A descriptor and voting scheme for fast 3D self-localization in man-made environments

Marvin Gordon, Marcus Hebel and Michael Arens
Fraunhofer Institute of Optronics, System Technologies and Image Exploitation IOSB
Ettlingen, Germany
{*marvin.gordon,marcus.hebel,michael.arens*}*@iosb.fraunhofer.de*

***Abstract*—In contrast to the increasing availability of affordable and lightweight 3D sensors, navigation sensors are still big, expensive (IMU), and prone to GPS errors. In view of a lightweight, affordable and robust 3D mapping solution, it is preferable to aim at a low-cost IMU and GPS-less system. Therefore, some capabilities provided by navigation hardware should be replaced by methodical solutions.**

We present an approach for data-based self-localization in a large-scale 3D model of a man-made environment (e.g., an urban area, an indoor environment), which solves substantial parts of this problem. Our approach uses a rotation invariant descriptor and a 3D voting scheme to determine the own position and orientation within available 3D data of the environment. While our methods can support loop closing during mapping, the main result is the ability for fast and GPS-less initial self-localization.

***Keywords*-geometric validation; Hough voting; 3D descriptors; MLS; LIDAR;**

I. Introduction

Active 3D vision sensors get more accurate, smaller, and less expensive. One recent example is the Velodyne VLP-16. The availability of such small and lightweight devices is one reason for the increasing usage of such sensors. They can be mounted to UAVs (like quadcopters), autonomous driving vehicles or other robots. Due to the increasing demand for driver-assistance and safety systems, even non-autonomous vehicles are equipped with such sensors. Additional other applications are the creation of maps (e.g. for simulations) or change detection (cf. [1]).

In the field of robotic motion systems, the entire group of topics containing *map creation* and *localization* is well known under the term *SLAM*: simultaneous localization and mapping (e.g. [2]). This is an important task in robotics, at least in cases where no external navigation reference is available (e.g. in GPS-denied environments). Even under availability of GPS/IMU navigation solutions not all problems are solved: low-cost navigation systems tend to be less accurate. On the other hand, even for highly accurate GPS/IMU systems the accuracy in position is only achievable with additional DGPS correction data. If used in urban areas, multiple reflections of GPS signals typically lead to so-called multipath effects, which result in a significant loss of accuracy. Another problem is the size and weight of highly accurate navigation systems, especially if they need to be mounted to small UAVs. Therefore the potentials offered by SLAM are relevant for both indoor and outdoor mapping.

In this paper we focus on the "L" part of SLAM and present an approach for automatic 3D self-localization in a given 3D model of the actual environment. The proposed methods can also be used for loop detection, which enables a more accurate map creation. Our approach is optimized for a man-made environment, where planar shapes are considered to be the typical geometric primitives to occur. Special attention is paid for the handling of the huge amount of data which needs to be managed by the method.

The localization problem has some similarities to model-based object recognition. In the case of object recognition, a comparison of a model catalog and the scene data is performed. Typically it is uncertain if one or more models are contained in the scene. For self-localization there are a few differences: the model is typically much bigger than the scene and the scene characteristics are matched to a single model in order to find the correct location of the scene within the model.

Our approach adopts the typical approach of model-based object recognition: we extract local descriptors in the model as well as in the scene. Then we match the scene descriptors to the model descriptors. The transformations resulting from these matches are evaluated in a voting process to estimate the most probable transformation, which finally results in an estimate of the own 3D pose in the model.

II. Related work

Solutions to place recognition have been presented for the case of 2D LIDAR data (e.g. [3] or [4]). Another approach published in [5] also works with 3D data, where they detect and describe interest points in range images. After matching them, for each match a transformation is calculated and evaluated using validation points. Another solution for forest and urban area place recognition is presented in [6], which includes a voting scheme for places. In [7] a *normal distribution transform* (NDT) based histogram descriptor is used for loop detection. This descriptor is not based on points, instead the data are stored in grid cells and each cell is classified as linear, planar or spherical. The descriptor is a

histogram of the given classes in the neighborhood that also incorporates the distance. To achieve rotation invariance, a normalization step is performed.

Most of the approaches mentioned above use local shape descriptors. As already stated in [8], local shape descriptors are a common approach for object recognition. Approaches for model based object recognition can be found in the literature: in [9] a matching point pair (corresponding points) votes for a model. Afterwards the models with the highest votes are selected and the matched pairs are grouped. Two pairs are grouped if the difference of the distances between the points on the model and the scene ($d_{C_1,C_2} = |d_{S_1,S_2} - d_{M_1,M_2}|$) is close to zero. The solution presented in [10] calculates a centroid for each model. Then a displacement vector is saved for every descriptor, which is given with respect to a local coordinate frame of the feature point. The features extracted from the scene are matched against the descriptors of the models. Then the saved displacement vector is used to vote in a 3D Hough space, but prior to this the displacement vector is transformed from the local feature coordinate system to a global one. The maxima in Hough space yield the position of the models present in the scene.

