IMPROVING GOOGLE'S CARTOGRAPHER 3D MAPPING BY CONTINUOUS-TIME SLAM

Andreas Nüchter [a,b], Michael Bleier [b], Johannes Schauer [a,b], Peter Janotta [c]

[a] Informatics VII – Robotics and Telematics, Julius Maximilian University of Würzburg, Germany - (johannes.schauer, andreas.nuechter)@uni-wuerzburg.de
[b] Zentrum für Telematik e.V., Würzburg, Germany - michael.bleier@telematik-zentrum.de
[c] Measurement in Motion GmbH, Theilheim, Germany - peter.janotta@mim3d.de

KEY WORDS: SLAM, trajectory optimization, backpack, personal laser scanner, 3D point clouds

ABSTRACT:

This paper shows how to use the result of Google's SLAM solution, called Cartographer, to bootstrap our continuous-time SLAM algorithm. The presented approach optimizes the consistency of the global point cloud, and thus improves on Google's results. We use the algorithms and data from Google as input for our continuous-time SLAM software. We also successfully applied our software to a similar backpack system which delivers consistent 3D point clouds even in absence of an IMU.

1 INTRODUCTION

On October 5, 2016 Google released the source code of its real-time 2D and 3D simultaneous localization and mapping (SLAM) library Cartographer [1]. The utilized algorithms for solving SLAM in 2D have been described in a recent paper by the authors of the software (Hess et al., 2016). It can deliver impressive results — especially considering that it runs in real-time on commodity hardware. A publication describing the 3D mapping solution is still missing. The released software however, solves the problem. In addition, Google published a very demanding, high-resolution data set to the public for testing their algorithms. Also custom data sets are easy to process, as Google's software comes with an integration into the robot operating system (ROS) (Quigley et al., 2009). ROS is the de-facto standard in the robotic community as middleware. It allows to connect heterogeneous software packages via a standardized inter-process communication (IPC) system and is available on recent GNU/Linux distributions.

Google's sample data set was recorded in the museum "Deutsches Museum" in München, Germany. It is the world's largest museum of science and technology, and has about 28,000 exhibited objects from 50 fields of science and technology. The data set was recorded with a backpack system, which features an inertial measurement unit (IMU) and two Velodyne PUCK (VLP-16) sensors. The trajectory we processed was 108 meters long and contained 300,000 3D scans from the PUCK sensors.

Due to a high demand on flexible mobile mapping systems, mapping solutions on pushcarts, on trolleys, on mobile robots, and backpacks have recently been developed. Human carried systems offer the advantage to overcome doorsteps and that the operator can open closed doors etc. To this end, several vendors build human carried systems which are also often called personal laser scanners.

This paper shows how to use the result of Google's SLAM solution to bootstrap our continuous-time SLAM algorithm. Our approach optimizes the consistency of the global point cloud, and thus improves on Google's results. We use the algorithms and data from Google as input for our continuous-time SLAM solution, which was recently published in (Elseberg et al., 2013). We present successful applications of the software to our own similar backpack system which delivers consistent 3D point clouds even in absence of an IMU (Nüchter et al., 2015).

[1] https://opensource.googleblog.com/2016/10/introducing-cartographer.html

2 STATE OF THE ART

2.1 Laser Scanner on Robots and Backpacks

Mapping environments received a lot of attention in the robotics community, especially after the appearance of cost effective 2D laser range finders. Seminal work with 2D profiles in robotics was performed by Lu and Milios (1994). After deriving 2D variants of the by now well-known ICP algorithm they derived a PosegraphSLAM solution (Lu and Milios, 1997), that considers all 2D scans in a global fashion. Afterwards, many other approaches to SLAM have been presented, including extended Kalman filters, particle filters, expectation maximization and GraphSLAM. These SLAM algorithms aimed at enabling mobile robots to map the environments where they have to carry out user specific tasks. Thrun et al. (2000) presented a system, where a horizontally mounted scanner performed FastSLAM –a particle filter approach to SLAM– while an upward looking scanner was used to acquire 3D data, exploiting the robot motion to construct environments in 3D. Lu and Milios' approach was extended to 3D point clouds and poses with six degree of freedom (DoF) in (Borrmann et al., 2008).

