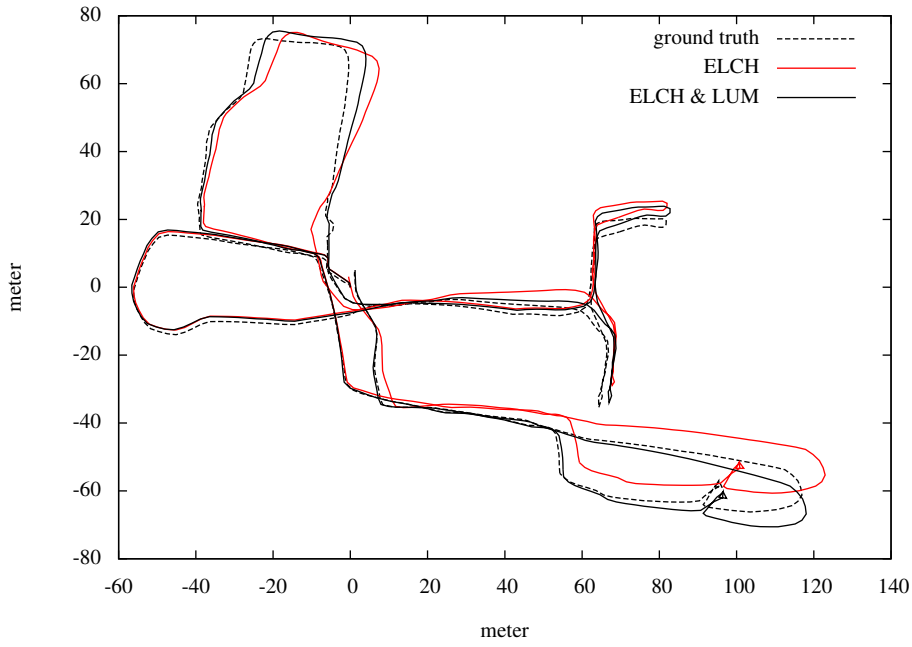


Fig. 19. Plot of the final trajectories of the data set HANNOVER 1*, after processing with ELCH only and a combination of ELCH and LUM, against ground truth.*

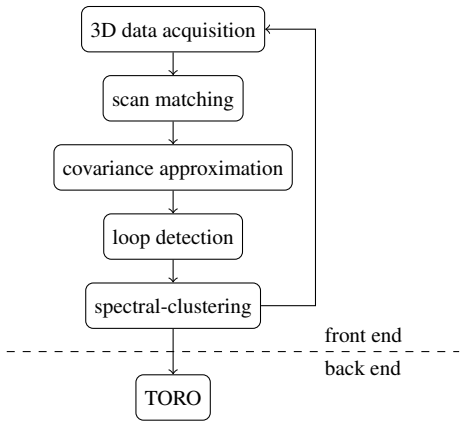


Fig. 20. Schematic overview of the TORO front and back end as described in [33].

matching. Second, loops are detected via covariance approximation, and third for each scan within a 3σ distance from every scan, scan matching is applied to generate all possible loop closing edges. In a fourth step spectral clustering is used to extract a consistent set of edges. As a post processing step, the resulting SLAM graph is optimized by algorithms like TORO or HOG-Man [23, 33, 34]. Please bear in mind that, to our knowledge, there is no publicly available implementation of this SLAM front end as well as no information about the runtime.

We extend this approach by addressing the issue of separation of the front and back end. Any such splitting that does not correct loops at least approximately during front end operation, has to cope with a steadily growing σ ellipse where all scans have to be matched against each other. But, as scan matching is sensible to the initial poses, one has to try many different loop correction candidates because of the lack of context information, i.e., other matchings from surrounding scans. On the other hand, as σ is approximative, there is no guarantee that all loops are detected if the robot has traveled far enough. The ELCH heuristic is applied at little cost during online processing in order to continue scan matching based on pose estimations that are improved by the information from loop closing. In that sense, our approach interleaves the front and back end operations of Figure 20 in order to improve the overall quality of the starting pose estimations for loop detection.

The SLAM back ends TORO and HOG-Man are available from [35]; the original SLAM front end as described in [33] does not appear to be available. To make a comparison, we used our own SLAM front end, namely the same as for ELCH and LUM. For the comparison with ELCH, we generated a SLAM graph with only one loop closing edge, found by scan matching and executed the back end whenever a loop was detected. This was done by replacing LOA (part (IX) in Figure 3) with TORO and HOG-Man, respectively. With this approach we where able to compare the graph optimization part of ELCH against TORO and HOG-Man. Results are presented in Figure 13 and Tables 1 and 2. For an additional comparison, we used TORO and HOG-Man as a back end to our LUM algorithms. Thus, we were able to test it as a post processing step after ELCH. For this we used the same SLAM graph as for LUM, but iterated the scan matching part of the front end first and then optimized the resulting graph with TORO and HOG-Man. This allowed us to get comparable results with respect the the original control flow of the related approaches, as depicted in Figure 20. Results are given in Figure 14 and Tables 1 and 2.