A. 3D shape descriptors

Different 3D descriptors have been presented in the literature. They are often combined with a feature/keypoint detector, which selects distinguished points. 3D descriptors can be categorized into global (some examples given in [11]) and local descriptors. Some descriptors include texture information (e.g. [12]), but we focus on the more general texture-less type. A *Spin Image* (SI) is an example for such a local descriptor [13]. It uses oriented points, i.e. points with an associated local normal vector. For each point in the neighborhood of a given oriented point, a 2D histogram is incremented depending on the distance perpendicular and parallel to the normal vector. Another 3D descriptor is presented in [14], the *fast point feature histogram* (FPFH). It requires a local normal vector for each point, which is taken to create a standardized frame between two points. Then three angles are calculated utilizing this frame. This triple is calculated between a query point and each point in its neighborhood. The triples are saved in a 3D histogram. In an iteration over all points, each histogram is recalculated by integrating the histogram of the neighboring points weighted by the distance. A descriptor for range images is the so-called *NARF* descriptor [15]. The local normal vector is used as the z-axis of a new coordinate system. Then a range image is calculated by using the z-axis as line-of-sight. A star is projected onto the resulting image. The descriptor consists of the accumulated change along each beam of the star. Finally, the descriptor is shifted according to a unique orientation to achieve rotation invariance. A descriptor for triangles is presented in [16]. The descriptor for a triangle is a 2D histogram, "which describes its pairwise relationship with each of the other surrounding facets within a predefined distance". The features considered are the angle between the two triangles and the range of perpendicular distances.

III. Proposed method

Our overall objective is the development of a LIDAR mapping system which achieves high accuracy, although it uses no more than a low-cost IMU. Such a low-cost IMU provides accurate data only for a short period of time, say a few seconds. Hence we combine a few seconds of IMU and LIDAR data to create a so-called *scene*, which gets linked to previously recorded data (which we call *model*). Beside avoiding GPS-related problems, this approach enables for indoor applications. In this work we concentrate on a model-based coarse localization, which also solves the loop detection problem. However, in future work, the presented methods will be supplemented by verification, fine registration, and loop closing.

Most previous approaches work directly with points, but currently available LIDAR sensors produce millions of (i.e., too many) points in a few seconds. Furthermore, we mainly consider urban or indoor environments, where the existence of planar shapes can be expected. Therefore the first step of our approach consists in a local plane extraction in a voxel grid, which keeps the data easy to handle.

Our solution to model-based 3D self-localization in man-made environments comprises several steps. At first, planes are extracted in both parts, in the *model* and in the *scene*. Then we calculate the descriptors and randomly select two voxels in the scene. For these voxels we search for the corresponding descriptors in the model. By using a geometric consistency check, a first validation of the matching pairs is done. If the check succeeds, a Euclidean transformation is estimated and a vote is cast for this transformation. After these steps have been repeated many times, a maximum is extracted from the voting space that gives the (globally) most probable transformation. Because we work with data sub-sampled to a voxel grid, the transformation will be supplemented by a fine registration.

In the following sections the details of the different steps are explained: plane extraction, the 3D descriptor, estimation of the transformation, and voting scheme.

A. Plane extraction

This step reduces the amount of data. A voxel grid that is axis-aligned to the local coordinate system is used to assign the points to grid cells (voxels). A RANSAC-based plane fit is carried out for each cell. Then we refine the plane parameters with a local *principal component analysis* (PCA) (e.g. [17]), which also yields the *centroid* of the cell (cf. Fig. 1).

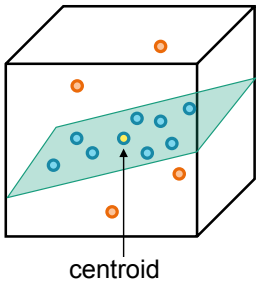

Figure 1. Inlier (blue) and outlier (orange) of RANSAC-based plane estimation in a voxel grid cell. The plane parameters are refined and the centroid is calculated by using the PCA.

B. The proposed 3D descriptor

In this subsection we present our local 3D descriptor, which is invariant to Euclidean transformations. The descriptor is a 2D histogram H with $b_a \times b_d$ bins, which counts the pairwise differences in angle and distance. Given a voxel grid cell c, we determine all neighboring cells c_i within a given radius r. We use the corresponding centroids p and normal vectors n to calculate the angle difference between the normal vectors and the centroid distance. Following that step, the corresponding histogram cell is determined (cf. (1) and (2)) and increased (3) by one. Finally the histogram is normalized to a histogram sum of 1. Fig. 2 shows an exemplary constellation and the resulting histogram.

$$a = \left\lfloor \frac{\arccos(\langle \boldsymbol{n}, \boldsymbol{n}_i \rangle)}{\pi} \cdot b_a \right\rfloor \quad (1)$$

$$d = \left\lfloor \frac{\|\boldsymbol{p} - \boldsymbol{p}_i\|}{r} \cdot b_d \right\rfloor \quad (2)$$

$$\mathrm{H}(a,d) := \mathrm{H}(a,d) + 1 \quad (3)$$

C. Estimation of the transformation

An important step is the estimation of the Euclidean transformation between the scene and the model coordinate system, given a set of matched cell pairs between these coordinate systems. Each cell contains a centroid and a normal vector. Therefore the minimum number of matched pairs to estimate this transformation is two, since one pair retains one rotational degree of freedom, i.e. the rotation angle around the normal vector would be missing.

