Towards Spherical Robots for Mobile Mapping in Human Made Environments*

Fabian Arzberger, Anton Bredenbeck, Jasper Zevering, Dorit Borrmann, Andreas Nüchter

Informatics VII: Robotics and Telematics, Julius-Maximilians-University Würzburg, Germany
e-mail: dorit.borrmann@uni-wuerzburg.de

Abstract

Spherical robots are a format that has not been thoroughly explored for the application of mobile mapping. In contrast to other designs, it provides some unique advantages. Among those is a spherical shell that protects internal sensors and actuators from possible harsh environments, as well as an inherent rotation for locomotion that enables measurements in all directions. Mobile mapping always requires a high-precise pose knowledge to obtain consistent and correct environment maps. This is typically done by a combination of external reference sensors such as Global Navigation Satellite System (GNSS) measurements and inertial measurements or by coarsely estimating the pose using inertial measurement units (IMUs) and post processing the data by registering the different measurements to each other. In indoor environments, the GNSS reference is not an option. Hence many mobile mapping applications turn to the second option. An advantage of indoor environments is that human-made environments usually have a certain structure, such as parallel and perpendicular planes. We propose a registration procedure that exploits this structure by minimizing the distance of measured points to a corresponding plane. Further, we evaluate the procedure on a simulated dataset of an ideal corridor and on an experimentally acquired dataset with different motion profiles. We show that we nearly reproduce the ground truth for the simulated dataset and improve the average point-to-point distance to a reference scan in the experimental dataset. The presented algorithms are required to work completely autonomously.

Keywords: `Mobile Mapping`, `SLAM`, `Spherical Robot`
2021 MSC: 00-01, 99-00

1. Introduction

Today's robots for mobile mapping come in all shapes and sizes. State of the art for urban environments are laser scanners mounted to cars. Smaller robotic systems are particularly used when cars no longer have access. Examples for this are human operated systems such as Zebedee [1], a small Hokuyo 2D scanner on a spring, that is carried through the environment, VILMA [2], a rolling FARO scanner operating in profiler mode, RADLER [3], a SICK 2D laser scanner mounted to a unicycle, or a backpack mounted "personal laser scanning system" as in [4] or [5]. Recently more and more autonomous systems gained maturity. A stunning example is Boston Dynamics' quadruped "Spot" that autonomously navigates and maps human environments [6]. Also, the mobile mapping approaches implemented on the ANYmal platform such as [7] were very successful. Of all these formats, one has not been explored thoroughly in the scientific community: The spherical mobile mapping robot. Yet this provides some very promising advantages over the other formats. For one, the locomotion of a spherical robot inherently results in rotation. That way, a sensor fixed inside the spherical structure will cover the entire environment, given the required locomotion without the need for additional actuators for the sensors. This requires a solution for the spherical simultaneous localization and mapping (SLAM) problem, given the six degrees of freedom of the robot. Secondly, a spherical shell that encloses all sensors protects these from possible hazardous environments. For example, the shell stops any dust that deteriorates sensors or actuators when settling at sensitive locations. In contrast to an usual enclosing the shell can separate the sensors entirely from the environment without the need for a number of points-of-connection. A strict requirement then is that the shell is very durable. This is particularly useful for unknown or dangerous environments. E.g., old buildings that are in danger of collapsing, narrow underground tunnels, construction sites, or mining shafts. The spherical format is, in fact, also suited for space applications. In the DAEDALUS study [8], such a robot is proposed that is to be lowered into a lunar cave and create a 3D map of the environment. The authors choose this format as the moon regolith is known to damage instruments and other components. They also present an approach to protect the shell from accumulating dust and dirt. However, the spherical format also comes with some disadvantages. Lidar sensors often have a minimum scan distance, resulting in a less dense (or empty) point cloud when the scanner looks

*To abide the FAIR-Principles of science, all data and code used to produce the results are given in our repository (`https://github.com/fallow24/SphereTDP`).

on the ground, whereas density is higher when looking in other directions. The ground itself is likely to be less populated with points, due to weak angles of incidence while mapping it. Furthermore, relying on IMU based odometry as a localization technique alone yields inaccurate and noisy pose measurements. Therefore, a robust registration procedure is needed for spherical robots, that is able to cope with vast differences is point cloud density and high noise regarding pose measurements.

This paper proposes to use such a spherical robot for mobile mapping man-made environments. In such environments, one advantage are architectural shapes following standard conventions arising from tradition or utility. In particular, there are many flat surfaces such as walls, floors, etc. that are sensed. Exploiting this fact yields more opportunities for registration as point-to-plane correspondences can be used. The proposed registration method minimizes the distances of each point to its corresponding plane as an objective function.

The next sections introduce you to the state of the art SLAM algorithms for spherical robots and plane based registration. We then present our approach, which includes global plane extraction, point to plane correspondences, and an optimized gradient descent that minimizes point to plane distances. Furthermore, we show the results on simulated, as well on real world datasets.

2. State Of The Art

2.1. Spherical SLAM

While there exist camera-based approaches [9], to the best of our knowledge, SLAM with spherical robots and laser scanners was not done before. However, laser-based SLAM algorithms for motions in six degrees of freedom (DoF) have been thoroughly studied. For outdoor environments [10] provides a first baseline. Adding a heuristic for closed-loop detection and a global relaxation Borrmann et al. yield highly precise maps of the scanned environment [11]. Thereby they reduce the position and orientation error of the scanning poses by orders of magnitude in the range of centimeters and hundreds of a degree. Zhang et al. propose a real-time solution to the SLAM problem in [12]. They achieve the performance at a lower computational load by dividing the SLAM algorithm into two different algorithms: one performing odometry at a high frequency but low fidelity and another running at a lower frequency performing fine matching and registration of the point clouds. More recently Dröschel et al. also propose an online method using a novel combination of a hierarchical graph structure with local multi-resolution maps to overcome problems due to sparse scans [13]. Another intriguing example is the NavVis VLX system [14]. They perform real time SLAM on a mobile, wearable platform, producing colored point clouds with high accuracy by combining it with a camera system. However, this platform has to be moved by a human operator. In particular, they are able to achieve one sigma of measurements within three millimeters difference the measurements of a ground truth terrestrial laser scan [15]. They also employ artificial markers that can be placed in the environment, which get recognized by the camera system to further improve registration accuracy. Obviously, this is not possible in inaccessible, or dangerous environments.

Since these approaches are based on point-to-point correspondences, they require a rather high point density to achieve precise registration. For low-cost LiDARs, this implies slow motion and long integration time.

2.2. Point Cloud Registration Using Plane Based Correspondences

The de-facto standard for many SLAM algorithms is the Iterative-Closest-Point (ICP) algorithm [16] that employs point-to-point correspondences using closest points, as the name suggests. To overcome the requirements on point-density imposed by the point-to-point correspondences instead other correspondences are used. In human-made environments, planes are abundantly available and hence provide an attractive base for correspondences.

Pathak et al. [17] propose to reduce the complexity of the registration by using correspondences between planar patches instead of points. They demonstrate the effectiveness of their approach even with noisy data in cluttered environments. However, their approach is designed for data acquired in stop-scan-go fashion and not for mobile mapping applications. As they use a region growing procedure with randomized initialization for detecting planar patches the distortions in the data introduced by pose uncertainties are likely to effect the shape of the planar patches and lead to faulty correspondences.

Förster et al. use the planarity of human-made environment successfully in [18]. They register point clouds using plane-to-plane correspondences and include uncertainty measures for the detected planes and the estimated motion. Thereby, they propose a costly exact algorithm and cheaper approximations that yield high-quality maps. Favre et al. [19] use point-to-plane correspondences after preprocessing the point clouds using plane-to-plane correspondences to register two scans with each other successfully.

Both approaches use plane-to-plane correspondences to pre-register the scans. However, for pre-registration the classical point-to-point registration is also very effective. One advantage that point-to-point correspondences have over plane-to-plane correspondences is that they do not require a long stop in order to obtain enough points to detect planes in each pose. For plane-to-plane correspondences, this is necessary to gather enough data to measure planes in each scan robustly. In particular this pause is needed when the field-of-view (FOV) is limited, as the detected planes are thin slices of the true planes which are difficult to find correct correspondences for. The resulting scan procedure is stop-scan-and-go. In particular, for the application of a spherical robot this standstill in each

pose cannot be guaranteed or even approximated, making continuous-time approaches using point-to-plane correspondences the method of choice.

Lidar odometry and mapping (LOAM) [12] is the baseline-algorithm that provides a real-time and lowdrift solution based on two parallel registration algorithms using planes and lines. Unfortunately, it is not open-source anymore. LOAM livox extends the LOAM framework to the rotating prism scanner with small FoV [20]. Zhou, Wang, and Kaess write that it also adopts the parallel computing to achieve real-time global registration. Parallel computing needs a powerful CPU, it may be not suitable for an embedded system which has limited computational resources [21]. Thus, they extend their smoothing and mapping to the LiDAR and planes case. In their experiments they used a VLP-16 LiDAR to collect indoor datasets.

Further recent planar SLAM approaches include [22, 23, 24, 21, 25]. While Wei et al. uses only the ground plane in outdoor experiments. Indoors, Jung et al. used in 2015 several Hokuyo laser scanners for their kinematic scanning system [22], while Grant used a single Velodyne HDL-32E sensor with 32 rotating laser/receiver pairs mounted on a backpack and walked through different environments [23]. The LIPS system [24] employes a so-called Closest Point Plane Representation with an Anchor Plane Factor. Random sample consensus (RANSAC) is used to find the planes. Their system couples an eight channel Quanergy M8 LiDAR operating at 10Hz with a Microstrain 3DM-GX3-25 IMU attached to the bottom of the LiDAR operating at 500Hz.

In the following, we propose this combination of point-to-point based pre-registration followed by a point-to-plane based optimization.

3. Registration Algorithm

This section describes the presented algorithm. We first pre-register all individual scans to align them in one global coordinate system. Then, planes are extracted from this global scan model. The key to improving map quality is to find a model for each individual scan, where each point is correctly identified to belong to one of the globally extracted planes. Then, each scan is optimized by finding a 6D pose-transformation that minimizes the distance of each correspondence. Note that all of this takes place on a reduced version of the scan, i.e. an octree based reduction where each voxel is only allowed to have a certain amount of points. This ensures a homogenous point density, which is favourable for the error function. It also decreases runtime, while preserving the planar features of the environment.