In 2004 an early version of a backpack system was presented. Saarinen et al. (2004) used the term *Personal Localization And Mapping*. They used a horizontally mounted SICK LMS200 scanner in front of the human carried and put additional sensors and the computing equipment into a backpack. Chen et al. (2010) presented a backpack system that feature a number of lightweight 2D profilers (Hokuyo scanner) mounted in different viewing directions. In previous work, we tried to apply our algorithms to a backpack system without an IMU (Nüchter et al., 2015). The system consists of a horizontally mounted SICK LMS100 scanner and a spinning Riegl VZ400. Similarly to the work of Thrun et al. a horizontal scanner is used to estimate an initial trajectory that is afterwards updated to regard the 6 DoF motion. The term Personal Laser Scanning System was shaped by Liang et al. (2014). They use a single FARO scanner and rely on the global navigation satellite system (GNSS) system. Similarly, the commercially available ROBIN system features a RIEGL VUX scanner

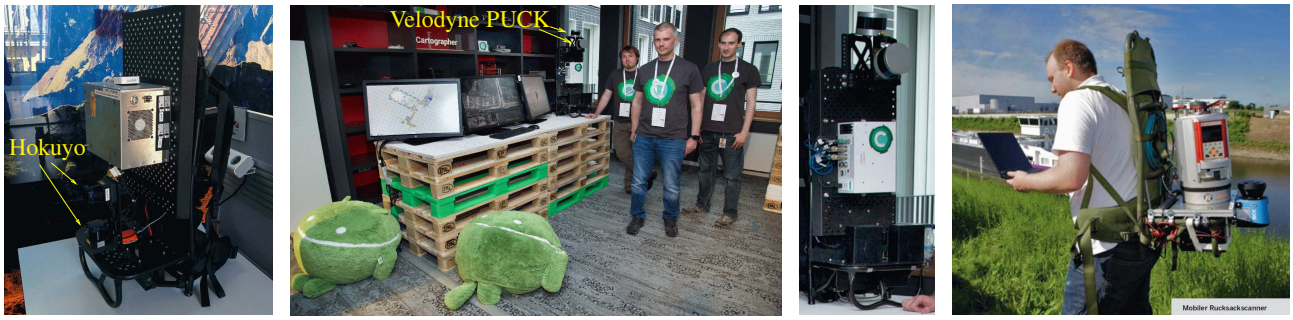

Figure 1: Left: Google's Cartographer system featuring two Hokuyo laser scanners (image: Google blog). Middle: Google's Cartographer system with two Velodyne PUCKs and the Cartographer team (image courtesy of the Cartographer team). Right: Second author operating Würzburg's backpack scanner.

and GNSS. In contrast, the Leica Pegasus is a commercially available backpack wearable mobile mapping solution, which is composed of *two* Velodyne PUCK scanners, cameras and a GNSS. The PUCKs scan 300.000 points per second and have a maximal range of 100 meters. 16 profilers are combined to yield a vertical field of view of ± 15 degree.

The Google Cartographer backpack was initially presented in September 2014. Back then, the backpack system was based on two Hokuyo profilers and an internal measurement unit (IMU). The current version features two Velodyne PUCK scanners. Figure 1 shows the system from Google and our backpack solution.

2.2 Calibration, Referencing, and SLAM

To acquire high quality range measurement data with mobile laser scanner the position and orientation of every individual sensors have to be known. Traditionally, scanners, GPS and IMU are calibrated against other positioning devices whose pose in relation to the system is already known. The term Boresight calibration is used for the technique of finding the rotational parameters of the range sensor with respect to the already calibrated IMU/GPS unit. In the airborne laser scanning community, automatic calibration approaches are known (Skaloud and Schaer, 2007), and similarly vehicle-based kinematic laser scanning has been considered (Rieger et al., 2010). In the robotics community there exist approaches for calibrating several range scanners semi-automatically, i.e., with manually labeled data (Underwood et al., 2009) or using automatically computed quality metrics (Sheehan et al., 2011; Elseberg et al., 2013). Often vendors do not make their calibration methods public and unfortunately, the authors of this paper have no information on the calibration of the Google Cartographer backpack. In general, calibration inaccuracies can to some extend be compensated with a SLAM algorithm.

Aside from sensor misalignment, a second source of errors are timing related issues. All subsystems on a mobile platform need to be synchronized to a common time frame. This is often accomplished with pure hardware via triggering or with mixes of hard and software like pulse per second (PPS) or the network time protocol (NTP). However, good online synchronization is not always available for all sensors. Olson (2010) has developed a solution for the synchronization of clocks that can be applied after the fact. In ROS, sensor data is time-stamped, when it arrives and it is recorded in an open file format (`.bag` files). Afterwards, one works with the time-stamped data using nearest values or interpolation.