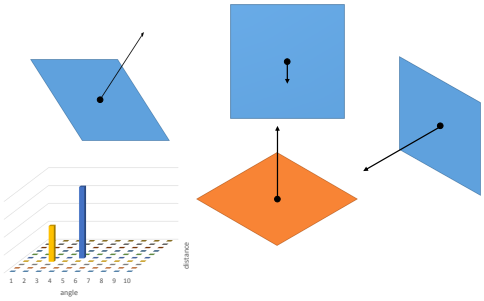

Figure 2. 2D histogram descriptor: the voxel cells are represented by their planes, centroids and normal vectors. The orange plane represents the center cell and the blue planes its neighborhood. Additionally, the histogram of the center cell is shown.

We consider two coordinate systems (CS) – one for the *scene* and one for the *model* –, for which we determine the transformation between them. For now, we suppose to have two corresponding point pairs, and it is assumed that these match perfectly. For both given CS, a new CS is constructed applying the following construction rule. Because the point pairs are assumed to match perfectly, both constructed (intermediate) CS are identical. Then we can compute a transformation between the two original CS by using the intermediate CS.

Given two points $\boldsymbol{a}$ and $\boldsymbol{b}$ as well as the normal vector $\boldsymbol{n}_a$, with the equations (4) - (7) we calculate the rotation $\boldsymbol{R}$ between the intermediate CS and the CS of $\boldsymbol{a}$ and $\boldsymbol{b}$, provided that $\boldsymbol{k}$ and $\boldsymbol{n}_a$ are not parallel (e.g. $\langle \boldsymbol{k}, \boldsymbol{n}_a \rangle < 0.9$). The normalized connecting vector $\boldsymbol{k}$ between $\boldsymbol{a}$ and $\boldsymbol{b}$ solves the problem of the missing rotation angle around the normal vector $\boldsymbol{n}_a$. The x-axis of the new CS is the normal vector $\boldsymbol{n}_a$. The y-axis cannot be represented by $\boldsymbol{k}$, since the two axes need to be perpendicular to one another. Therefore we subtract the "none perpendicular part" of $\boldsymbol{k}$ (5), like it is done within the well-known *Gram-Schmidt process* (e.g. [18]). After normalization the y-axis is represented by $\boldsymbol{m}$ (6). The resulting rotation matrix is shown in (7). As origin we select the point between $\boldsymbol{a}$ and $\boldsymbol{b}$ (8).

$$\boldsymbol{k} = \frac{1}{\|\boldsymbol{b} - \boldsymbol{a}\|}\boldsymbol{b} - \boldsymbol{a} \quad (4)$$

$$\boldsymbol{l} = \boldsymbol{k} - \langle \boldsymbol{k}, \boldsymbol{n}_a \rangle \boldsymbol{n}_a \quad (5)$$

$$\boldsymbol{m} = \frac{1}{\|\boldsymbol{l}\|}\boldsymbol{l} \quad (6)$$

$$\boldsymbol{R} = \begin{bmatrix} \boldsymbol{n}_a & \boldsymbol{m} & (\boldsymbol{n}_a \times \boldsymbol{m}) \end{bmatrix} \quad (7)$$

$$\boldsymbol{t} = 1/2\,(\boldsymbol{a} + \boldsymbol{b}) \quad (8)$$

The whole process is shown in Fig. 3 for two matching point pairs $((\boldsymbol{a}_1, \boldsymbol{a}_2)$ and $(\boldsymbol{b}_1, \boldsymbol{b}_2))$.

To transform a point $\boldsymbol{p}_{im}$ from the intermediate CS to the respective source CS, we apply the equation $\boldsymbol{p}_s = \boldsymbol{R}\boldsymbol{p}_{im} + \boldsymbol{t}$.

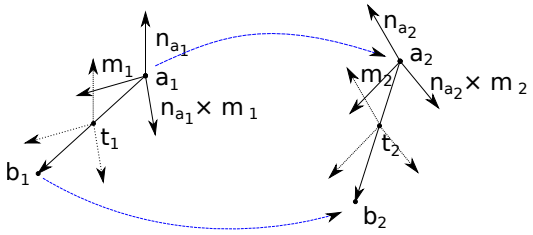

Figure 3. Construction of an intermediate coordinate system for transformation estimation: matched point pairs $(\boldsymbol{a}_1, \boldsymbol{a}_2)$ and $(\boldsymbol{b}_1, \boldsymbol{b}_2)$. The points $\boldsymbol{a}_1$ and $\boldsymbol{b}_1$ together with $\boldsymbol{a}_2$ and $\boldsymbol{b}_2$ are used to construct the coordinate system. The doted lines show both intermediate CS at their true origin.