Consider a line scan as the smallest chunk of range data we obtain from the scanner device driver. In the case of a SICK LMS1xx it is a line, and in case of a Livox scanner it has a flower shape. More details are provided in the following sections. We start with transforming each line scan into the project coordinate system, which is defined by the pose of the first acquired line scan. For map improvement, the individual scans need to be registered to another. We propose an algorithm that consists of multiple steps outlined in algorithm 1. Based on ideas described in [26] we first pre-register the scans and then further improve the overall map by exploiting the fact that human-made environments often consist of planes. We then find the planes in the pre-registered point cloud and then optimize the poses associated with the scans to minimize the distance of all points to their respective planes.

Algorithm 1: Registration algorithm for man-made environments

Result: A corrected map and 6D path.

1. Pre-register the scans;
2. Extract planes from the registered map globally;
3. Further improve map by solving the optimization that minimizes the distance of all points to their respective planes;

3.1. Pre-Registration

For the plane extraction the linescans need to be transformed into the project coordinate system. Various pre-registration methods are suitable for this. The most simple method is to use the data from an intertial measurement unit for a coarse alignment. Alternative solutions combine multiple linescans into a metascan and perform registration methods known from terrestrial laser scanning and distribute the error over the trajectory [27, 28] or use a small number of iterations of continuous-time SLAM approaches such as the one presented in [29]. This yields maps that can be used for plane registration and hence a point-to-plane correspondences based registration.

3.2. Plane Extraction

After having done the pre-registration, the scans are aligned well enough to make statements about the potential planes in the environment. We extract planes only once from the global scan model, or a portion of that global scan model. The first few line scans, e.g. the first half of the global model, is sufficient for plane extration. This is because within the initial movements of the robot system, pose error has not accumulated for very long, resulting in a less dense, yet less distorted representation of the environment. For mobile robots this yields a quite consistent strategy for global plane extraction, since all the subsequent scans (where pose error keeps accumulating) get matched with the initial planar representation. However, especially for large datasets, it is necessary to update the global plane model multiple times while establishing point to plane correspondences. Currently, this is not done, and is one of the primary objectives for future works.

To find the planes in the environment, a Randomized Hough transform (RHT) with an accumulator ball as described in [30] is used as this method prefers dominant planes such as the building structure over smaller planar surfaces. The RHT is combined with a region growing approach, similar plane merging, and a flatness filter that is based on principal component analysis (PCA). Once a plane is identified, we calculate the convex hull of that plane, representing the extent in all directions. This way, large planar structures such as walls or the ceiling get identified. Note that smaller, non-perpendicular planes such as doors or cupboards potentially get identified, too, if they pass the PCA filter.

In the next subsection, the Hesse-normal form of the plane is required. For an ideal plane the orthogonal distance from the origin $\rho_{\mathcal{P}_k}$ is computed via $\mathbf{n}_{\mathcal{P}_k} \cdot \mathbf{p}_i$, where $\mathbf{p}_i = \begin{bmatrix} x_i & y_i & z_i \end{bmatrix}^T$ is an arbitrary point on the plane and $\mathbf{n}_{\mathcal{P}_k} = \begin{bmatrix} n^x_{\mathcal{P}_k} & n^y_{\mathcal{P}_k} & n^z_{\mathcal{P}_k} \end{bmatrix}^T$ is the normal vector of that plane. To find this point, we use the convex hull, as it is defined by the points that lie on the plane with the furthest distance to one another. We choose the center point of the convex hull of the plane as $\mathbf{a}_{\rho_k}$.

3.3. Point-to-Plane Correspondences

There are many ways of solving the segmentation problem of assigning each point to a plane. They are often based on the RANSAC method [31], [32] but also deep learning approaches [33] have been successful. The quality of the preregistration directly influences the quality of the plane detection and therefore the quality of the final registration result. We assume small enough errors in pre-registration to allow for plane detection. We employ two distance models to represent the distance from a point to a plane, and combine them with a local planar clustering (LPC) approach to establish point-to-plane correspondences.

3.3.1. Distance Models

The Hesse-normal form describes the distance from the i-th transformed point $T(\mathbf{p}_i)$ to infinitely extending, k-th plane in 3D vector space

$$D_h^{i,k} = \mathbf{n}_{\mathcal{P}_k} \cdot \left[T(\mathbf{p}_i) - \mathbf{a}_{\mathcal{P}_k} \right], \tag{1}$$

with normal $\mathbf{n}_{\mathcal{P}_k}$ and supporting point $\mathbf{a}_{\mathcal{P}_k}$ of the k-th plane. The distance $D_h^{i,k}$ of the i-th point to the k-th plane reflects the length of the line segment, constructed from the i-th point to its projection on the k-th plane. Therefore, by definition the line segment is parallel to the normal vector of the plane. Thus, the i-th points projection $\widetilde{T}(\mathbf{p}_i)$ onto the k-th plane, which is required later, is easily calculated by shifting the point against normal direction:

$$\widetilde{T}(\mathbf{p}_i) = T(\mathbf{p}_i) - D_h^{i,k} \mathbf{n}_{\mathcal{P}_k} \tag{2}$$

The simplicity of the Hesse distance is at contrast with its inability to take the extent of the plane into account. Therefore, a second distance model is introduced: the polygon projection distance (PPD). The convex hull represents the extend of a plane by forming a polygon with the outer most points that are assigned to a given plane. It is able to represent the expansion of the plane in all directions, and thus is utilized as a distance model. To find the PPD, first a corresponding point gets projected onto the infinitely extending plane representation given by the Hesse form, i.e. $\widetilde{T}(\mathbf{p}_k)$ is found from eq. (2) (green point in figure 1). Then, the 3D polygon, i.e. the points that make up the convex hull, as well as the corresponding 3D point projection, are projected again into a 2D vector space, using the maximal component of the normal vector of the plane. E.g. in figure 1, the blue planes normal vector has its mayor component in z-direction (upwards), thus the projection polygon distance is calculated in the xy-plane. Using the maximum of the absolute magnitude of the normal vector ensures that the most sensible 2D projection is used for every direction. Sunday et al. [34] presents an optimal algorithm to check whether the point projection lies inside the polygon in 2D. If the point is inside the polygon, its PPD is set to zero. If, however, the point is outside the polygon, the shortest distance to the polygon is calculated by looking at the minimum distance of the i-th point to each line segment, making up the convex hull of the k-th plane in 3D. Let the polygon P_k be made up of line segments $s_{m,j}$. The line segment $s_{m,j}$ consists of the points p_m and p_j. The line segment is parameterized by

$$s_{m,j}(t) = p_m + t \cdot (p_j - p_m), \tag{3}$$

where $t \in [0, 1]$. We set up a distance function, which measures the distance from the query point p_i to an arbitrary point on the line segment.

$$d_{m,j}(t) = \| s_{m,j}(t) - p_i \| \tag{4}$$

It is now possible to find the shortest distance to the line segment by finding the argument of the function that minimizes this distance. Therefore, we set

$$\frac{\partial}{\partial t} d_{m,j}(t)^2 \stackrel{!}{=} 0, \tag{5}$$

resulting in

$$t_0 = \frac{(p_j - p_m) \cdot (p_i - p_m)}{(p_j - p_m)^2}. \tag{6}$$

Note the possibility of $t_0 \notin [0, 1]$. In that case, the projection of the point p_i onto the line given by eq. (3) is not between p_j and p_m. Instead, its projection onto the line falls outside of the segment. By constraining t_0 we get

$$\hat{t} = \min\left(\max\left(t_0, 0\right), 1\right), \tag{7}$$

which is the argument that gives the shortest distance to the line segment, when inserted to eq. (4). We find the PPD from the point p_i to the polygon P_k by calculating the minimum shortest distance over all line segments that make up the polygon.

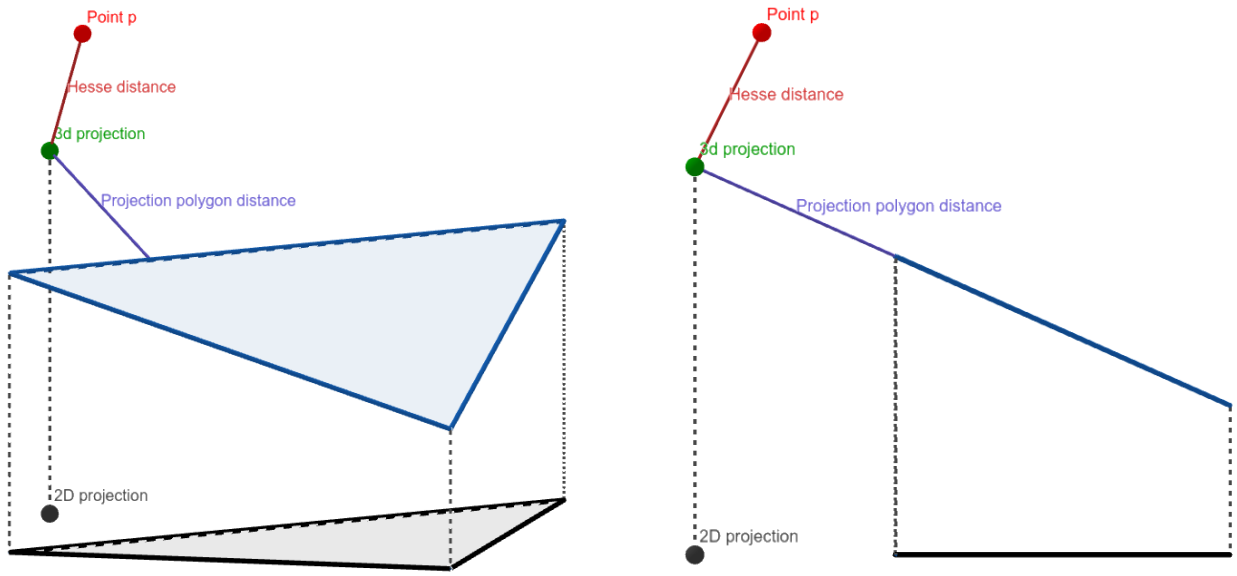

Figure 1: Illustration of the Hesse- and polygon projection distance. The point p (red) gets projected onto the infinitely extending global plane. Both the point projection (green) and the global planes convex hull (blue) get projected into 2D space (grey). The Hesse distance, i.e. the shortest distance to the infinitely extending plane (red), is shown as well as the minimum distance from the 3D point projection to the polygon (purple).

