Fast Planar Surface 3D SLAM Using LIDAR

Kruno Lenac[a,*], Andrej Kitanov[a], Robert Cupec[b], Ivan Petrović[a]

[a] *University of Zagreb, Faculty of Electrical Engineering and Computing, Unska 3, HR-10000 Zagreb, Croatia*
[b] *University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technologies Osijek, Kneza Trpimira 2B, HR-31000 Osijek, Croatia*

Abstract

In this paper we propose a fast 3D pose based SLAM system that estimates a vehicle's trajectory by registering sets of planar surface segments, extracted from 360° field of view (FOV) point clouds provided by a 3D LIDAR. Full FOV and planar representation of the map gives the proposed SLAM system the capability to map large-scale environments while maintaining fast execution time. For efficient point cloud processing we apply image-based techniques to project it to three two-dimensional images. The SLAM backend is based on Exactly Sparse Delayed State Filter as a non-iterative way of updating the pose graph and exploiting sparsity of the SLAM information matrix. Finally, our SLAM system enables reconstruction of the global map by merging the local planar surface segments in a highly efficient way. The proposed point cloud segmentation and registration method was tested and compared with the several state-of-the-art methods on two publicly available datasets. Complete SLAM system was also tested in one indoor and one outdoor experiment. The indoor experiment was conducted using a research mobile robot Husky A200 to map our university building and the outdoor experiment was performed on the publicly available dataset provided by the Ford Motor Company, in which a car equipped with a 3D LIDAR was driven in the downtown Dearborn Michigan.

Keywords: mapping, pose estimation, point cloud segmentation, planar surface registration, planar map, ESDS filter

1. Introduction

In the last two decades sensors capable of providing real-time 3D point clouds which represent the environment opened many possibilities in the field of mobile robotics and environment sensing in general. Using real-time 3D information, autonomous robots are capable to safely navigate in unknown complex environments and perform wider variety of tasks. One of the best examples comes from the automotive industry. Today, automotive industry invests a lot in research of fully autonomous driving capability, which does not require the driver to intervene and take control of the vehicle. Self-driving cars could revolutionize the transportation system known today and bring many advantages to it, such as increased efficiency and lower accidents rate. However, autonomous driving opens many new challenges for the robotic systems and the full autonomy is still far away. Some of the challenges are design of perception systems that operate reliably in changing environmental conditions and navigation in an unknown and GPS restricted complex environments. There are different sensors for environment perception (e.g. stereo cameras, depth cameras, radars, LIDARs etc.) but LIDARs are by far the most used ones. Therefore, development of a real-time Simultaneous Localization And Mapping (SLAM) solution based on 3D LIDAR measurements, that can work in large-scale environments, is of great importance.

3D LIDAR provides 360° field of view (FOV) 3D point clouds representing the surrounding environment. The main question is how to process those point clouds. Many attempts [1, 2, 3] had the philosophy that the environment is its best representation and operate directly on raw 3D point clouds. In almost all of these works, matching 3D point clouds of the environment is performed with algorithms derived from the iterative closest point (ICP) algorithm [4]. Although ICP has an advantage of implicitly solving data association problem, it suffers from premature convergence to local minima, especially when the overlap between scene samples decreases [5]. Furthermore the main problem of ICP comes from the sheer size of the raw data provided by 3D sensors that even modern computing systems cannot process in real-time.

Since raw 3D point cloud processing is computationally intensive, a question arises what is the best environment representation that would allow real time processing while preserving precision up to a certain level. Answer to this question is especially important for SLAM systems since they must simultaneously build the map and perform

*Corresponding author: Kruno Lenac
Email addresses: kruno.lenac@fer.hr (Kruno Lenac), andrej.kitanov@fer.hr (Andrej Kitanov), rcupec@etfos.hr (Robert Cupec), ivan.petrovic@fer.hr (Ivan Petrović)
URL: http://www.fer.unizg.hr/kruno.lenac (Kruno Lenac), http://www.fer.unizg.hr/andrej.kitanov (Andrej Kitanov), http://www.etfos.unios.hr/fakultet/imenik-djelatnika/rcupec (Robert Cupec), http://www.fer.unizg.hr/ivan.petrovic (Ivan Petrović)

localization using sensors information about the environment. Ability to work in real-time increases SLAM localization performance and enables it to build a meaningful map. One solution to the problem is in segmenting the sensor data into higher level features. During segmentation of raw data into higher level features some precision is always lost but the SLAM system can be designed so that this does not significantly impact its overall performance. The maps consisting of higher level features such as polygons can be almost as good for the navigation tasks as more precise maps consisting of raw data, but also enable execution in real-time which is more important in those tasks.

The main contribution of this paper is the new 3D SLAM system capable of working in real-time and in large-scale environments. We have designed every component of our SLAM system with computational and memory efficiency in mind to be suitable for navigation in large-scale complex environments. We have chosen planar surface segments to represent the environment because they are prevalent in indoor and outdoor urban spaces. Three important distinctive novelties can be pointed out. First, we have adapted point cloud segmentation algorithm from [6], which was originally developed for RGB-D sensors, to operate on full field of view 3D LIDAR point clouds. We did this by dividing and projecting full FOV 3D point clouds onto three image planes which allows fast 2.5D point cloud segmentation based on recursive 2D Delaunay triangulation and region merging. Second, we have changed algorithm presented in [7], which calculates relative pose between two local maps by registering planar surface segments contained within them, to include initial guess of the relative pose from the SLAM trajectory. We have also adapted its planar segment model and registration algorithm to take into account both the uncertainty model and 360° FOV of 3D LIDAR. By doing this we were able to significantly reduce the number of outliers in pose constraint calculation and increase process speed at the same time. Third, we have developed a global planar map building algorithm which reduces the number of required planar surface segments for map representation. Number of segments is reduced by taking into account that several planar surface segments extracted at different time can belong to one single planar segment and that same planar surface segment can be re-observed so consequently it does not form a new planar surface segment in the global map.

The rest of the paper is organized as follows. The related work and the overall concept of the proposed SLAM system are presented in Section 2. In Section 3, our point cloud segmentation algorithm is described. In Section 4, five components of our SLAM system are explained. Experimental results are presented in Section 5 and conclusion is given in Section 6.

2. Related work and overall system concept

In this section, first we present previous work related to point cloud segmentation and 3D planar SLAM, and then give the overall concept of the proposed SLAM system together with notations used throughout the paper.

2.1. Related work

Algorithms that segment point clouds into planar surface segments can be divided into organized and unorganized, based on the type of used point clouds, and structured and unstructured, based on the environments they can work on. In [8] two different algorithms are presented i) a subwindow based region growing (SBRG) algorithm for structured environments and ii) a hybrid region growing (HRG) algorithm for unstructured environments. Both SBRG and HRG work only on organized point clouds which are first divided into subwindows and then classified as planar or non planar based on their shape. Only planar subwindows are used in SBRG while both planar and non planar subwindows are used in HRG. In [9] Cached Octree Region Growing (CORG) algorithm is presented which computes planar segments from unorganized point clouds in both structured and unstructured environments. The main idea of CORG is to accelerate the neighbour searching algorithm in the octree by requiring a single nearest neighbour search trial for each point in the octree. As a result, a compromise is made between memory and speed by caching the indices of the nearest neighbours searched for each point. The solution used in our paper is based on the modified version of [6]. We used this solution because it works on unorganized point clouds as input and achieves comparable results to the techniques based on organized point clouds.

Some of the earliest 3D SLAM solutions based on planar segments use a mobile robot equipped with a rotating 2D LIDAR producing three dimensional point clouds. The paper [10] presents a feature-based SLAM approach based on the Extended Kalman Filter (EKF), where the scans are directly converted into a planar representation composed of polygons and plane parameters with associated uncertainty in the framework of Symmetries and Perturbation model (SPmodel). The resulting maps are very detailed and compact but the approach is computationally very demanding and therefore not suitable for real-time applications. Work described in [11] is interested only in the local aspect of mapping, doing sequential surface capture of the workspace, while global pose corrections are propagated on-line in an elastic graph with a bounded number of elastic sub-maps. A feature based graph SLAM that uses rectangles in order to build a global map of indoor environments is presented in [12]. Algorithm allows extraction of rectangles from LIDAR measurements even in conditions of only partial visibility.

Besides LIDAR based SLAM solutions that use higher level features, there are several SLAM solutions based on dense point clouds obtained from RGB-D sensors [13, 14,

15, 16, 17] and by dense multi-view stereo reconstruction [13, 18]. These methods are different from LIDAR based approaches due to significantly smaller operating range and field of view. In order to address sensor limitations of the range cameras, [19] makes use of 2D laser range finders in combination with a range camera sensor in a planar surface 3D SLAM system. In that work, 2D LIDAR line measurements are used to constrain planar landmark poses in graph optimization.

The approach that is the most similar to ours is presented in [5]. It uses a fast pose-graph relaxation technique for enhancing the consistency of the three-dimensional maps created by registering large planar surface segments. The surface segments are extracted from point clouds acquired from a 2D rotating LIDAR. The approach accurately determines the rotation, although a lack of predominant surfaces in certain directions may result in translational uncertainty in those directions. Hence, a loop-closing and relaxation problem is formulated that gains significant speed by relaxing only the translational errors and utilizes the full-translation covariance determined during pairwise registration. In other words, the approach assumes that the orientation error obtained by plane registration is neglectable and minimizes only the translational errors to produce constant time updates. There are three main differences between [5] and work presented in this paper. First, we do not minimize only the translational errors, but optimize all states of the trajectory. Our method takes the advantage of the information space parametrization without incurring any sparse approximation error. Second, in [5] planar surface segments are extracted from raw point clouds, while we base our method on projection of the point clouds into lower dimensional space before extracting planar surface segments. This makes our approach by the order of the magnitude faster. Third, planar surface segments in [5] are not merged in any way and are all added to the global map, while we perform merging of extracted planar surface segments that lie on the same plane which significantly reduces complexity of the global map making it suitable to represent large-scale environments.

2.2. The overall concept of the proposed SLAM system

The layout of the proposed SLAM system with key components is shown in Fig. 1. Hereafter, we briefly describe the functions of each component, while detailed descriptions are given in Sections marked in the Fig. 1. ***SLAM backend*** is responsible for vehicle localization and trajectory building. Trajectory consists of discrete states X_i, $i = 0, \ldots, n-1$, where each state is represented by vehicle's pose (3D position and orientation quaternion) expressed in the coordinate frame assigned to the first state. A new state X_k is added to the trajectory when the pose difference between the last added state in the trajectory and the current pose exceeds predefined thresholds. Once new state is augmented two things occur: i) point cloud acquired in this state is sent to the ***Local map building*** module and ii) ***Loop detection*** algorithm begins to search for possible loop closings between the state X_k and other states in the trajectory. ***Local map building*** module then segments received point cloud into planar surface segments (PSS) and builds the local map. If loop closing is detected between states X_k and X_i indexes (k, i) are sent to the ***Local maps registration (LMR)*** module alongside with the current trajectory for predicting relative pose between the states X_k and X_i. The LMR uses SLAM trajectory and the segmented planar surface segments for the initial guess and precise estimation of the relative pose between the states X_k and X_i. Once the relative pose is calculated, it is reported back to the ***SLAM backend*** which then incorporates it as a pose constraint into the pose graph and performs trajectory update. Updated trajectory and indexes (k, i) are sent to the ***Planar surface segments update*** module which uses new trajectory to update existing planar surface segments in the local maps. After the local maps are updated, that trajectory is also sent to the ***Global map building*** module which then incorporates the updated planar surface segments together with the newly segmented planar surface segments into the global map. If no loop closing was detected the trajectory is sent to the ***Global map building*** module immediately after augmentation which then adds the newly segmented planar surface segments to the global map. In the background, ***SLAM backend*** continuously predicts the vehicle's pose using odometry information.

2.3. Notation used in this paper

Here we give the overview of commonly used notations with short descriptions for easier reference.

- X_i - State in SLAM trajectory taken at time step i
- P_l - Point cloud associated with l-th trajectory state
- M_j - Local map that consists of planar surface segments segmented from j-th point cloud
- S_j - Coordinate frame of j-th local map (the same as coordinate frame of j-th point cloud)
- $(R_{i,j},\, t_{i,j})$ - Rotation matrix and translation vector which transform S_j into S_i
- ${}^{g}F_{i,j}$ - j-th planar surface segment in i-th local map which belongs to g-th global planar surface
- $S_{F_{i,j}}$ - Local coordinate frame of j-th planar surface segment in i-th local map
- $({}^{F}R_{i,j},\, {}^{F}t_{i,j})$ - Rotation matrix and translation vector which transform $S_{F_{i,j}}$ into S_i
- ${}^{F}n_{i,j}$ - Unit normal of $F_{i,j}$ expressed in $S_{F_{i,j}}$ (expected value is [0 0 1])
- ${}^{F}\rho_{i,j}$ - Distance of $F_{i,j}$ from $S_{F_{i,j}}$ (expected value is 0)

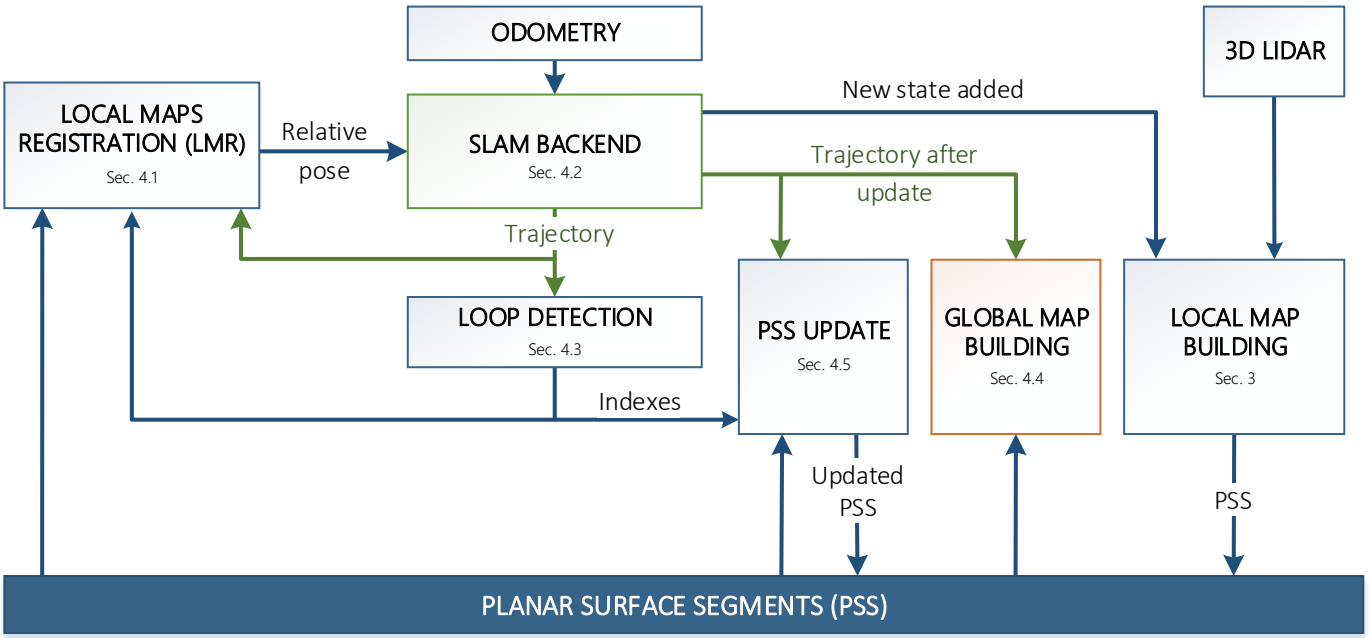

Figure 1: The overall concept of the proposed SLAM system.

- $\Sigma_{q_{i,j}}$ - Covariance matrix of perturbation vector q representing uncertainty of ${}^{F}n_{i,j}$ and ${}^{F}\rho_{i,j}$ in $S_{F_{i,j}}$
- ${}^{n}G_l$ - l-th global surface with assigned n-th local map
- S_{G_l} - Local coordinate frame of l-th global surface
- $({}^{G}R_l, {}^{G}t_l)$ - Rotation matrix and translation vector which transform S_{G_l} into coordinate frame of local map M_n assigned to ${}^{n}G_l$
- $({}^{G}R_l^0, {}^{G}t_l^0)$ - Rotation matrix and translation vector which transform S_{G_l} into coordinate frame S_0
- ${}^{G}n_l$ - Unit normal of G_l expressed in S_{G_l} (expected value is [0 0 1])
- ${}^{G}\rho_l$ - Distance of G_l from S_{G_l} (expected value is 0)
- Σ_{G_l} - Covariance matrix of perturbation vector representing uncertainty of ${}^{G}n_l$ and ${}^{G}\rho_l$ in S_{G_l}

3. Local map building

Local maps consist of planar surface segments extracted from one point cloud. In this section, we first describe a method for detection of planar surface segments from 3D LIDAR point clouds and then give their mathematical description and uncertainty model. Local maps are used for creation of the global environment map and for calculating relative poses used as the pose constraints in the SLAM pose graph.

3.1. Detection of planar surface segments

The approach used in the presented system for extracting planar surface segments is based on 2D Delaunay triangulation, which requires an appropriate 2D projection of the input point cloud. One possibility is to project the point cloud onto a cylindrical surface. However, projections of straight lines in 3D space onto a cylindrical surface are not straight lines. Therefore, triangles obtained by Dealunay triangulation applied to such projection don't represent triangular surfaces in reality. This is the reason why we choose to project point clouds onto three image planes I_1, I_2 and I_3, each covering a field of view of 120° in horizontal direction, as shown in Fig. 2(a). Point cloud projection is performed using the pinhole model. First, we define two sets: $s_n = \{0, \sin(2\pi/3), -\sin(2\pi/3)\}$ and $c_n = \{1, \cos(2\pi/3), \cos(2\pi/3)\}$. Then for each point $p(X, Y, Z)$ in the point cloud P_l the following equations are evaluated for $i = 1, 2, 3$:

$$x_i = -s_n(i)Z + c_n(i)X \tag{1}$$

$$y_i = Y \tag{2}$$

$$z_i = c_n(i)Z + s_n(i)X \tag{3}$$

$$u_i = f_u \frac{x_i}{z_i} + u_c \tag{4}$$

$$v_i = f_v \frac{y_i}{z_i} + v_c. \tag{5}$$

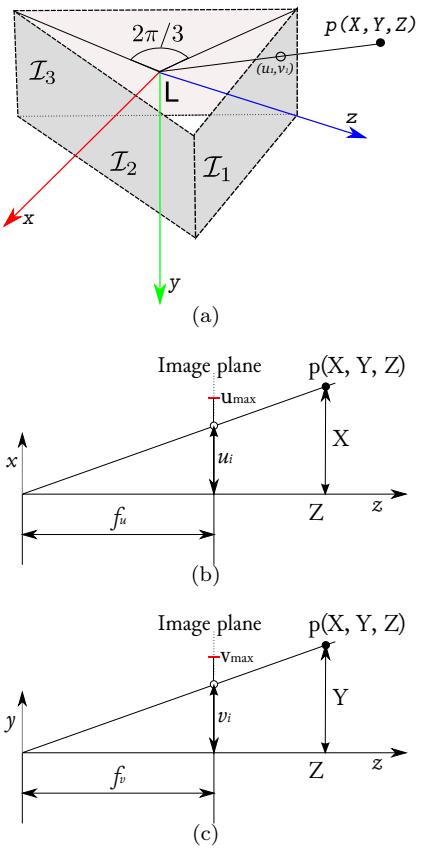

Figure 2: Point cloud division and projection onto three image planes: a) three image planes with their coordinate frame, b) determination of the horizontal pixel coordinate u_i of the point's p projection onto the image plane, c) determination of the vertical pixel coordinate v_i of the point's p projection onto the image plane.