Finally we find the transformation between CS 1 and CS 2 using the intermediate CS. Therefore we apply the previous equation in (9) and get the final rotation matrix (11) and translation vector (12).

$$\boldsymbol{p}_2 = \boldsymbol{R}_2 \left(\boldsymbol{R}_1^\mathsf{T} \boldsymbol{p}_1 - \boldsymbol{R}_1^\mathsf{T} \boldsymbol{t}_1 \right) + \boldsymbol{t}_2 \tag{9}$$

$$\boldsymbol{p}_2 = \boldsymbol{R}_2 \boldsymbol{R}_1^\mathsf{T} \boldsymbol{p}_1 - \boldsymbol{R}_2 \boldsymbol{R}_1^\mathsf{T} \boldsymbol{t}_1 + \boldsymbol{t}_2 \tag{10}$$

$$\boldsymbol{R} = \boldsymbol{R}_2 \boldsymbol{R}_1^\mathsf{T} \tag{11}$$

$$\boldsymbol{t} = \boldsymbol{t}_2 - \boldsymbol{R} \boldsymbol{t}_1 \tag{12}$$

D. Voting scheme

Annotation: Although we call it "point" in this section, this term stands for a voxel cell with its centroid and normal vector, respectively.

The following steps are relevant for our voting scheme:

1) Random selection of two points in the *scene*.
2) For both point's descriptor the k nearest neighbors in the descriptor set of the *model* are determined.
3) Geometric consistency check of all possible pairs (filtering).
4) Estimation of the transformation for the remaining matching pairs.
5) Application of the transformation to a point and voting for the resulting point.

A single position is assigned to each scene. We use the sensor position belonging to the first measured 3D point. This allows us to transform the position by using the estimated transformation resulting from a point pair and its matched pair. Hence we can carry out the voting in 3D space and do not need a 6D space, which would be required for the transformation. In a RANSAC-manner we randomly select many point pairs (around $50\ 000$) and perform the above steps. In the following we describe some more details.

The k nearest neighbors of each scene point's descriptor are determined within the available set of descriptors of the model. After this step all possible matching point pairs are constructed and prefiltered by the geometric consistency check. Details of this consistency check are given in the next subsection.

Two point pairs are used for the estimation of the transformation (cf. Section III-C). After that, the transformation is applied to the position for voting. Every vote is saved in a 3D voxel grid – aligned with the grid used for the model processing. Each vote consists of the transformed position and the associated transformation, which are both saved. Finally, we search for the cell with the maximum number of votes. The transformation of the vote with the shortest distance to the average vote position in that cell is selected as the result.

1) Geometric consistency check: The geometric consistency is checked with the inequalities (13) to (15). At first we compare the distance of the points (13). Only if the distance between the point pairs is similar enough, the following checks are performed, otherwise the pairs are discarded. With the second test the angles between the normal vector and the connecting vector are compared, because a simple comparison $\left| \langle \boldsymbol{n}_{a_1}, \boldsymbol{n}_{b_1} \rangle - \langle \boldsymbol{n}_{a_2}, \boldsymbol{n}_{b_2} \rangle \right|$ between the dot product of the normal vectors would not be sufficient to distinguish between the two cases shown in Fig. 4. Therefore the more complex inequalities (14) and (15) are evaluated.

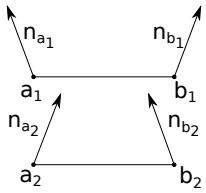

Figure 4. Two constellations to distinguish between.

$$\Big| \, \|\boldsymbol{a}_1 - \boldsymbol{b}_1\| - \|\boldsymbol{a}_2 - \boldsymbol{b}_2\| \, \Big| < \varepsilon \tag{13}$$

$$\left| \frac{\langle \boldsymbol{a}_1 - \boldsymbol{b}_1, \boldsymbol{n}_{b_1} \rangle}{\|\boldsymbol{a}_1 - \boldsymbol{b}_1\|} - \frac{\langle \boldsymbol{a}_2 - \boldsymbol{b}_2, \boldsymbol{n}_{b_2} \rangle}{\|\boldsymbol{a}_2 - \boldsymbol{b}_2\|} \right| < \alpha \tag{14}$$

$$\left| \frac{\langle \boldsymbol{b}_1 - \boldsymbol{a}_1, \boldsymbol{n}_{a_1} \rangle}{\|\boldsymbol{b}_1 - \boldsymbol{a}_1\|} - \frac{\langle \boldsymbol{b}_2 - \boldsymbol{a}_2, \boldsymbol{n}_{a_2} \rangle}{\|\boldsymbol{b}_2 - \boldsymbol{a}_2\|} \right| < \alpha \tag{15}$$

E. Paramters

The presented approach incorporates eleven parameters. The following list explains these parameters:

scene size	Number of scans combined to a scene.
voxel size	Edge length of the voxels in each dimension.
r	Search radius of the neighborhood of each cell for descriptor calculation.
k	Number of nearest neighbors to be found in the model descriptor set.
tries	Number of randomly selected voxel cell pairs.
ε	Distance threshold for geometric consistency check.
α	Angle threshold for geometric consistency check.
b_d	Size of the histogram dimension for the distance.
b_a	Size of the histogram dimension for the angle.
dist. thresh.	Threshold for the RANSAC-based plane extraction.
min. pt. nr.	Minimal number of points needed to start plane extraction.