As the term direct referencing or direct Geo-referencing implies, it is the direct measurement of the position and orientation of a mapping sensor, i.e., the laser scanner, such that each range value can be referenced without the need for collecting additional information. This means that the trajectory is then used to "unwind" the laser range measurements to produce the 3D point cloud. This approach has been taken by Liang et al. (2014).

Some systems employ a horizontally mounted scanner and perform 2D SLAM on the acquired profiles. Thrun et al. (2000) used FastSLAM, Nüchter et al. (2015) used SLAM based on the truncated signed distance function (TSD SLAM), or alternatively, HectorSLAM (Kohlbrecher et al., 2011). These 2D SLAM approaches produce 2D grid maps. Similarly, Google's Cartographer code is for creating 2D grid maps (Hess et al., 2016). Afterwards, the computed trajectory and the IMU measurements are used to "unwind" the laser range measurements to produce the 3D point cloud.

The Google Cartographer library achieves its outstanding performance by grouping scans into probability grids that they call submaps and by using a branch and bound approach for loop closure optimization. While new scans are matched with and subsequently entered into the current submap in the foreground, in the background, the library matches scans to nearby submaps to create loop closure constraints and to continually optimize the constraint graph of submaps and scan poses. The authors differentiate between local scan matching which inserts new scans into the current submap and which will accumulate errors over time and global SLAM which includes loop closing and which removes the errors that have accumulated in each submap that are part of a loop. Both, local and global matching are run at the same time.

During local scan matching, the Cartographer library matches each new scan against the current submap using a Ceres-based scan matcher (Agarwal et al., n.d.). A submap is a regular probability grid where each discrete grid point represents the probability that the given grid point is obstructed or free. These two sets are disjoint. A grid point is obstructed if it contains an observed point. Free points are computed by tracing the laser beam from the estimated scanner location to the measured point through the grid. The optimization function of the scan matcher makes use of the probability grid as part of its minimization function.

During global SLAM, finished submaps (those that no longer change) and the scans they contain are considered for loop closing. Just as during local scan matching, the problem is passed to Ceres as a nonlinear least squares problem. The algorithm is accurate down to the groups of points defined by the regular probability grid of each submap. By taking the submap grid size as translation step delta and the angle under which a grid point is seen at maximum range as the rotation step delta, a finite set of possible transformations is created. This solution space is searched using a branch and bound approach where nodes are

traversed using a greedy depth first search and the upper bound of the inner nodes being defined in terms of computational effort and quality of the bound. To compute the upper bound efficiently, grids are precomputed for each height of the tree that overlay the involved submaps and store for each obstructed grid point the maximum values of possible scores. This operation is done in $O(n)$ with n being the number of obstructed grid points in each precomputed grid.

Hess et al. (2016) describe the 2D version of the algorithm, which uses the horizontally mounted 2D profiler. The provided data sets contain also data from a setup with Velodyne PUCK scanners (cf. Figure 1 middle). Their algorithm is able to process 3D data and to output poses with 6 DoF, however, a description of their 3D approach is missing from their paper. Nevertheless, we understand from their published source code that their 3D implementation is mostly an extension of their 2D approach to three dimensions with a 3D probability grid. Some changes have been made to improve performance. For example, the 3D grid is not fully traversed to find free grid cells but only a configurable distance up to the measured point is checked.

Only a few approaches optimize the whole trajectory in a continuous fashion. Stoyanov and Lilienthal (2009) presented a non rigid optimization for a mobile laser scanning system. They optimize point cloud quality by matching the beginning and the end of a single scanner rotation using ICP. The estimate of the 3D pose difference between the two points in time is then used to optimize the robot trajectory in between. In a similar approach Bosse and Zlot (2009) use a modified ICP with a custom correspondence search to optimize the pose of six discrete points in time of the trajectory of a robot during a single scanner rotation. The trajectory in between is modified by distributing the errors with a cubic spline. The software of Riegl RiPRECISION MLS automatically performs adjustments of GNSS/INS trajectories to merge overlapping mobile scan data based on planar surface elements. Our own continuous-time SLAM solution improves the entire trajectory of the data set simultaneously based on the raw point cloud. The algorithm is adopted from Elseberg et al. (2013), where it was used in a different mobile mapping context, i.e., on platforms like the LYNX mobile mapper or the Riegl VMX-250. As no motion model is required, it can be applied to any continuous trajectory.