3.3.2. Local Planar Clustering

The presented algorithm employs LPC on each individual line scan. The idea is to find planar features utilizing an optimized approximate k nearest neighbor (AKNN) search for normal calculation as in [35]. Thus, normals are calculated for every point in the respective line scan. Then, the local normal model is combined with a region growing approach. The region growing is implemented by storing the k-nearest neighbors in a queue, which represents the iteration order of the clustering algorithm. A region grows if one of these neighbors has a distance shorter then a threshold d_{growth} and additionally has a similar normal. We define similarity of two normals via the angle between them, which is:

$$\alpha(\mathbf{n}_1, \mathbf{n}_2) = \arccos(\mathbf{n}_1 \cdot \mathbf{n}_2) \; . \tag{8}$$

However, the *smallest* angle between two *normal* vectors is

$$\hat{\alpha}(\mathbf{n}_1, \mathbf{n}_2) = \begin{cases} 2\pi - \alpha(\mathbf{n}_1, \mathbf{n}_2) & \text{if } \alpha(\mathbf{n}_1, \mathbf{n}_2) > \frac{3}{2}\pi \\ \alpha(\mathbf{n}_1, \mathbf{n}_2) - \pi & \text{if } \alpha(\mathbf{n}_1, \mathbf{n}_2) > \pi \\ \pi - \alpha(\mathbf{n}_1, \mathbf{n}_2) & \text{if } \alpha(\mathbf{n}_1, \mathbf{n}_2) > \frac{\pi}{2} \end{cases} , \tag{9}$$

because the opposing normals $-\mathbf{n}_1$ and $-\mathbf{n}_2$ always have to be considered, since they correspond to the same plane. We say that two normals are similar if the angle $\hat{\alpha}$ from eq. (9) is smaller than a threshold ε_α.

Planar areas like walls are therefore connected in one large cluster, while non-feature parts, i.e. non-planar or detached parts of the scan will have their own, smaller cluster. Finally the clusters are filtered in two steps. First, to combine clusters that are similar, i.e. have a short distance (d_{growth}) to each other and similar mean normals (similarity defined as before). Second, to filter out all clusters that contain non-feature points. Since all the non-feature points belong to small clusters, the presented algorihtm does this by putting a minimum threshold $N_{c,min}$ on the cluster size. Figure 2 shows this concept on the real world dataset described later in section 5.

Point-to-Plane correspondences are now established. The distance models (Hesse and PPD) are utilized to check whether a cluster overlaps with any of the global planes, i.e. any point in the cluster has a Hesse distance smaller than some threshold ε_H and a PPD smaller than some additional threshold ε_P, to the global plane. We observe the noise level around the expected planes by the naked eye. Then we set the thresholds accordingly. For each overlap, we find the smallest angle between the corresponding cluster normal and the global plane normal, according to eq. (9). Finally, a correspondence is established between all points in the cluster and the global plane, if the minimum angle between their normals is smaller than ε_α.

3.4. Optimization

Assuming we now know the Hesse-normal form of all planes and all assigned points to these planes, we register the points to have a minimal distance to their respective planes. The transformation $T(\mathbf{p}_i)$ of each point $\mathbf{p}_i$ with respect to a 6 DoF motion is described in homogeneous coordinates using the roll-pitch-yaw ($\varphi - \vartheta - \psi$) Tait-Brian angles as in [36]. Transforming the result back from homogeneous coordinates and using C_a and S_a to denote the cosine and sine of the angle in the subscript, and t_a denot-

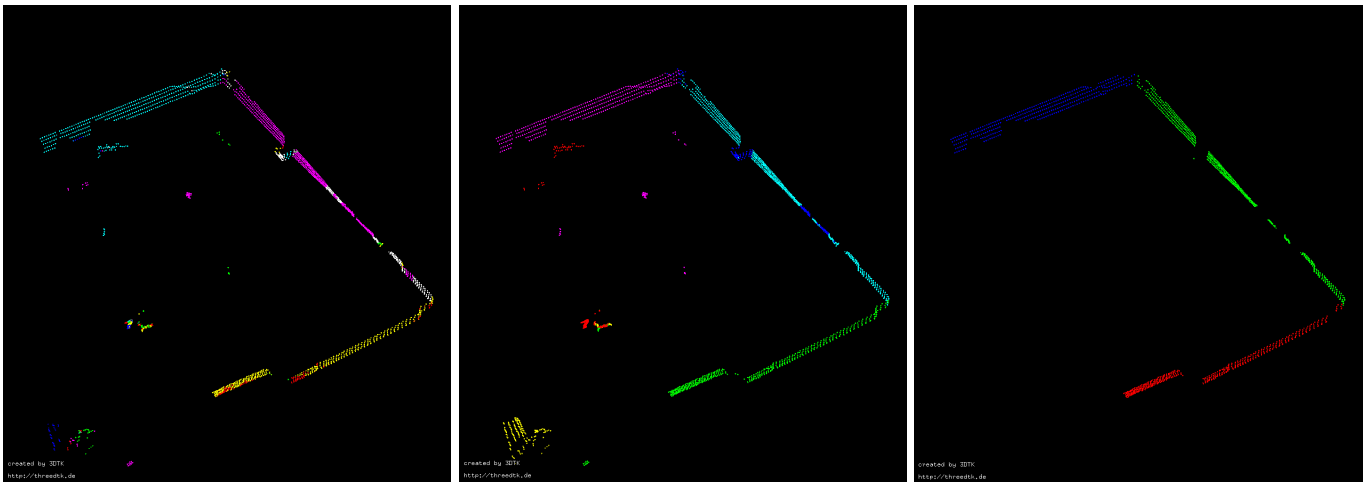

Figure 2: (Left) Point normals using the AKNN method with $K = 20$ (number of nearest neighbors). (Middle) Result of the region growing, before applying the second filter. (Right) Resulting clusters after applying the second filter, using $N_{c,min} = 300$. In each image, one color represents a segment.

ing the translation along the axis in the subscript yields:

$$T(\mathbf{p}_i) = \begin{bmatrix} x_i C_\vartheta C_\psi - y_i C_\vartheta S_\psi + z_i S_\vartheta + t_x \\ x_i(C_\varphi S_\psi + C_\psi S_\varphi S_\vartheta) + y_i(C_\varphi C_\psi - S_\varphi S_\vartheta S_\psi) - z_i C_\vartheta S_\varphi + t_y \\ x_i(S_\varphi S_\psi - C_\varphi C_\psi S_\vartheta) + y_i(C_\psi S_\varphi + C_\varphi S_\vartheta S_\psi) + z_i C_\varphi C_\vartheta + t_z \end{bmatrix} \quad (10)$$

From this we define the function $D(\varphi, \vartheta, \psi, t_x, t_y, t_z, \mathbf{p}_i)$ that computes the distance of a point $\mathbf{p}_i$ to its corresponding plane $\mathcal{P}_k$. Omitting the arguments of the function for simplicity:

$$\begin{aligned} D &= T(\mathbf{p}_i) \cdot \mathbf{n}_{\mathcal{P}_k} \\ &= n^x_{\mathcal{P}_k}(x_i C_\vartheta C_\psi - y_i C_\vartheta S_\psi + z_i S_\vartheta + t_x) \\ &+ n^y_{\mathcal{P}_k}(x_i(C_\varphi S_\psi + C_\psi S_\varphi S_\vartheta) \\ &\quad + y_i(C_\varphi C_\psi - S_\varphi S_\vartheta S_\psi) - z_i C_\vartheta S_\varphi + t_y) \\ &+ n^z_{\mathcal{P}_k}(x_i(S_\varphi S_\psi - C_\varphi C_\psi S_\vartheta) \\ &\quad + y_i(C_\psi S_\varphi + C_\varphi S_\vartheta S_\psi) + z_i C_\varphi C_\vartheta + t_z) \\ &- \rho_{\mathcal{P}_k} \end{aligned} \quad (11)$$

This distance function is what we want to minimize for all points and their respective planes. Hence the error function E is chosen as the square of the L2-norm of the distance:

$$E = \sum_{\forall \mathcal{P}_k} \sum_{\mathbf{p}_i \in \mathcal{P}_k} ||D(\varphi, \vartheta, \psi, t_x, t_y, t_z, \mathbf{p}_i)||_2^2 \quad (12)$$

Its gradient follows then immediately:

$$\nabla E = \sum_{\forall \mathcal{P}_k} \sum_{\mathbf{p}_i \in \mathcal{P}_k} \left[\frac{\partial}{\partial\varphi}E \quad \frac{\partial}{\partial\vartheta}E \quad \frac{\partial}{\partial\psi}E \quad \frac{\partial}{\partial t_x}E \quad \frac{\partial}{\partial t_y}E \quad \frac{\partial}{\partial t_z}E\right]^T \quad (13)$$

$$\Rightarrow \nabla E = \textstyle\sum_{\forall \mathcal{P}_k} \sum_{\mathbf{p}_i \in \mathcal{P}_k} 2D(\mathbf{\Pi}, \mathbf{p}_i) \left[\nabla E_\varphi \quad \nabla E_\vartheta \quad \nabla E_\psi \quad n^x_{\mathcal{P}_k} \quad n^y_{\mathcal{P}_k} \quad n^z_{\mathcal{P}_k}\right]^T \quad (14)$$