Equations (1) - (3) represent the transformation of the point $p(X, Y, Z)$ from the LIDAR coordinate frame into the coordinate frame of the image plane I_i, while the equations (4) and (5) represent the projection of the transformed point into the pixel (u_i, v_i) of the I_i. Values of u_{max} and v_{max} define the image resolution which affects the precision of the projection. Higher image resolution corresponds to greater projection precision. However, the processing speed drops with increase of resolution. Pixel (u_c,v_c) represents the center of the projection while (f_u, f_v) represent vertical and horizontal focal lengths of the pinhole model, respectively. Their values are determined based on the LIDAR FOV (Fig. 2(b) and 2(c)) in order to capture the whole scan. Value of u_c is always set at the horizontal middle of the image plane $(u_{max}/2)$ with horizontal focal length set to $f_u = u_c/\sqrt{3}$. Values of v_c and f_v are set according to the vertical FOV of the used LIDAR.

Let the point's p projection to the image plane I_i at the image pixel (u_i, v_i) be such that conditions $u_i >= 0$, $u_i <= u_{max}$, $v_i >= 0$ and $v_i <= v_{max}$ are satisfied. We say that the point p belongs to I_i and we assign to its image pixel the distance $r = \sqrt{X^2 + Y^2 + Z^2}$ of the point p from the projection center L. For points belonging to the same image plane whose projections fall at the same image pixel, r is set to the range of the closest point from L. In this way we form a triplet (u_i, v_i, r) for every pixel in the image plane I_i corresponding to a point in the point cloud, thus obtaining 2.5D input for our segmentation method. The projected point clouds are then segmented into connected approximately planar subsets using a split-and-merge algorithm based on the approach proposed by [20], which consists of an iterative Delaunay triangulation method followed by region merging. An example of the point cloud segmentation to planar 3D surface segments is shown in Fig. 3. The obtained planar surface segments represent features which are used in the presented system for environment representation and trajectory estimation.

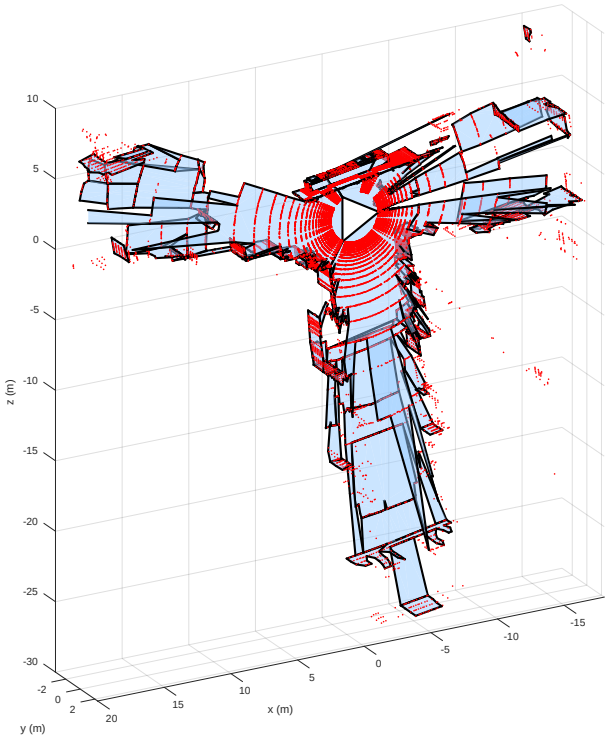

Figure 3: Local planar 3D map. Red dots represent points from the point cloud acquired by the LIDAR.

The parameters of the plane supporting a planar surface segment are determined by least-square fitting of the plane to supporting points of the segment. Each surface segment is assigned a reference frame S_F with an origin in the centroid of the supporting point set and z-axis parallel to the supporting plane normal. The orientation of the x-axis and y-axis in the supporting plane are defined by the eigenvectors of the covariance matrix Σ_p representing the distribution of the supporting points within this plane. The purpose of assigning reference frames to surface segments is to provide a framework for surface segment matching and EKF-based pose estimation explained in Section 4.1.

3.2. Representation of planar surface segments

Planar surface segments in our work are represented by sets of 2D polygons in the same way as described in [7]. Each 2D polygon is defined with its outer and inner contours and with supporting 3D plane which is defined by the equation

$$({}^F n)^{\mathrm{T}} \cdot {}^F p = {}^F \rho, \tag{6}$$

where ${}^F n$ is the unit normal of the plane represented in the planar surface segment reference frame S_F, ${}^F \rho$ is the distance of the plane from the origin of S_F and ${}^F p \in \Re^3$ is an arbitrary point of the plain represented in S_F. The uncertainty of the supporting plane parameters is described by three random variables that form the disturbance vector $q = [s_x, s_y, r]^{\mathrm{T}}$. These three variables describe the deviation of the true plane parameters from the measured plane parameters. In the ideal case, where the measured plane is identical to the true plane, the true plane normal is identical to the z-axis of S_F, which means that ${}^F n = [0, 0, 1]^{\mathrm{T}}$, while ${}^F \rho = 0$. In a general case, however, the true plane normal deviates from the z-axis of S_F and this deviation is described by the random variables s_x and s_y, representing the deviation in directions of the x-axis and y-axis of S_F respectively, as illustrated in Fig. 4 for the x direction.

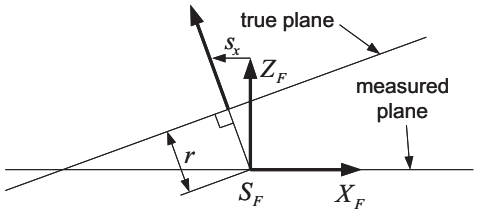

Figure 4: Deviation of the true plane from the measured plane.

The unit normal vector of the true plane can then be written as

$${}^F n = \frac{1}{\sqrt{s_x^2 + s_y^2 + 1}} \begin{bmatrix} s_x & s_y & 1 \end{bmatrix}^{\mathrm{T}}. \tag{7}$$

The random variable r represents the distance of the true plane from the origin of S_F, i.e.

$${}^F \rho_{i,j} = r. \tag{8}$$

We use the Gaussian uncertainty model, where the disturbance vector q is assumed to be normally distributed with 0 mean and covariance matrix Σ_q. Covariance matrix Σ_q is a diagonal matrix with variances σ_{sx}^2, σ_{sy}^2 and σ_r^2 on its diagonal. Computation of covariance matrices Σ_q is explained in [21]. Finally, a planar surface segment denoted by the symbol F segmented from the point cloud P_l is associated with the quadruplet

$$F = \left({}^F R,\ {}^F t,\ \Sigma_q,\ \Sigma_p \right), \tag{9}$$

where ${}^F R$ and ${}^F t$ are respectively the rotation matrix and translation vector defining the pose of S_F relative to the coordinate system S_l of the point cloud P_l.

From this point on, it is important to distinguish between different planar surface segments, so we shall briefly explain our notation. Whenever the point cloud P_l is segmented, a local map M_l is created. The local map has the coordinate frame S_l equal to the coordinate frame of P_l and consists of planar surface segments segmented from P_l. Every planar surface segment $F_{i,j}$ in M_l is identified by two indexes (i and j), where the value of index i represents the ID number of the planar surface segment in M_l and the value of index j identifies the local map which the planar surface segment belongs to (i.e. all planar surface segments contained within M_l have value of index $j = l$). According to this notation, planar surface segment $F_{i,j}$ is represented by quadruplet

$$F_{i,j} = ({}^F R_{i,j}, {}^F t_{i,j}, \Sigma_{q_{i,j}}, \Sigma_{p_{i,j}}) \tag{10}$$

4. Planar Surface 3D SLAM

In this section, we describe in details each of the five components introduced in Section 2.2 that constitute our 3D SLAM system: (i) local maps registration, (ii) SLAM backend, (iii) loop closing detection, (iv) global map building and (v) update of planar surface segments.

4.1. Local maps registration

In order to perform trajectory and map update, the relative pose between two states X_i and X_j must be estimated. This is accomplished by the local map registration (LMR) module based on the approach described in [7]. Given two local maps representing two sets of 3D planar surface segments, LMR searches for the relative pose between these two local maps which maximizes overlapping between the surface segments of the first local map and the surface segments of the second local map, transformed by this pose into the reference frame of the first local map. The approach presented in [7] is primarily designed for place recognition using RGB-D cameras, i.e. matching of a currently acquired local map to a relatively large database of previously acquired local maps. LMR applied in this paper represents an adaptation of this approach to registration of two local maps acquired by the 3D LIDAR. Since registration of planar surface sets is not in the focus of this work, we only provide a brief description of the applied registration approach with emphasis on the differences between the method presented in [7] and the LMR applied in this paper.

The considered LMR approach generates multiple hypotheses about the pose of the reference frame S_j corresponding to the state X_j with respect to the reference frame S_i corresponding to the state X_i. The hypotheses are generated by selecting small sets of pairs $(F_{a,i},\ F_{b,j})$, where $F_{a,i}$ is a planar surface segment belonging to the local map M_i and $F_{b,j}$ is a planar surface segment belonging

to the local map M_j, such that the surface segments selected from M_i have similar geometric arrangement to the corresponding segments selected from M_j. Then, EKF-based registration of the selected surface segments from M_i to the corresponding surface segments from M_j is performed, resulting in a hypothetical relative pose between X_i and X_j, which can be represented by $w_{i,j} = (R_{i,j}, t_{i,j})$, where $R_{i,j}$ is the rotation matrix defining the orientation and $t_{i,j}$ is the translation vector defining the position of X_j relative to X_i. In general, the number of possible combinations of feature pairs, which can be used for generating hypotheses, is very large. In order to achieve computational efficiency, a strategy for selection of feature pairs proposed in [6] and described in detail in [7] is applied. It is based on the feature ranking according to a measure of their usefulness in the pose estimation process. The result of this hypothesis generation process is a set of hypotheses about the pose $w_{i,j}$. Besides the correct hypothesis, this set usually contains many false hypotheses. The number of false hypotheses is significantly reduced by using the coarse information about the pose $w_{i,j}$ provided by SLAM trajectory to constrain the selection of planar surface segment pairs ($F_{a,i}$,$F_{b,j}$) to those which satisfy coplanarity criterion and overlapping criterion with respect to this initial pose estimate ${}^{I}w_{i,j}$, as proposed in [7].

The initial pose estimate ${}^{I}w_{i,j}$ is defined by rotation matrix ${}^{I}R_{i,j}$, translation vector ${}^{I}t_{i,j}$ and covariance of the transformation ${}^{I}P_{i,j}$. In order to calculate ${}^{I}w_{i,j}$, unscented transform is used. Inputs to the unscented transform are vector v_P

$$v_P = [t_i \; q_i \; t_j \; q_j]^{\mathrm{T}}, \tag{11}$$

where t_i, q_i, t_j and q_j are positions and orientation quaternions assigned to trajectory states X_i and X_j, and its covariance matrix Σ_P. Both v_P and Σ_P are obtained from the SLAM trajectory. Nonlinear function used in the unscented transform is equal to the measurement model (24) modified to operate on Euler angles instead of quaternions for representing rotation.

In addition to planar surface segments, line features representing straight edges along depth discontinuity contours can also be used. The LMR algorithm described in [7] uses line features detected by segmenting depth discontinuity contours in RGB-D images. However, for reliable detection of line features, a dense point clouds must be available, such as those obtained by RGB-D camera or by dense LIDAR scanning. Since the system proposed in this paper is designed to use sparse point clouds obtained by Velodyne HDL-32 LIDAR , line features are not considered.

The main difference between the method applied in this paper and the original LMR approach presented in [7] is in the hypothesis evaluation stage. In this stage, each generated hypothesis $w_{i,j}$ is assigned a conditional probability $P(w_{i,j}|M_j, M_i)$ representing the estimated probability that $w_{i,j}$ is the pose of X_j relative to X_i if M_j is a local map consisting of planar surface segments segmented in the state X_j and M_i is a local map consisting of planar surface segments segmented in the state X_i. The hypothesis with the highest estimated probability is selected as the final solution.

Assuming that the prior probabilities of all hypotheses are equal, maximizing $P(w_{i,j}|M_j, M_i)$ is equivalent to maximization of likelihood $p(M_j|w_{i,j}, M_i)$, i.e. the conditional probability of segmenting the local map M_j with particular parameters in the state X_j if the local map M_i is detected in the state X_i and the pose of X_j relative to X_i is $w_{i,j}$. The probability $p(M_j|w_{i,j}, M_i)$ is computed as follows:

$$p(M_j|w_{i,j}, M_i) = \prod_{F_{b,j} \in M_j} \max\{\max_{F_{a,i} \in M_i} (p(F_{b,j}|w_{i,j}, F_{b,j} \equiv F_{a,i})), p(F_{b,j})\}, \tag{12}$$

where $p(F_{b,j}|w_{i,j}, F_{b,j} \equiv F_{a,i})$ is the probability of detecting a planar surface segment $F_{b,j}$ with particular parameters in the state X_j if this segment represents the same surface as the segment $F_{a,i}$ detected in the state X_i, while $p(F_{b,j})$ represents the prior probability of detecting a planar surface segment with parameters equal to $F_{b,j}$.

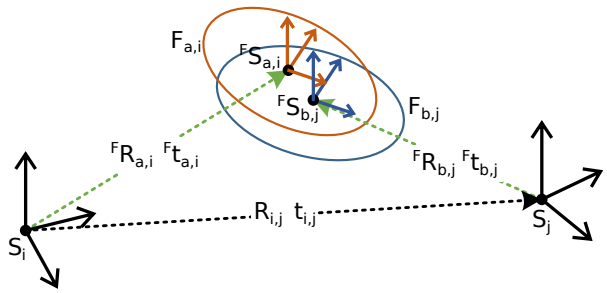

Figure 5: Relative pose between planar surface segments $F_{a,i}$ and $F_{b,j}$.

Probabilities $p(F_{b,j}|w_{i,j}, F_{b,j} \equiv F_{a,i})$ and $p(F_{b,j})$ are computed using the approach proposed in [7]. While in [7] only plane normal is considered in probability computation, in this paper, the plane offset is also considered in the same manner. The probability $p(F_{b,j}|w_{i,j}, F_{b,j} \equiv F_{a,i})$ for a particular pair $(F_{a,i}, F_{b,j})$ is computed only if surface segment $F_{b,j}$, transformed to the reference frame S_i of the local map M_i using the rotation matrix $R_{i,j}$ and translation vector $t_{i,j}$, overlaps sufficiently with $F_{a,i}$. Reference frames S_i and S_j of the local maps M_i and M_j are shown in Fig. 5, together with two planar surface segments $F_{a,i}$ and $F_{b,j}$ and their reference frames $S_{F_{a,j}}$ and $S_{F_{b,j}}$.

The approach presented in [7] is originally designed for RGB-D cameras. Due to relatively narrow FOV of RGB-D cameras, in many cases, relatively small number of dominant surfaces is captured within the camera FOV. Therefore, in addition to supporting plane parameters of the detected planar surface segments, the segment shape must also be considered in the place recognition process in

order to allow reliable distinction between geometrically similar places. In order to include the information about the surface shape in the hypothesis evaluation stage, in [7] planar surface segments are represented by sets of square patches approximating the surface shape. The 3D LIDAR used in this paper, however, has a 360° FOV, allowing it to capture many dominant surfaces in the local environment. The information provided by the plane parameters of the detected surfaces has shown to be sufficient for a reliable registration of two views. Furthermore, in contrast to the research presented in [7] where the discussed approach is used to recognize the particular place among a relatively large number of possible candidate local maps, in this paper, the considered approach is used for registration of only two close local maps, making the discrimination of surface segments according to their precise surface shape unnecessary. Therefore, time consuming surface sampling is not applied in this paper and overlapping of the scene surface segment with a corresponding planar surface segment is measured by an alternative approach. Instead of counting overlapping surface patches, which is the approach applied in [7], overlapping of two corresponding planar surface segments is measured in this paper by approximating these surface segments with ellipsoids whose radii are defined by the eigenvalues of their covariance matrices Σ_p.

Overlapping between two planar surface segments $F_{a,i}$ and $F_{b,j}$ is measured by Mahalanobis distance between distributions of points of these two surface segments represented by their centroids ${}^F t_{a,i}$ and ${}^F t_{b,j}$ and covariance matrices $\Sigma_{p_{a,i}}$ and $\Sigma_{p_{b,j}}$, which is computed by

$$d_p(F_{a,i}, F_{b,j}; w_{i,j}) = \\ ({}^F t_{a,i} - {}^F t_{b,j})^{\mathrm{T}} (\Sigma_{p_{a,i}} + \Sigma_{p_{b,j}} + \Sigma_w)^{-1} ({}^F t_{a,i} - {}^F t_{b,j}), \tag{13}$$

where Σ_w represents the uncertainty of the estimated pose $w_{i,j}$. Covariance matrix Σ_w is added to the covariance matrix $\Sigma_{p_{i,j}}$ of the surface segment $F_{b,j}$ because this feature is transformed in the reference frame S_i using pose $w_{i,j}$. Covariance matrix Σ_w is defined using a simplifying assumption that the uncertainty in translation is equal in all directions and that the uncertainty in rotation around all three axes is equal. It is computed as follows

$$\Sigma_w = \sigma_t^2 I^{3\times 3} + \sigma_R^2 (r^{\mathrm{T}} r I^{3\times 3} - r r^{\mathrm{T}}), \tag{14}$$

where $I^{3\times 3}$ is a unit matrix, σ_t and σ_R represent uncertainty of the estimated translation and rotation, respectively, and

$$r = {}^F t_{b,j} - t_{i,j}. \tag{15}$$

Parameters σ_t and σ_R are determined experimentally. Pair of planar surface segments $F_{a,i}$ and $F_{b,j}$ is considered as successful match if the following condition is satisfied

$$d_p(F_{a,i}, F_{b,j}; w_{i,j}) \leq \epsilon_p, \tag{16}$$

where the threshold ϵ_p is computed according to a desired probability assuming χ^2 distribution of d_p distance.

4.2. SLAM backend

The SLAM backend algorithm responsible for trajectory estimation and based upon Exactly Sparse Delayed State Filter (ESDSF) is developed partly in [22]. The main extension made within this paper is generalization of the motion model to 3D and a detailed analysis of the SLAM time complexity. ESDSF is used for estimation of Gaussian vehicle's trajectory X which consists of n pose samples $X_i = [q_i \ t_i]^{\mathrm{T}},\ i = 0, \ldots, n-1$:

$$X = \begin{pmatrix} X_0 \\ X_1 \\ . \\ . \\ . \\ X_{n-1} \end{pmatrix},\ X \sim\ \mathcal{N}(\mu, \Sigma) = \mathcal{N}^{-1}(\eta, \Lambda),$$

where relation between state mean μ and information vector η is: $\eta = \Lambda\mu$, and relation between state covariance Σ and information matrix Λ is: $\Lambda = \Sigma^{-1}$. q_i represents vehicle's orientation in the global frame in the form of quaternion, and t_i denotes its position in that frame. As elaborated in [22], the information matrix Λ of this system has sparse structure which makes SLAM computationally and memory efficient when implemented to use sparse algebraic system solvers and therefore significantly gains execution speed.