4.3 Interpretation of the Results

Large scale SLAM with many 3D scans yields a complex optimization system. In previous own work, we have extended the ICP algorithm to solve SLAM by globally consistent scan matching, i.e., 6D SLAM [13]. The advantage of this approach is the iterative fashion. It results in correct and consistent maps. Previously we iterate scan matching, i.e., closest point calculation, i.e., covariance estimation, with a SLAM back end. Other state of the art SLAM solutions, like Toro and HOG-Man used for comparison here, strictly separate the SLAM back and front end and are typically faster [33, 34] the the original 6D SLAM. However, modeling the uncertainties in the map only by one mean and covariance per pose, this is fragile and prone to yield imprecise maps. The iteration of SLAM front and back end calculates mean and covariances several times, thus capturing the underlying statistical process much better. More precisely, $c * l$ iterations are needed for l detected loops and a maximal number of LUM iterations c.

With the presented ELCH heuristics we reduce the number of iterations between SLAM front and back end to $c+l$, thus lowering the overall runtime significantly in comparison to competing approaches (cf. Figure 17), yet yielding at least the same map quality (cf. Table 1) – as was proven by evaluation against independently acquired ground truth. Further experiments have been carried out, e.g., by Pellenz et al. to process a dataset of Disaster City [36].

5 CONCLUSION AND OUTLOOK

This paper has introduced a novel approach to scan matching based GraphSLAM. The usual approach to loop closing 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 heuristically over the SLAM graph

such that a minimal error occurs, with respect to the uncertainties in correlated poses. To validate the correctness of your SLAM-built maps in an outdoor setting, we have refined the evaluation method presented in [1].

The ELCH algorithm proposed in this paper yields an improved estimation of scan poses during online scan processing 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.

FURTHER EXPERIMENTS

Results of experiments with numerous other datasets of different robots can be found at `http://kos.informatik.uni-osnabrueck.de/download/elch/`. Speed and accuracy are comparable to the ones described in this paper.

APPENDIX A 6D SLAM

In this section, we briefly recapitulate our previously published 6D SLAM method [13, 27], to make this paper self-sufficient. Its basis is a fast and reliable scan matching algorithm for ICP and Lu/Milios style relaxation (see Figure 3). We will refer to this method as LUM that includes the estimation of the covariance matrices and the equation solver, the latter one is our current SLAM back end. The back end is based on a sparse Cholesky decomposition [27] and is therefore similar to the LU decomposition that yields the most accurate results in [24, 25].

A.1 6D SLAM with ICP Based Scan Matching

We use the ICP algorithm [12] to calculate the transformation while the robot is acquiring a sequence of 3D scans. ICP calculates point correspondences iteratively. In each iteration step, the algorithm selects the closest points as correspondences and calculates the transformation $(\mathbf{R}, \mathbf{t})$ for minimizing the equation

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

where the tuples $(\mathbf{m}_i, \mathbf{d}_i)$ of corresponding model and data points are given by minimal distance, i.e., $\mathbf{m}_i$ is the closest point to $\mathbf{d}_i$ within a close limit [12]. The underlying assumption of the ICP algorithm is that the point correspondences are correct in the last iteration. In each iteration, the transformation is calculated by the quaternion based method of Horn [37].

To digitalize environments without occlusions, multiple 3D scans have to be registered. Consider a robot traveling along a path, and traversing $(n + 1)$ 3D scan poses $\mathbf{X}_0, \ldots, \mathbf{X}_n$. A straightforward method for aligning several 3D scans taken from the poses $\mathbf{X}_0, \ldots, \mathbf{X}_n$ is *pairwise ICP* , i.e., matching the scan taken from pose $\mathbf{X}_1$ against the scan from pose $\mathbf{X}_0$, matching the scan from $\mathbf{X}_2$ against the one from $\mathbf{X}_1$, and so on. The detection of closes loops operates on the registered scans thus far, as sketched in Section 3.2.