3 CONTINUOUS-TIME SLAM

3.1 6D SLAM

To understand the basic idea of our continuous-time SLAM, we summarize its foundation, 6D SLAM, which was designed for a high-precision registration of terrestrial 3D scans, i.e., globally consistent scan matching (Borrmann et al., 2008). It is available in *3DTK – The 3D Toolkit*[2]. The globally consistent scan matching is a full SLAM solution for 3D point clouds.

6D SLAM works similarly to the well-known iterative closest points (ICP) algorithm, which minimizes the following error function

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

to iteratively solve for an optimal rotation $\mathbf{T} = (\mathbf{R},\mathbf{t})$, where the tuples $(\mathbf{m}_i, \mathbf{d}_i)$ of corresponding model $\mathbf{M}$ and data points $\mathbf{D}$ are given by minimal distance, i.e., $\mathbf{m}_i$ is the closest point to $\mathbf{d}_i$ within a close limit (Besl and McKay, 1992). Instead of the two-scan-Eq. (1), we look at the n-scan case

$$E = \sum_{j\to k}\sum_{i} \left|\mathbf{R}_j\mathbf{m}_i + \mathbf{t}_j - (\mathbf{R}_k\mathbf{d}_i + \mathbf{t}_k)\right|^2, \qquad (2)$$

where j and k refer to scans of the SLAM graph, i.e., to the graph modeling the pose constraints in SLAM or bundle adjustment. If they overlap, i.e., closest points are available, then the point pairs for the link are included in the minimization. We solve for all poses at the same time and iterate like in the original ICP. The derivation of a GraphSLAM method using a Mahalanobis distance that describes the global error of all the poses

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

where $\mathbf{E}'_{j,k}$ is the linearized error metric and the Gaussian distribution is $(\bar{\mathbf{E}}_{j,k}, \mathbf{C}_{j,k})$ with computed covariances from scan matching as given in (Borrmann et al., 2008). $\mathbf{X}'_j$ and $\mathbf{X}'_k$ denote the two poses linked in the graph and related by the linear error metric. Minimizing Eq. (2) instead of (3) does not lead to different results (Nüchter et al., 2010). In matrix notation W in Eq. (3) becomes

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

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 submatrices. Minimizing this function yields new optimal pose estimates.

Please note, while there are four closed-form solutions for the original ICP Eq. (1), linearization of the rotation in Eq. (2) or (3) seams to be always required.

Globally consistent scan matching is a full SLAM solution for 3D point clouds. It is an offline algorithm and thus optimizes *all* poses in the SLAM pose graph. As a result, processing more scans will increase the algorithm's run-time.

3.2 Continuous-time SLAM

We also developed an algorithm that improves the entire trajectory of the backpack simultaneously. The algorithm is adopted from Elseberg et al. (2013), where it was used in a different mobile mapping context, i.e., on wheeled platforms. Unlike other state of the art algorithms, like (Stoyanov and Lilienthal, 2009) and (Bosse and Zlot, 2009), it is not restricted to purely local improvements. We make no rigidity assumptions, except for the computation of the point correspondences. We require no explicit motion model of a vehicle for instance, thus it works well on backpack systems. The continuous-time SLAM for trajectory optimization works in full 6 DoF, which implies that even planar trajectories are turned into poses with 6 DoF. The algorithm requires no high-level feature computation, i.e., we require only the points themselves.

In case of mobile mapping, we do not have separate terrestrial 3D scans. In the current state of the art in the robotics community developed by Bosse and Zlot (2009) for improving overall map quality of mobile mappers, the time is coarsely discretized. This results in a partition of the trajectory into sub-scans that are treated rigidly. Then rigid registration algorithms like the ICP and other solutions to the SLAM problem are employed. Obviously, trajectory errors within a sub-scan cannot be improved

[2] `http://threedtk.de`

in this fashion. Applying rigid pose estimation to this non-rigid problem directly is also problematic since rigid transformations can only approximate the underlying ground truth. When a finer discretization is used, single 2D scan slices or single points result that do not constrain a 6 DoF pose sufficiently for rigid algorithms.