Where

$$\begin{aligned} \nabla E_\varphi =& n^y_{\mathcal{P}_k}(x_i[-S_\varphi S_\psi + C_\varphi C_\psi S_\vartheta] \\ &+ y_i[-S_\varphi C_\psi - C_\varphi S_\vartheta S\psi] - z_i C_\varphi C_\vartheta) \\ &+ n^z_{\mathcal{P}_k}(x_i[C_\varphi S_\psi + C_\varphi C_\psi S_\vartheta] \\ &+ y_i[C_\varphi C_\psi - S_\varphi S_\vartheta S\psi] - z_i S_\varphi C_\vartheta) \end{aligned} \quad , \quad (15)$$

$$\begin{aligned} \nabla E_\vartheta =& n^x_{\mathcal{P}_k}(-x_i S_\vartheta C_\psi + y_i S_\vartheta S_\psi + z_i C_\vartheta) \\ &+ n^y_{\mathcal{P}_k}(x_i C_\psi S_\varphi C_\vartheta - y_i S_\varphi C_\vartheta S_\psi + z_i S_\vartheta S_\varphi) \\ &+ n^z_{\mathcal{P}_i}(-x_i C_\varphi C_\psi C_\vartheta + y_i C_\varphi C_\vartheta S_\psi - z_i C_\varphi) \end{aligned} \quad , \quad (16)$$

$$\begin{aligned} \nabla E_\psi =& n^x_{\mathcal{P}_k}(-x_i C_\vartheta S_\psi - y_i C_\vartheta C_\psi) \\ &+ n^y_{\mathcal{P}_k}(x_i[C_\varphi C_\psi - S_\varphi S_\vartheta S_\psi] \\ &+ y_i[-C_\varphi S_\psi - S_\varphi S_\theta C_\psi]) \\ &+ n^z_{\mathcal{P}_k}(x_i[S_\varphi C_\psi + C_\varphi S_\psi S_\theta] \\ &+ y_i[-S_\psi S_\varphi + C_\varphi S_\vartheta C_\psi]) \end{aligned} \quad (17)$$

and $\mathbf{\Pi} = \begin{bmatrix}\varphi & \vartheta & \psi & t_x & t_y & t_z\end{bmatrix}^T$.

As the gradient is well-defined we minimize the error function with any gradient based method. The commonly used, well-known stochastic gradient descent (SDG) algorithm computes

$$\mathbf{\Pi}_{k+1} = \mathbf{\Pi}_k - \alpha \nabla E \quad (18)$$

where α is the learning rate. To accelerate convergence and to improve the found solution further modifications are made.

Since we have vastly different effects on the error function by each dimension, the first consideration for improving the SDG is the following: Typically, changes in orientation, i.e., the first three elements of the gradient vector

$\frac{\partial}{\partial\varphi}E$, $\frac{\partial}{\partial\vartheta}E$, and $\frac{\partial}{\partial\psi}E$, have much more impact on the error function than a change in position. This is intuitively explained since translating the scan makes the error grow linearly for all points. However, when rotating the scan, points with a larger distance to the robot are moved drastically, leading to a higher sensibility on the error function. For this reason, the α applied on orientation has to be much smaller than the α applied on the position. It becomes obvious that α needs to be extended into vector form, $\boldsymbol{\alpha}$, therefore weighting each dimension differently.

Another consideration to speed up SDG is to adaptively recalculate $\boldsymbol{\alpha}$ for each iteration. We employ and modify ADADELTA as a technique to do so, which is described in detail in [37]. The main idea is the following: It extends the SDG algorithm by two terms. First, an exponentially decaying average of past gradients $\mathbf{G}_k$, which is recursively defined as

$$\mathbf{G}_{k+1} = \zeta\mathbf{G}_k + (1-\zeta)\nabla E^2 \tag{19}$$

and second, an exponentially decaying average of past changes $\mathbf{X}_k$, which is defined as

$$\mathbf{X}_{k+1} = \zeta\mathbf{X}_k + (1-\zeta)\boldsymbol{\alpha}\nabla E^2 \tag{20}$$

where $\zeta \leq 1$ is a decay constant, typically close to 1. The root mean squared (RMS) of these quantities are

$$RMS[\mathbf{G}]_k = \sqrt{\mathbf{G}_k + \varepsilon_{num}} \tag{21}$$

and

$$RMS[\mathbf{X}]_k = \sqrt{\mathbf{X}_k + \varepsilon_{num}} \tag{22}$$

where $\varepsilon_{num} > 0$ is a very small constant, typically close to 0 (Note: this is a different threshold as the one used in section 3.3). It will prevent dividing by zero in the recalculation of $\boldsymbol{\alpha}$, which is as follows:

$$\boldsymbol{\alpha}_k = \frac{RMS[X]_{k-1}}{RMS[G]_k} \tag{23}$$

For our particular application, ADADELTA behaves a little too aggressively. Despite giving a good measure on how to adapt $\boldsymbol{\alpha}$, the algorithm sometimes overshoots, and does not converge. Therefore, we employ another scaling factor, typically not found in ADADELTA, extending eq. (23) to:

$$\boldsymbol{\alpha}_k = \boldsymbol{\alpha}_0 \cdot \frac{RMS[X]_{k-1}}{RMS[G]_k} \tag{24}$$

where $\boldsymbol{\alpha}_0$ holds the scaling factors for each dimension.

Finally, the SDG model is improved using eq. (24) and extends to

$$\boldsymbol{\Pi}_{k+1} = \boldsymbol{\Pi}_k - \boldsymbol{\alpha}_0 \frac{RMS[X]_{k-1}}{RMS[G]_k} \cdot \nabla E \tag{25}$$

Using this algorithm once after finding correspondences from points to planes leads to convergence to a local minimum, which is often not an optimal solution. Even if we increase the number of iterations dramatically, no better solution than the local minimum is found. That is unless you consider updating the correspondence model after i iterations of gradient descent. Re-assigning point-to-plane correspondences this way k times, with a large enough k, leads to an optimal solution after maximum $n = k \cdot i$ iterations of gradient descent.

3.5. Further Optimizations of the Algorithm

The algorithm was extended by two further optimizations, which are helpful in different scenarios.

Firstly, we introduce the possibility to choose a lock on some dimensions from being optimized by setting the corresponding α_i to zero. Although 6D optimization generally works, reducing the optimization space is particularly useful if the source of error in the system is known and a model exists. That way, e.g., as we expect a spherical robot to move on a plane the position along the axis perpendicular to the plane is constant and should not be used for optimization.

Secondly, we employ a sequential iteration, where the scans are processed one after another. This is especially useful for mobile systems, where pose errors accumulate due to increasing tracking uncertainties. The assumption is that the error in one scan is also present in the following scan, plus some unknown new error. We eliminate the error from scan m which is also present in scan $m+1$ by applying the pose change from scan m, $\boldsymbol{\Pi}_{n,m} - \boldsymbol{\Pi}_{0,m}$ after n gradient descent iterations, to scan $m+1$, before restarting gradient descent. To quantify the quality of the proposed registration method, it is tested on simulated and real-world data.

4. Simulation

For the simulated data, we implemented a noisy world-robot-sensor model. The simulated sensor is modeled after a Livox-Mid100 [38] laser scanner with customizable noise level on the range measurements inside a robot with different motion capabilities, subject to noise in its pose estimation. This yields scan results similar to the ones obtained in the real world.

4.1. Simulated Sensor

The Livox-Mid100 laser scanner consists of three Mid40 scanners arranged to provide an overall field-of-view of 98.4° (horizontal) by 38.4° (vertical). It provides a rate of 300.000 points/s with a range precision of 2 cm up to 20 m range [38]. A particular advantage of the Livox laser scanners is their unique scanning pattern. The scanners use two motorized, so-called Risley prisms to steer the beam deflectance. For the specific ratio of rotation speeds used in the Livox scanners, the beam traces out a non-repetitive, flower-shaped scanning pattern [39]. Since the pattern is non-repetitive, the point density increases with integration time, enabling denser measurements.

4.2. Range Noise Model

Most real laser scanners are subjected to range measurement noise that is proportional to the measured range. Therefore the simulated sensors are as well. To achieve this, we sample a noise percentage n_P from some normal distribution $\mathcal{N}(\mu, \sigma^2)$. The resulting range r given the true range r_t is then

$$r = r_t(1 + n_P) \tag{26}$$

For each measurement ray the noise percentage is sampled independently.

4.3. Pose Estimation Noise Model

Regarding the pose measurement, one cannot simply add white noise to the current pose estimate as this does not capture the integration error, which is common among inertial measurement units (IMU). Therefore we assume a disturbance torque about the two axes that lie in the ground plane. This is equivalent to a case where a slightly shifted ground plane or an unbalanced locomotion system of the robot is present.

Accordingly, a spherical robot that is assumed to be translating exactly along one axis by rotating about one other axis at an angular velocity ω experiences random disturbance torques at each time instant that accelerate the rotation of the sphere about the respective axes. This additional motion is determined by integrating the disturbance torques, which are sampled from some normal distribution $n_\phi, n_\psi \in \mathcal{N}(\mu, \sigma^2)$ each. The pose update after a discrete time step Δt in the simulation is therefore given by:

$$\begin{bmatrix} \varphi \\ \vartheta \\ \psi \\ x \\ y \\ z \end{bmatrix}_{k+1} = \begin{bmatrix} \varphi \\ \vartheta \\ \psi \\ x \\ y \\ z \end{bmatrix}_{k} + \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ R \end{bmatrix} \omega\Delta t + \Delta t \underbrace{\sum_{i=0}^{k} \begin{bmatrix} n_\varphi[i]\Delta t \\ 0 \\ n_\psi[i]\Delta t \\ n_\varphi[i]\Delta t R \\ 0 \\ n_\psi[i]\Delta t R \end{bmatrix}}_{\substack{\text{Difference in} \\ \text{angular velocity}}} \tag{27}$$

where Δt is generally chosen as some small value (e.g. 1 ms)

Assuming intended locomotion along the z-axis and unintended locomotion along the x-axis where R is the sphere radius. No upwards movement is possible since the robot moves on a flat ground, hence the entry is zero.

4.4. Simulated Datasets

Figure 3 shows a noise-free trajectory through a simulated environment and a trajectory with noisy sensors while figure 4 shows a rendering of the simulation sequence. In the noise-free case, the trajectory slightly bends in the direction of disturbance torque sideways while also varying in velocity in the intended direction of travel. In the noisy case, the ideal straight trajectory is assumed. Hence the planes enclosing the room are sensed multiple times. In particular, we see that the trajectory of the robot leads through one of the sensed planes. Also, the further the sensed points are, the noisier they appear, which is consistent with the larger influence of the pose error at higher distances.