A.2 6D SLAM with Global Relaxation

Once a closed loop is detected, a 6 DoF graph optimization algorithm for global relaxation is employed, a variant of GraphSLAM. Our method relies on a notion of the uncertainty of the poses, calculated by the registration algorithm. The following method extends the probabilistic approach first proposed in [19] to 6 DoF. For a more detailed description of the extension refer to [13] and [27]. For each pose $\mathbf{X}$, the term $\bar{\mathbf{X}}$ denotes a pose estimate, and $\Delta\mathbf{X}$ is the pose error.

The positional error of two poses $\mathbf{X}_j$ and $\mathbf{X}_k$ is described by:

$$E_{j,k} = \sum_{i=1}^{m} \left\| \mathbf{X}_j \oplus \mathbf{d}_i - \mathbf{X}_k \oplus \mathbf{m}_i \right\|^2 = \sum_{i=1}^{m} \left\| \mathbf{Z}_i(\mathbf{X}_j, \mathbf{X}_k) \right\|^2.$$

Here, $\oplus$ is the compounding operation that transforms a point into the global coordinate system. For small pose differences, $E_{j,k}$ can be linearized by use of a Taylor expansion:

$$\begin{aligned}\mathbf{Z}_i(\mathbf{X}_j, \mathbf{X}_k) \approx{}& \bar{\mathbf{X}}_j \oplus \mathbf{d}_i - \bar{\mathbf{X}}_k \oplus \mathbf{m}_i \\ &- \left(\nabla_j \mathbf{Z}_i(\bar{\mathbf{X}}_j, \bar{\mathbf{X}}_k)\Delta\mathbf{X}_j - \nabla_k \mathbf{Z}_i(\bar{\mathbf{X}}_j, \bar{\mathbf{X}}_k)\Delta\mathbf{X}_k\right)\end{aligned}$$

where ∇_j, ∇_k denotes the derivative with respect to $\mathbf{X}_j$ and $\mathbf{X}_k$ respectively. Utilizing the matrix decompositions $\mathbf{M}_i\mathbf{H}_j$ and $\mathbf{D}_i\mathbf{H}_k$ of the respective derivatives that separates the poses from the associated points gives:

$$\begin{aligned}\mathbf{Z}_i(\mathbf{X}_j, \mathbf{X}_k) &\approx \mathbf{Z}_i(\bar{\mathbf{X}}_j, \bar{\mathbf{X}}_k) - (\mathbf{M}_i\mathbf{H}_j\Delta\mathbf{X}_j - \mathbf{D}_i\mathbf{H}_k\Delta\mathbf{X}_k) \\ &\approx \mathbf{Z}_i(\bar{\mathbf{X}}_j, \bar{\mathbf{X}}_k) - \left(\mathbf{M}_i\mathbf{X}'_j - \mathbf{D}_i\mathbf{X}'_k\right)\end{aligned}$$

Appropriate decompositions are given for Euler angles, quaternion representation and the Helix transformation in [38]. In the following, we will work with the pose representation as Euler angles. This matrix decomposition cannot be derived from first principles and was first presented in [13]. Since $\mathbf{M}_i$ as well as $\mathbf{D}_i$ are independent of the pose, the positional error $E_{j,k}$ is minimized with respect to the new pose difference $\mathbf{E}'_{j,k}$:

$$\begin{aligned}\mathbf{E}'_{j,k} &= (\mathbf{H}_j\Delta\mathbf{X}_j - \mathbf{H}_k\Delta\mathbf{X}_k) \\ &= (\mathbf{X}'_j - \mathbf{X}'_k).\end{aligned}$$

$\mathbf{E}'_{j,k}$ is linear in the quantities $\mathbf{X}'_j$ that will be estimated so that the minimum of $\mathbf{E}_{j,k}$ and the corresponding covariance are given by

$$\bar{\mathbf{E}}_{j,k} = (\mathbf{M}^T\mathbf{M})^{-1}\mathbf{M}^T\mathbf{Z}$$
$$\mathbf{C}_{j,k} = s^2(\mathbf{M}^T\mathbf{M}).$$

where s^2 is the unbiased estimate of the covariance of the identically, independently distributed errors of $\mathbf{Z}$:

$$s^2 = (\mathbf{Z} - \mathbf{M}\bar{\mathbf{E}})^T(\mathbf{Z} - \mathbf{M}\bar{\mathbf{E}})/(2m-3).$$

Here $\mathbf{Z}$ is the concatenated vector consisting of all $\mathbf{Z}_i(\bar{\mathbf{X}}_j, \bar{\mathbf{X}}_k)$ and $\mathbf{M}$ the concatenation of all $\mathbf{M}_i$'s.

