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

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. Examples for this are human operated systems such as Zebedee (Bosse et al., 2012), a small Hokuyo 2D scanner on a spring, that is carried through the environment, VILMA (Lehtola et al., 2016), a rolling FARO scanner operating in profiler mode, RADLER (Borrmann et al., 2020), a SICK 2D laser scanner mounted to a unicycle, or a backpack mounted “personal laser scanning system” as in (Lauterbach et al., 2015) or (Leica, 1067). Recently more and more autonomous systems gained accessibility. Examples for this are human operated systems such as Zebedee (Bosse et al., 2012), a small Hokuyo 2D scanner on a spring, that is carried through the environment, VILMA (Lehtola et al., 2016), a rolling FARO scanner operating in profiler mode, RADLER (Borrmann et al., 2020), a SICK 2D laser scanner mounted to a unicycle, or a backpack mounted “personal laser scanning system” as in (Lauterbach et al., 2015) or (Leica, 1067). Recently more and more autonomous systems gained maturity. A stunning example is Boston Dynamics’ quadruped “Spot” that autonomously navigates and maps human environments (Boston Dynamics and Spot rob, 2021). Also, the mobile mapping approaches implemented on the ANYmal platform such as (Fankhauser et al., 2018) 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

* 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).
* Corresponding author.
** Corresponding author.
E-mail addresses: anton.bredenbeck@stud-mail.uni-wuerzburg.de (A. Bredenbeck), borrmann@informatik.uni-wuerzburg.de (D. Borrmann).

https://doi.org/10.1016/j.isprsjrhp.2021.100004
Received 14 May 2021; Received in revised form 30 August 2021; Accepted 30 August 2021
Available online 6 September 2021
space applications. In the DAEDALUS study (Rossi et al., 2021), 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 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 in 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 (Guo et al., 2020), 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 (Nüchter et al., 2007) 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 (Borrmann et al., 2008). 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 (Zhang and Singh, 2014). 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 Droschel 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 (Droschel and Behnke, 2018). Another intriguing example is the NavVis VLX system (NavVis, 2021a). 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 3 mm difference the measurements of a ground truth terrestrial laser scan (NavVis, 2021b). 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 (Besl and McKay, 1992) 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. (2010) 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.

Forster et al. use the planarity of human-made environment successfully in (Forstner and Khoshelham, 2017). 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. (2021) 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) (Zhang and Singh, 2014) 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 (Lin and Zhang, 2020). 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 (Zhou et al., 2021). 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 (Jung et al., 2015; Grant et al., 2019; Geneva et al., 2018; Zhou et al., 2021; Wei et al., 2021). 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 (Jung et al., 2015), while Grant used a single Velodyne HDL-32E sensor with 32 rotating laser/receiver pairs mounted on a backpack and walked through different environments (Grant et al., 2019). The LIPS system (Geneva et al., 2018) employs 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 (Elseberg et al., 2010) 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 inertial 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 (Stoyanov and Lilienthal, 2009; Bosse and Zlot, 2009) or use a small number of iterations of continuous-time SLAM approaches such as the one presented in (Elseberg et al., 2013). 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 extraction. 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 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 (Borrmann et al., 2011) 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 \( \rho_{P_k} \) is computed via \( n_{P_k} \cdot p_i / C_1 \), where \( p_i = [x_i, y_i, z_i]^T \) is an arbitrary point on the plane and \( n_{P_k} = [n_{xP_k}, n_{yP_k}, n_{zP_k}]^T \) 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 \( a_{P_k} \).