5. Experimental Setup

We also tested our algorithm on two real-world datasets that adhere to different motion profiles and laser scanner types with different scanning patterns.

The first dataset is collected by a line scanner, in particular a SICK LMS141 industrial scanner, inside a sphere of a diameter of 20 cm that lies on a floating desk, which is air-pressurized, such that the sphere is hovering. On this floating desk, the sphere can rotate freely about all axis while being fixed in position. Hence, in this experiment the optimization space can be reduced to a rotation. Further, without the motion, the 2D laser cannot obtain a 3D model from the environment, thus requiring registration. The sphere is equipped with a low-cost IMU, i.e., a PhidgetSpatial Precision 3/3/3, to estimate the orientation, a battery pack and a raspberry pie for processing. This IMU provides a precision of 76.3 µg in linear acceleration, $0.02\,^\circ\,\mathrm{s}^{-1}$ in angular velocity about the x and y axis, and $0.013\,^\circ\,\mathrm{s}^{-1}$ in angular velocity about the z axis [40]. Figure 5 shows the experimental setup. The left column of figure 8 shows the pre-registered resulting point cloud. We see that the point cloud is rather noisy, and in particular, the walls are sensed multiple times, hence appearing very blurry.

The second datasets is collected by a 3D scanner, namely the LIVOX MID 100, which is the same sensor the simulation uses. We put the laser scanner into an acrylic glass spherical shell together with three IMUs (same specifications as in the first dataset) and manually roll it with a slow, constant motion in a home indoor environment. Thus, this time nearly all six degrees of freedom are used for optimization. We exclude a translation in the global up- or downwards direction as the sphere rolls on the ground surface, without possibility for up- or downwards motion- figure 6 shows the prototype. The left column of figure 9 shows the pre-registered point cloud, applying only the transformation defined by the recorded poses. Although the robot has been moved carefully, you see that pose error accumulates in such a way that the same wall is sensed multiple times.

6. Results

6.1. Simulated Results

The plane-based registration is applied to the simulated dataset with noisy pose and range measurements (cf. figure 3) without further processing. Assuming this represents a coarsely pre-registered 3D point cloud, the distances to the ground truth were evaluated before and after

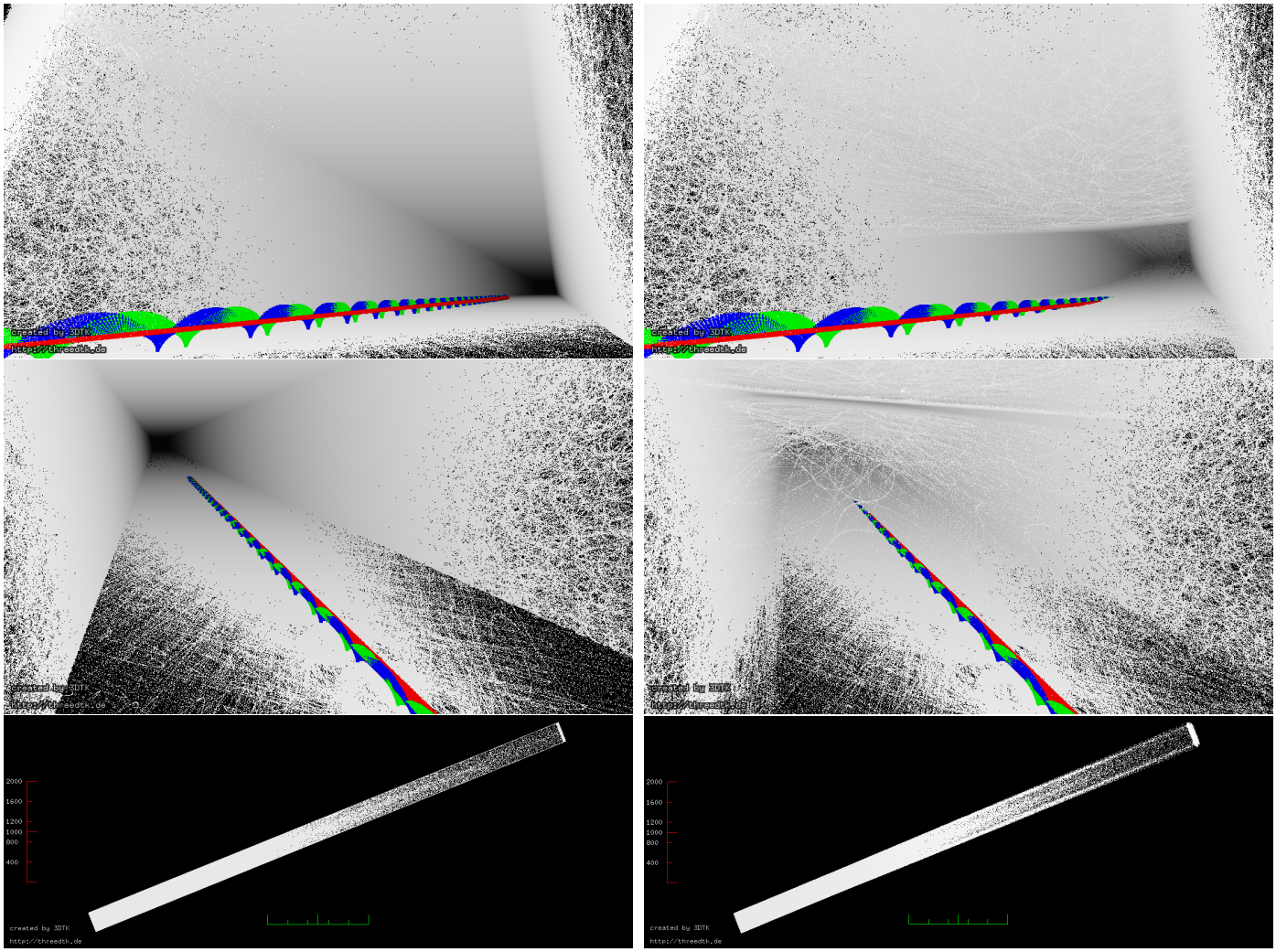

Figure 3: Two simulated datasets of a long hallway of size $4\,\mathrm{m} \times 3\,\mathrm{m} \times 100\,\mathrm{m}$. Ideal noise free (left), noisy pose ($\mathcal{N}_\varphi = \mathcal{N}_\psi(\mu = 0.0001, \sigma = 0.00001)$) and noisy range measurements ($\mathcal{N}_r(\mu = 0, \sigma = 0.001)$) (right). The normal distributions are define in such a way, that in the resulting dataset plane detection is still possible.

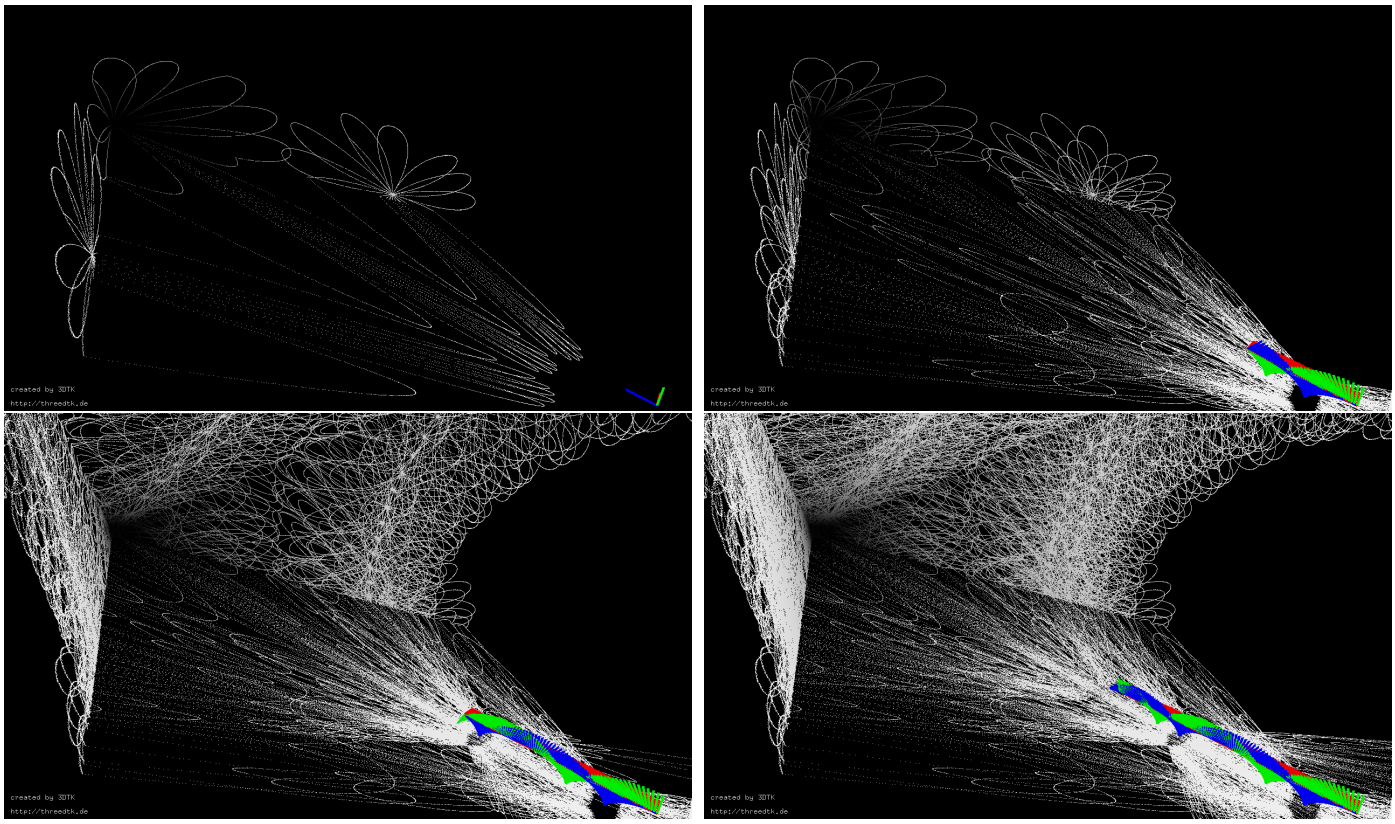

Figure 4: Top: Initial sequence of the simulation of the ideal dataset showing the simulated motion of the robot.