Let's say that we control the vehicle by applying inputs to its motors that generate translational velocity

$$v = \begin{pmatrix} v_x & v_y & v_z \end{pmatrix}^{\mathrm{T}},$$

and rotational velocity

$$\omega = \begin{pmatrix} \omega_0 & \omega_1 & \omega_2 \end{pmatrix}^{\mathrm{T}},$$

relative to the vehicle's body coordinate system. Both v and ω are represented with quaternions where the vector part of a quaternion corresponds to a direction of motion or axis of rotation, respectively, and its length to a speed amplitude. Kinematics of such system can be described by

$$\begin{pmatrix} 0 \\ \dot{x} \\ \dot{y} \\ \dot{z} \end{pmatrix} = q \begin{pmatrix} 0 \\ v_x \\ v_y \\ v_z \end{pmatrix} q^{-1} \tag{17}$$

$$\dot{q} = \frac{1}{2} q * \omega \tag{18}$$

$$\dot{q} = M \cdot q \tag{19}$$

$$\begin{pmatrix} \dot{q_0} \\ \dot{q_1} \\ \dot{q_2} \\ \dot{q_3} \end{pmatrix} = \begin{pmatrix} 0 & -\frac{\omega_0}{2} & -\frac{\omega_1}{2} & -\frac{\omega_2}{2} \\ \frac{\omega_0}{2} & 0 & \frac{\omega_2}{2} & -\frac{\omega_1}{2} \\ \frac{\omega_1}{2} & -\frac{\omega_2}{2} & 0 & \frac{\omega_0}{2} \\ \frac{\omega_2}{2} & \frac{\omega_1}{2} & -\frac{\omega_0}{2} & 0 \end{pmatrix} \begin{pmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{pmatrix}, \quad (20)$$

where $[x\ y\ z]$ is the vehicle's position and $[q_0\ q_1\ q_2\ q_3]$ its orientation quaternion in a global frame. In (18) "$*$" denotes quaternion multiplication while "$\cdot$" in (19) denotes matrix multiplication which is then expanded in (20).

We discretize the vehicle's kinematics model (17)-(18) using Euler method and describe its uncertainty with additive Gaussian white process noise w_n with mean value 0 and covariance Q_n. In this way, the non-linear first order Markov process is obtained and used in ESDSF:

$$X_{n+1} = f(X_n, u_n) + w_n, \quad (21)$$

where control signals u_n consists of changes in vehicle's position and angle in time step n determined by integrating translational v and rotational velocity ω between time steps $n-1$ and n.

Motion model (21) defines tridiagonal information matrix which remains sparse when adding new states:

$$\Lambda = \begin{pmatrix} \Lambda_{X_{n+1}X_{n+1}} & \Lambda_{X_{n+1}X_n} & & \\ \Lambda_{X_nX_{n+1}} & \Lambda_{X_nX_n} & \Lambda_{X_nX_{n-1}} & \\ & \Lambda_{X_{n-1}X_n} & \Lambda_{X_{n-1}X_{n-1}} & \\ & & \cdot & \cdot \\ & & \cdot & \cdot \\ & & \cdot & \cdot \end{pmatrix}.$$

During prediction, the state X_{n+1} is marginalized with state X_n. Since X_n is connected only to states X_{n+1} and X_{n-1}, only four blocks of information matrix need to be changed. These are: $\Lambda_{X_{n+1}x_{n+1}}, \Lambda_{X_{n+1}X_{n-1}}, \Lambda_{X_{n-1}X_{n+1}}$, $\Lambda_{X_{n-1}X_{n-1}}$.

A measurement model h in the pose SLAM system is given in a form of relative pose between two states (X_i, X_j). The relative pose describes rotation q_{ij}^L and translation t_{ij}^L of the sensor, which records 3D point clouds between poses that sensor had when it recorded data in states X_i and X_j.

$$q_{ij} = q_i^{-1} q_j \quad (22)$$

$$t_{ij} = q_i^{-1}(t_j - t_i) q_i \quad (23)$$

$$h \left\{ \begin{array}{c} q_{ij}^L = q_{RL}^{-1} q_{ij} q_{RL} \\ t_{ij}^L = q_{RL}^{-1}(q_{ij} t_{RL} q_{ij}^{-1} + t_{ij} - t_{RL}) q_{RL} \end{array} \right\} \quad (24)$$

q_{RL} and t_{RL} describe the relative pose between coordinate frames of the LIDAR sensor and the vehicle.

Measurement update in ESDSF is constant time as it affects only blocks sharing information associated to X_i and X_j. This is because the Jacobian H of the measurement function y_n, given as

$$y_n(X) = h(X_i, X_j) + v_n, v_n \sim \mathcal{N}(0, P_{i,j}) \quad (25)$$

$$H = (0, ..., \frac{\partial h}{\partial X_i}, ..., \frac{\partial h}{\partial X_j}, ..., 0) \quad (26)$$

is always sparse, since y_n depends only on these two states. The measurement arrives at time stamp n from the LMR algorithm described in Section 4.1 by registration of planar surface segments between local maps M_i and M_j. The relative pose error v_n, determined by registration, acts as a measurement noise in the estimator and is assumed to be white with normal probability distribution

In Appendix A we give an overview of ESDSF equations for the operations of state augmentation, time prediction and measurement update. Their derivation, the benefit of the exact sparsity of the delayed-state framework used in ESDSF and the advantage of the information space parametrization without incurring any sparse approximation error can be found in [23].

However, parts of the state mean vector μ and its covariance matrix Σ are still needed for motion prediction and measurement update in order to linearize nonlinear process and measurement models and to perform map registration. These can be obtained by solving a sparse, symmetric, positive-definite, linear systems of equations

$$\eta = \Lambda\ \mu \quad (27)$$

$$\Lambda \Sigma_{*i} = e_i, \quad (28)$$

where Σ_{*i} and e_i denote the ith columns of the covariance matrix and identity matrix, respectively.

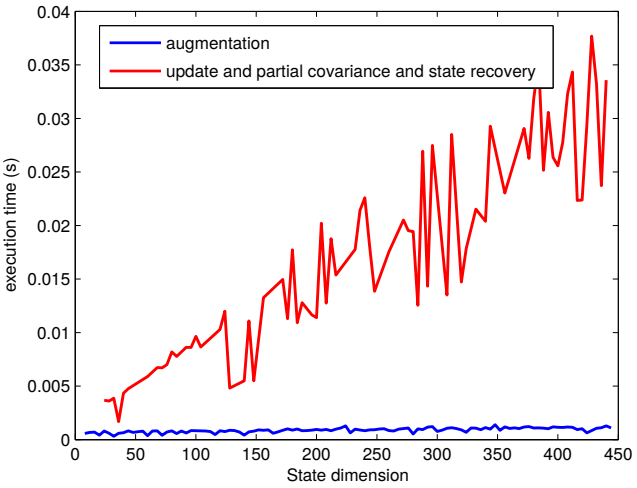

Figure 6: Execution time of the proposed SLAM system.

Figure 6 shows execution times for the operations of state augmentation and update with partial covariance and state recovery in dependence on the state dimension. Augmentation and update operation are $O(1)$ and therefore this graph shows that covariance and state recovery

are the most complex operations in information filter, and behave almost like $O(n)$, a theoretical limit. This is the result of optimizations achieved by using sparse information matrix structure in our implementation.

4.3. Loop closing detection

Whenever the new state is added to the trajectory, the loop closing algorithm begins to search for possible loop closings between newly added state X_i and all other states X_j, $j \neq i$, in the trajectory. In order to be chosen for the loop closing state X_j has to satisfy two conditions. First condition is that its Euclidean distance from X_i has to be lower than the predefined value d_{min}. This condition imposes high probability of registration between states X_j and X_i since it is likely that there are enough similar planar surface segments between local maps assigned to them. Second condition is that the resulting SLAM update will have high enough impact on the trajectory accuracy. Although every loop closing would increase accuracy of the trajectory, SLAM update and the global map update after loop closing are performance costly operations that should not be executed if the impact on the accuracy is too small. In general, update impact on the trajectory accuracy is proportional to the sum of cost functions f_c between all neighboring states in the trajectory moving from the state X_j to the state X_i. The cost function f_c takes into account both angle and distance differences between two states and is calculated as

$$\begin{aligned} f_c(i,j) =& d(X_i, X_j) + \alpha(|\Delta_{Yaw}(X_i, X_j)| + \\ &+ |\Delta_{Pitch}(X_i, X_j)| + |\Delta_{Roll}(X_i, X_j)|), \end{aligned} \tag{29}$$

where $d(X_i, X_j)$ is Euclidean distance between the states X_i and X_j, Δ_{Yaw}, Δ_{Pitch} and Δ_{Roll} are differences of Euler angles between those states and α is the scaling factor.

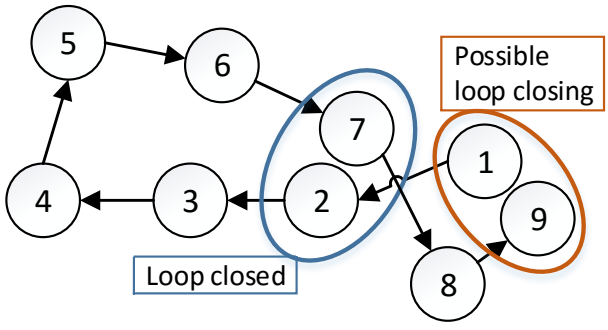

Figure 7: Topological distance as a measure of information gain.

However, the problem with always using sum of all cost functions between neighboring states to calculate accuracy impact of loop closing, arises if there were previous loop closing detections. For example let's consider situation illustrated in Fig. 7, where we want to measure accuracy impact of loop closing between states X_1 and X_9. Although the sum of cost functions between all states from X_9 to X_1 is high, since the update occurred between states X_7 and X_2, the overall impact on trajectory accuracy from closing loop between states X_9 and X_1 is small. This is because a lot of information that would be gained from loop closing between states X_1 and X_9 was already gained by loop closing between states X_2 and X_7. This is why we use approach similar to [24] as a measure of accuracy impact of a loop closing. The accuracy impact is determined using the topological distance. The topological distance is calculated from the graph ${}^{T}g$ generated from the pose graph incidence matrix ${}^{T}I$. The pose graph incidence matrix ${}^{T}I$ is $n \times n$ matrix, where n is the number of states in the trajectory, which elements are given by

$${}^{T}I_{ij} = \begin{cases} 1, & \text{if state } X_i \text{ is connected with state } X_j \\ 0, & \text{otherwise} \end{cases} \tag{30}$$

The states are connected if they are neighboring states (i.e. the state X_i is connected with states X_{i+1} and X_{i-1}) or if pose constraint was formed between them (i.e. loop closing was detected and trajectory update executed). Nodes in the graph ${}^{T}g$ are represented by the states and connections between nodes i and j exist if the element (i,j) in the matrix ${}^{T}I$ is 1. Weight of the connection $w_{i,j}$ between the states i and j is

$$w_{i,j} = \begin{cases} f_c(i,j), & \text{if } |i-j| = 1 \\ 0, & \text{otherwise} \end{cases} \tag{31}$$

This ensures that connections made by previous loop closing have 0 weight and will not increase measured topological distance between two states. Topological distance between states (X_i,X_j) is calculated as the shortest path from the node i to the node j in the graph ${}^{T}g$. The shortest path is calculated using the A* algorithm. For example, topological distance from the node 1 to the node 9 (Fig. 7) would be the sum of cost functions between the states (X_1,X_2), (X_7,X_8) and (X_8,X_9). Now when we can calculate topological distance between two states (X_i,X_j), we can form second condition that states have to satisfy in order to be chosen for the loop closing:

$${}^{T}d(X_i, X_j) > D_{max}, \tag{32}$$

where ${}^{T}d$ is the topological distance between the states X_i and X_j and D_{max} is the predefined topological distance threshold. For all states that satisfy both conditions, the state with the highest topological distance from X_i is selected for loop closing.

4.4. Global map building

In this section we present our approach to building the environment global map which consists of global planar surfaces. The reference coordinate frame of the global map is the same as the coordinate frame of the local map M_0 associated with the first state X_0 in the trajectory. Each global planar surface joins all planar surface segments from

the local maps that approximately lie on the same plane in the environment. That is why the global map has much less planar surface segments than the total number of segments in all local maps, which makes it faster to process and requires less memory to store. For example, entire floor, roof or wall is represented by only one global planar surface in the global map.

In our work, the global map is represented with a hierarchical graph. Nodes in that graph are: i) global planar surfaces ${}^{j}G_g$, at the highest level, where values of indexes g and j represent respectively the ID number of the global planar surface and the local map assigned to the g-th global planar surface, ii) local maps M_j, at the middle level, where value of index j represents the ID number of the local map, and iii) planar surface segments, at the lowest level ${}^{g}F_{i,j}$, where the value of index i represents the ID number of the planar surface segment in the local map M_j. The triplet (i, j, g) uniquely identifies the planar surface segment and its belonging to a certain local map and a certain global planar surface. For example ${}^{1}F_{3,4}$ means that ID of the planar surface segment is 3, and that it is part of the global planar surface 1 and the local map 4. If a planar surface segment does not belong to any global planar surface we omit index g from the notation. Connections between planar surface segments and local maps are formed during the segmentation process and are kept unchanged afterwards.

Every global planar surface ${}^{n}G_l$, $l \in \{1, \ldots, N_G\}$, is defined by

$$ {}^{n}G_l = \left({}^{G}R_l,\ {}^{G}t_l,\ {}^{G}\Sigma_l,\ {}^{G}R_l^0,\ {}^{G}t_l^0\right) \tag{33} $$

where N_G is the total number of global planar surfaces in the global map, ${}^{G}R_l$ and ${}^{G}t_l$ are respectively the rotation matrix and translation vector defining the transformation between local coordinate frame S_{G_l} of the global surface ${}^{n}G_l$ and the coordinate frame S_n of the local map M_n which is assigned to ${}^{n}G_l$, ${}^{G}\Sigma_l$ is the covariance matrix defining uncertainties of the perturbation vector of the unit-normal ${}^{G}n_l$ and the plane distance ${}^{G}\rho_l$ from the origin of S_{G_l}, and ${}^{G}R_l^0$ and ${}^{G}t_l^0$ are rotation and translation vector defining the transformation between S_{G_l} and S_0.

Updating of the global map occurs every time a new state is added to the trajectory. The update process differs depending on whether or not loop closing has been detected with the newly added state. Below we shall first explain how the global map updating is performed if a new state is added to the trajectory and loop closing is not detected, and than we shall describe additional steps that are performed if the loop closing is detected.

4.4.1. Global map update after trajectory augmentation without loop closing

Let's assume that there are $k-1$ states in the trajectory and that at time step k the state X_k was added. After point cloud P_k has been segmented, the local map M_k is created. At this point none of the planar surface segments from M_k is present in the global map. The next step is to determine pairs between the planar surface segments already included in the global map and the newly extracted segments from M_k. The exact way would be to try to match every planar surface segment from M_k with all previously extracted segments. However, this approach would be far too slow for real time application. Our solution is to try to match the planar surface segments from M_k only with the planar surface segments from M_{k-1} and to update the global map accordingly. With this approach the accuracy deterioration is neglectable if no loop closing is detected because it is reasonable to assume that the newly extracted planar surface segments in M_k mostly originate from the same planes as the surface segments extracted from M_{k-1}. So the first step in the global map update is to perform planar surface segments matching between M_k and M_{k-1}. Two planar surface segments are matched if they lie on approximately the same plane. This matching process is different than the segments matching when estimating relative poses between local maps in LRM. While LRM searches for planar surface segments between two local maps that are coplanar and overlap, in the case of building the global map, it is not necessary for planar surface segments to overlap. For example, if we consider a vehicle moving through a corridor, every new state will have one additional wall segment that needs to be connected with the previous segments. These segments do not overlap in the environment, but they do lie on the same plane and should represent one global planar surface. This is why only coplanarity condition is checked. The entire algorithm for checking if surfaces are coplanar is described in [7] and only final expressions and brief description are given here. In order to check if ($F_{i,k-1}$,$F_{j,k}$) are coplanar, the Mahalanobis distance $d(e)$ is used

$$ d(e) = e^{\mathrm{T}}(E\Sigma_{q_{j,k}}E^{\mathrm{T}} + CP_{k-1,k}C^{\mathrm{T}} + \Sigma_{q_{k-1,i}})^{-1}e, \tag{34} $$

where E represents the Jacobian matrix propagating the uncertainty of planar surface segment parameters and C represents the Jacobian matrix propagating the uncertainty of the transformation between two local maps (the final expressions for E and C are given in Appendix B), $P_{k-1,k}$ is the uncertainty of the transformation between local maps (calculated from the SLAM trajectory using unscented transform the same way as ${}^{I}P$) and e is the random variable given by

$$ {}^{F}\tilde{n}_{j,k} = \begin{bmatrix} {}^{F}\tilde{n}_{j,k}^{x} \\ {}^{F}\tilde{n}_{j,k}^{y} \\ {}^{F}\tilde{n}_{j,k}^{z} \end{bmatrix} = {}^{F}R_{i,k-1}^{\mathrm{T}} R_{k-1,k}\, {}^{F}R_{j,k}\, {}^{F}n_{j,k} \tag{35} $$

$$ {}^{F}\tilde{\rho}_{j,k} = ({}^{F}t_{j,k} - ({}^{F}t_{i,k-1} - t_{k-1,k})R_{k-1,k}){}^{F}R_{j,k}\, {}^{F}n_{j,k}) \tag{36} $$

$$ e = \begin{bmatrix} {}^{F}\tilde{n}_{j,k}^{x} \\ {}^{F}\tilde{n}_{j,k}^{y} \\ {}^{F}\tilde{\rho}_{j,k} \end{bmatrix} \tag{37} $$

where ${}^{F}\tilde{n}_{j,k}$ and ${}^{F}\tilde{\rho}_{j,k}$ represent the expected values of ${}^{F}n_{j,k}$ and ${}^{F}\rho_{j,k}$ transformed into $S_{F_{i,k-1}}$ and $[{}^{F}\tilde{n}^{x}_{j,k},\ {}^{F}\tilde{n}^{y}_{j,k},\ {}^{F}\tilde{n}^{z}_{j,k}]$ represent x, y, z-coordinates of ${}^{F}\tilde{n}_{j,k}$ in $S_{F_{i,k-1}}$. The pair ($F_{i,k-1}$,$F_{j,k}$) is considered coplanar if the following condition is satisfied

$$d(e) < \epsilon, \tag{38}$$

where ϵ is a measure of coplanarity calculated using χ^2-distribution with three degrees of freedom.

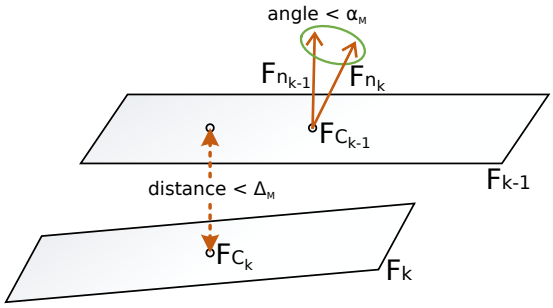

Figure 8: Second condition for matching planar surface segments.

The problem with using only coplanar condition based on the Mahalanobis distance is that covariances (especially covariance P which describes uncertainty of transformation between two local maps) can become large due to the uncertainty in the current pose, i.e., when a vehicle travels on a difficult terrain and no loop closings are detected. That is why, in order to ensure global map accuracy, all surface pairs that pass condition (38) also have to pass an additional condition based on the absolute values of their parameters differences (e). This condition acts as a cut-off threshold in the matching process and ensures that the angle between normals $\angle({}^{F}\tilde{n}_{j,k}, {}^{F}n_{i,k-1})$ and distance $F\tilde{\rho}_{j,k}$ are smaller than the predefined thresholds α_M, Δ_M, respectively (Fig. 8). Mathematically, the condition can be expressed as

$$|cos^{-1}({}^{F}\tilde{n}_{j,k}{}^{F}n_{i,k-1})| < \alpha_M \qquad |F\tilde{\rho}_{j,k}| < \Delta_M. \tag{39}$$

After forming pairs of planar surface segments between M_{k-1} and M_k, all pairs are sorted into groups. Pairs belong to the same group if one or both of the following conditions are satisfied: i) pairs have one planar surface segment in common, and ii) both pairs have at least one planar surface segment in the same global planar surface. The result is a list L_F of N_L groups: $L_F = \{l_1 ... l_{N_L}\}$. Each group in L_F contains planar surface segments from M_{k-1} and M_k.