IV. Experiments and results

Our aim is a concept not only for loop detection but also for model based self-localization. Therefore we use two

Figure 5. Sensor vehicle used for the experiments.

datasets covering the same area, but which were recorded at different days. They also contain some sort of clutter like moving persons or moved cars. One set is the *model* and different parts of the other one are used as *scene*.

We compare the estimated transformation to the ground-truth given different parts of different size from the scene dataset. More details of the data used for the experiments are given in the next subsection.

A. Experimental setup and data acquisition

For the model creation and ground-truth generation an accurate mapping system is needed. Thus the data used for the experiments were recorded with a GPS/IMU augmented sensor vehicle (cf. Fig. 5). On this vehicle, two Velodyne HDL-64E laser scanners are located over the front corners of the vehicle roof, and are positioned on a wedge with a 25 degree angle to the horizontal. This configuration guarantees a good coverage of the roadway in front of the car and allows scanning of building facades alongside and behind it. For direct georeferencing an Applanix POS LV inertial navigation system is built into the van. It comprises the following sensor components: an IMU (inertial measurement unit), two GPS antennas and a DMI (distance measuring indicator).

Each one of the laser scanners has a rotation rate of 10 Hz and a data rate of approximately 1.3 million points per second. The navigation system has a data rate of 200 Hz. For each single laser range measurement, the position and orientation of the vehicle were calculated by linear interpolation from the navigational data. Then the point is transformed to a local *ENU* (east-north-up) coordinate system with the z-axis representing the height. The CS is identical for both datasets. Though it is identical by construction, there is an offset between the datasets

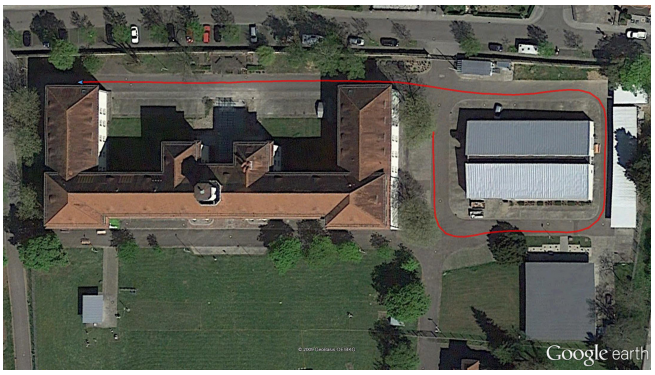

Figure 6. Path driven by the sensor vehicle during the data acquisition (image data: Google Earth, Image © 2016 GeoBasis-DE/BKG).

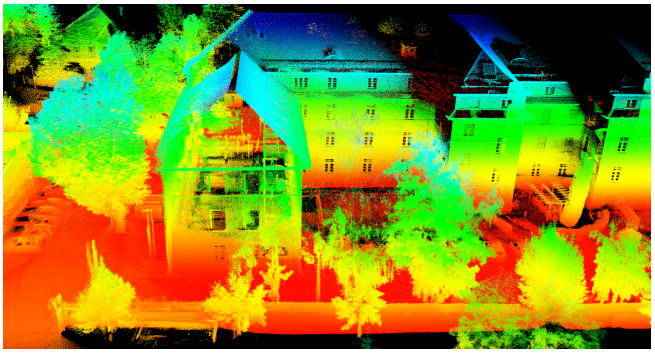

Figure 7. Plotted accumulated 3D data of the *scene* dataset.

resulting from GPS-related errors. The rotational part of this offset can be ignored, but the translation has a notable magnitude. For the two datasets it was determined by using ICP as $\boldsymbol{t} = (0.81543, -1.46302, 0.734457)^\mathsf{T}$ m, which is considered to be the "ground-truth" transformation $\boldsymbol{T}_{GT}$.

The data acquisition took place at a small test site, which is the outside area around our institute building. The main part of the scene is a big building surrounded by neighboring structures and some trees.

The path driven during the acquisition of the *model* dataset is shown in Fig. 6. It took about 1.5 minutes and resulted in approximately 140 million 3D points. This value is smaller than the actual data rate of the laser scanners, since some points measured in close vicinity of the sensors (e.g., the vehicle roof) were filtered out during the point generation process.

A similar track was driven to generate the *scene* dataset. It took approximately 1 minute to record around 110 million 3D points. The points are separated to 1 376 files where each file contains one rotation of the head of a Velodyne LIDAR.