Figure 5: Data acquisition using floating sphere. Top: A sequence of orientations the sphere assumes during data acquisition. Bottom: Hardware setup of the sphere.

	P90	P95	P98
Uncorrected	372.1 cm	553.4 cm	827.9 cm
Corrected	35.9 cm	64.1 cm	122.8 cm

Table 1: Comparison of point-distances in the uncorrected and corrected simulated dataset.

the plane-based registration. Figure 7 shows the different point-to-point distances.

Before the registration, the corridor is only represented acceptably in the front part. The further into the corridor, i.e., the longer the robot accumulates errors, the more imprecise the data becomes. Finally, we see that many points exceed the threshold of 1 m and thus being mapped to the same color value. After registration, we see that, qualitatively, the ideal corridor was nearly restored from the noisy data. In particular a very large portion of points (90%) have distances of less than 35.9 cm. Table 1 shows the comparison of further percentiles of both datasets.

Further, the square and straight shape of the corridor is restored well, and especially the large amount of points with an errors of greater than 1 m is removed. Any such errors tend to occur at the back and the front of the corridor where the measured range is the largest hence has the largest contribution of the range error.

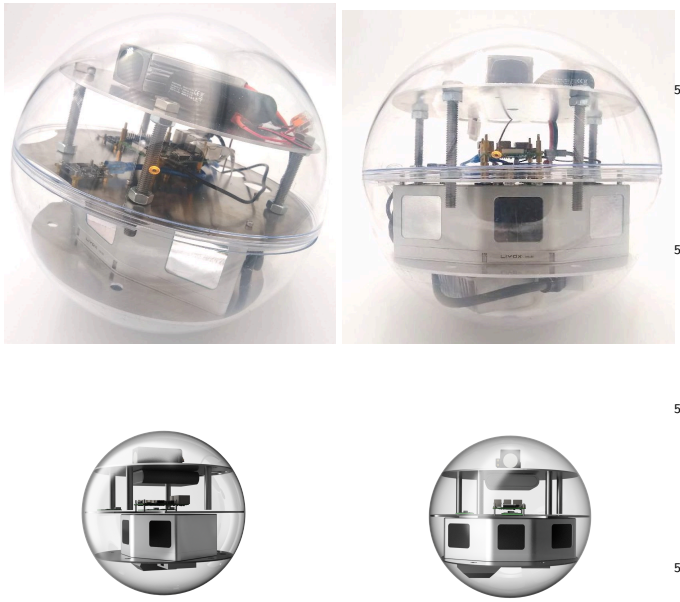

Figure 6: Top: Prototype used for data acquisition with the rolling sphere. The main payload is the Livox Mid-100 laser scanner. For pose-estimation, three IMUs of the manufacturer Phidget are placed inside and a Raspberry Pi 4 for the calculations. On the top are two batteries, and on the bottom one voltage stabilizer and the breakout box of the laser scanner. Below: Rendering of the simulated robot, a Livox laser scanner insider a spherical shell.

6.2. Floating Sphere Results

Figure 8 shows the results obtained before and after employing the plane based registration on the dataset acquired by the floating sphere. The pre-registration is obtained by determining the orientation of the sphere via Madgwick filtered [41] IMU measurements. To increase the number of possible point-to-plane correspondences we combine twenty temporally successive scans into one meta-scan, which is then globally registered. We always use the scan at the median index as a reference coordinate system. This does not only speed up convergence due to the proportional effect on the error function but also decreases the risk of transforming a single line scan incorrectly. Transformations like these happen in particular for a small collection of points as outliers have more influence.

After the registration, the walls of the room are significantly more prominent in the point cloud. In particular, the noise in the top left corner of the top view (cf. figure 8) has been visibly reduced. Further, the deviation of points around the walls is notably smaller as the points are moved on the plane.

6.3. Rolling Sphere Results

Figure 9 shows the results obtained before and after employing the presented algorithm on the dataset, acquired by the prototype in figure 6. We again combine twenty temporally successive scans into one meta-scan, using the scan at the median index as a reference coordinate system, which is then globally registered. We used the first seven of these meta-scans, corresponding to the first full rotation of the sphere, to extract global planes. All subsequent meta-scans are then matched against the initial global plane model. We use both optimizations from section 3.5, introducing a global up/down lock for gradient descent and applying sequential iteration to eliminate accumulating pose error.

After the registration, the walls of the environment are no longer ambiguous, since all the subsequent scans are matched with the initial plane representation. When considering the resulting path, it looks distorted. This is because for some scans, especially empty ones where the sensor points to the ground, no planar features got matched with the global plane model, thus their pose is not optimized. Improving the path quality is an objective for future works.

7. Conclusion

In this paper, we proposed an approach to mobile mapping using a spherical robot. Given that a spherical robot inherently rotates for locomotion, we use this movement to also rotate a laser scanner, that consequently measures the robots entire environment. We propose to post-process the acquire data using a novel registration method for man made environments that exploits the structure of those environments. In human-made environments straight planes are abundantly available, hence we employ point-to-plane correspondences to improve a pre-registered 3D point cloud. We have evaluated the procedure on a simulated dataset and on two experimentally acquired datasets with different laser scanners and motion profiles. In this evaluation, we have shown that the procedure improves all datasets and yields maps that better resemble human-made, structures. In particular, the qualitative structure of the environments is reconstructed well. In the resulting maps, the parallel walls are clearly improved. The simulation results show that the algorithm has the potential to improve the map quality based on point-distances by approximately a factor of ten.

Right now, not all steps in the procedure are autonomous, in particular the parameter tuning. In the future, one goal is to increase the autonomy of the system. One approach is to introduce soft-locks for the optimization dimensions. I.e., instead of locking some dimensions entirely from being used for optimization, they are weighted based on the dynamics of the system that encode which noise source is more likely.

The biggest issue with the presented algorithm currently is that the global plane model is established only once at initialization, but never updated afterwards. Utilizing the local planar clustering (LPC) of the individual line scans for updating the global plane model step by step is currently the primary objective of future studies. One

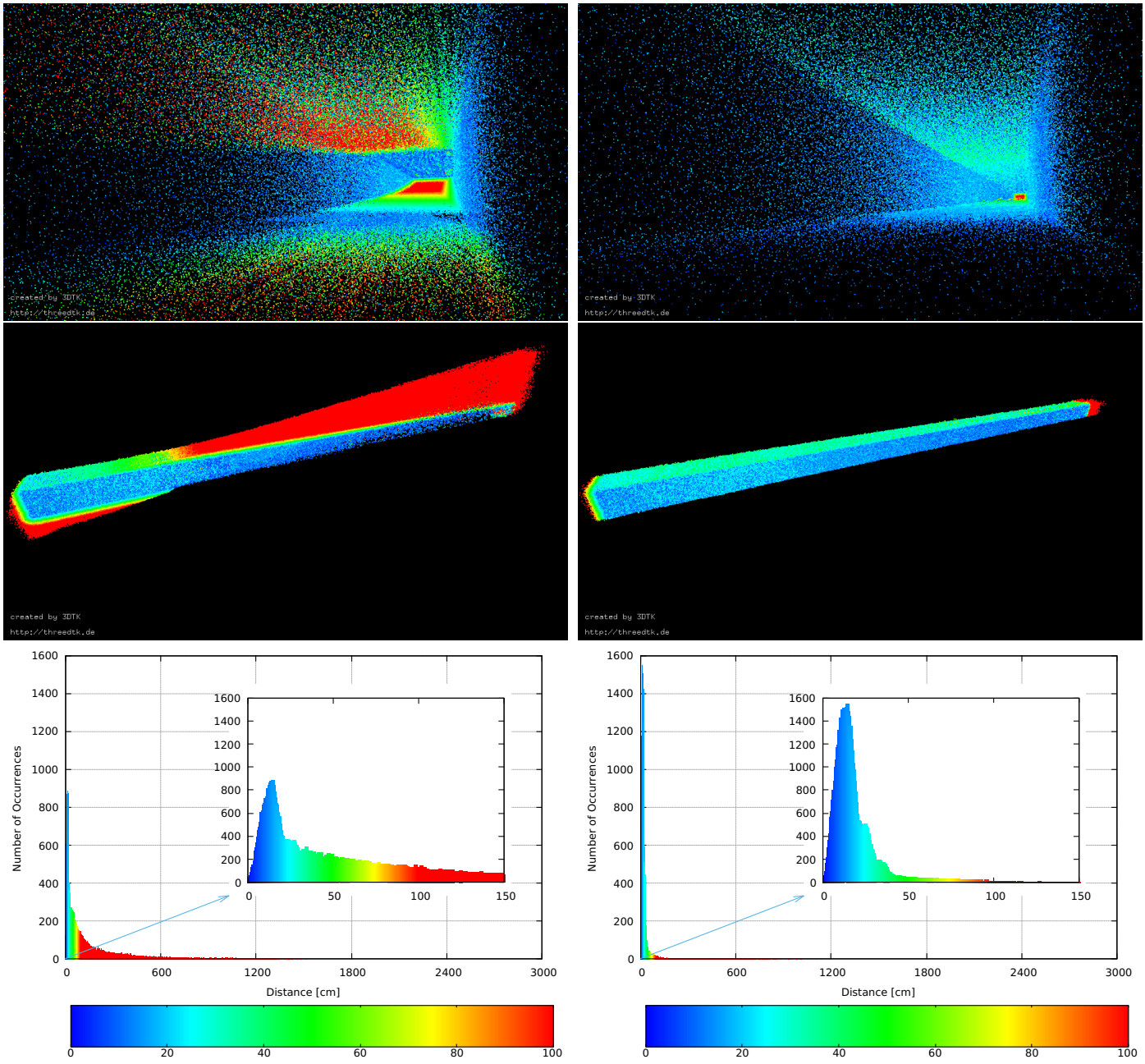

Figure 7: Evaluation of point distances before (left) and after (right) the plane-based registration on a simulated dataset. Lateral images always have the same orientation. A maximal distance of 30 m is set, such that all points that display a higher distance value are excluded from the analysis. Both point-clouds were reduced before evaluating point-to-point distances to the ground-truth. Further, the color-space maps all values with a distance greater than 1 m to the same color. The top two columns show a heat map of distances, while the bottom shows the corresponding histogram. The color mapping is equivalent in both. An animation of the matching process is given at `https://youtu.be/7igQdEeCsYk`.