Now for each group from L_F we determine whether it represents a new global planar surface. Group l_c, $c \in \{1 \dots N_L\}$ represents a new global surface if none of the planar surface segments from l_c are already contained within another global planar surface. For every group that represents a new global planar surface ${}^{n}G_l$, parameters of ${}^{n}G_l$ are estimated and ${}^{n}G_l$ is added to the global map.

The parameters of ${}^{n}G_l$ are estimated directly from the parameters of all planar surface segments contained within l_c. Therefore, before estimating the parameters of ${}^{n}G_l$, all parameters of all segments from the group l_c have to be transformed into the same coordinate frame $S_{F_{m,n}}$ of one of the planar surface segments from l_c. The transformation of the expected values of planar surface segment parameters is given by (35)-(37) and their covariance is

$$\tilde{\Sigma}_{q_{i,j}} = E\Sigma_{q_{i,j}}E^{\mathrm{T}} + CP_{n,i}C^{\mathrm{T}}. \tag{40}$$

Based on equations (37) and (40) the parameters of ${}^{n}G_l$ are estimated using maximum likelihood estimator. The problem which arises in the uncertainty model of plane perturbations is that uncertainty of z-coordinate of the normal is lost by imposing unit-normal constraint. To solve this issue, we add this unit-constraint

$${}^{F}\tilde{n}^{z} = \sqrt{1 - ({}^{F}\tilde{n}^{x})^2 - ({}^{F}\tilde{n}^{y})^2}$$

to the state model (37) and obtain expanded state model $e^z = [{}^{F}\tilde{n}^{x}\ {}^{F}\tilde{n}^{y}\ {}^{F}\tilde{n}^{z}\ {}^{F}\tilde{\rho}]$. Unscented transform is used to determine its covariance Σ_{q^z}. Now we apply maximum likelihood estimator to estimate ${}^{G}\tilde{n}_l$ and ${}^{G}\tilde{\rho}_l$ together with their covariances:

$$\Sigma_G = (\sum_{k=1}^{N_{l_c}} (\tilde{\Sigma}^{-1}_{q^z_k}))^{-1} \tag{41}$$

$$\begin{bmatrix} {}^{G}\tilde{n}^{x}_{l} \\ {}^{G}\tilde{n}^{y}_{l} \\ {}^{G}\tilde{n}^{z}_{l} \\ {}^{G}\tilde{\rho}_{l} \end{bmatrix} = \Sigma_G (\sum_{k=1}^{N_{l_c}} (\tilde{\Sigma}^{-1}_{q^z_k} e^z_k)), \tag{42}$$

where ${}^{G}\tilde{n}_l$ is the normal of ${}^{n}G_l$ in $S_{F_{m,n}}$, ${}^{G}\rho_l$ is the distance of ${}^{n}G_l$ from $F_{m,n}$ in $S_{F_{m,n}}$ and N_{l_c} is the number of planar surface segments in l_c. At the end, we normalize estimated normal ${}^{G}\tilde{n}_l$.

The last four parameters that need to be estimated for ${}^{n}G_l$ are rotation matrices ${}^{G}R_l$ and ${}^{G}R^0_l$ and translation vectors ${}^{G}t_l$ and ${}^{G}t^0_l$ which transform the coordinate frame S_{G_l} into the coordinate frame S_n and S_0, respectively. In order to estimate ${}^{G}R_l$, the rotation matrix R_{min} that transforms unit normal ${}^{G}\tilde{n}_l$ into unit-normal $F_{n_{m,n}}$, has to be calculated. To preserve orientation of all planar surface segments, the rotation matrix R_{min}, which corresponds to minimum Euler angles, is determined using the following equation

$$R_{min} = I + [v]_\times + [v]^2_\times \frac{1-c}{s^2} \tag{43}$$

$$[v]_\times = \begin{bmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{bmatrix}, \tag{44}$$

where $v = {}^{G}\tilde{n}^{x}_{l} - {}^{G}\tilde{n}^{y}_{l}$, $c = {}^{G}\tilde{n}^{z}_{l}$ and $s = \sqrt{({}^{G}\tilde{n}^{x}_{l})^2 + ({}^{G}\tilde{n}^{y}_{l})^2}$.

Once the matrix R_{min} is calculated, the final rotation matrix ${}^{G}R_l$ can be determined as

$$ {}^{G}R_l = {}^{F}R_{m,n} R_{min}. \tag{45} $$

A vector ${}^{l}t_G$ can be calculated using the following equation

$$ {}^{G}t_l = {}^{F}R_{m,n} \begin{bmatrix} 0 \\ 0 \\ {}^{G}\tilde{\rho}_l \end{bmatrix} + t_{F_{m,n}}. \tag{46} $$

Finally, ${}^{G}R_l^0$ and ${}^{G}t_l^0$ can be easily calculated as

$$ {}^{G}R_l^0 = R_{0,n}{}^{G}R_l \tag{47} $$

$$ {}^{G}t_l^0 = R_{0,n}{}^{G}t_l + t_{0,n}, \tag{48} $$

Next step is to perform plane merging in order to reduce the number of planar surface segments in the global map. First, we transform all contours of planar surface segments within l_c to S_{G_l}, and then perform the union of the transformed contours which are 2D polygons. The contours are transformed to S_{G_l} by transforming their vertexes using the following transformation

$$ {}^{F}\tilde{p}_{i,j} = {}^{G}R_l{}^{-1}(R_{n,j}({}^{F}R_{i,j}{}^{F}p_{i,j} + {}^{F}t_{i,j}) + t_{n,j} - {}^{G}t_l), \tag{49} $$

where ${}^{F}p_{i,j}$ is the vertex of the contour of planar surface segment $F_{i,j}$ contained within l_c and ${}^{F}\tilde{p}_{i,j}$ represents its coordinates in the coordinate frame S_{G_l}. After all local planar surface segment from l_c have been transformed using equation (49), their normal will be aligned with ${}^{G}n_l$ and their contours will be expressed in S_{G_l}. The union of contours is done using 2D polygons which are generated from the transformed contours by simply taking their x, y-coordinates. Finally, when all 2D polygons have been generated, union of polygons is performed using ***Boost*** *C++ libraries* [25]. ***Boost*** is used for creating the union because it offers fast performance and supports the union of multi-polygons and polygons with multiple holes. Once the union is complete, the resulting 2D polygon is assigned to the global planar surface ${}^{n}G_l$ as its contour.

For groups from L_F that have at least one planar surface segment ${}^{s}F_{i,j}$ which belongs to the global planar surface ${}^{n}G_s$, the only difference in the described process is that instead of transforming contour of the planar surface segment, the contour of the ${}^{n}G_s$ is transformed and used in the merging step. Here it becomes apparent why it is very important to assign a local map to the global planar surface since now all the equations used for transforming planar surface segments can be applied to the global planar surfaces by simply using ${}^{G}R_s$, ${}^{G}t_s$ and ${}^{G}\Sigma_l$ instead of ${}^{F}R_{i,j}$, ${}^{F}t_{i,j}$ and ${}^{G}\Sigma_{q_{i,j}}$ respectively. After the union has been created all the global planar surfaces used in the union are deleted because newly created global planar surface replaces them in the global map.

An example of the global map update after trajectory augmentation is shown in Fig. 9. Initially, we have two local maps M_1 and M_2. The groups generated after the matching phase are shown in Table 1. Three global planar surfaces are generated (${}^{1}G_1$, ${}^{2}G_2$ and ${}^{2}G_3$, Fig. 9, left diagram). When the state X_3 was added to the trajectory, the local map M_3 was segmented and after that the global map was updated by matching planar surface segments from M_2 and M_3. Only one segment from M_3 was matched, and the result of grouping is shown in Table 2. Since planar surface segment ${}^{3}F_{2,2}$ is already part of ${}^{2}G_3$ and ${}^{2}F_{3,2}$ is already part of ${}^{2}G_2$ the resulting global planar surface ${}^{1}G_4$ was estimated from ${}^{2}G_2$, ${}^{2}G_3$ and $F_{1,3}$. After the new global surface ${}^{1}G_4$ was generated, global surfaces ${}^{2}G_2$ and ${}^{2}G_3$ are deleted and the planar surface segments connected to them are connected to the ${}^{1}G_4$ (Fig. 9, right diagram).

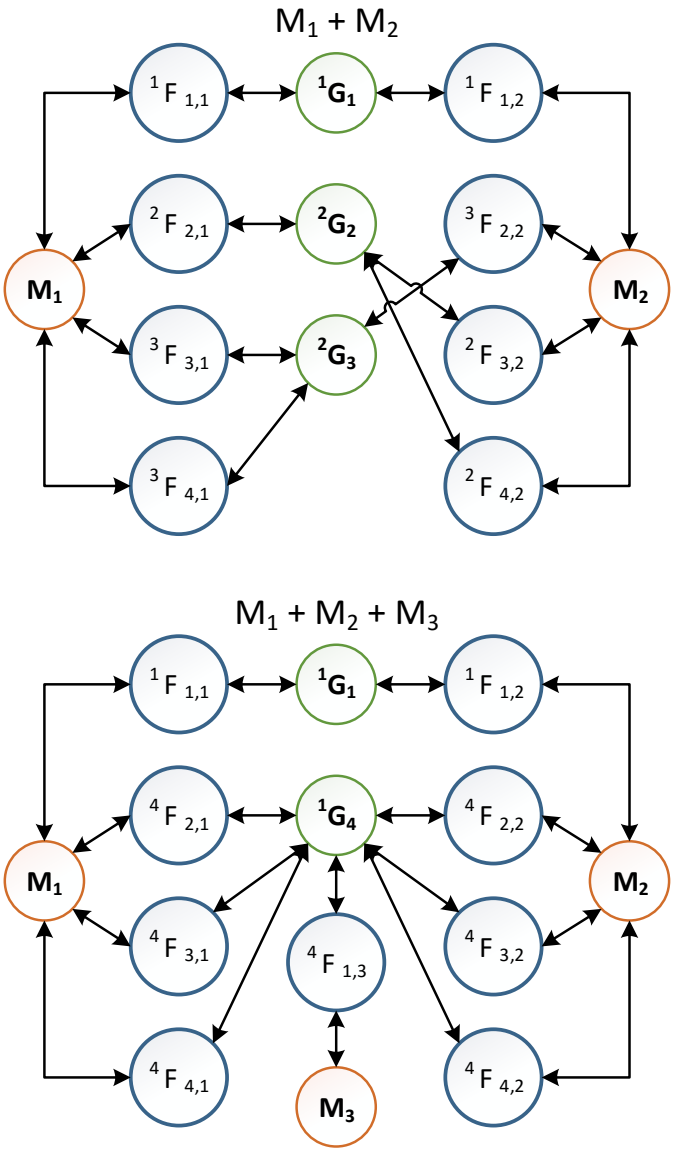

Figure 9: Example of the global map update: after update between M_1 and M_2 (top), and after update between M_2 and M_3 (bottom).

Group	M_1 IDs	M_2 IDs
1	1	1
2	2	3,4
3	3,4	2

Table 1: Groups from matches between M_1 and M_2.

Group	M_1 IDs	M_2 IDs	M_3 IDs
1		2, 3	1

Table 2: Groups from matches between M_2 and M_3.

4.4.2. Global map update after loop closing

If the loop closing is detected between a newly added state X_k to the the trajectory and an already existing trajectory state X_i, three additional steps to those explained in Section 4.4.1 are executed during the global map update:

1. Updating planar surface segments parameters in M_k and M_i
2. Reformatting the global planar surfaces if necessary
3. Updating the global map by matching planar surface segments between M_k and M_i

The first step is initiated immediately after the trajectory is updated. The planar surface segments from M_k and M_i are matched using the algorithm described in Section 4.4.1. Every matched pair is reported to the planar surface segments update algorithm which uses new SLAM trajectory to update their parameters (algorithm is described in Section 4.5). After the segments have been updated, the second step, i.e. reformatting the global planar surfaces, is performed. The global planar surface ${}^nG_l$ needs to be reformatted for two reasons: i) a planar surface segment within ${}^nG_l$ has been updated or, ii) one or more local maps containing planar surface segment that is part of ${}^nG_l$ are associated with trajectory states which poses were significantly changed. Reformatting the global planar surface ${}^nG_l$ ensures that all planar surface segments contained within ${}^nG_l$ satisfy the coplanarity conditions after the trajectory has been updated. The reformatting step is similar to the estimation of a new global planar surface. The difference is that, instead of matching planar surface segments from the local maps, planar surface segments contained within ${}^nG_l$ are matched with one-another instead. The result of the matching are groups of planar surface segments from ${}^nG_l$ that satisfy coplanarity conditions with one another. Although absolute poses of the trajectory states could significantly change due to the update, their relative positions could remain the same. In that case the result of the matching will be one group which represents the same global surface ${}^nG_l$. However, if relative positions change then several groups could exist, each representing a new global planar surface. If only one group exists there is no need to merge planar surface segments again and only values of parameters ${}^GR_l^0$, ${}^Gt_l^0$ are re-estimated. If several groups exist, parameters of the new global planar surface for each group are estimated in the same way as described in Section 4.4.1. Newly estimated global planar surfaces are added to the global map and the original global planar surface ${}^nG_l$ is deleted.

After reformatting is completed, the third and final additional step is performed. Since the states, between which loop closing is detected, are presumed to have similar local maps, matching them will result in numerous pairs that will connect newly segmented planar surface segments from M_k with global planar surfaces containing planar surface segments from M_j, thus creating ground for connecting planar surface segments from the future local maps with the same global planar surfaces. Updating of the global map by matching segments from M_k and M_i is the same process as the global map update between M_k and M_{k-1}, described in Section 4.4.1. After this step, update of the global map occurs by matching local maps M_k and M_{k-1} just like in the case of no loop closing detection.

4.4.3. Final global map generation

In order to complete the update of the global map the only thing left to do is to transform all the newly built global planar surfaces into the coordinate frame S_0. This is done by transforming every point ${}^Gp$ in contours of all newly created global planar surfaces using equation

$$ {}^G\tilde{p} = {}^GR_l^0({}^Gp) + {}^Gt_l^0 \tag{50} $$

After this step, the update of the global map is complete. However, there is a possibility that some of the global planar surfaces can still be merged with one another. The reason for this is explained by the following example. Let's say that a vehicle travels through a corridor and during that path it adds three new local maps (M_k, M_{k+1}, M_{k+2}). According to the algorithm, each wall of the corridor should be one surface. If during the recording of the LIDAR measurements for model M_{k+1} e.g. a group of people moves between corridor wall and the vehicle, the corridor surface will not be present in the model M_{k+1}. Since loop closing algorithm does not match local maps M_k and M_{k+2} there will be no connection established with the corridor planar surface segments from M_k and M_{k+2} and they will be displayed as two separate global planar surfaces in the global map. In order to solve this problem the global map has to be refined. Refinement process is done in the same way as updating the global map between two local maps, except that this time the input to the matching algorithm are global planar surfaces contained within the global map. The algorithm than matches only new global planar surfaces and global planar surfaces that were updated after the last state was added, with all unchanged global planar surfaces, and generates new groups of matched global planar surfaces. The rest of the process is the same as for the group of planar surface segments, described in Section 4.4.1. Once all the global planar surfaces from the group have been merged, the global planar surfaces within the group are deleted

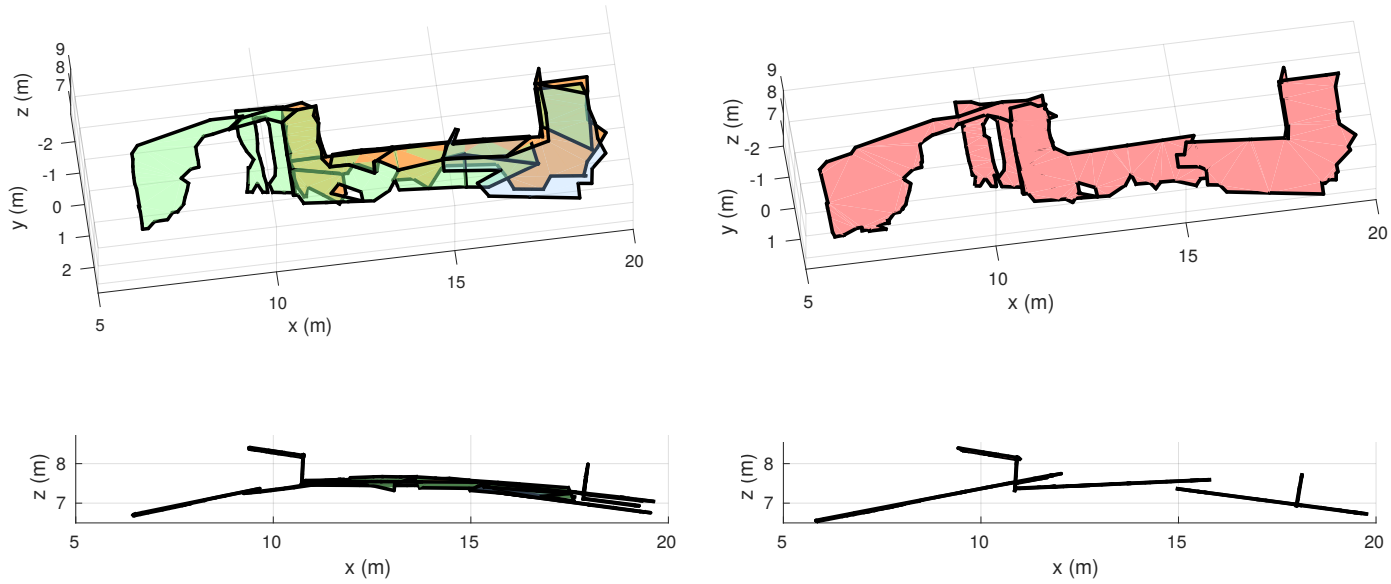

Figure 10: Left: Side and top view of curved wall consisting of planar surface segments from three consecutive local maps (blue are segments from M_{k-1}, orange are segments from M_k and green are segments from M_{k+1}); Right: Side and top view of curved wall represented in the global map by global planar surfaces.

from the global map and replaced with the merged global planar surface. When the refinement process is completed, the global map is ready to be used in any algorithm that works in combination with the SLAM. Figure 10 shows the example of a curved wall in the global map after three consecutive states are added to the trajectory and global map update process is completed.

4.5. Updating planar surface segments

Pose based SLAM systems rely on the fact that a trajectory estimation and global map building are conditionally independent. This enables separation of a trajectory estimation and environment mapping process as local maps are independent when conditioned on a specific trajectory. The joint posterior density of a trajectory X and local maps M_i and M_k can be factorized as

$$\begin{aligned} p(X, M_i, M_k) &= p(M_i, M_k|X)p(X) \\ &= p(M_i|X)p(M_k|X)p(X). \end{aligned} \tag{51}$$

The first two factors in (51) are conditional posterior density functions of the parameters of the local maps and the last one is a posterior density function of the trajectory. We have already elaborated trajectory estimation in Section 4.2. Here we consider estimation of the local maps based on the estimate of the trajectory. In our SLAM system we estimate plane perturbations in the local maps to avoid singularities of minimal parametrizations of SO(3) group. Local maps estimation occurs every time after trajectory optimization, i.e. after a relative pose constraint (measurement) is generated between M_i and M_j by their registration and the trajectory updated.