More mathematical details of our algorithm are given in (Elseberg et al., 2013). Essentially, we first split the trajectory into sections, and match these sections using the automatic high-precision registration of terrestrial 3D scans, i.e., globally consistent scan matching that is the 6D SLAM core. Here the graph is estimated using a heuristic that measures the overlap of sections using the number of closest point pairs. After applying globally consistent scan matching on the sections the actual continuous-time or semi-rigid matching as described in (Borrmann et al., 2008; Elseberg et al., 2013) is applied, using the results of the rigid optimization as starting values to compute the numerical minimum of the underlying least square problem. To speed up the calculations, we make use of the sparse Cholesky decomposition by (Davis, 2006).

Given a trajectory estimate, we "unwind" the point cloud into the global coordinate system and use nearest neighbor search to establish correspondences at the level of single scans (those can be single 2D scan profiles). Then, after computing the estimates of pose differences and their respective covariances, we optimize the trajectory. In a predependend step, we consider trajectory elements every k steps and fuse the trajectory elements around these steps l temporarily into a meta-scan.

A key issue in continuous-time SLAM is the search for closest point pairs. We use an octree and a multi-core implementation using OpenMP to solve this task efficiently. A time-threshold for the point pairs is used, i.e., we match only to points if they were recorded at least t_d time steps away. This time corresponds to the number of separate 3D point clouds acquired by the PUCKs. It is set to 0.1 sec. ($k = 300$, $l = 300$). In addition, we use a maximal allowed point-to-point-distance which has been set to 50 cm.

Finally, all scan slices are joined in a single point cloud to enable efficient viewing of the scene. The first frame, i.e., the first 3D scan slice from the PUCKs scanner defines the arbitrary reference coordinate system.

4 BOOTSTRAPPING CONTINUOUS-TIME SLAM WITH GOOGLE'S SLAM SOLUTION

For improving the Cartographer 3D mapping, the graph is estimated using a heuristics that measures the overlap of sections using the number of closest point pairs. After applying globally consistent scan matching on the sections for several iterations the actual continuous-time SLAM is started.

The data set provided by Google is challenging in several ways: Due to the enormous amount of data, clever data structures are needed to store and access the point cloud. We split the trajectory every 300 PUCK-scans and join $\pm$150 PUCK-scans into a meta-scan. These meta-scans are processed with our octree where we use a voxel size of 10 cm to reduce the point cloud by selecting one point per voxel. We prefer a data structure that stores the raw point cloud over a highly approximative voxel representation. While the latter one is perfectly justifiable for some use cases, it is incompatible with tasks that require exact point measurements like scan matching. Our implementation of an octree prioritizes memory efficiency. We use pointers in contrast to serialized pointer-free encodings in order to efficiently access the large amounts of data. The octree is free of redundancies and is nevertheless capable of fast access operations. Our implementation allows for access operations in $O(\log n)$. We use 6-bytes for pointers as this is sufficient to address a total of 256 terabyte. Two bit fields signal if there is a child or leaf node, thus our implementation needs a few bit operations and the 8 byte are sufficient for an octree node.

The preprocessing step of the continuous-time SLAM runs for 20 iterations, where the edges in the graph are added, when more than 400 point pairs between these meta-scans are present. The maximal allowed point-to-point distance is set to 50 cm. Figure 3 and present 4 results where the consistency of the point cloud has been improved. Figure 5 shows the modifications in the 3D point cloud, while Figure 6 details the changes in the trajectory's position and orientation. It is an open traverse, thus the changes are mainly at the end of the trajectory. Processing was done on a server featuring four Intel Xeon CPUs E7-4870 with 2.4 GHz (40 cores, 80 threads). We have optimized the Google data set for 12 to 15 days (few interruptions).

5 FURTHER CARTOGRAPHER SLAM RESULTS

In further experiments, we evaluated the Cartographer. As the library is fully integrated into ROS, we are able to exchange on our backpack HectorSLAM or TSD SLAM with Cartographer. Indoor office-like environments are no challenge for all the named algorithms. Featureless environments such as long tunnels or outdoor environments are problematic. Figure 2 shows a typical erroneous map obtained with default parameters of Cartographer vs. HectorSLAM in the environment where the photo Figure 1 (right) has been taken.

6 SUMMARY, CONCLUSION AND FUTURE WORK

This paper revisits a continuous-time SLAM algorithm and its application on Google's Cartographer sample data. The algorithm starts with splitting the trajectory into sections, and matches these sections using the automatic high-precise registration of terrestrial 3D scans.

Needless to say, a lot of work remains to be done. First of all, we plan to evaluate 2D mapping method as we have indicated above. Secondly, as calibration is as crucial as SLAM, we will apply our calibration framework (Elseberg et al., 2013) to the data files provided by Google. Furthermore, we will transfer our continuous-time SLAM to different application areas, e.g., underwater and aerospace mapping applications.