Up to now all considerations have been on a local scale. With the linearized error metric $\mathbf{E}'_{j,k}$ and the Gaussian distribution $(\bar{\mathbf{E}}_{j,k}, \mathbf{C}_{j,k})$ a Mahalanobis distance that describes the global error of all the poses is constructed:

$$\begin{aligned}\mathbf{W} &= \sum_{j\to k}(\bar{\mathbf{E}}_{j,k} - \mathbf{E}'_{j,k})^T\mathbf{C}^{-1}_{j,k}(\bar{\mathbf{E}}'_{j,k} - \mathbf{E}'_{j,k}) \\ &= \sum_{j\to k}\big(\bar{\mathbf{E}}_{j,k} - (\mathbf{X}'_j - \mathbf{X}'_k)\big)\mathbf{C}^{-1}_{j,k}\big(\bar{\mathbf{E}}'_{j,k} - (\mathbf{X}'_j - \mathbf{X}'_k)\big).\end{aligned} \tag{3}$$

In matrix notation, $\mathbf{W}$ becomes:

$$\mathbf{W} = (\bar{\mathbf{E}} - \mathbf{H}\mathbf{X})^T\mathbf{C}^{-1}(\bar{\mathbf{E}} - \mathbf{H}\mathbf{X}). \tag{4}$$

Here $\mathbf{H}$ is the signed incidence matrix of the pose graph, $\bar{\mathbf{E}}$ is the concatenated vector consisting of all $\bar{\mathbf{E}}'_{j,k}$ and $\mathbf{C}$ is a block-diagonal matrix comprised of $\mathbf{C}^{-1}_{j,k}$ as sub matrices. For deriving (4) from (3) we used an incidence matrix and stacked the matrices $\bar{\mathbf{E}}'_{j,k}$ and $\mathbf{C}^{-1}_{j,k}$. For the latter stacking must proceed in a diagonal fashion. Minimizing function (4) yields new optimal pose estimates. The minimization of $\mathbf{W}$ is accomplished by the following linear equation system:

$$\begin{aligned}(\mathbf{H}^T\mathbf{C}^{-1}\mathbf{H})\mathbf{X} &= \mathbf{H}^T\mathbf{C}^{-1}\bar{\mathbf{E}} \\ \mathbf{B}\mathbf{X} &= \mathbf{A}.\end{aligned}$$

The symmetrical matrix $\mathbf{B}$ consists of the sub matrices

$$\mathbf{B}_{k,j} = \mathbf{B}_{j,k} = \begin{cases}\sum_{l=0}^{n}\mathbf{C}^{-1}_{j,l} & (j=k) \\ -\mathbf{C}^{-1}_{j,k} & (j\neq k).\end{cases}$$

The entries of $\mathbf{A}$ are given by:

$$\mathbf{A}_j = \sum_{\substack{k=0\\k\neq j}}^{n}\mathbf{C}^{-1}_{j,k}\bar{\mathbf{E}}_{j,k}.$$

In addition to solving for $\mathbf{X}$ this allows us to compute the associated covariance $\mathbf{C}_X$ of $\mathbf{X}$:

$$\mathbf{C}_X = \mathbf{B}^{-1}.$$

The results have to be transformed to obtain the optimal pose estimates as follows:

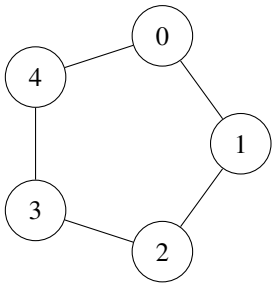
$$\mathbf{X}_j = \bar{\mathbf{X}}_j - \mathbf{H}_j^{-1}\mathbf{X}'_j,$$
$$\mathbf{C}_j = (\mathbf{H}_j^{-1})\mathbf{C}_j^X(\mathbf{H}_j^{-1})^T.$$

Figure 21 shows a simple graph containing five vertices and five directed edges. Each edge denotes a scan matching, where the model set corresponds to the 3D point cloud with an outgoing edge and the data set corresponds the point cloud with the incoming edge. For all points in the data set the closest point in the model set is calculated. Based on these point pairs the covariance matrices are estimated as stated above. The matrix $\mathbf{B}$ features 4 entries, since the first 3D scan, i.e., scan 0, defines the coordinate system and is not transformed. However, the covariance matrices with index 0 appear in the loop closing and at $\mathbf{B}_{1,1}$.