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 (Honti et al., 2018), (Gaspers et al., 2011) but also deep learning approaches (Engelmann et al., 2018) 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 $T(p_i)$ to infinitely extending, $k$-th plane in 3D vector space

$$D_i^k = n_k \cdot [(T(p_i) - a_k),$$

with normal $n_k$ and supporting point $a_k$ of the $k$-th plane. The distance $D_i^k$ 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 $\tilde{T}(p_i)$ onto the $k$-th plane, which is required later, is easily calculated by shifting the point against normal direction:

$$\tilde{T}(p_i) = T(p_i) - D_i^k n_k \quad (2)$$

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. $\tilde{T}(p_i)$ is found from eq. (2) (green point in Fig. 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 Fig. 1, the blue planes normal vector has its major 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. (Sunday, 2021) 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 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 Pk be made up of line segments smj. The line segment smj consists of the points pm and pj. The line segment is parameterized by

\[ s_{mj}(t) = pm + t \cdot (pj - pm), \]  

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.

\[ d_{mj}(t) = \| s_{mj}(t) - p_i \| \]  

(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

\[ \frac{\partial}{\partial t} d_{mj}(t)^2 = 0, \]  

(5)

resulting in

\[ t_0 = \frac{(pj - pm) \cdot (pj - pm)}{(pj - pm)^2} \]  

(6)

Note the possibility of \( t_0 \not\in [0, 1] \). 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

\[ i = \min(\max(t_0, 0), 1), \]  

(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.

### 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 (Beis and Lowe, 1997). 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 than a threshold \( d_{growth} \) and additionally has a similar normal. We define similarity of two normals via the angle between them, which is:

\[ \alpha(n_1, n_2) = \arccos(n_1 \cdot n_2). \]  

(8)

However, the smallest angle between two normal vectors is

\[ \hat{\alpha}(n_1, n_2) = \begin{cases} 2\pi - \alpha(n_1, n_2) & \text{if } \alpha(n_1, n_2) > \pi/2 \\ \alpha(n_1, n_2) - \pi & \text{if } \alpha(n_1, n_2) > \pi \\ \pi - \alpha(n_1, n_2) & \text{if } \alpha(n_1, n_2) > \pi/2 \end{cases}, \]  

(9)

because the opposing normals \(-n_1\) and \(-n_2\) always have to be...
considered, since they correspond to the same plane. We say that two normals are similar if the angle $\alpha$ from eq. (9) is smaller than a threshold $\epsilon_\alpha$.

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 algorithm does this by putting a minimum threshold $N_{c,min}$ on the cluster size. Fig. 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 $\epsilon_H$ and a PPD smaller than some additional threshold $\epsilon_P$, 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 $\epsilon_\alpha$.

3.4. Optimisation

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 $T(p_i)$ of each point

| Table 1 | Comparison of point-distances in the uncorrected and corrected simulated dataset. |
|---------|---------------------------------|
|         | P90      | P95      | P98      |
| Uncorrected | 372.1 cm | 553.4 cm | 827.9 cm |
| Corrected   | 35.9 cm  | 64.1 cm  | 122.8 cm |

Fig. 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: $\epsilon_H = 200$, $\epsilon_P = 300$, $\epsilon_\alpha = \frac{\pi}{4}$, $K = 200$ for AKNN, $d_{growth} = 50$, $N_{c,min} = 800$, $\alpha_0 = [0.001,0.001,0.001,1,0,1]^T$, $\ell = 1500$, $k = 1$. An animation of the registration process is given at https://youtu.be/Mkil0vLk8f8.
\[ p_i \] with respect to a 6 DoF motion is described in homogeneous coordinates using the roll-pitch-yaw (\( \phi - \theta - \psi \)) Tait-Bryan angles as in (Diebel, 2006). Transforming the result back from homogeneous coordinates and using \( C_{\phi} \) and \( S_{\phi} \) to denote the cosine and sine of the angle in the subscript, and \( t_o \) denoting the translation along the axis in the subscript yields:

\[
T(p_i) = \begin{bmatrix}
  x_i C_{\phi} C_{\psi} - y_i C_{\phi} S_{\psi} + z_i S_{\phi} + t_o \\
  x_i (C_{\phi} S_{\psi} + S_{\phi} S_{\psi}) + y_i (C_{\phi} C_{\psi} S_{\phi} - S_{\phi} C_{\psi}) - z_i C_{\phi} S_{\psi} + t_o \\
  x_i (S_{\phi} S_{\psi} - C_{\phi} C_{\psi}) + y_i (C_{\phi} C_{\psi} S_{\phi} + S_{\phi} C_{\psi}) + z_i C_{\phi} C_{\psi} + t_o
\end{bmatrix}
\] (10)