From the relative pose measurement $(R_{i,j},\ t_{i,j})$ between X_i and X_j, all the planar feature pairs $(F_{a,i},\ F_{b,j})$ in M_i and M_j consistent with that pose are found. The process of calculating $(R_{i,j},\ t_{i,j})$ and its uncertainty, and selecting suitable pairs, is the same as the one used in the global map estimation. Parameters of one planar surface segment are considered as observations to its paired planar surface segment. According to that, to estimate parameters of the $F_{a,i}$ a priori state estimate $\hat{x}$, and its covariance matrix estimate P_x, are set equal to the expected normal and offset of that planar surface and its uncertainty:

$$\hat{x} = [0\ 0\ 1\ 0]^{\mathrm{T}} \tag{52}$$

$$P_x = \Sigma_{q^z_{a,i}} \tag{53}$$

Measurement z and covariance R_z are set by transforming parameters of $F_{b,j}$ into the $S_{F_{a,i}}$:

$$z = x = \begin{bmatrix} {}^F\tilde{n}^x_{b,j} \\ {}^F\tilde{n}^y_{b,j} \\ {}^F\tilde{n}^z_{b,j} \\ {}^F\tilde{\rho}_{b,j} \end{bmatrix} \tag{54}$$

$$R_z = \tilde{\Sigma}_{q^z_{b,j}}. \tag{55}$$

Finally, we apply EKF a posteriori update which results in a new perturbation parameters x' of the planar feature $F_{a,i}$. $R'_{F_{a,i}}$ and $t'_{F_{a,i}}$ are estimated by aligning x with x' using minimal rotation matrix in the same way as in the global plane estimation. Once parameters of $F'_{a,i}$ are estimated, the same process is repeated for $F_{b,j}$, taking transformed parameters of $F_{a,i}$ as measurements and parameters of $F_{b,j}$ as a priori state estimate.

4.6. Functional flow diagram of the proposed SLAM algorithm

To summarize our SLAM algorithm, in Fig. 11 we present functional flow diagram of the entire process, from predicting current vehicle pose using odometry, to refinement of the global map. Functional flow diagram consists of the same major components as the overall system concept diagram shown in Fig. 1, but presents them in more detailed view by segmenting them to several function blocks. All blocks represented with green color belong to the ***SLAM backend*** module and all orange blocks belong to the ***Global map building*** module. Each block marked with blue color corresponds to related module of the overall system concept diagram. It is important to notice that since map building and localization are two separate processes, the orange blocks are executed in parallel with the green blocks. This means that once trajectory update is complete, ***SLAM backend*** does not wait for the global map building to finish but continues to augment and update the vehicles trajectory in the meantime. This ensures that timely global map operations, like update after the loop closing, do not affect pose accuracy.

5. Experimental results

We have divided the testing of our SLAM solution into two stages. First we present test results of our point cloud segmentation and registration algorithm and then provide test results for our SLAM system.

5.1. Test results for point cloud segmentation and registration

We compare our point cloud segmentation and registration algorithm with the state-of-the-art local 3D registration algorithms: two variants of Normal Distributions Transform (NDT) and two variants of ICP, using standardized benchmarking protocol [26] on the structured environment datasets. Also, we have compared our method to one global alternative, Minimally Uncertain Maximum Consensus (MUMC) method proposed in [5], using the "Collapsed car park" dataset[1]. The "Collapsed car park" dataset consists of 35 point clouds collected with a mobile platform and a 2D LIDAR mounted on a pan-tilt unit. Actually, we made the comparison with currently available implementation of MUMC[2] which works more accurately than the original one described in [5], but does not have graph relaxation method. Because of that we only tested both algorithms for planar segmentation and registration on a consecutive point clouds and constructed a trajectory based on them; no loop closing constraints were added nor pose graph optimizations performed. One of

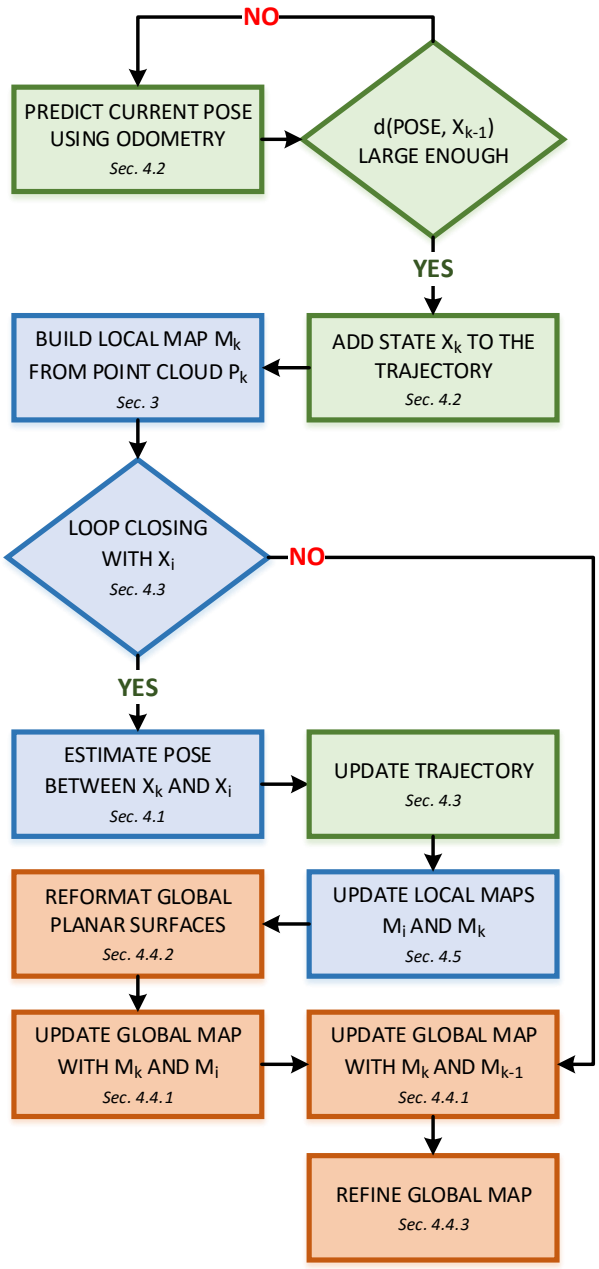

Figure 11: Functional flow diagram of the entire SLAM system.

the main parameters which impacts the MUMC performance is the number of planar surface segments used in matching, i.e. "filter percentage threshold". In MUMC, the planar surface segments are sorted in the decreasing order of their statistical certainty, and only the top filter-percentage-threshold is used for matching. This means that the lower threshold values increase computation speed but decrease the accuracy.

We chose this parameter to be the lowest possible which still produces relatively accurate registrations based on qualitative analysis of the trajectory and the map since neither a ground truth trajectory nor a 3D model is available. Figure 12 shows MUMC trajectories for three differ-

[1] http://robotics.jacobs-university.de/node/292

[2] by the courtesy of Prof. Pathak, Prof. Pfingsthorn and Prof. Vaskevicius

Table 3: Mean, maximum and minimum registration times of consecutive point clouds for our method and MUMC with different filter thresholds. All times are in seconds.

	$MUMC_{60}$	$MUMC_{40}$	$MUMC_{20}$	Our
Mean	23.30	8.02	2.83	0.29
Max	123.84	35.00	7.02	0.42
Min	3.54	1.73	0.95	0.11

ent threshold values and the trajectory of our algorithm. As can be seen, the trajectories of our algorithm and the MUMC trajectory for the threshold of 60% ($MUMC_{60}$) are the most similar, MUMC trajectory for the threshold set at 40% ($MUMC_{40}$) is still close to the first two, while MUMC trajectory for the threshold of 20% ($MUMC_{20}$) is severely degraded. Since computation time for $MUMC_{40}$ is significantly lower than for $MUMC_{60}$% (Table 3), and there is no exact way to assess the accuracy of the trajectories, we compared our method to the $MUMC_{40}$. The resolution of each of the three image planes used to project a point cloud in our method was set to 512×663.

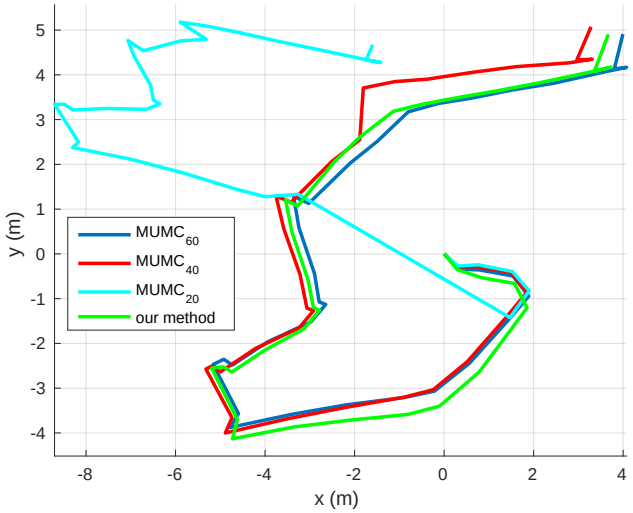

Figure 12: Estimated trajectories using our method and using MUMC with different filter percentage threshold values.

The trajectory estimation accuracy was tested indirectly by observing the quality of mapping of dominant planar structures in the scene, e.g the walls, the floor, the ceiling etc. Once the trajectory has been estimated, we used it to transform the extracted planar surface segments to the coordinate frame of the first scan. Then, we have compared how well the planar surface segments extracted from the point clouds using our method align with each other in the case we used the trajectory acquired from $MUMC_{40}$ and in the case we used our trajectory. The planar map of the area constructed from all 35 scans using our method is shown on Fig. 13. Some elements are not shown because we have filtered out smaller planes too improve visibility. All planar surface segments with the same color belong to the same point cloud. We can distinguish three main groups of planar segments: first is the wall, second is the floor and third is the ceiling group. Figure 13 also shows zoomed parts of the map. From the zoomed parts we can see how well the planar surface segments from different scans align with each other for each of the three groups. Figure 14 shows the same map, but built with trajectory estimated using $MUMC_{40}$. Both our method and the MUMC work well on this dataset although our method does align planes somewhat better which can be best seen on the ceiling group.

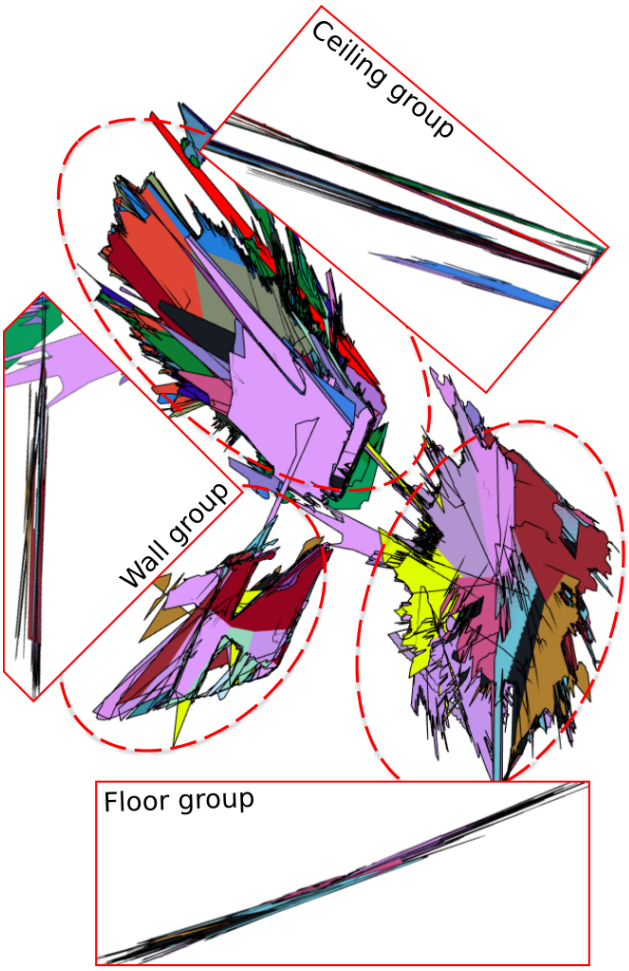

Figure 13: Planar map built using trajectory estimated by our algorithm.

Table 4 shows computation times for our method and for $MUMC_{40}$. We can see that our method is about 4 times faster in the segmentation and 26 times faster in the registration. Such high computation times for $MUMC_{40}$ method are due to the fact that MUMC is a global method, i.e. does exhaustive search for the most consistent set of planar surface correspondences without an initial guess, while we set the initial guess of our method to zero pose with high uncertainty.

The dataset used to compare our point cloud segmentation and registration method with local 3D registration algorithms is the "Challenging data sets for point cloud

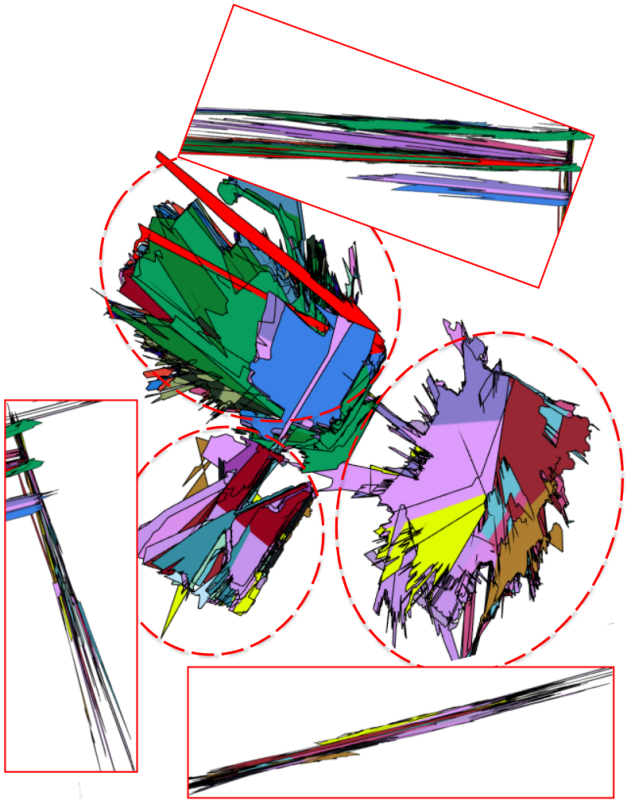

Figure 14: Planar map built using trajectory estimated by $MUMC_{40}$.

Table 4: Mean, maximum and minimum computation times for segmentation (seg). and registration (reg.) of point clouds. All times are in seconds.

	Our method			$MUMC_{40}$		
	Seg.	Reg.	Total	Seg.	Reg.	Total
Mean	0.17	0.29	0.46	0.70	8.02	8.72
Max	0.22	0.42	0.62	0.82	36.00	36.78
Min	0.13	0.11	0.22	0.59	1.73	2.35

registration algorithms"[3] described in [27]. This dataset consists of point clouds from 6 distinctive environments and covers both indoor and outdoor situations as well as structured and unstructured environments. The exact pose of every point cloud in relation to the first point cloud is provided by a highly accurate differential GPS solution. The dataset was used in [28] to several state of the art approaches for 3D scan matching. In total 5 algorithms were tested: Point based ICP variant described in [29], Plane based ICP variant described in [4], P2D-NDT method described in [30], D2D-NDT method described in [31] and MUMC method from [5]. The protocol for testing is available on the dataset website. From each environment, 35 different pairs of scans are selected and for each pair 196 initial transforms are given. The initial transforms are derived from the ground truth transforms by adding different magnitude pose offsets and, depending on its difference from the ground truth, a pair is marked as easy, medium or hard for matching. Details of the entire test protocol including the initial pose generation can be found in [26]. The results of the testing are available on the dataset web page for each algorithm except for MUMC, since MUMC does not make use of initial transform. The results for MUMC method are available in [28] and show how well the method performs depending on the overlap between two point clouds for each of the 35 pairs.

We have tested our algorithm only on indoor datasets since it is designed to work in structured environments which contain enough planar structure (i.e. indoor and outdoor urban environments), while outdoor environments of the considered dataset are deficient in dominant planar surfaces, required for our approach, and hence on these environments we don't consider it competitive with other methods. Two indoor environments we provide results for are the Apartment and the Stairs datasets. Apartment is the largest of the 6 datasets, consisting of 45 scans averaging 365000 points per scan. It includes dynamic conditions resulting from moving furniture between scans and consists of multiple reflecting surfaces. The Stairs dataset consists of 31 scans with average of 191000 points per scan. It is intended to test registration algorithms when there are rapid variations in scanned volumes and when the hypothesis of the planar scanner motion is incorrect.

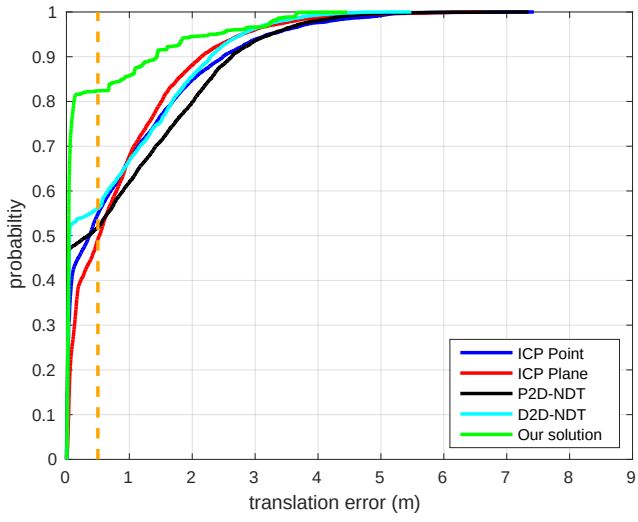

Figure 15: Cumulative translation error probability for the Apartment dataset (dashed line represents error of 0.5 m).

Figures 15 and 16 show the cumulative probability of the translation and orientation error, respectively, for the Apartment dataset after all 35*196 combinations were matched. We can see that our method outperforms all other methods in accuracy both for translation and orientation. More than 82% of translation errors are lower than 0.5 m, and more than 90% of rotation errors are lower than 20°. However in the Apartment dataset our method

[3] http://projects.asl.ethz.ch/datasets/doku.php?id=laserregistration:evaluations:home

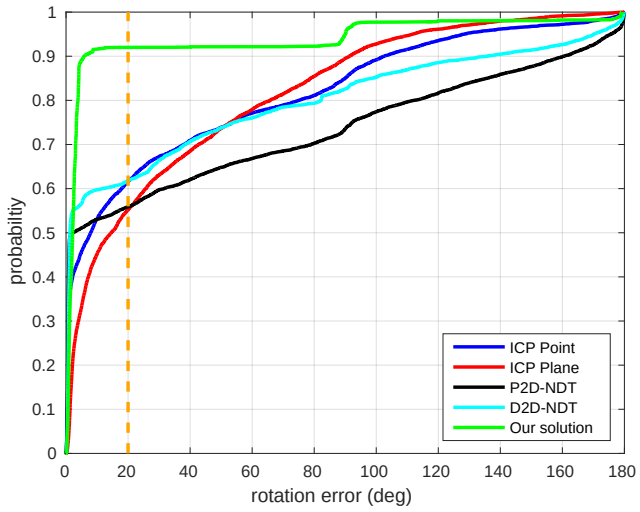

Figure 16: Cumulative rotation error probability for the Apartment dataset (dashed line represents error of 20°).