Pose Representation using Euler Angles Following the convention in [13], we represent a pose $\mathbf{X}$, as well as its estimate and error, in Euler angles

$$\mathbf{X} = \begin{pmatrix}t_x\\t_y\\t_z\\\theta_x\\\theta_y\\\theta_z\end{pmatrix} \qquad \bar{\mathbf{X}} = \begin{pmatrix}\bar{t}_x\\\bar{t}_y\\\bar{t}_z\\\bar{\theta}_x\\\bar{\theta}_y\\\bar{\theta}_z\end{pmatrix} \qquad \Delta\mathbf{X} = \begin{pmatrix}\Delta t_x\\\Delta t_y\\\Delta t_z\\\Delta\theta_x\\\Delta\theta_y\\\Delta\theta_z\end{pmatrix}.$$

The matrix decomposition $\mathbf{M}_i\mathbf{H} = \nabla\mathbf{Z}_i(\bar{\mathbf{X}})$ is given in Figure 22. As required, $\mathbf{M}_i$ contains all point information, while $\mathbf{H}$ expresses the pose information. Thus, this matrix decomposition constitutes a pose linearization similar to that proposed in the preceding section. While the matrix decomposition is arbitrary with respect to the column and row ordering of $\mathbf{H}$, this particular description was chosen due to its similarity to the 3D pose solution given in [19]. Finally, a system of $6n$ equations (n denotes the number of poses to be estimated) has to be solved, but since the pose graph is sparse the resulting equation system is sparse, too. We use a sparse Cholesky decomposition as SLAM back end [27].



$$\mathbf{B} = \begin{pmatrix} \mathbf{C}_{0,1}^{-1} + \mathbf{C}_{1,2}^{-1} + \mathbf{C}_{4,0}^{-1} & -\mathbf{C}_{1,2}^{-1} & 0 & -\mathbf{C}_{4,0}^{-1} \\ -\mathbf{C}_{1,2}^{-1} & \mathbf{C}_{1,2}^{-1} + \mathbf{C}_{2,3}^{-1} & -\mathbf{C}_{2,3}^{-1} & 0 \\ 0 & -\mathbf{C}_{2,3}^{-1} & \mathbf{C}_{2,3}^{-1} + \mathbf{C}_{3,4}^{-1} & \mathbf{C}_{3,4}^{-1} \\ -\mathbf{C}_{4,0}^{-1} & 0 & \mathbf{C}_{3,4}^{-1} & \mathbf{C}_{3,4}^{-1} + \mathbf{C}_{4,0}^{-1} \end{pmatrix}$$

$$\mathbf{A} = \begin{pmatrix} -\mathbf{C}_{0,1}^{-1}\bar{\mathbf{E}}_{0,1} + \mathbf{C}_{1,2}^{-1}\bar{\mathbf{E}}_{1,2} \\ -\mathbf{C}_{1,2}^{-1}\bar{\mathbf{E}}_{1,2} + \mathbf{C}_{2,3}^{-1}\bar{\mathbf{E}}_{2,3} \\ -\mathbf{C}_{2,3}^{-1}\bar{\mathbf{E}}_{2,3} + \mathbf{C}_{3,4}^{-1}\bar{\mathbf{E}}_{3,4} \\ -\mathbf{C}_{3,4}^{-1}\bar{\mathbf{E}}_{3,4} + \mathbf{C}_{4,0}^{-1}\bar{\mathbf{E}}_{4,0} \end{pmatrix}$$

Fig. 21. Simple loop containing 5 vertices. The corresponding system of linear equations is shown on right side.

$$\mathbf{H} = \begin{pmatrix} 1 & 0 & 0 & 0 & -\bar{t}_z\cos(\bar{\theta}_x) + \bar{t}_y\sin(\bar{\theta}_x) & \bar{t}_y\cos(\bar{\theta}_x)\cos(\bar{\theta}_y) + \bar{t}_z\cos(\bar{\theta}_y)\sin(\bar{\theta}_x) \\ 0 & 1 & 0 & \bar{t}_z & -\bar{t}_x\sin(\bar{\theta}_x) & -\bar{t}_x\cos(\bar{\theta}_x)\cos(\bar{\theta}_y) + \bar{t}_z\sin(\bar{\theta}_y) \\ 0 & 0 & 1 & -\bar{t}_y & \bar{t}_x\cos(\bar{\theta}_x) & -\bar{t}_x\cos(\bar{\theta}_y)\sin(\bar{\theta}_x) - \bar{t}_y\sin(\bar{\theta}_y) \\ 0 & 0 & 0 & 1 & 0 & \sin(\bar{\theta}_y) \\ 0 & 0 & 0 & 0 & \sin(\bar{\theta}_x) & \cos(\bar{\theta}_x)\cos(\bar{\theta}_y) \\ 0 & 0 & 0 & 0 & \cos(\bar{\theta}_x) & -\cos(\bar{\theta}_y)\sin(\bar{\theta}_x) \end{pmatrix}$$

$$\mathbf{M}_i = \begin{pmatrix} 1 & 0 & 0 & 0 & -d_{y,i} & d_{z,i} \\ 0 & 1 & 0 & -d_{z,i} & d_{x,i} & 0 \\ 0 & 0 & 1 & d_{y,i} & 0 & -d_{x,i} \end{pmatrix}.$$