From this we define the function \( D(\phi, \theta, \psi, t_x, t_y, t_z, p_i) \) that computes the distance of a point \( p_i \) to its corresponding plane \( P_i \). Omitting the arguments of the function for simplicity:

\[
D = T(p_i) \cdot n_P
\]

\[
= n_{P_x}(x_i C_{\phi} C_{\psi} - y_i C_{\phi} S_{\psi} + z_i S_{\phi} + t_o) \\
+ n_{P_y}(x_i (C_{\phi} S_{\psi} + S_{\phi} S_{\psi}) + y_i (C_{\phi} C_{\psi} S_{\phi} - S_{\phi} C_{\psi}) - z_i C_{\phi} S_{\psi} + t_o) \\
+ n_{P_z}(x_i (S_{\phi} S_{\psi} - C_{\phi} C_{\psi}) + y_i (C_{\phi} C_{\psi} S_{\phi} + S_{\phi} C_{\psi}) + z_i C_{\phi} C_{\psi} + t_o)
\] (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:

\[
E = \sum_{i=0}^{num} \|D(\phi, \theta, \psi, t_x, t_y, t_z, p_i)\|^2
\]

Its gradient follows then immediately:

\[
\nabla E = \sum_{i=0}^{num} \sum_{p_i \in \Omega_i} \nabla D_k(p_i) \nabla D_{k'}(p_i) n^2_{P_x} n^2_{P_y} n^2_{P_z}
\]

Where

\[
\nabla D_k(p_i) = n_{P_x}(x_i \{-S_{\phi} S_{\psi} + C_{\phi} C_{\psi}\} + y_i \{-S_{\phi} C_{\psi} + C_{\phi} S_{\psi}\} + z_i S_{\phi} - C_{\phi} S_{\psi}) \\
+ n_{P_y}(x_i \{C_{\phi} S_{\psi} + S_{\phi} S_{\psi}\} + y_i \{C_{\phi} C_{\psi} S_{\phi} - S_{\phi} C_{\psi}\} - z_i C_{\phi} S_{\psi} + C_{\phi} S_{\psi}) \\
+ n_{P_z}(x_i \{S_{\phi} S_{\psi} - C_{\phi} C_{\psi}\} + y_i \{C_{\phi} C_{\psi} S_{\phi} + S_{\phi} C_{\psi}\} + z_i C_{\phi} C_{\psi} - S_{\phi} C_{\psi})
\]

and

\[
\Pi = \begin{bmatrix}
  \phi \\
  \theta \\
  \psi \\
  t_x \\
  t_y \\
  t_z
\end{bmatrix}
\]

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

\[
\Pi_{i+1} = \Pi_i - \alpha \nabla E
\] (18)

where \( \alpha \) 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 SGD is the following: Typically, changes in orientation, i.e., the first three elements of the gradient vector \( \nabla E, \nabla E, \) and \( \nabla E \), 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 \( \alpha \) applied on orientation has to be much smaller than the \( \alpha \) applied on the position. It becomes obvious that \( \alpha \) needs to be extended into vector form, \( \alpha \), therefore weighting each dimension differently.

Another consideration to speed up SDG is to adaptively recalculate \( \alpha \) for each iteration. We employ and modify ADADELTA as a technique to do so, which is described in detail in (Zeiler, 2012). The main idea is the following: It extends the SDG algorithm by two terms. First, an exponentially decaying average of past gradients \( G_k \), which is recursively defined as

\[
G_{k+1} = \zeta G_k + (1 - \zeta) \nabla E^2
\]

and second, an exponentially decaying average of past changes \( X_k \), which is defined as

\[
X_{k+1} = \alpha X_k + (1 - \zeta) \alpha \nabla E^2
\]