Exemplary input data – i.e. measured points – are shown in Fig. 7. A typical voting space combined with the model planes is shown in Fig. 8. The model is shown again in Fig. 9 in combination with the correctly positioned scene (i.e., after applying the resulting transformation).

B. Details of the experiments

With parts of the experiments we evaluate the impact of the number of scans subsumed as *scene*, where two scans represent a tenth of a second. In our case this correlates to the size of the scene, because we drove with an almost constant vehicle speed. A widespread scene increases the probability of an unique geometrical constellation, which has a better chance to be found in the model. We examine this dependency for the three values of 30, 50 and 100 scans – corresponding to 1.5, 2.5 and 5 seconds of measuring time. The other parameters of our approach were set according to Table I.

Table I
PARAMETER SETTINGS FOR THE EXPERIMENTS.

Parameter	Value
voxel size	2 m
r	6 m
k	30
tries	50 000
ε	0.5 m
α	0.05
b_d	10
b_a	10
dist. thresh.	0.2 m
min. pt. nr.	200

Our approach uses a local plane extraction in a voxel grid. The resulting planes and thus the descriptors may depend on the alignment (position, orientation) of the voxel grid. To examine the amount of independence with respect to the grid, we apply four different intentional ("faked") transformations $\boldsymbol{T}_S$ to the *scene* dataset:

1) no translation and rotation at all,
2) only a translation of half the value of the voxel cell size in each dimension,
3) translation as previous, slight rotation around x- and y-axis and major rotation around the z-axis (0.01 rad and 0.4 rad),
4) also major rotation around x- and y-axis (0.5 rad).

We did 129 runs for each scene size, each time selecting a different start scan and iterating in steps of 10 through the scene dataset – containing 1 376 scans.

C. Results

The estimated transformations are assessed according to two measures: first the distance of the sensor position p_{sp} used for voting (cf. Section III-D) and secondly the rotation angle between the following transformations: we compare the "ground-truth" transformation $\boldsymbol{T}_{GT}$ with $\boldsymbol{T}_{Est} \circ \boldsymbol{T}_S$, which is the transformation resulting from concatenating the intentionally applied transformation with the estimated transformation. In case of a perfect result they are equal and (16) should hold, therefore we use the distance (17) as a quantity for the assessment.

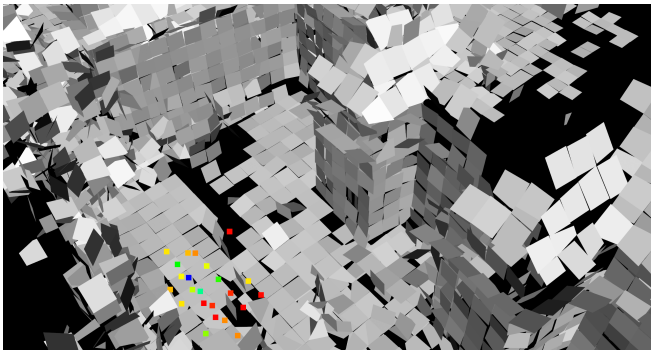

Figure 8. Voting cells with more than 9 votes and the model, where the color represents the number of votes in the cell – between 10 = red and 40 = blue.

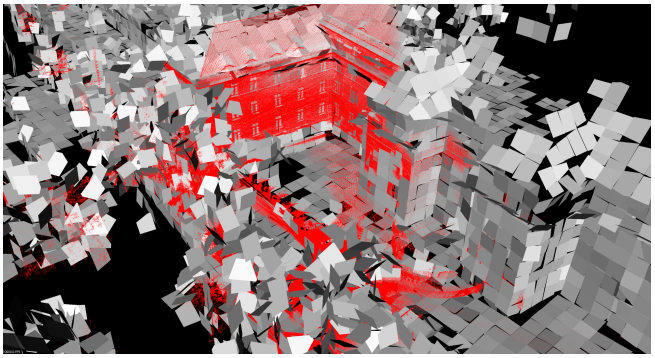

Figure 9. Estimated position of a *scene* in the *model*. The scene point cloud is depicted in red, whereas the gray facets represent the model.

$$\boldsymbol{T}_{GT} \stackrel{!}{=} \boldsymbol{T}_{Est} \circ \boldsymbol{T}_S \tag{16}$$

$$\|\boldsymbol{T}_{GT} \cdot p_{sp} - (\boldsymbol{T}_{Est} \circ \boldsymbol{T}_S) \cdot p_{sp}\| \tag{17}$$

Parts of the results are shown in Fig. 10 to 12. Each figure shows the results for transformation type 3 (s. previous subsection) for the 129 sub-scenes. Especially Fig. 10 and 11 unveil the existence of different regions with a different chance for successful localization.

All results are accumulated in Fig. 13: it shows the proportion of successful localization with respect to the scene size and the transformation type. In a summary the following statement holds: the more comprehensive the scene, the larger is the proportion of successful localization. Transformation type 4 generates the least number of correct estimations followed by type 1, type 3 and type 2.