Fig. 22. Definition of the matrix decomposition $\mathbf{M}$ *and* $\mathbf{H}$*.*

APPENDIX B ERROR COMPUTATION AND DISTRIBUTION

The theoretical examples in Section 3.3 had always the first scan of the loop in the origin of the global coordinate system. Thus it was easy to compute the offset and distribute the error. In practice the origin is normally defined by the scan, i.e. where the robot was turned on. To keep this constraint, we have to transform all ELCH corrections into the coordinate system of the first scan. The needed computations are detailed in the following section, using matrices for ease of writing.

For this section $\mathbf{P}_i^0$ defines the $\mathbb{R}^{4\times4}$ transformation matrix of the i^{th} scan in the coordinate system of the first one. The inverse transformation is written as $\overline{\mathbf{P}}$. With this definitions one can transform between the coordinate systems of the different scans with the following equation:

$$\mathbf{P}_i^0 = \mathbf{P}_j^0\mathbf{P}_i^j, \text{ or } \mathbf{P}_i^j = \overline{\mathbf{P}_j^0}\mathbf{P}_i^0. \tag{5}$$

The ICP offset, resulting from matching the last scan l of the loop against the first one f, is computed in the coordinate system of the first scan and one can compute the δ by multiplying the pose of the last scan before using ICP ($\mathbf{P}_l^f$) with the pose thereafter ($\mathring{\mathbf{P}}_l$):

$$\delta^f = \mathring{\mathbf{P}}_l^f\overline{\mathbf{P}_l^f}.$$

With (5) this is transformed into the coordinate system of the first scan:

$$\delta^f = \overline{\mathbf{P}_f^0}\mathring{\mathbf{P}}_l^0\overline{\mathbf{P}_f^0}\mathbf{P}_l^0. \tag{6}$$

Next, a δ for every scan is computed using LOA (Section 3.5) resulting in a δ_i for every scan i. This is now used to compute the new poses for every scan:

$$\hat{\mathbf{P}}_i^f = \delta_i^f\mathbf{P}_i^f.$$

As indicated above, this would transform the first scan as well, so it would not define the origin anymore. To move the origin back to where it was before, we transform all other scans with the inverse of δ_0:

$$\tilde{\mathbf{P}}_i^f = \overline{\delta_0^f}\delta_i^f\mathbf{P}_i^f.$$

Using (5) we can transform the equations into the coordinate system of the first scan:

$$\tilde{\mathbf{P}}_i^0 = \mathbf{P}_f^0\tilde{\mathbf{P}}_i^f,$$

and for the right side:

$$\tilde{\mathbf{P}}_i^0 = \mathbf{P}_f^0 \overline{\delta_0^f} \delta_i^f \overline{\mathbf{P}_f^0} \mathbf{P}_i^0. \quad (7)$$

For some scans we can simplify these equations. For scan f (with $\delta_f^f = \mathbf{I}$) it results into:

$$\tilde{\mathbf{P}}_f^0 = \mathbf{P}_f^0 \overline{\delta_0^f}.$$

The equation for scan l with $\delta_l^f = \delta^f$, using (7) and (6):

$$\tilde{\mathbf{P}}_l^0 = \mathbf{P}_f^0 \overline{\delta_0^f} \overline{\mathbf{P}_f^0} \mathring{\mathbf{P}}_l^0 \overline{\overline{\mathbf{P}_f^0} \mathbf{P}_l^0} \overline{\mathbf{P}_f^0} \mathbf{P}_l^0,$$

or, shortened:

$$\tilde{\mathbf{P}}_l^0 = \mathbf{P}_f^0 \overline{\delta_0^f} \overline{\mathbf{P}_f^0} \mathring{\mathbf{P}}_l^0.$$

ACKNOWLEDGMENTS

We would like to thank Jan Elseberg (Jacobs University Bremen) for fruitful discussions on the topic.

REFERENCES

[1] A. Nüchter, K. Lingemann, J. Hertzberg, and H. Surmann, "6D SLAM – 3D Mapping Outdoor Environments," *Journal of Field Robotics (JFR), Special Issue on Quantitative Performance Evaluation of Robotic and Intelligent Systems*, vol. 24, pp. 699–722, August/September 2007.

[2] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L.-E. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. van Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, and P. Mahoney, "Winning the darpa grand challenge," *Journal of Field Robotics (JFR)*, vol. 23, pp. 661–692, August 2006.