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

\[
RMS[G]_k = \sqrt{G_k + \epsilon_{\text{num}}}
\]

and

\[
RMS[X]_k = \sqrt{X_k + \epsilon_{\text{num}}}
\]

where \( \epsilon_{\text{num}} > 0 \) 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 \( \alpha \), which is as follows:

\[
\alpha_i = \frac{RMS[X]_{i-1}}{RMS[G]_i}
\] (23)

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

\[
\alpha_i = \alpha_0 \cdot \frac{RMS[X]_{i-1}}{RMS[G]_i}
\] (24)

where \( \alpha_0 \) holds the scaling factors for each dimension.

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

\[
\Pi_{i+1} = \Pi_i - \alpha \frac{RMS[X]_{i-1}}{RMS[G]_i} \cdot \nabla E
\] (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 optimisations 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 \( q_i \) 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 processed one after another. This is especially useful for mobile systems, on a plane the position along the axis perpendicular to the plane is constant. Further, for mobile systems, the 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.

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 (Livox, 2021) 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 scans 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 (Livox, 2021). 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 deflect on a plane. This is equivalent to a case where a slightly shifted ground plane or unbalanced locomotion system of the robot is present.

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 \( N(\mu, \sigma^2) \). The resulting range \( r \) given the true range \( r_t \) is then

\[
r = r_t(1 + n_p)
\]

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 \( \omega \) 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 \( n_o, n_p \in N(\mu, \sigma^2) \) each. The pose update after a discrete time step \( \Delta t \) in the simulation is therefore given by:

\[
\begin{bmatrix}
\phi \\
\theta \\
\psi \\
x \\
y \\
z
\end{bmatrix}_{t+1} = \begin{bmatrix}
\phi_t \\
\theta_t \\
\psi_t \\
x_t \\
y_t \\
z_t
\end{bmatrix} + \begin{bmatrix}
1 \\
0 \\
0 \\
0 \\
0 \\
0
\end{bmatrix} \omega \Delta t + \Delta t \sum_{i=0}^{n_p} \begin{bmatrix}
R_{n_o} \| \Delta t \\
R_{n_p} \| \Delta R \\
0 \\
0 \\
\| \Delta R
\end{bmatrix}
\]

(27)

where \( \Delta t \) 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

Fig. 3 shows a noise-free trajectory through a simulated environment and a trajectory with noisy sensors while Fig. 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-pressureurized, such that the sphere is hovering. On this floating desk, the sphere can rotate freely about all axes 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 µs in linear acceleration, 0.02° s⁻¹ in angular velocity about the \( x \) and \( y \) axis, and 0.013° s⁻¹ in angular velocity about the \( z \) axis (Phidget, 2021).

Fig. 5 shows the experimental setup. The left column of Fig. 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.

Fig. 6 shows the prototype. The left column of Fig. 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. Fig. 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 the plane-based registration. Fig. 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.

6.2. Floating sphere results

Fig. 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 (Madgwick, 2010) 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. Fig. 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

Fig. 9 shows the results obtained before and after employing the presented algorithm on the dataset, acquired by the prototype in Fig. 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 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.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

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

Beis, J.S., Lowe, D.G., 1997. Shape Indexing Using Approximate Nearest-Neighbour Search in High-Dimensional Spaces, CVPR ’97. IEEE Computer Society. https://doi.org/10.1109/CVPR.1997.609451.
Beis, P.J., McKay, N.D., 1992. A method for registration of 3-d shapes. IEEE Trans. Pattern Anal. Mach. Intell. 14 (2), 239–256. https://doi.org/10.1109/34.121791.
Biermann, D., Elseberg, J., Lingemann, K., Nüchter, A., Hertzberg, J., 2008. Globally consistent 3d mapping with scan matching. Journal Robotics and Autonomous Systems (JRAS) 56 (2), 130–142. https://doi.org/10.1016/j.robot.2007.07.002. URL. https://robotik.informatik.uni-wuerzburg.de/telematics/download/ras2007.pdf