D. Runtime and enviroment

The implementation was done in *C++* and the *Point Cloud Library* (PCL) [19] is mainly used for plane estimation and visualization purposes. Nearest neighbor search was performed using the *FLANN*-library [20]. Up to now, none of the parts are parallelized or optimized in any way, therefore a typical runtime for the scene voxel grid filling (from

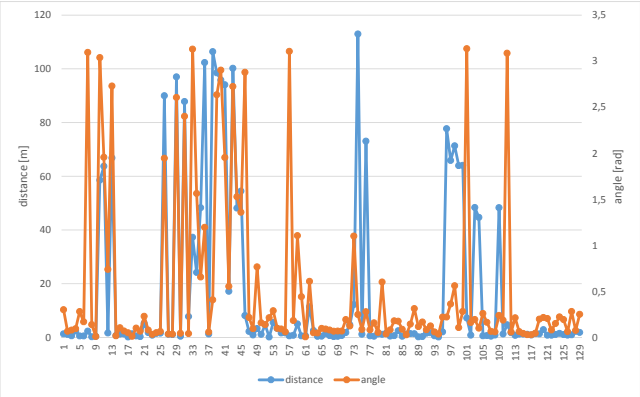

Figure 10. Distance and angle errors of the estimated transformations for a scene size of 30 scans and transformation type 3.

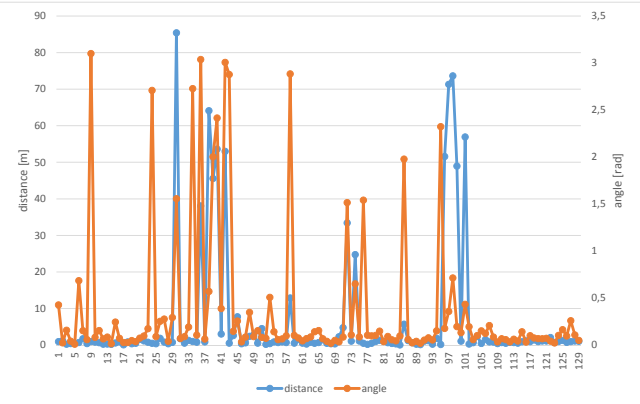

Figure 11. Distance and angle errors of the estimated transformations for a scene size of 50 scans and transformation type 3.

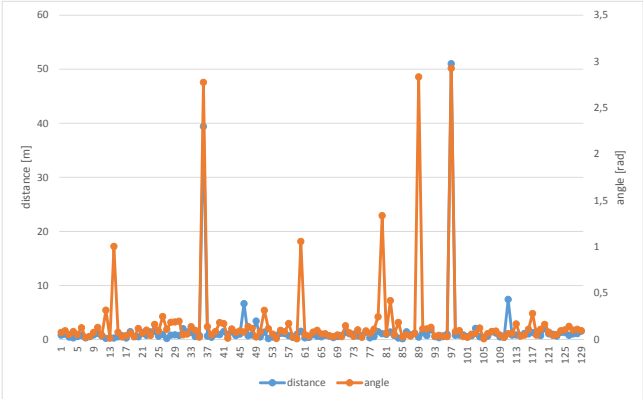

Figure 12. Distance and angle errors of the estimated transformations for a scene size of 100 scans and transformation type 3.

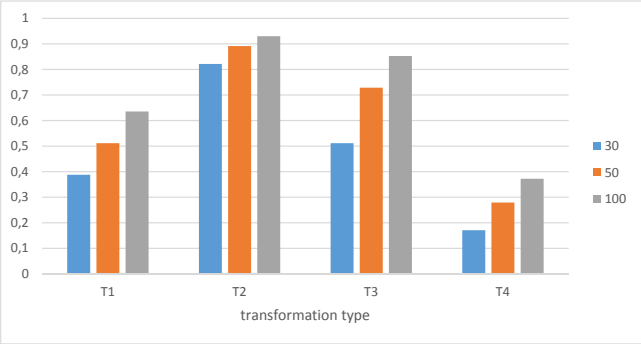

Figure 13. Proportion of correctly estimated transformations for all evaluated cases: four transformations and three scene sizes. A transformation is considered as correct, if the distance is below 5 m and the angle is below 10°.

recorded data), plane estimation and final transformation estimation took around 83 seconds. First results with a parallelized version indicate a speedup of factor 8 using a 12 core processor.

V. Conclusion

Our results show that the presented approach allows for a successful global localization in the model. Although the voxel grid position and orientation as well as the scene size considerably affect the localization performance, localization at adequate locations is possible even under poor conditions. Especially if the z-axis (should correspond to the direction of gravity) is nearly identical for the *model* and the *scene*, which was the case with transformation type 3, the success rate is above 50 %. The alignment of the z-axes can be achieved by using a low-cost IMU.

Transformation type 2 yields better results than type 1, because the faked translation has similarities with the "ground-truth" translation. Thus the resulting voxel grid alignment is more similar to the one of the model.