[3] The RoboCup Federation, "`http://www.robocup.org/`," 2008.

[4] The European Robot Trial (ELROB), "`http://www.elrob.org/`," 2008.

[5] DARPA Grand Challenge, "`http://www.darpa.mil/grandchallenge/`," 2007.

[6] J. Sprickerhof, A. Nüchter, K. Lingemann, and J. Hertzberg, "An explicit loop closing technique for 6d Slam," in *Proceedings of the European Conference on Mobile Robotics (ECMR '09)*, pp. 229–234, September 2009.

[7] D. Hähnel and W. Burgard, "Probabilistic Matching for 3D Scan Registration," in *Proceedings of the second German conference on robotics (ROBOTIK '02)*, (Ludwigsburg, Germany), June 2002.

[8] H. Surmann, K. Lingemann, A. Nüchter, and J. Hertzberg, "A 3D laser range finder for autonomous mobile robots," in *Proceedings of the of the 32nd International Symposium on Robotics (ISR '01)*, (Seoul, Korea), pp. 153–158, April 2001.

[9] O. Wulf and B. Wagner, "Fast 3D-scanning methods for laser measurement systems," in *International Conference on Control Systems and Computer Science (CSCS '03)*, pp. 312–317, 2003.

[10] O. Wulf, A. Nüchter, J. Hertzberg, and B. Wagner, "Benchmarking Urban Six-Degree-of-Freedom Simultaneous Localization and Mapping," *Journal of Field Robotics (JFR)*, vol. 25, pp. 148–163, March 2008.

[11] A. Nüchter and K. Lingemann, "SLAM software." `http://slam6d.sourceforge.net/`, 2009.

[12] P. Besl and N. McKay, "A method for registration of 3–d shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 14, pp. 239–256, February 1992.

[13] D. Borrmann, J. Elseberg, K. Lingemann, A. Nüchter, and J. Hertzberg, "Globally consistent 3d mapping with scan matching," *Journal of Robotics and Autonomous Systems (JRAS)*, vol. 65, no. 2, pp. 130–142, 2008.

[14] S. Thrun, "Robotic mapping: A survey," in *Exploring Artificial Intelligence in the New Millenium* (G. Lakemeyer and B. Nebel, eds.), pp. 1–34, Morgan Kaufmann, 2002.

[15] U. Frese, "A Discussion of Simultaneous Localization and Mapping," *Autonomous Robots*, vol. 20, no. 1, pp. 25–42, 2006.

[16] 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 (TRA)*, vol. 17, pp. 229–241, June 2001.

[17] J. Folkesson and H. I. Christensen, "Graphical SLAM for Outdoor Applications," *Journal of Field Robotics (JFR)*, vol. 24, pp. 51–70, February 2007.

[18] P. Newman and K. Ho, "SLAM-loop closing with visually salient features," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '05)*, pp. 635–642, 2005.

[19] F. Lu and E. Milios, "Globally consistent range scan alignment for environment mapping," *Autonomous Robots*, vol. 4, pp. 333–349, April 1997.

[20] J.-S. Gutmann and K. Konolige, "Incremental Mapping of Large Cyclic Environments," in *Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA '00)*, pp. 318–325, 2000.

[21] K. Konolige, "Large-scale map-making," in *Proceedings of the National Conference on AI (AAAI '04)*, pp. 457–463, 2004.

[22] P. Pfaff, R. Triebel, and W. Burgard, "An efficient extension to elevation maps for outdoor terrain mapping and loop closing," *International Journal of Robotics Research (IJRR)*, vol. 26, pp. 217–230, 2007.

[23] G. Grisetti, C. Stachniss, and W. Burgard, "Non-linear constraint network optimization for efficient map learning," *IEEE Transaction on Intelligent Transportation Systems*, vol. 10, pp. 428–439, 2009.

[24] E. Olson, J. Leonard, and S. Teller, "Fast iterative alignment of pose graphs with poor initial estimates," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '06)*, pp. 2262–2269, 2006.

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

[26] M. Kaess, A. Ranganathan, and F. Dellaert, "iSAM: Fast incremental smoothing and mapping with efficient data association," in *IEEE International Conference on Robotics and Automation (ICRA '07)*, (Rome, Italy), pp. 1670–1677, Apr 2007.

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

[28] L. Paz, J. Tardos, and J. Neira, "Divide and Conquer: EKF SLAM in $O(n)$," *IEEE Transactions on Robotics (TRO)*, vol. 24, pp. 1107–1120, October 2008.

[29] U. Frese, "Efficient 6-DOF SLAM with Treemap as a Generic Backend," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '07)*, (Rome, Italy), pp. 4814–4819, April 2007.