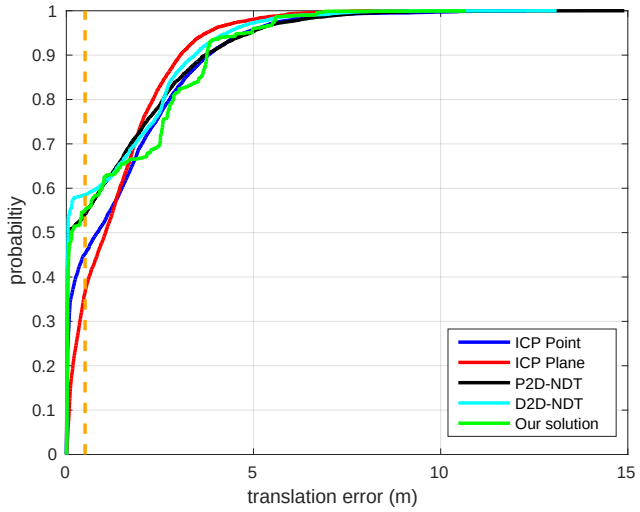

Figure 17: Cumulative translation error probability for the Stairs dataset (dashed line represents error of 0.5 m).

failed to provide relative pose estimate in 24 out of 6720 matchings (0.36%) which was automatically detected by the method itself.

Figures 17 and 18 show the cumulative probability of the translation and rotation error, respectively, for the Stairs dataset after all 6720 combinations were matched. We assume that pose estimates which differ from the true values for more than 0.5 m with respect to translation or more than 20 degrees with respect to rotation are not useful for robot navigation and focus our analysis to data within these error bounds. We can see that our algorithm has translation error greater than 0.5 m for approximately 5% measurements more than D2D-NDT, but has more accurate rotation estimation. Rotation estimation error of our algorithm is less than 20° in approximately 80% cases, while D2D-NDT achieves this result in approximately 65% cases. For the Stairs dataset there were only 3 matchings

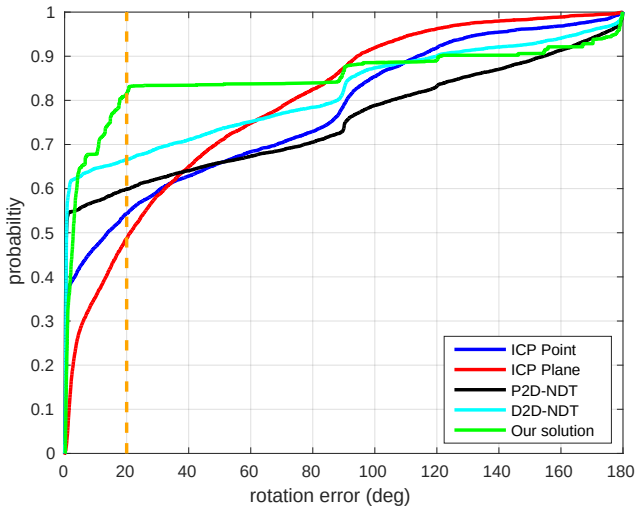

Figure 18: Cumulative rotation error probability for the Stairs dataset (dashed line represents error of 20°).

Table 5: Mean, maximum and minimum computation times of all 6720 relative poses for the Apartment dataset.

	ICP Point	ICP Plane	P2D	D2D	Our
Mean	4.37	2.35	0.62	0.22	0.36
Max	25.62	15.16	1.35	0.44	0.64
Min	0.71	0.26	0.09	0.13	0.23

without estimate (0.045%).

Computation times for the Apartment and Stairs datasets are shown in Tables 5 and 6. We can see that only D2D-NDT is slightly faster than our method. We computed the time for our method by combining the segmentation times for two point clouds and the time needed to perform registration. However, in reality almost always only one newly arrived point cloud has to be segmented while all past point clouds have already been segmented before, i.e. when doing odometry. This is why a practical application computation time of our algorithm would be even smaller. Furthermore, it has to be mentioned that D2D-NDT and P2D-NDT methods were tested using Intel Core i7 @ 3.5 Ghz while we tested our method on Intel Core i7 @ 2.6 Ghz. One more advantage of our method is that the map is represented with planar surface segments resulting in much smaller memory consumption in comparison to map representations which consist of points.

Table 6: Mean, maximum and minimum computation times of all 6720 relative poses for the Stairs dataset.

	ICP Point	ICP Plane	P2D	D2D	Our
Mean	2.45	1.50	0.86	0.22	0.35
Max	17.60	11.08	3.19	0.78	0.42
Min	0.21	0.15	0.06	0.07	0.29

5.2. Test results for the SLAM system

We tested our SLAM system on two datasets. One dataset was acquired indoor while driving a mobile robot through our university building. The other, outdoor, dataset is available online[4] and was acquired by Ford Motor Company while driving their specially equipped Ford F-250 pickup truck. All algorithms were implemented in ROS (Robot Operating System) using C++ programming language and both experiments were executed on Lenovo Thinkpad P50 with 8GB RAM and Intel Core i7-6700HQ processor at 2.6Ghz running 64-bit Ubuntu 14.04 LTS operating system.

5.2.1. Indoor experiment

The indoor experiment was conducted using equipment shown in Fig. 19. A mobile platform Husky A200 was driven with an average speed of 1m/s through our university building. It was equipped with Velodyne HDL-32E LIDAR and Xsens MTi-G-700 IMU sensor. Velodyne HDL-32E has a vertical field of view of 40° with angular resolution of 1.33°. Its measurement rate was set to 10Hz. IMU sensor and the encoders of the mobile platform were used to estimate its rotational and translational motion, respectively. Resolution of all three image planes used for projecting point cloud was set to 320 × 240.

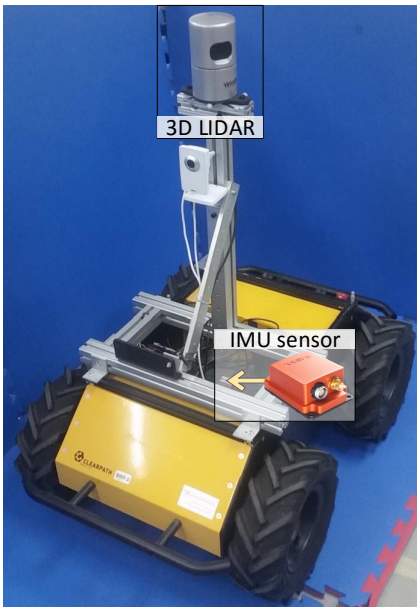

Figure 19: Equipment used in the indoor experiment.

The ground plan of the testing environment and the simplified robot trajectory are depicted in Fig. 20. This environment provides challenging conditions for SLAM including many reflective surfaces (i.e. windows and marble floors) and moving people as shown in Fig. 21. During

[4]http://robots.engin.umich.edu/SoftwareData/Ford

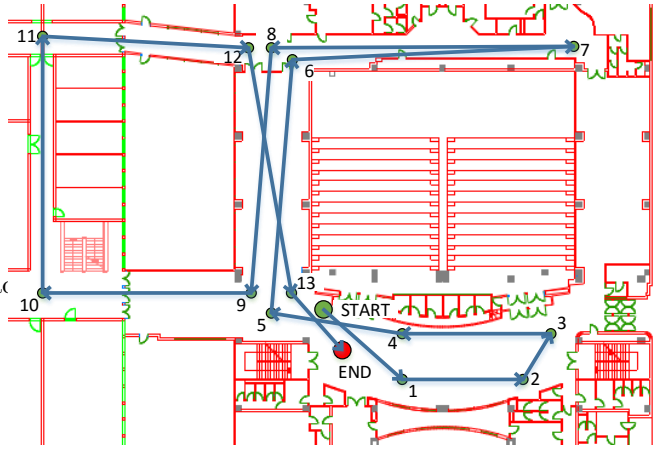

Figure 20: Ground plan and simplified trajectory.

Figure 21: Conditions in indoor environment.

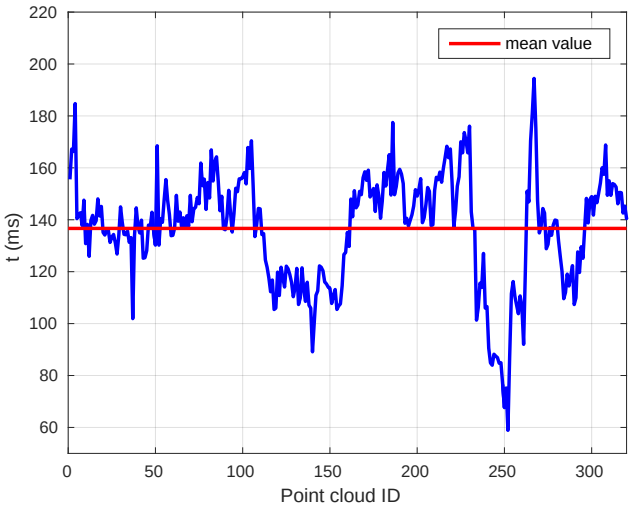

Figure 22: Point cloud segmentation time.

the experiment, 320 states were added to the SLAM trajectory. The graph presented in Fig. 22 shows the computation time for the point cloud segmentation process for every state added to the trajectory. It can be seen that it is always below 200 ms. Although we could not provide ground truth trajectory, the actual SLAM trajectory presented in Fig. 24 shows that the robot passed accurately through narrow passages like doors and corridors.

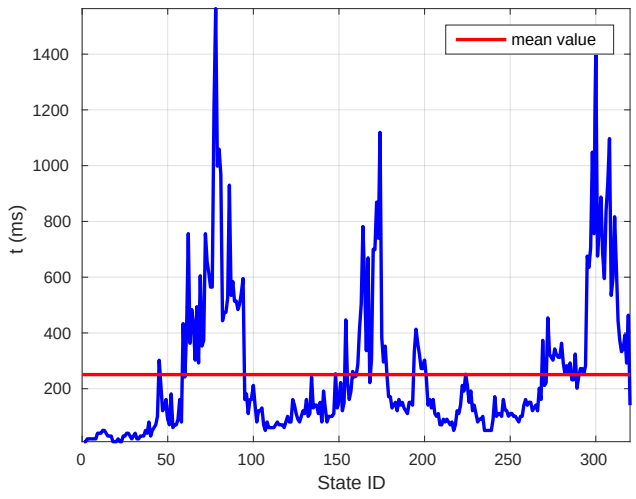

Figure 23: Global map update time.

Figure 24 also shows comparison between final SLAM and odometry trajectories. From the comparison we can see that SLAM has managed to estimate accurate trajectory although odometry used for prediction has accumulated high amount of error during the experiment. Both SLAM and odometry trajectory start from the initial robot's pose marked with green dot, and the ending poses are marked with red dot.

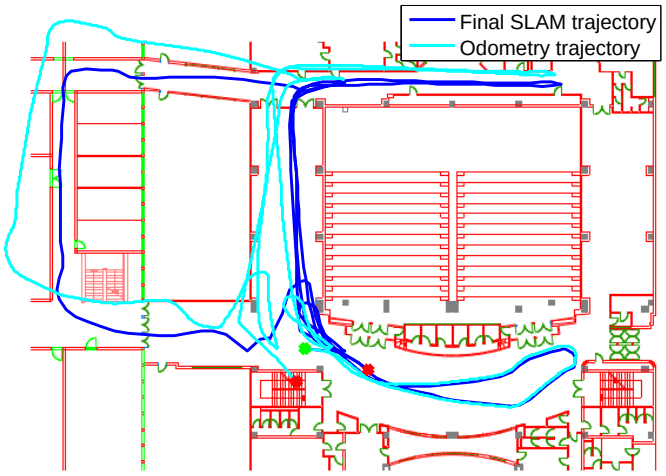

Figure 24: Odometry and SLAM trajectory comparison.

Figure 23 shows computation time of updating the global map after the new state is added to the trajectory. We can see that the mean time of the global map update is around 250 ms, but there are spikes in the computation time which are the result of loop closing detection after which additional steps described in Section 4.4.2 were performed. In ideal case, there would be only one spike after every loop closing and computation time would return back to low values at the next global map update between consecutive local maps. However in reality, computation time increases/decreases gradually before/after the loop

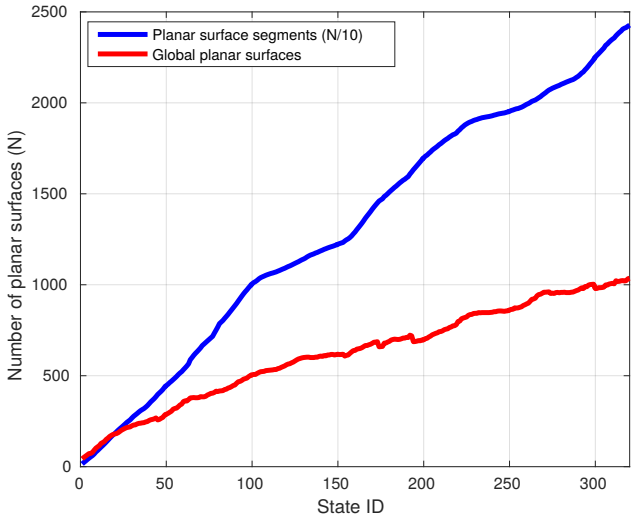

Figure 25: Total number of planar surface segments (scaled by 10) and global planar surfaces.

closure. Because the LIDAR has very long range upon entering already mapped areas, planar surface segments that belong to the surfaces already included in the global map are segmented from newly acquired point clouds before the loop closing happens, and consequently planar surface segments are matched only with previous local maps which results in new global planar surfaces being added to the global map. They are combined with already existing ones in the refinement step, which is why computation time rises even before the loop closing. When the loop closing happens, additional steps further extend the computation time, especially if the trajectory changed significantly. Since mapping works in parallel with the trajectory estimation, new states are added to the trajectory while update of the global map after the loop closing is being performed. Because of this, local maps from newly added states are not immediately incorporated into the global map. When the global map update is done, there are more than one local map that has to be incorporated into the global map using algorithm described in Section 4.4.1. They are incorporated at the next update of the global map, but since there are several of them, this update also takes longer than when incorporating only one local map, and consequently more than one states could be added to the trajectory before the update is complete. This is why the computation time gradually decreases after the loop closing. From the map update time, we can also see that our loop closing technique performs well by selecting only highly informative loop closing states. Although the platform was moving along previously traversed trajectory over significant distances and came close to several of previously added states, only one loop closing was initiated.

Figure 25 shows the total number of planar surface segments contained within all global planar surfaces (blue) and the total number of global planar surfaces after every global map update (red). At the end of the experiment,

out of the 24273 planar surface segments in all global planar surfaces, the final global map contains only 1036 surfaces, which proves that the number of planar surface segments merged in the global map is significant. It can also be seen that the number of planar segments in the global map sometimes decreases after the global map is updated with new local maps. This is direct result of merging multiple global planar surfaces in the same global planar surface.

Figure 26 shows the complete global 3D map of the area built by our SLAM system, after the last state was added to the SLAM trajectory. The roof plane was removed from the map in order to show interior structure. It can be seen that reobserving places does not introduce duplicate planar surface segments into the map. This is because localization has remained accurate and all re-observed planar surface segments are correctly merged into one global plane. It can also be seen that the final global map contains only static environment features, i.e. our map building method managed to filter out the moving objects because either they represent outliers in the matching process or can not be described as strong planar features. Video of the experiment is available online[5]. In order to at least qualitatively estimate modeling accuracy by our SLAM system, 2D ground plan of the test area was extracted from the global 3D model and plotted over 2D CAD ground plan of the building in Fig. 27. It can be seen that they align well.

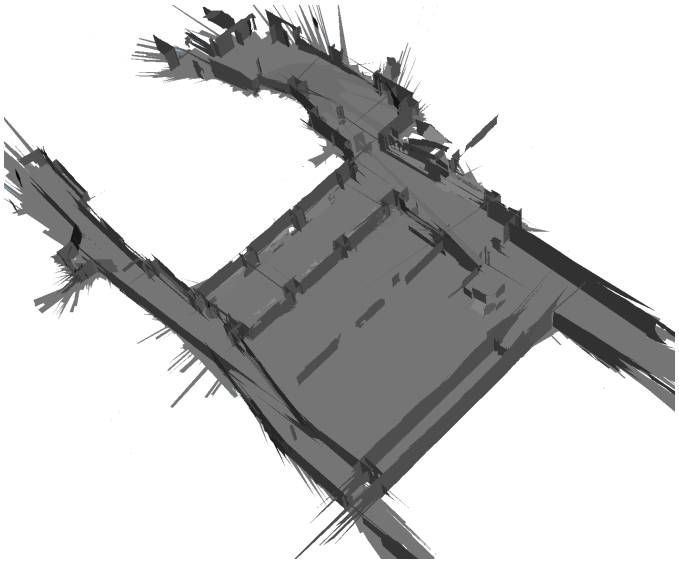

Figure 26: 3D model of the test area.

5.2.2. Outdoor experiment

Outdoor experiment was conducted using publicly available dataset from Ford Motor Company [32]. The dataset was acquired with a Ford F-250 pickup truck driving through

[5]https://youtu.be/vWoS_9wSNJw

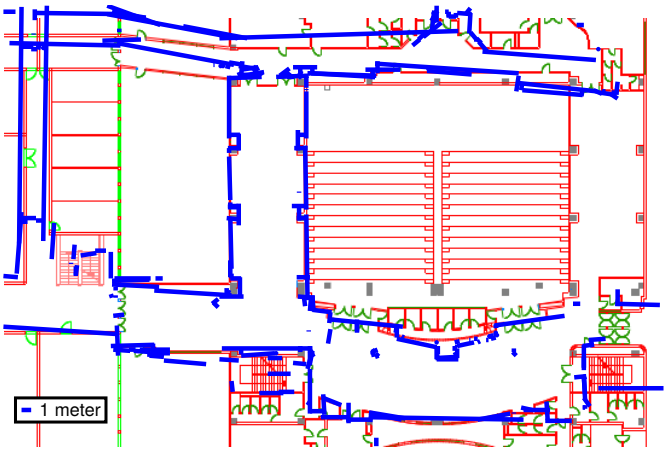

Figure 27: Ground plan from SLAM (blue) overlayed over the CAD ground plan (red).

downtown Dearborn. The vehicle was equipped with a professional (Applanix POS LV) and consumer (Xsens MTI-G) IMU, a differential GPS, a Velodyne HDL-64E 3D LIDAR, two push-broom forward looking Riegl LIDARs and a Point Grey Ladybug3 omnidirectional camera system. Velodyne HDL-64E has 64 vertical laser beams which is twice as much as Velodyne HDL-32E LIDAR used in the indoor experiment presented in Section 5.2.1. The vertical FOV of the Velodyne HDL-64E is 40°, which is the same as Velodyne HDL-32E. Since rotation rate is also 10Hz it generates double the number of points per scan. We have used the data from the differential GPS as ground truth. Odometry for prediction was acquired the same way as in the indoor experiment by fusing the data from Xsens IMU and the wheel encoders data contained within the Applanix POS LV raw sensor measurements. Figure 28 shows the test vehicle Ford F-250 equipped with sensors and sample images taken by the vehicle's camera while driving downtown Dearborn. It can be seen that there were multiple moving objects in the area as well as that the dataset was collected during daytime and represents real world scenario for urban environment. Average velocity of the vehicle was 20km/h while the maximum velocity was 45km/h. Total distance travelled was around 1.5km. Resolution of all three image planes used for projecting point cloud was set to 1024×297.