A widespread scene increases the probability of a successful localization, because the scene covers a greater area and more likely contains distinct geometrical constellations. E.g., a scene containing only a long building parallel to the street is less distinct than a crossing. This can be observed in the details of the results: in many places the localizations work well, however, a local lack of geometrical complexity in the scene could avoid a successful localization.

We presented a successful approach for model based self-localization, which for the sake of performance works on voxels rather than directly on points. Furthermore it accomplishes a global search and does not need any pre-localization. While the performance is dependent on the voxel grid alignment, it is still sufficient: a global localization is not needed for new every position, instead it can be tracked over time and corrected as soon as a (initial) localization succeeded.

Further work will concentrate on verification and plane based fine registration with special consideration of the voxel cell planes. Beside this, the localization will be combined with a system for handling multiple states and their probabil-

ity – like a *particle filter*. Additional tests will be performed with publicly available datasets.

REFERENCES

[1] M. Hebel, M. Arens, and U. Stilla, "Change detection in urban areas by object-based analysis and on-the-fly comparison of multi-view ALS data," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 86, pp. 52–64, 2013.

[2] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part I," *Robotics Automation Magazine, IEEE*, vol. 13, no. 2, pp. 99–110, 2006.

[3] M. Bosse and J. Roberts, "Histogram Matching and Global Initialization for Laser-only SLAM in Large Unstructured Environments," in *Robotics and Automation, 2007 IEEE International Conference on*, 2007, pp. 4820–4826.

[4] M. Himstedt, J. Frost, S. Hellbach, H. J. Böhme, and E. Maehle, "Large scale place recognition in 2D LIDAR scans using Geometrical Landmark Relations," in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, 2014, pp. 5030–5035.

[5] B. Steder, G. Grisetti, and W. Burgard, "Robust place recognition for 3D range data based on point features," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, 2010, pp. 1400–1405.

[6] M. Bosse and R. Zlot, "Place recognition using keypoint voting in large 3D lidar datasets," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, 2013, pp. 2677–2684.

[7] M. Magnusson, H. Andreasson, A. Nüchter, and A. J. Lilienthal, "Automatic appearance-based loop detection from three-dimensional laser data using the normal distributions transform," *Journal of Field Robotics*, vol. 26, no. 11-12, pp. 892–914, 2009.

[8] M. Bosse and R. Zlot, "Place Recognition Using Regional Point Descriptors for 3D Mapping," in *Field and Service Robotics*, ser. Springer Tracts in Advanced Robotics, B. Siciliano, O. Khatib, F. Groen, A. Howard, K. Iagnemma, and A. Kelly, Eds. Springer Berlin Heidelberg, 2010, vol. 62, pp. 195–204.

[9] H. Chen and B. Bhanu, "3D free-form object recognition in range images using local surface patches," *Pattern Recognition Letters*, vol. 28, no. 10, pp. 1252–1262, 2007.

[10] F. Tombari and L. Di Stefano, "Hough voting for 3d object recognition under occlusion and clutter," *IPSJ Transactions on Computer Vision and Applications*, vol. 4, no. 0, pp. 20–29, 2012.

[11] A. Aldoma, Z.-C. Marton, F. Tombari, W. Wohlkinger, C. Potthast, B. Zeisl, R. B. Rusu, S. Gedikli, and M. Vincze, "Tutorial: Point Cloud Library: Three-Dimensional Object Recognition and 6 DOF Pose Estimation," *Robotics Automation Magazine, IEEE*, vol. 19, no. 3, pp. 80–91, 2012.

[12] F. Tombari, S. Salti, and L. Di Stefano, "A combined texture-shape descriptor for enhanced 3D feature matching," in *Image Processing (ICIP), 2011 18th IEEE International Conference on*, 2011, pp. 809–812.

[13] A. Johnson, "Spin-Images: A Representation for 3-D Surface Matching," Ph.D. dissertation, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, 1997.

[14] R. B. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (FPFH) for 3D registration," in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, 2009, pp. 3212–3217.

[15] B. Steder, R. B. Rusu, K. Konolige, and W. Burgard, "Point feature extraction on 3D range scans taking into account object boundaries," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, 2011, pp. 2601–2608.

[16] A. P. Ashbrook, R. B. Fisher, C. Robertson, and N. Werghi, "Finding surface correspondence for object recognition and registration using pairwise geometric histograms," in *Computer Vision — ECCV'98*, ser. Lecture Notes in Computer Science, H. Burkhardt and B. Neumann, Eds. Springer Berlin Heidelberg, 1998, vol. 1407, pp. 674–686.

[17] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, "Surface Reconstruction from Unorganized Points," in *Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '92. ACM, 1992, pp. 71–78.

[18] Ş. S. Bayin, *Essentials of mathematical methods in science and engineering*. Hoboken, N.J.: Wiley, 2008.

[19] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, Shanghai, China, May 9-13 2011, pp. 1–4.

[20] M. Muja and D. G. Lowe, "Scalable nearest neighbor algorithms for high dimensional data," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 36, 2014.