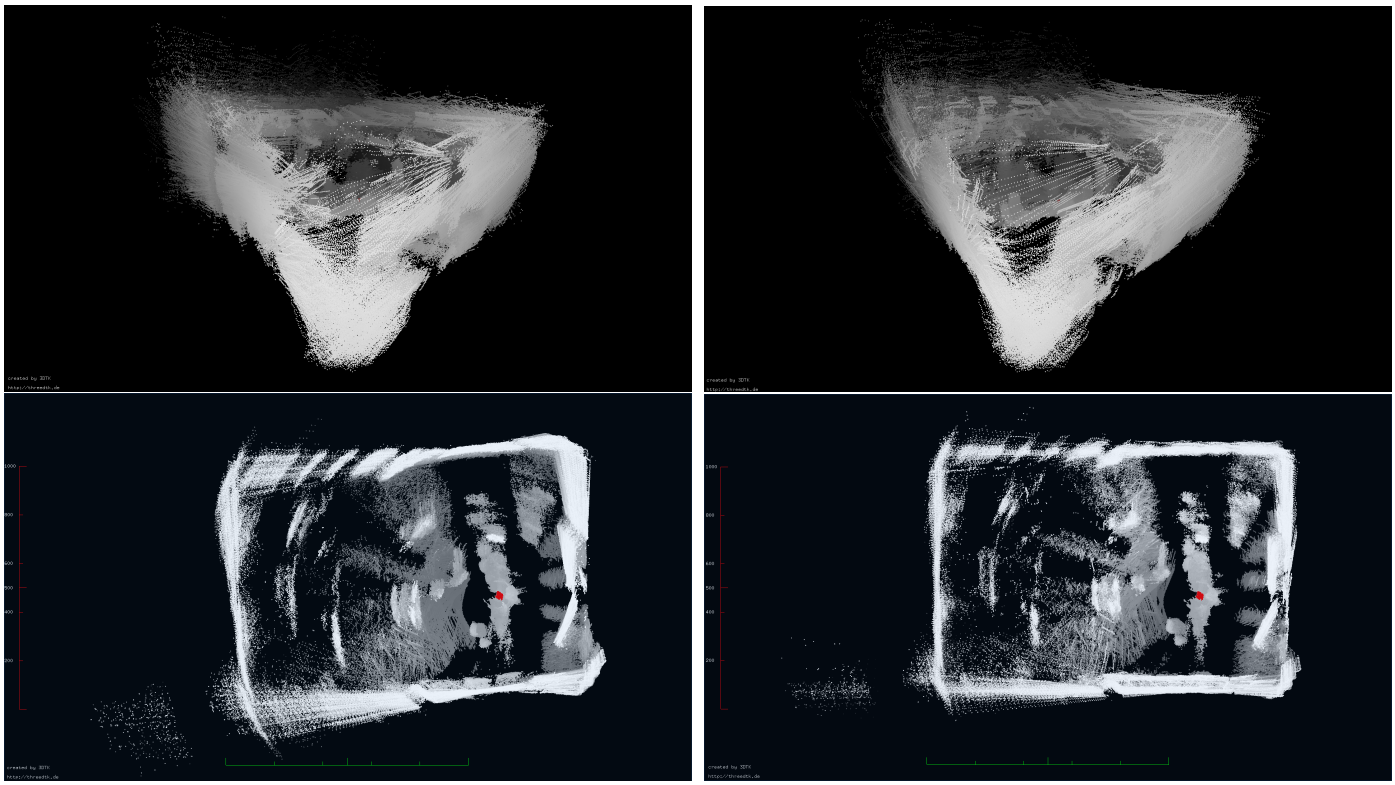

Figure 8: The point cloud acquired by the floating sphere before (left) and after (right) applying the plane based registration. View from the interior (top) and a birds-eye view (bottom). Parameters used for optimization: $\varepsilon_H = 50$, $\varepsilon_P = 200$, $\varepsilon_\alpha = \frac{\pi}{4}$, $K = 20$ for AKNN, $d_{growth} = 50$, $N_{cmin} = 200$, $\alpha_0 = [0.01, 0.01, 0.01, 0, 0, 0]^\tau$, $i = 8$, $k = 100$. An animation of the registration process is given at `https://youtu.be/Qa2Qi8z0KEk` .

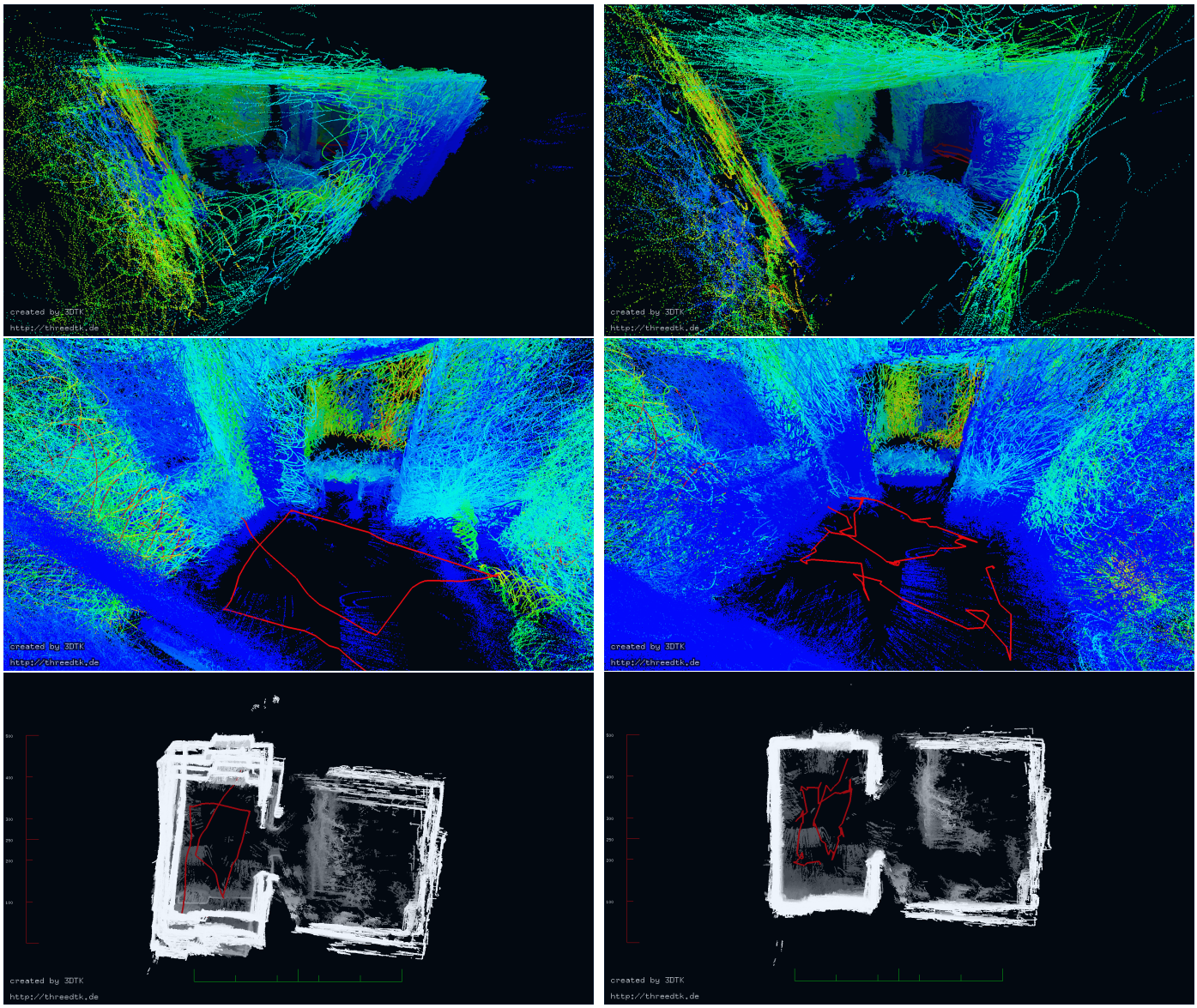

Figure 9: The point cloud acquired by the rolling sphere before (left) and after (right) applying the plane based registration. View from the interior (top) and a birds-eye view (bottom). Parameters used for optimization: $\varepsilon_H = 200$, $\varepsilon_P = 300$, $\varepsilon_\alpha = \frac{\pi}{4}$, $K = 200$ for AKNN, $d_{growth} = 50$, $N_{c_min} = 800$, $\alpha_0 = [0.001, 0.001, 0.001, 1, 0, 1]^\tau$, $i = 1500$, $k = 1$. An animation of the registration process is given at `https://youtu.be/2J8y7MN10JM`.

idea is to calculate the convex hulls of the LPC, and combine them with the global planes sequentially. Therefore, we expect to increase robustness and autonomy of the registration procedure.

Furthermore, more experimental evaluation is required, hence we will test our rolling and scanning spheres on a rail system to achieve repeatable trajectories.

Acknowledgement

The authors thank the Cylon team (Ignacio Dorado Llamas, Timothy Randolph, Camilo Andres Reyes Mantilla) for designing and building the prototype and Timo Burger for acquiring the floating table data set. Special thanks to Dieter Ziegler and Sergio Montenegro for supporting our work. We acknowledge funding from the ESA Contract No. 4000130925/20/NL/GLC for the study "DAEDALUS – Descent And Exploration in Deep Autonomy of Lava Underground Structures" within the Open Space Innovation Platform (OSIP) lunar caves-system and the Elite Network Bavaria (ENB) for providing funds for the academic program "Satellite Technology".