[30] K. Shoemake, "Animating rotation with quaternion curves," *ACM SIGGRAPH Computer Graphics*, vol. 19, pp. 245–254, July 1985.

[31] J. Bondy and U. Murty, *Graph theory*. Springer, 2008.

[32] A. Nüchter and K. Lingemann, "3D scan repository." `http://kos.informatik.uni-osnabrueck.de/3Dscans/`, 2009.

[33] G. Grisetti, R. Kümmerle, C. Stachniss, U. Frese, and C. Hertzberg, "Hierarchical optimization on manifolds for online 2d and 3d mapping," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 273–278, 2010.

[34] G. Grisetti, S. Grzonka, C. Stachniss, P. Pfaff, and W. Burgard, "Efficient estimation of accurate maximum likelihood maps in 3d," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3472–3478, 2007.

[35] C. Stachniss, U. Frese, and G. Grisetti, "Openslam." `http://openslam.org/`, 2010.

[36] J. Pellenz, D. Lang, F. Neuhaus, and D. Paulus, "Real-time 3d mapping of rough terrain: A field report from disaster city," in *IEEE International Workshop on Safety, Security and Rescue Robotics*, 2010.

[37] B. K. P. Horn, "Closed–form solution of absolute orientation using unit quaternions," *Journal of the Optical Society of America A (JOSA)*, vol. 4, pp. 629–642, April 1987.

[38] A. Nüchter, J. Elseberg, P. Schneider, and D. Paulus, "Study of parameterizations for the rigid body transformations of the scan registration problem," *Journal Computer Vision and Image Understanding (CVIU)*, 2010 (accepted).

Jochen Sprickerhof is a research associate at the University of Osnabrück. He received a Diploma degree in applied System Science from the University of Osnabrück in 2009. In his research he focuses on large scale 3D scan matching algorithms and their applications.

Andreas Nüchter holds an assistant professorship of Computer Science at Jacobs University Bremen. Before he joined Jacobs was a research associate at University of Osnabrück. Further past affiliations were with the Fraunhofer Institute for Autonomous Intelligent Systems (AIS, Sankt Augustin), the University of Bonn, from which he received the diploma degree in computer science in 2002 (best paper award by the German society of informatics (GI) for his thesis) and the Washington State University. He holds a doctorate degree (Dr. rer. nat) from University of Bonn. His thesis was shortlisted for the EURON PhD award. Andreas works on robotics and automation, cognitive systems and artificial intelligence. His main research interests include reliable robot control, 3D environment mapping, 3D vision, and laser scanning technologies, resulting in fast 3D scan matching algorithms that enable robots to perceive and map their environment in 3D representing the pose with 6 degrees of freedom. The capabilities of these robotic SLAM approaches were demonstrated at RoboCup Rescue competitions, ELROB and several other events. is a member of the GI and the IEEE.

Kai Lingemann is a research associate at the University of Osnabrück. Past affiliations were with the Fraunhofer Institute for Autonomous Intelligent Systems (AIS, Sankt Augustin), University of Kyoto and University of Bonn, from which he received the diploma degree in computer science in 2004. His research interests include reliable robot control, 3D environment mapping, 3D vision, and laser scanning technologies.

Joachim Hertzberg is a full professor for computer science at the University of Osnabrück, where he is heading the Knowledge-Based Systems lab. He graduated in Computer Science (U. Bonn, 1982; Dr. rer. nat. 1986, U. Bonn; habilitation 1995, U. Hamburg). Former affiliations were with GMD and with Fraunhofer AIS in Sankt Augustin. His areas of scientific interest are Artificial Intelligence and Mobile Robotics, where he has contributed to action planning, robot localization and mapping, plan-based robot control, active sensing, robot control architectures, temporal reasoning, logical reasoning about action and change, constraint-based reasoning, and applications of these. In these areas, he has written or edited six books and published over 90 refereed or invited papers in books, journals or conferences.

AUTHORS' ADDRESSES

Jochen Sprickerhof,
Kai Lingemann,
Prof. Joachim Hertzberg, Ph.D.,
University of Osnabrück,
Institute of Computer Science,
Albrechtstraße 28,
49076 Osnabrück, Germany
emails: {sprickerhof, limngemann, hertzgerg} @ informatik.uni-osnabrueck.de

Prof. Andreas Nüchter, Ph.D.,
Jacobs University Bremen gGmbH,
School of Engineering and Science,
Campus Ring 12,
28759 Bremen, Germany
email: andreas@nuechti.de

Received: 2010-04-10
Accepted: 2011-04-10