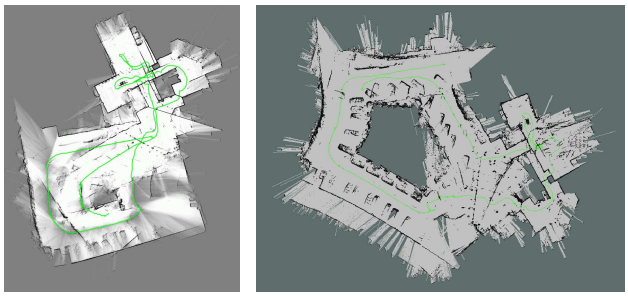

Figure 2: Left: Cartographer. Right: HectorSLAM.

Figure 3: Results of continuous-time SLAM on Google's Cartographer sample data set Deutsches Museum in München. Left: input. Right: output of our solution. Shown are 3D views (perspective) of the scene. Major changes in the point cloud are highlighted in red. Continued in Figure 4

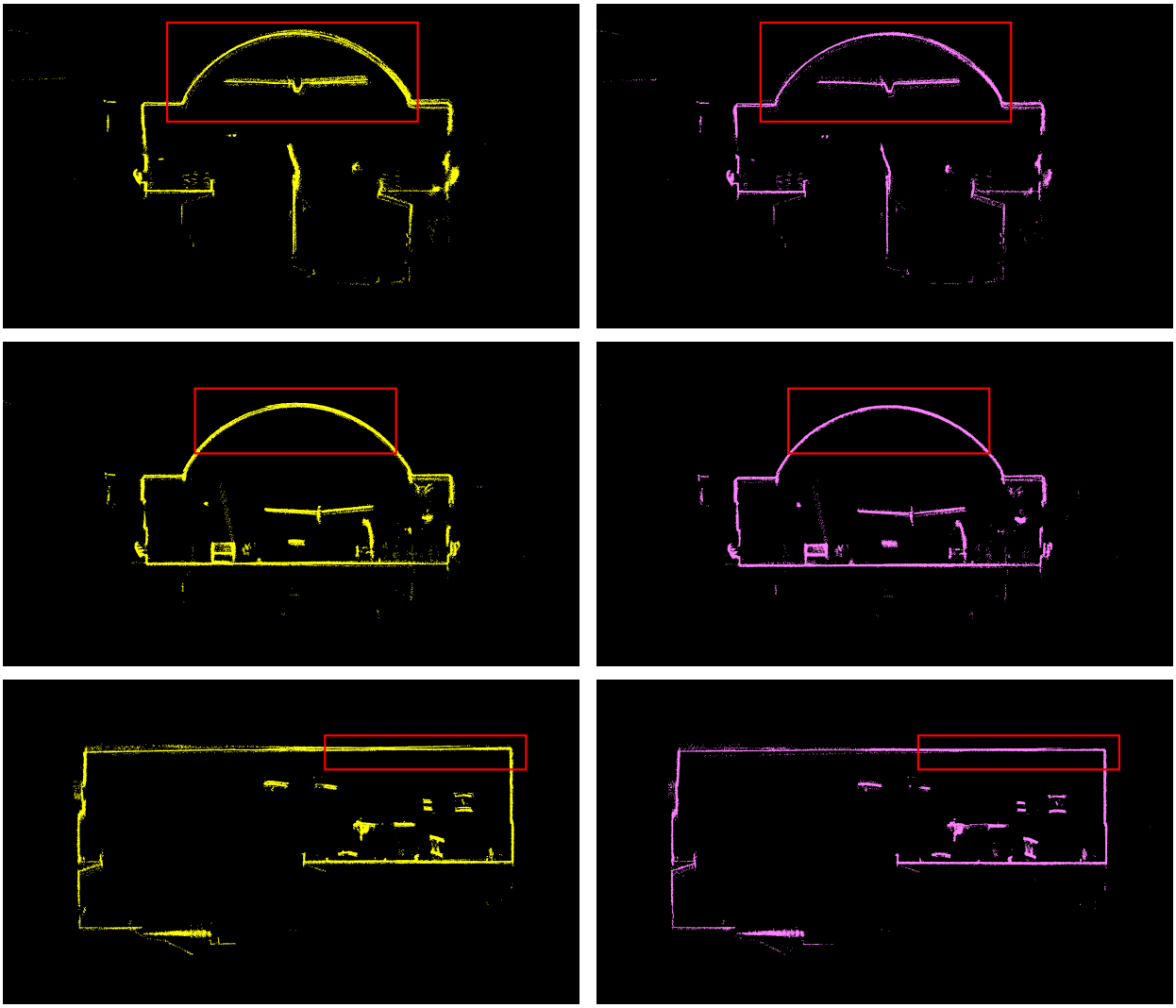

Figure 4: Results of continuous-time SLAM on Google's Cartographer sample data set Deutsches Museum in München. Left: input. Right: output of our solution. Shown are sectional views of the museum hall. Major changes in the point cloud are highlighted in red.