In order to show real time capability of our SLAM system even with LIDAR with higher resolution, we segmented every acquired point cloud and calculated relative poses between consecutive local maps. Figure 29 shows segmentation times for point clouds (mean/max value is around 250ms/375ms) and Figure 30 shows time required to compute relative poses (mean/max value is around 1.8ms/3ms). Segmentation time is larger than in indoor experiment since the resolution of every image plane used for projecting point cloud had to be increased in order to accommodate for higher resolution of Velodyne HDL-64E compared to Velodyne HDL-32E (64 instead of 32 verti-

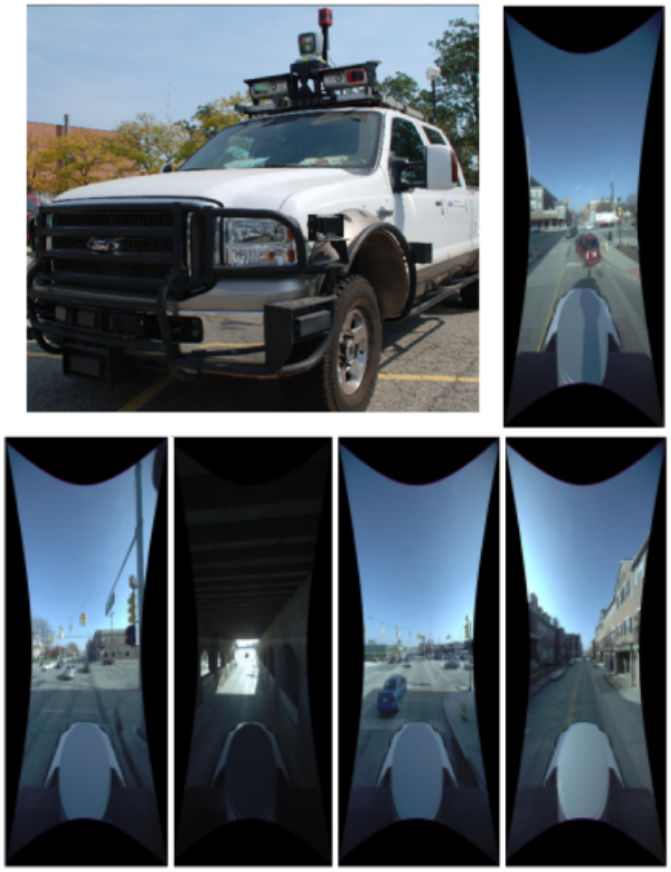

Figure 28: Vehicle used in the dataset collection and sample images taken from the environment.

cal laser beams) and to allow projection of planar surface segments from larger distances.

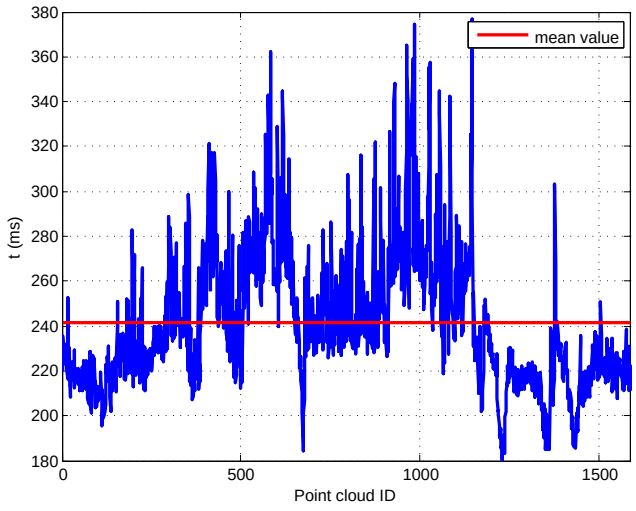

Figure 29: Point cloud segmentation time.

In total 280 states were added to the SLAM trajectory. SLAM, odometry and ground truth trajectories are shown in Fig. 31. Since loop closing is possible only at the end of the drive, first part of the trajectory does not change much because loop closing at the end has very little information gain for that part of the trajectory. Two loop closings were detected, first between states (X_{267}, X_{19}) and second between states (X_{280}, X_0). Corrections that SLAM made can be best seen on lower part of the trajectories (marked with dashes in Fig. 31) where the SLAM trajectory is much more precise than odometry trajectory and is almost identical to the ground truth trajectory. We have calculated RMS error of SLAM and odometry trajectories with the respect to the ground truth trajectory:

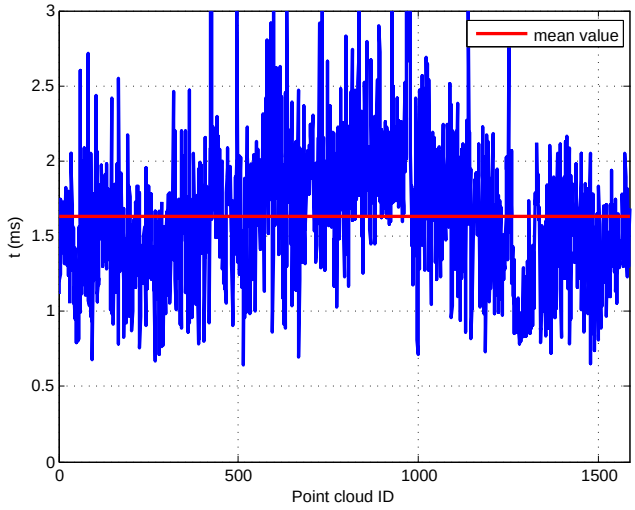

Figure 30: Relative pose computation time.

$$RMSE_{SLAM} = \sqrt{\frac{\sum_{k=1}^{n} d(X_k, GT_k)}{n}},$$
$$RMSE_{odom} = \sqrt{\frac{\sum_{k=1}^{n} d(O_k, GT_k)}{n}}, \quad (56)$$

where n is the number of states in the trajectory, $d(X_k, GT_k)$ is Euclidean distance between state X_k of the SLAM trajectory and the ground truth at time step k and $d(O_k, GT_k)$ is Euclidean distance between odometry and the ground truth trajectories at time step k. The calculated RMS errors are:

$$RMSE_{SLAM} = 4.48m \qquad RMSE_{odom} = 11.22m$$

The error of the SLAM trajectory is about 2.5 times smaller than the error of the odometry trajectory. What is more important, the final poses of the SLAM and the ground truth trajectories are almost the same, which means that if the vehicle had continued to move, its estimated pose would remain accurate whereas odometry error would increase further. This would result in even more expressed odometry versus SLAM RMS error ratio in the second lap with additional big loop closing events. Figure 32 shows the absolute error of the final odometry and the SLAM trajectories. The absolute error is calculated as Euclidean distance between the ground truth pose at time step k

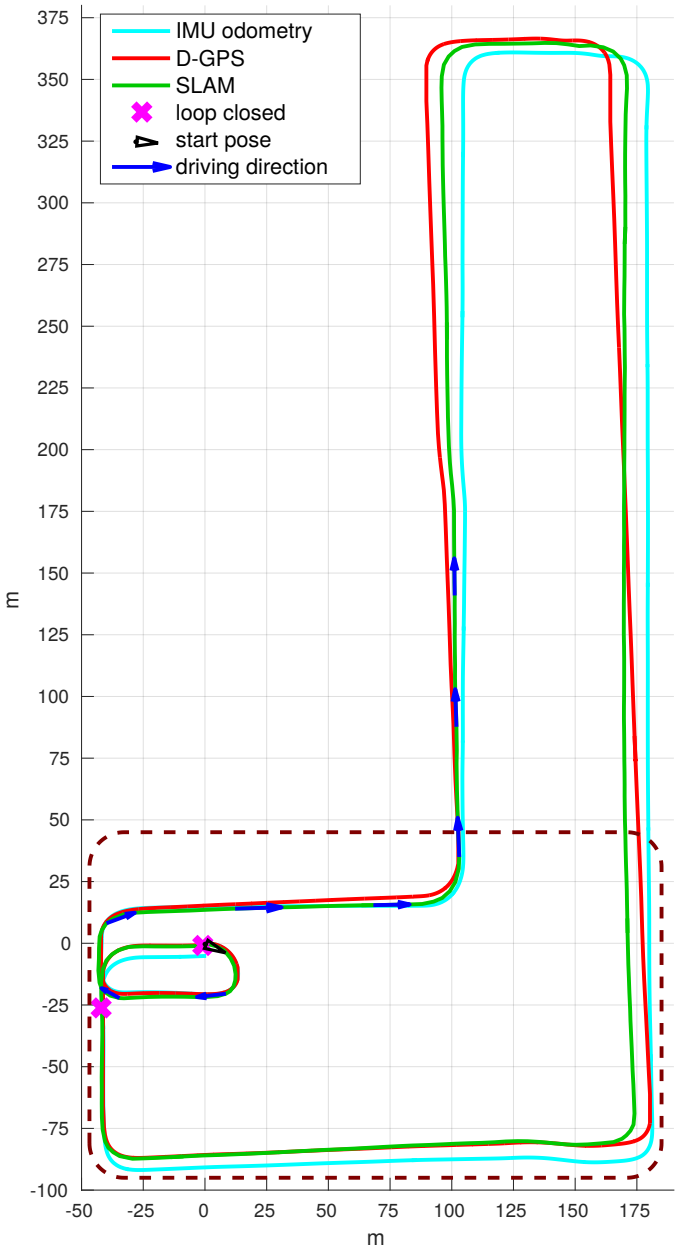

Figure 31: Comparison between odometry, ground truth and SLAM trajectory. Dashes mark the area where SLAM correction is most significant.

and SLAM state X_k (red) / odometry pose at time step k (blue). It can be seen that our SLAM system is able to recover from relatively large localization drift accumulated in large-scale environments.

The global map update time is shown in Fig. 33. We can see that the time increases as more global planes are added to the map but remains in real-time domain throughout the experiment. Spikes in the update time are due to the loop closing detections. The reasons for the residual spikes are the same as in the indoor experiment. Figure 34 shows the number of global planar surfaces compared to the number of planar surface segments within them after each map update. At the end of the experi-

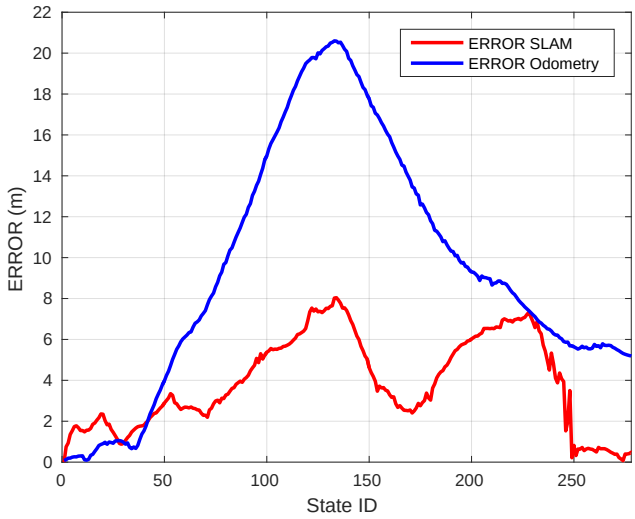

Figure 32: Absolute errors of final odometry and SLAM trajectories.

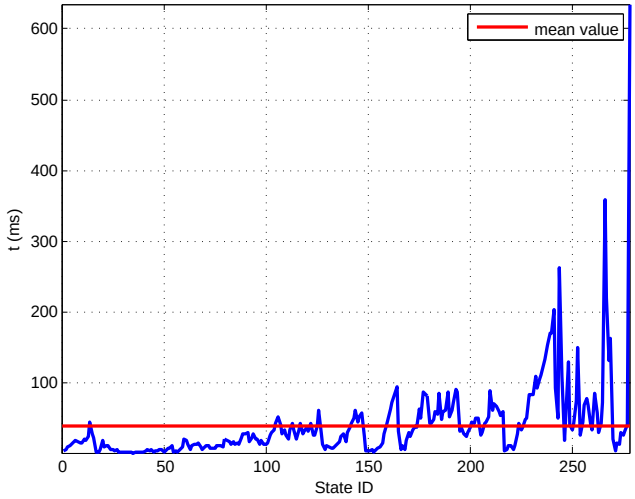

Figure 33: Global map update time time after each augmented state.

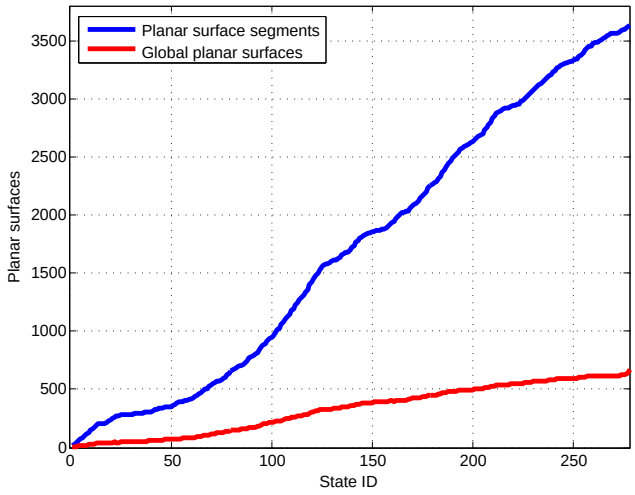

Figure 34: Total number of planar surface segments and global planar surfaces.

ment, there were 3631 planar surface segments and only 667 global planar surfaces. Our merging algorithm has reduced the number of planar surfaces in the global map by 5.45 times. Since there is no ground truth for the map in this dataset, we perform qualitative map accuracy analysis by showing our generated planar global map together with the SLAM trajectory in Fig. 35. Several areas of the global map are zoomed in for better representation. As in the indoor experiment, it can be seen that there are almost no duplicate planes. However, some of the moving objects (e.g. cars) are present in the map (marked with dotted circles) since their speed was too low when first observed and could not be differentiated from the static objects. Video of the outdoor experiment is also available online[6].

6. Conclusion

In this paper, we have presented a fast planar surface 3D SLAM solution that is designed to work on full field of view 3D point clouds obtained from 3D LIDAR measurements. There are four key improvements that allow our SLAM algorithm to work fast in large-scale environments. First is efficient processing of 3D point clouds achieved by projecting them onto 2D image planes and then performing segmentation of projected point clouds into planar surface segments. Second is the use of the information space parametrization of the SLAM filter and exploiting the sparsity of the SLAM information matrix without incurring any sparse approximation error. In that way, we can optimize the trajectory of the robot in a non-iterative way. Third is adaptation of pose constraint calculation algorithm developed in [7], which was initially intended for use with RGB-D cameras. We have modified the algorithm to respect new uncertainty model of the 3D-LIDAR and to work with planar surface segments extracted from 360° field of view. Also, it takes into account SLAM trajectory as initial guess for generating pose constraint which reduces the number of outliers and speeds up the calculation time. Fourth is improvement of the planar global map generation. Instead of simply transforming segmented planar surfaces into one coordinate frame, we have developed a new technique that combines all planar surface segments that lie on the same plane in the environment into one global planar surface whose parameters are estimated based on the uncertainty models of every planar surface segment contained within. Using this approach we have significantly reduced the number of planar surface segments in the global map since all re-observed segments and segments that belong to the same plane are represented as one global planar surface. The result is the global map which requires much less memory and consequently allows fast processing.

We have demonstrated the effectiveness of our SLAM solution on two experiments, one conducted indoors and the other outdoors, under real-world conditions. Experimental results confirmed that the proposed SLAM algorithm managed to significantly improve accuracy of vehicle trajectories as well as generate planar global map with high reduction in the number of planar surface segments compared to the total number of planar surface segments.

Acknowledgments

This work has been supported by the Unity Through Knowledge Fund under the project Cooperative Cloud based Simultaneous Localization and Mapping in Dynamic Environments. We would also like to thank professors Kaustubh Pathak, Max Folkert Pfingsthorn and Narunas Vaskevicius from Jacobs University for providing us with the source code of their method and helping us with running it on our machine.

[6] https://youtu.be/HboixGB2umY

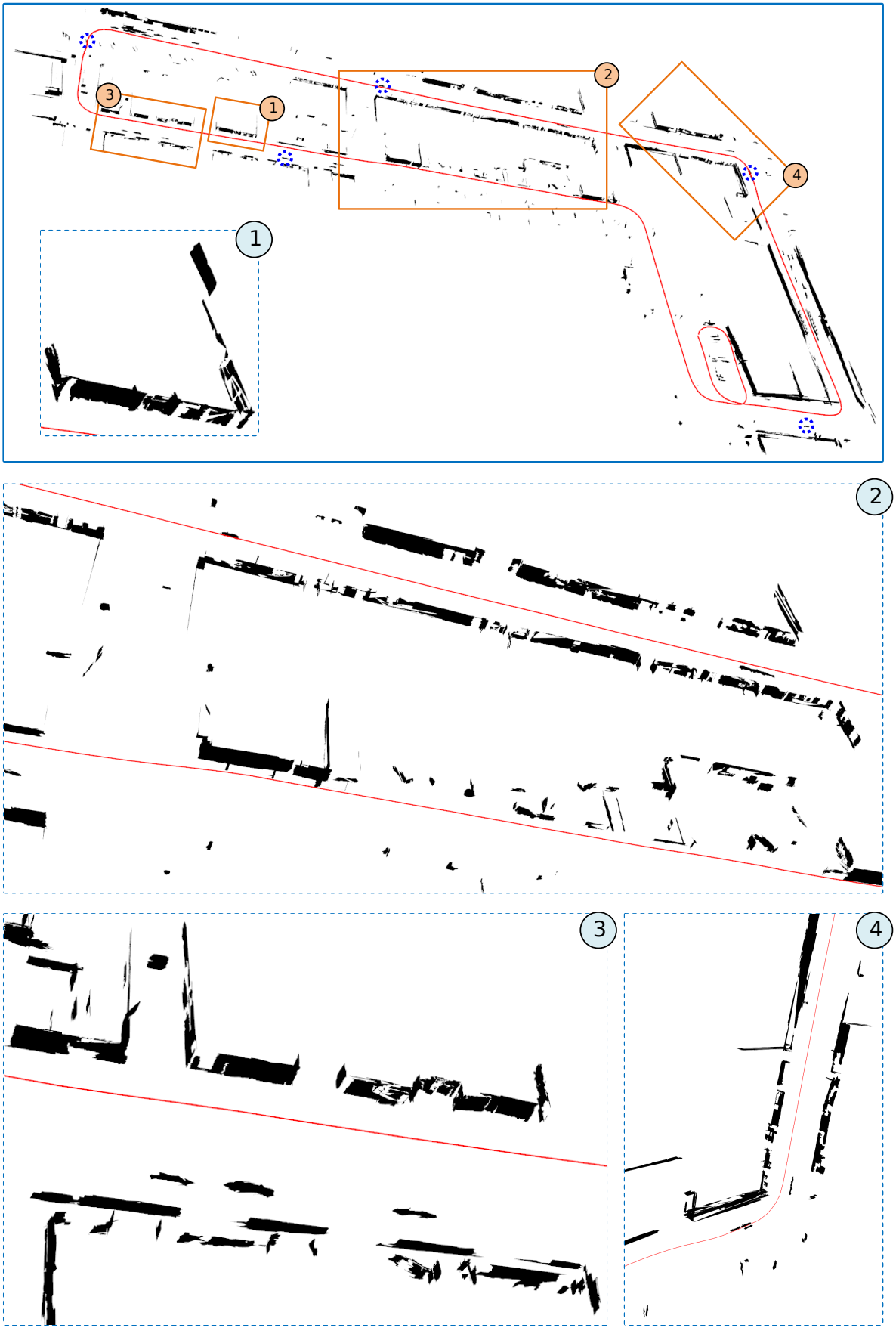

Figure 35: Global map generated from Ford dataset. Dotted circles represent moving objects present in the map.