References

[1] M. Bosse, R. Zlot, P. Flick, Zebedee: Design of a spring-mounted 3-d range sensor with application to mobile mapping, IEEE Transactions on Robotics 28 (5) (2012) 1104–1119. doi:10.1109/TRO.2012.2200990.

[2] V. V. Lehtola, J.-P. Virtanen, M. T. Vaaja, H. Hyyppä, A. Nüchter, Localization of a Mobile Laser Scanner via Dimensional Reduction, ISPRS Journal of Photogrammetry and Remote Sensing (JPRS) 121 (2016) 48–59. doi:10.1016/j.isprsjprs.2016.09.004.
URL https://robotik.informatik.uni-wuerzburg.de/telematics/download/jprs2016.pdf

[3] D. Borrmann, S. Jörissen, A. Nuchter, RADLER – A RADial LasER scanning device, in: Proceedings of the International Symposium on Experimental Research, Buenos Aires, Argentina, 2020, pp. 655–664. doi:10.1007/978-3-030-33950-0_56.

[4] H. A. Lauterbach, D. Borrmann, R. Heß, D. Eck, K. Schilling, A. Nüchter, Evaluation of a backpack-mounted 3d mobile scanning system, Remote Sensing 7 (10) (2015) 13753–13781. doi:10.3390/rs71013753.
URL https://www.mdpi.com/2072-4292/7/10/13753

[5] Leica, Leica pegasus:backpack, www.leica-geosystems.com/de/Leica-PegasusBackpack_106730.htm (May 2015).

[6] Boston Dynamics, Spot robot, https://www.bostondynamics.com/spot (2021).

[7] P. Fankhauser, M. Bloesch, M. Hutter, Probabilistic terrain mapping for mobile robots with uncertain localization, IEEE Robotics and Automation Letters 3 (4) (2018) 3019–3026. doi:10.1109/LRA.2018.2849506.

[8] A. P. Rossi, F. Maurelli, V. Unnithan, H. Dreger, K. Mathewos, N. Pradhan, D.-A. Corbeanu, R. Pozzobon, M. Massironi, S. Ferrari, C. Pernechele, L. Paoletti, E. Simioni, P. Maurizio, T. Santagata, D. Borrmann, A. Nüchter, A. Bredenbeck, J. Zevering, F. Arzberger, C. A. R. Mantilla, Daedalus - descent and exploration in deep autonomy of lava underground structures, Tech. Rep. 21, Institut für Informatik (2021). doi:10.25972/OPUS-22791.

[9] J. Guo, X. Chen, S. Guo, J. Xu, Study on map construction of spherical robot based on statistical filtering, in: 2020 IEEE International Conference on Mechatronics and Automation (ICMA), 2020, pp. 1269–1274. doi:10.1109/ICMA49215.2020.9233654.

[10] A. Nüchter, K. Lingemann, J. Hertzberg, H. Surmann, 6D SLAM – 3D Mapping Outdoor Environments, Journal of Field Robotics (JFR), Special Issue on Quantitative Performance Evaluation of Robotic and Intelligent Systems 24 (8–9) (2007) 699–722. doi:10.1002/rob.20209.
URL https://robotik.informatik.uni-wuerzburg.de/telematics/download/jfr2007.pdf

[11] D. Borrmann, J. Elseberg, K. Lingemann, A. Nüchter, J. Hertzberg, Globally consistent 3d mapping with scan matching, Journal Robotics and Autonomous Systems (JRAS) 56 (2) (2008) 130–142. doi:10.1016/j.robot.2007.07.002.
URL https://robotik.informatik.uni-wuerzburg.de/telematics/download/ras2007.pdf

[12] J. Zhang, S. Singh, Loam: Lidar odometry and mapping in real-time, in: Robotics: Science and Systems Conference (RSS), 2014. doi:10.15607/RSS.2014.X.007.

[13] D. Droeschel, S. Behnke, Efficient continuous-time slam for 3d lidar-based online mapping, 2018 IEEE International Conference on Robotics and Automation (ICRA)doi:10.1109/icra.2018.8461000.
URL http://dx.doi.org/10.1109/ICRA.2018.8461000

[14] NavVis, The navvis vlx mobile mapping system, https://www.navvis.com, accessed on 05.07.2021.

[15] NavVis, Testing navvis vlx side-by-side with a terrestrial laser scanner, https://www.navvis.com/blog/testing-navvis-vlx-side-by-side-with-a-terrestrial-laser-scanne accessed on 20.08.2021.

[16] P. J. Besl, N. D. McKay, A method for registration of 3-d shapes, IEEE Transactions on Pattern Analysis and Machine Intelligence 14 (2) (1992) 239–256. doi:10.1109/34.121791.

[17] K. Pathak, A. Birk, N. Vaskevicius, J. Poppinga, Fast registration based on noisy planeswith unknown correspondences for 3d mapping, IEEE Transactions on Robotics 26 (3) (2010) 424–441. doi:10.1109/TRO.2010.2042989.

[18] W. Förstner, K. Khoshelham, Efficient and accurate registration of point clouds with plane to plane correspondences, in: 2017 IEEE International Conference on Computer Vision Workshops (ICCVW), 2017, pp. 2165–2173. doi:10.1109/ICCVW.2017.253.

[19] K. Favre, M. Pressigout, E. Marchand, L. Morin, A Plane-based Approach for Indoor Point Clouds Registration, in: ICPR 2020 - 25th International Conference on Pattern Recognition, Milan (Virtual), Italy, 2021.
URL https://hal.archives-ouvertes.fr/hal-03108891

[20] J. Lin, F. Zhang, Loam livox: A fast, robust, high-precision LiDAR odometry and mapping package for LiDARs of small FoV, in: proceedings of the IEEE International Conference on Robotics and Automation (ICRA), 2020, pp. 3126–3131. doi:10.1109/ICRA40945.2020.9197440.

[21] L. Zhou, S. Wang, M. Kaess, π-LSAM: LiDAR smoothing and mapping with planes, in: Proceedings of IEEE International Conference on Robotics and Automation (ICRA '21), Xi'an, China, 2021, to appear.

[22] J. Jung, S. Yoon, S. Ju, J. Heo, Development of kinematic 3D laser scanning system for indoor mapping and as-built BIM using constrained SLAM, Sensors 15 (10) (2015) 26430–26456. doi:10.3390/s151026430.

[23] W. S. Grant, R. C. Voorhies, L. Itti, Efficient Velodyne SLAM with point and plane features, Autonomous Robots 43 (2019) 1207–1224. doi:10.1007/s10514-018-9794-6.

[24] P. Geneva, K. Eckenhoff, Y. Yang, G. Huang, LIPS: LiDAR-Inertial 3D Plane SLAM, in: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '18), 2018, pp. 123–130. doi:10.1109/IROS.2018.8594463.

[25] X. Wei, J. Lv, J. Sun, S. Pu, Ground-SLAM: Ground Constrained LiDAR SLAM for Structured Multi-Floor Environments (2021). arXiv:2103.03713.

[26] J. Elseberg, D. Borrmann, K. Lingemann, A. Nüchter, Non-

rigid registration and rectification of 3d laser scans, in: 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2010, pp. 1546–1552. doi:10.1109/IROS.2010.5652278.

[27] T. Stoyanov, A. J. Lilienthal, Maximum likelihood point cloud acquisition from a mobile platform, in: 2009 International Conference on Advanced Robotics (ICAR), 2009, pp. 1–6.

[28] M. Bosse, R. Zlot, Continuous 3d scan-matching with a spinning 2d laser, in: 2009 IEEE International Conference on Robotics and Automation, 2009, pp. 4312–4319. doi:10.1109/ROBOT.2009.5152851.

[29] J. Elseberg, D. Borrmann, A. Nüchter, Algorithmic solutions for computing accurate maximum likelihood 3D point clouds from mobile laser scanning platforms, Remote Sensing 5 (11) (2013) 5871–5906. doi:10.3390/rs5115871.
URL https://robotik.informatik.uni-wuerzburg.de/telematics/download/remotesensing2013.pdf

[30] D. Borrmann, J. Elseberg, A. Nüchter, K. Lingemann, The 3D Hough Transform for Plane Detection in Point Clouds – A Review and A new Accumulator Design, Journal of 3D Research 2 (2) (2011) 1–13. doi:10.1007/3DRes.02(2011)3.
URL https://robotik.informatik.uni-wuerzburg.de/telematics/download/3dresearch2011.pdf

[31] R. Honti, J. Erdélyi, A. Kopacik, Plane segmentation from point clouds, Pollack Periodica 13 (2018) 159–171. doi:10.1556/606.2018.13.2.16.

[32] B. Gaspers, J. Stückler, J. Welle, D. Schulz, S. Behnke, Efficient multi-resolution plane segmentation of 3d point clouds, 2011, pp. 145–156. doi:10.1007/978-3-642-25489-5_15.

[33] F. Engelmann, T. Kontogianni, J. Schult, B. Leibe, Know what your neighbors do: 3d semantic segmentation of point clouds, CoRR abs/1810.01151. arXiv:1810.01151.
URL http://arxiv.org/abs/1810.01151

[34] D. Sunday, Practical Geometry Algorithms, Independently published, 2021.

[35] J. S. Beis, D. G. Lowe, Shape indexing using approximate nearest-neighbour search in high-dimensional spaces, CVPR '97, IEEE Computer Society, 1997. doi:10.1109/CVPR.1997.609451.

[36] J. Diebel, Representing attitude: Euler angles, unit quaternions, and rotation vectors, Matrix 58 (15-16) (2006) 1–35.

[37] M. D. Zeiler, Adadelta: An adaptive learning rate method (2012). arXiv:1212.5701.

[38] Livox, Livox Mid40 and Mid100, https://www.livoxtech.com/mid-40-and-mid-100 (2021).

[39] ThorLabs, Application note risley prism scanner, https://www.thorlabs.com/images/tabimages/Risley_Prism_Scanner_App_Note.pdf.

[40] Phidget, PhidgetSpatial Precision 3/3/3, https://www.phidgets.com/?&prodid=32 (2021).

[41] S. Madgwick, An efficient orientation filter for inertial and inertial/magnetic sensor arrays, Report x-io and University of Bristol (UK) 25 (2010) 113–118.