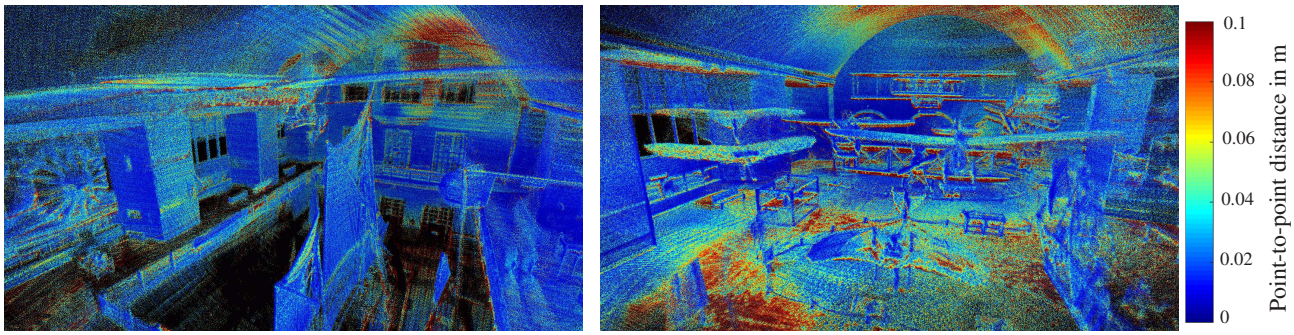

Figure 5: Visualization of the changes in the point cloud. Shown are two views of the optimized 3D point cloud, colored with the difference to the result from Google's Cartographer.

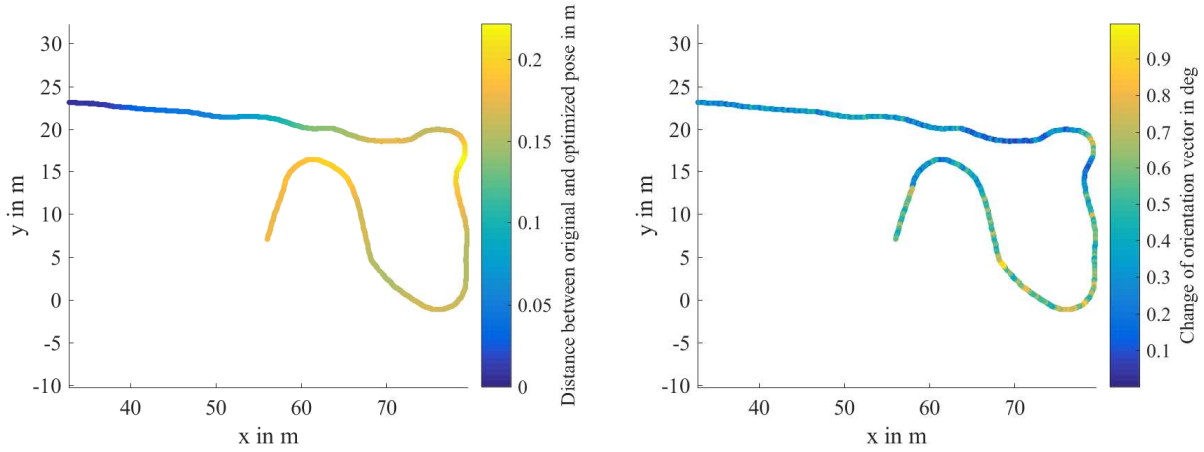

Figure 6: Visualization of the changes in the trajectory computed by our method to bootstraped trajectory. Left: distance. Right: orientation

References

Agarwal, S., Mierle, K. and Others, n.d. Ceres solver. `http://ceres-solver.org`.