References

[1] A. Nuchter, K. Lingemann, J. Hertzberg, 6D SLAM-3D Mapping Outdoor Environments, Journal of Field Robotics 24 (2007) 699–722. doi:10.1002/rob.20209.

[2] H. Surmann, A. Nüchter, J. Hertzberg, An autonomous mobile robot with a 3D laser range finder for 3D exploration and digitalization of indoor environments, Robotics and Autonomous Systems 45 (3-4) (2003) 181–198. doi:10.1016/j.robot.2003.09.004.

[3] S. Thrun, D. Hähnel, D. Ferguson, M. Montemerlo, R. Triebel, W. Burgard, C. Baker, Z. Omohundro, S. Thayer, W. Whittaker, A System for Volumetric Robotic Mapping of Abandoned Mines, Proceedings of the IEEE International Conference on Robotics and Automation (ICRA).

[4] P. Besl, N. 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.

[5] K. Pathak, A. Birk, N. Vaskevicius, M. Pfingsthor, S. Schwertfeger, Jann Poppinga, Online 3D SLAM by Registration of Large Planar Surface Segments and Closed Form Pose-Graph Relaxation, Journal of Field Robotics 27 (1) (2010) 52–84.

[6] R. Cupec, E. K. Nyarko, D. Filko, I. Petrović, Fast pose tracking based on ranked 3D planar patch correspondences, in: IFAC Proceedings Volumes (IFAC-PapersOnline), 2012, pp. 108–113.

[7] R. Cupec, E. K. Nyarko, D. Filko, A. Kitanov, I. Petrović, Place Recognition Based on Matching of Planar Surfaces and Line Segments, The International Journal of Robotics Research 34 (4-5) (2015) 674–704. doi:10.1177/0278364914548708.

[8] J. Xiaoa, J. Zhangb, B. Adlera, H. Zhangc, J. Zhanga, Threedimensional point cloud plane segmentation in both structured and unstructured environments, Robotics and Autonomous Systems 61 (12) (2013) 1641–1652.

[9] J. Xiao, B. Adler, J. Zhang, H. Zhang, Planar Segment Based Threedimensional Point Cloud Registration in Outdoor Environments., Journal of Field Robotics 30 (4) (2013) 552–582.

[10] J. Weingarten, R. Siegwart, 3D SLAM using planar segments, IEEE International Conference on Intelligent Robots and Systems (2006) 3062–3067doi:10.1109/IROS.2006.282245.

[11] P. Kohlhepp, P. Pozzo, M. Walther, R. Dillmann, Sequential 3D-SLAM for mobile action planning, 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566) 1 (2004) 722–729. doi:10.1109/IROS.2004.1389438.

[12] P. de la Puente, D. Rodriguez-Losada, Feature based graph {SLAM} with high level representation using rectangles, Robotics and Autonomous Systems 63, Part 1 (2015) 80 – 88. doi:http://dx.doi.org/10.1016/j.robot.2014.09.006.

[13] R. a. Newcombe, D. Molyneaux, D. Kim, A. J. Davison, J. Shotton, S. Hodges, A. Fitzgibbon, KinectFusion: Real-Time Dense Surface Mapping and Tracking, IEEE International Symposium on Mixed and Augmented Reality (2011) 127–136doi:10.1109/ISMAR.2011.6092378.

[14] R. F. Salas-Moreno, R. a. Newcombe, H. Strasdat, P. H. J. Kelly, A. J. Davison, SLAM++: Simultaneous localisation and mapping at the level of objects, Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (2013) 1352–1359doi:10.1109/CVPR.2013.178.

[15] R. F. Salas-Moreno, B. Glocken, P. H. J. Kelly, A. J. Davison, Dense Planar SLAM, Proc. of ISMAR (2014) 157–164doi:10.1109/ISMAR.2014.6948422.

[16] X. Gao, T. Zhang, Robust rgb-d simultaneous localization and mapping using planar point features, Robotics and Autonomous Systems 72 (2015) 1 – 14. doi:http://dx.doi.org/10.1016/j.robot.2015.03.007.

[17] T. Whelan, M. Kaess, H. Johannsson, M. Fallon, J. J. Leonard, J. Mcdonald, Real-time large-scale dense rgb-d slam with volumetric fusion, Int. J. Rob. Res. 34 (4-5) (2015) 598–626.
URL http://dx.doi.org/10.1177/0278364914551008

[18] M. Kaess, Simultaneous localization and mapping with infinite planes, in: IEEE Intl. Conf. on Robotics and Automation, ICRA, Seattle, WA, 2015, pp. 4605–4611.

[19] A. J. B. Trevor, J. G. Rogers, H. I. Christensen, Planar surface SLAM with 3D and 2D sensors, Proceedings - IEEE International Conference on Robotics and Automation (2012) 3041–3048doi:10.1109/ICRA.2012.6225287.

[20] F. Schmitt, X. Chen, Fast segmentation of range imagery into planar regions, Computer Vision, Graphics and Image Processing 45 (1) (1991) 42–60. doi:10.1016/0734-189X(89)90069-8.

[21] R. Cupec, E. K. Nyarko, D. Filko, A. Kitanov, I. Petrović, Global Localization Based on 3D Planar Surface Segments Detected by a 3D Camera, Proceedings of the Croatian Computer Vision Workshop, Year 1 (2013) 31–36.

[22] A. Kitanov, I. Petrović, Exactly sparse delayed state filter based robust {SLAM} with stereo vision, in: Robotics (ISR), 2010 41st International Symposium on and 2010 6th German Conference on Robotics (ROBOTIK), VDE, 2010, pp. 1 – 7.

[23] R. M. Eustice, H. Singh, J. J. Leonard, Exactly sparse delayed-state filters for view-based slam, IEEE Transactions on Robotics 22 (6) (2006) 1100–1114. doi:10.1109/TRO.2006.886264.

[24] C. Stachniss, D. Hahnel, W. Burgard, Exploration with active loop-closing for FastSLAM, 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566) 2 (2004) 1505–1510. doi:10.1109/IROS.2004.1389609.

[25] Boost, Boost library, http://www.boost.org (2016).
URL www.boost.org

[26] F. Pomerleau, F. Colas, R. Siegwart, S. Magnenat, Comparing ICP variants on real-world data sets, Autonomous Robots 34 (3) (2013) 133–148. doi:10.1007/s10514-013-9327-2.
URL http://dx.doi.org/10.1007/s10514-013-9327-2

[27] F. Pomerleau, M. Liu, F. Colas, R. Siegwart, Challenging data sets for point cloud registration algorithms, The International Journal of Robotics Research 31 (14) (2012) 1705–1711.

[28] M. Magnusson, N. Vaskevicius, T. Stoyanov, K. Pathak, A. Birk, Beyond points: Evaluating recent 3d scan-matching algorithms, in: International Conference on Robotics and Automation (ICRA), 2015.

[29] Y. Chen, G. Medioni, Object modelling by registration of multiple range images, Image and Vision Computing 10 (3) (1992) 145–155. doi:10.1016/0262-8856(92)90066-C.
URL http://dx.doi.org/10.1016/0262-8856(92)90066-C

[30] M. Magnusson, A. Lilienthal, T. Duckett, Scan Registration for Autonomous Mining Vehicles Using 3D-NDT, Journal of Field Robotics 24 (10) (2007) 803–827. doi:10.1002/rob.

[31] T. Stoyanov, M. Magnusson, H. Andreasson, A. J. Lilienthal, Fast and accurate scan registration through minimization of the distance between compact 3d ndt representations, The International Journal of Robotics Research 31 (12) (2012) 1377–1393. doi:10.1177/0278364912460895.

[32] G. Pandey, J. R. McBride, R. M. Eustice, Ford Campus vision and lidar data set, The International Journal of Robotics Research 30 (13) (2011) 1543–1552. doi:10.1177/0278364911400640.

Appendix A: ESDSF equations for the operations of state augmentation, time prediction and measurement update

The derivation of Exactly Sparse Delayed State Filter (ESDSF) is given in [23]. Here, we provide just final expressions with review how they impact the estimation process in SLAM regarding computational complexity.

In ESDSF, delayed states of the system trajectory $X = [X_0\ X_1 \dots X_n]$ are estimated using Extended Information Kalman filter. Let a system state transition model be

$$X_{n+1} = f(X_n, u_n) + w_n, w_n \sim\ \mathcal{N}(0, Q), \tag{57}$$

and a measurement model

$$y_n = h(X_i, X_j) + v_n, v_n \sim \mathcal{N}(0, R). \tag{58}$$

If we denote estimated pose in time step n by X_n, and all other previous poses with m, i.e. $m = [X_{n-1}, \dots, X_0]^T$ then we can partition the joint probability distribution of X_n and m given a history of measurements y^n and control inputs u^n up to time step n in the fallowing way

$$\begin{aligned} p(X_n, m|y^t, u^t) &= \mathcal{N}\left[\begin{pmatrix} \mu_{X_n} \\ \mu_m \end{pmatrix}, \begin{pmatrix} \Sigma_{X_n} & \Sigma_{X_n m} \\ \Sigma_{m X_n} & \Sigma_{mm} \end{pmatrix}\right] \\ &= \mathcal{N}^{-1}\left[\begin{pmatrix} \eta_{X_n} \\ \eta_m \end{pmatrix}, \begin{pmatrix} \Lambda_{X_n X_n} & \Lambda_{X_n m} \\ \Lambda_{m X_n} & \Lambda_{mm} \end{pmatrix}\right]. \end{aligned} \tag{59}$$

State augmentation acts on a distribution with the 1st order Markov process as

$$\begin{aligned} p(X_{n+1}, X_n, m|y^n, u^{n+1}) &= p(X_{n+1}|X_n, m, y^n, u^{n+1})\ p(X_n, m|y^n, u^{n+1}) \\ &= p(X_{n+1}|X_n, u_{n+1})\ p(X_n, m|y^n, u^n). \end{aligned} \tag{60}$$

If we denote by F Jacobian of the motion model (57) with respect to the state variables in expected value of states μ_X, augmentation yields

$$p(X_{n+1}, X_n, m|\, y^n, u^{n+1}) = \mathcal{N}^{-1}(\eta'_{n+1}, \Lambda'_{n+1}), \tag{61}$$

where

$$\eta'_{n+1} = \begin{pmatrix} Q^{-1}(f(\mu_{X_n}, u_{n+1}) - F\mu_{X_n}) \\ \eta_{X_n} - F^T Q^{-1}(f(\mu_{X_n}, u_{n+1}) - \mu_{X_n}) \\ \eta_m \end{pmatrix} \tag{62}$$

and

$$\Lambda'_{n+1} = \begin{pmatrix} \boxed{Q^{-1}} & \boxed{-Q^{-1}F} & 0 \\ \boxed{-F^T Q^{-1}} & \Lambda_{X_n X_n} + F^T Q^{-1} F & \Lambda_{X_n m} \\ 0 & \Lambda_{m X_n} & \Lambda_{mm} \end{pmatrix}. \tag{63}$$

We can see that by augmentation of the state trajectory, information matrix Λ exhibits tridiagonal form as only boxed block elements can be non-zero in (63) of all added rows and columns.

Motion prediction can be defined as augmentation with X_{n+1} and marginalization by X_n

$$\begin{aligned} p(X_{n+1}, m|y^n, u^{n+1}) &= \mathcal{N}(\bar{\mu}_{n+1}, \bar{\Sigma}_{n+1}) = \mathcal{N}^{-1}(\bar{\eta}_{n+1}, \bar{\Lambda}_{n+1}) \\ &= \int p(X_{n+1}, X_n, m|y^n, u^{n+1}) dX_n, \end{aligned} \tag{64}$$

$$\begin{aligned} \bar{\eta}_{n+1} &= \begin{pmatrix} Q^{-1}(f(\mu_{X_n}, u_{n+1}) - F\mu_{X_n}) \\ \eta_m \end{pmatrix} \\ &\quad - \begin{pmatrix} -Q^{-1}F \\ \Lambda_{m X_n} \end{pmatrix} \Omega^{-1}(\eta_{X_n} - F^T Q^{-1}(f(\mu_{X_n}, u_{n+1}) - F\mu_{X_n}) \\ &= \begin{pmatrix} Q^{-1}F\Omega^{-1}\eta_{X_n} + \Psi(f(\mu_{X_n}, u_{n+1}) - F\mu_{X_n}) \\ \eta_m - \Lambda_{m X_n}(\eta_{X_n} - F^T Q^{-1}(f(\mu_{X_n}, u_{n+1}) - F\mu_{X_n})) \end{pmatrix} \end{aligned} \tag{65}$$

and

$$\begin{aligned}\bar{\Lambda}_{n+1} &= \begin{pmatrix} Q^{-1} & 0 \\ 0 & \Lambda_{mm} \end{pmatrix} - \begin{pmatrix} -Q^{-1}F \\ \Lambda_{mX_n} \end{pmatrix} \Omega^{-1} \begin{pmatrix} -F^T Q^{-1} & \Lambda_{X_n m} \end{pmatrix} \\ &= \begin{pmatrix} \Psi & Q^{-1}F\Omega^{-1}\Lambda_{X_n m} \\ \Lambda_{mX_n}\Omega^{-1}F^T Q^{-1} & \Lambda_{mm} - \Lambda_{mX_n}\Omega^{-1}\Lambda_{X_n m} \end{pmatrix} \end{aligned} \tag{66}$$

where

$$\begin{aligned}\Psi =& Q^{-1} - Q^{-1}F(\Lambda_{X_n X_n} + F^T Q^{-1}F)^{-1}F^T Q^{-1} = (Q + F\Lambda^{-1}_{X_n X_n}F^T)^{-1} \\ \Omega =& (\Lambda_{X_n X_n} + F^T Q^{-1}F).\end{aligned}$$

Since X_n is only serially connected to X_{n+1} and X_{n-1}, marginalizing it out only requires modifying the information blocks associated with these elements, i.e. $\Lambda_{X_{n+1}X_{n+1}}, \Lambda_{X_{n+1}X_{n-1}}, \Lambda_{X_{n-1}X_{n-1}}$, and $\Lambda_{X_{n-1}X_{n+1}}$.

The measurement Jacobian with respect to the states has the form

$$H = \begin{pmatrix} 0 & \dots & \frac{\partial y_n}{\partial X_i} & \dots & 0 & \dots & \frac{\partial y_n}{\partial X_j} & \dots & 0 \end{pmatrix}. \tag{67}$$

Therefore, measurement update is constant time in ESDSF as it affects only blocks sharing information associated to X_i and X_j

$$\eta_n = \bar{\eta}_n + H^T R^{-1}(y_n - h(\bar{\mu}_{X_n}) + H\bar{\mu}_{X_n}) \tag{68}$$

$$\Lambda_n = \bar{\Lambda}_n + H^T R^{-1} H. \tag{69}$$

Appendix B: Planar surface segment covariance transformation

The parameter uncertainties of planar surface segment $F_{i,m}$ are transformed from $S_{F_{i,m}}$ into local coordinate frame $S_{F_{j,n}}$ of planar surface segment $F_{j,m}$ as (matrices E, $\Sigma_{q_{i,m}}$, C and $P_{n,i}$ are defined in Section 4.4.1).

$$\tilde{\Sigma}_{q_{j,n}} = E\Sigma_{q_{i,m}}E^T + CP_{n,i}C^T \tag{70}$$

In general case, given surface segment pair ($F_{i,m}$, $F_{j,n}$), their perturbation vectors ($q_{i,m}$, q_{jn}) and rotation matrix $R_{m,n}$ and translation vector $t_{m,n}$ between their local maps M_m and M_n we can define a non-linear function h that transforms parameters of $F_{j,n}$ into $S_{F_{i,m}}$:

$$h = \begin{bmatrix} \begin{bmatrix} {}^F x_{j,n}^T \\ {}^F y_{j,n}^T \end{bmatrix} R_{m,n} \frac{{}^F z_{i,m} + \begin{bmatrix} {}^F x_{i,m} & {}^F y_{i,m} \end{bmatrix} s_{i,m}}{\sqrt{1 + s_{i,m}^T s_{i,m}}} - \frac{s_{j,n}}{\sqrt{1 + s_{j,n}^T s_{j,n}}} \\ r_{i,m} + ({}^F t_{i,m}^T - ({}^F t_{j,n} - t_{m,n})^T R_{m,n}) \frac{{}^F z_{i,m} + \begin{bmatrix} {}^F x_{i,m} & {}^F y_{i,m} \end{bmatrix} s_{i,m}}{\sqrt{1 + s_{i,m}^T s_{i,m}}} - r_{j,n} \end{bmatrix} \tag{71}$$

where first two rows represent transformation of the x, y coordinates of the normal ${}^F n_{i,m}$, third row represents transformation of the distance ${}^F \rho_{i,m}$, vector $s = [s_x \; s_y]$ is a part of perturbation vector q describing uncertainties of ${}^F n$ in S_F, r represents uncertainty ${}^F \rho$, and vectors ${}^F x$, ${}^F y$ and ${}^F z$ represent columns of the rotation matrices:

$${}^F R_{i,m} = \begin{bmatrix} {}^F x_{i,m} & {}^F y_{i,m} & {}^F z_{i,m} \end{bmatrix} \quad {}^F R_{j,n} = \begin{bmatrix} {}^F x_{j,n} & {}^F y_{j,n} & {}^F z_{j,n} \end{bmatrix}. \tag{72}$$

In order to get expected values of the transformed parameters e defined in equation (37) we evaluate h for $s = 0$ and $r = 0$

$$e = h|_{s=0, r=0} = \begin{bmatrix} \begin{bmatrix} {}^F x_{j,n}^T \\ {}^F y_{j,n}^T \end{bmatrix} R_{m,n} {}^F z_{i,m} \\ ({}^F t_{i,m}^T - ({}^F t_{j,n} - t_{m,n})^T R_{m,n}) {}^F z_{i,m} \end{bmatrix} \tag{73}$$

Jacobian matrix C of the function h given with (71) can be calculated as

$$C = \frac{\partial h}{\partial w}\Big|_{q_{i,m}=0, q_{j,n}=0} = \begin{bmatrix} \begin{bmatrix} {}^F x_{j,n}^T \\ {}^F y_{j,n}^T \end{bmatrix} J_\phi(\phi, {}_z^F\, j, n) & 0^{2\times 3} \\ -({}^F t_{j,n} - t_{m,n})^T J_\phi(\phi, {}_z^F\, j, n) & {}^F z_{j,n}^T R_{m,n}^T \end{bmatrix} \tag{74}$$

where vector $w = [\phi_{m,n}\ t_{m,n}]$, vector $\phi_{m,n} = [\alpha\ \beta\ \Theta]$ represents Euler angles corresponding to rotation matrix $R_{m,n}$, and Jacobian $J_\phi(\phi, {}^F z_{j,n})$ is calculated as

$$J_\phi(\phi, p) = \frac{\partial(R(\psi)p)}{\partial \psi}|_{\psi=\phi} \tag{75}$$

Calculation of matrix E from the function h is done using the following equation

$$E = \frac{\partial h}{\partial q}\Big|_{q_{i,m}=0, q_{j,n}=0} = \begin{bmatrix} \begin{bmatrix} {}^F x_{j,n}^T \\ {}^F y_{j,n}^T \end{bmatrix} R_{m,n} \begin{bmatrix} {}^F x_{i,m}^T \\ {}^F y_{i,m}^T \end{bmatrix} & 0 \\ ({}^F t_{i,m}^T - ({}^F t_{j,n} - t_{m,n})^T R_{m,n}) \begin{bmatrix} {}^F x_{i,m}^T \\ {}^F y_{i,m}^T \end{bmatrix} & 1 \end{bmatrix} \tag{76}$$