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

Borrmann, D., Elseberg, J., Lingemann, K., Nüchter, A. and Hertzberg, J., 2008. Globally consistent 3d mapping with scan matching. *Journal Robotics and Autonomous Systems (JRAS)* 56(2), pp. 130–142.

Bosse, M. and Zlot, R., 2009. Continuous 3D Scan-Matching with a Spinning 2D Laser. *In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '09)*, pp. 4312–4319.

Chen, G., Kua, J., Shum, S., Naikal, N., Carlberg, M., and Zakhor, A., 2010. Indoor Localization Algorithms for a Human-Operated Backpack System. *In: Proceedings of the International Conference on 3D Data Processing, Visualization, and Transmission (3DPVT '10)*, Paris, France.

Davis, T. A., 2006. Direct Methods for Sparse Linear Systems. SIAM.

Elseberg, J., Borrmann, D. and Nüchter, A., 2013. Algorithmic solutions for computing accurate maximum likelihood 3D point clouds from mobile laser scanning platforms. *Remote Sensing* 5(11), pp. 5871–5906.

Hess, W., Kohler, D., Rapp, H. and Andor, D., 2016. Real-time loop closure in 2d lidar slam. *In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '16)*, Stockholm, Sweden.

Kohlbrecher, S., Meyer, J., von Stryk, O. and Klingauf, U., 2011. A flexible and scalable slam system with full 3d motion estimation. *In: Proceedings of the IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR ' 11)*.

Liang, X., Kukko, A., Kaartinen, H., andY. Xiaowei, J. H., Jaakkola, A. and Wang, Y., 2014. Possibilities of a personal laser scanning system for forest mapping and ecosystem services. *Sensors* 14(1), pp. 1228–1248.

Lu, F. and Milios, E., 1994. Robot Pose Estimation in Unknown Environments by Matching 2D Range Scans. *In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '94)*, pp. 935–938.

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

Nüchter, A., Borrmann, D., Koch, P., Kühn, M. and May, S., 2015. A man-portable, imu-free mobile mapping system. *In: Proceedings of the ISPRS Geospatial Week 2015, Laserscanning 2015*, ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci., II-3/W5, La Grande Motte, France, pp. 17–23.

Nüchter, A., Elseberg, J., Schneider, P. and Paulus, D., 2010. Study of Parameterizations for the Rigid Body Transformations of The Scan Registration Problem. *Journal Computer Vision and Image Understanding (CVIU)* 114(8), pp. 963–980.

Olson, E., 2010. A Passive Solution to the Sensor Synchronization Problem. *In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '10)*, pp. 1059–1064.

Quigley, M., Gerkey, B., Conley, K., Faust, J., Foote, T., Leibs, J., Berger, E., Wheeler, R. and Ng, A., 2009. Ros: an open-source robot operating system. *In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '09)*, Kobe, Japan.

Rieger, P., Studnicka, N. and Pfennigbauer, M., 2010. Boresight Alignment Method for Mobile Laser Scanning Systems. *Journal of Applied Geodesy (JAG)* 4(1), pp. 13–21.

Saarinen, J., Mazl, R., Kulich, M., Suomela, J., Preucil, L. and Halme, A., 2004. Methods For Personal Localisation And Mapping. *In: Proceedings of the 5th IFAC symposium on Intelligent Autonomous Vehicles (IAV '04)*.

Sheehan, M., Harrison, A. and Newman, P., 2011. Self-calibration for a 3D Laser. *International Journal of Robotics Research (IJRR)* 31(5), pp. 675–687.

Skaloud, J. and Schaer, P., 2007. Towards Automated LiDAR Boresight Self-calibration. *In: Proceedings of the 5th International Symposium on Mobile Mapping Technology (MMT '07)*.

Stoyanov, T. and Lilienthal, A. J., 2009. Maximum Likelihood Point Cloud Acquisition from a Mobile Platform. *In: Proceedings of the IEEE International Conference on Advanced Robotics (ICAR '09)*, pp. 1– 6.

Thrun, S., Fox, D. and Burgard, W., 2000. A Real-time Algorithm for Mobile Robot Mapping with Application to Multi Robot and 3D Mapping. *In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '00)*, San Francisco, CA, USA.

Underwood, J. P., Hill, A., Peynot, T. and Scheding, S. J., 2009. Error Modeling and Calibration of Exteroceptive Sensors for Accurate Mapping Applications. *Journal of Field Robotics (JFR)* 27(1), pp. 2–20.