

Article

3D Scene Reconstruction Using Omnidirectional Vision and LiDAR: A Hybrid Approach

Michiel Vlamincx ^{1,*}, Hiep Luong ¹, Werner Goeman ² and Wilfried Philips ¹

¹ Department of Telecommunications and Information Processing, Ghent University, Sint-Pietersnieuwstraat 41, iMinds, Ghent 9000, Belgium; hiep.luong@telin.ugent.be (H.L.); philips@telin.ugent.be (W.P.)

² Sweco/Grontmij, Ghent 9000, Belgium; werner.goeman@grontmij.be

* Correspondence: michiel.vlaminck@telin.ugent.be; Tel.: +32-473-413-613

Academic Editors: Gabriel Oliver-Codina, Nuno Gracias and Antonio M. López

Received: 13 September 2016; Accepted: 8 November 2016; Published: 16 November 2016

Abstract: In this paper, we propose a novel approach to obtain accurate 3D reconstructions of large-scale environments by means of a mobile acquisition platform. The system incorporates a Velodyne LiDAR scanner, as well as a Point Grey Ladybug panoramic camera system. It was designed with genericity in mind, and hence, it does not make any assumption about the scene or about the sensor set-up. The main novelty of this work is that the proposed LiDAR mapping approach deals explicitly with the inhomogeneous density of point clouds produced by LiDAR scanners. To this end, we keep track of a global 3D map of the environment, which is continuously improved and refined by means of a surface reconstruction technique. Moreover, we perform surface analysis on consecutive generated point clouds in order to assure a perfect alignment with the global 3D map. In order to cope with drift, the system incorporates loop closure by determining the pose error and propagating it back in the pose graph. Our algorithm was exhaustively tested on data captured at a conference building, a university campus and an industrial site of a chemical company. Experiments demonstrate that it is capable of generating highly accurate 3D maps in very challenging environments. We can state that the average distance of corresponding point pairs between the ground truth and estimated point cloud approximates one centimeter for an area covering approximately 4000 m². To prove the genericity of the system, it was tested on the well-known Kitti vision benchmark. The results show that our approach competes with state of the art methods without making any additional assumptions.

Keywords: 3D point cloud registration; Iterative Closest Point (ICP); LiDAR scanning; loop closure; surface reconstruction; Velodyne; Ladybug

1. Introduction

Reconstructing the 3D scene using a mobile observer, also referred to as mobile mapping, is used in a variety of applications ranging from navigational tasks for autonomous vehicles to facility condition assessment or cartography. The former application imposes a real-time constraint on the processing time of the used algorithms, whereas the latter allows the system to take more computation time. The application that we have in mind aims at construction site monitoring. The goal there is to obtain a 3D model of the environment in a reasonable amount of time, which spans the duration of the intervention. To this end, we propose a system that combines both a LiDAR sensor (Velodyne HDL32-e) and a panoramic camera system (Ladybug 3) to perceive the environment. Some systems exist that rely on the output of a regular camera to obtain 3D reconstructions [1,2], known in the literature as (visual) structure from motion (SfM). In the case of sequential (or incremental) SfM, the problem is also referred to as simultaneous localization and mapping (SLAM). However, visual SfM produces rather sparse 3D reconstructions, which require much post-processing to make them more

dense. Moreover, they are prone to drift and tend to fail when the images are overexposed because of the abundance of sunlight. For this reason, the core of our mobile mapping system is based on the output of a LiDAR sensor, aided by images of a regular camera to perform loop detection. Often times, mobile mapping systems are also relying on GPS to estimate the trajectory of the observer. However, in the environments we have in mind, i.e., construction sites with many overhanging or protruding structures (e.g., pipelines), the GPS signal is too weak. In closed indoor environments, it is even completely lacking. Therefore, we developed a system that is fully GPS-independent. In summary, we propose an entire 3D reconstruction system that covers both the odometry and mapping problem. It incorporates the alignment of consecutive point clouds, the fusion with a global 3D map and a loop closure mechanism to cope with drift. The main goal of our approach is to make the system as generic as possible and, hence, to make as little assumptions as possible about the environment and about the sensor set-up. Doing so, we are able to use our system in combination with a drone-mounted LiDAR sensor and camera in the future. The main contribution of this work is that our solution deals explicitly with the inhomogeneous density of point clouds produced by LiDAR scanners. This latter refers to the fact that some parts of the point cloud have a higher point density than others due to the inclination of the laser beams and the proximity of the objects present in the scene. Generally, regions close to the sensor origin will be densely sampled, whereas more distant objects will be sparsely sampled. As has been shown in numerous studies [3–5], traditional point-to-point registration techniques (e.g., Iterative Closest Point (ICP)) fail in these situations. For that reason, we propose a number of improvements over standard point-to-point strategies and incorporate them in one framework. The main contributions can be summarized as follows:

1. We propose a surface analysis technique that is used to build a topological space on top of the point cloud, providing for each point its ideal neighborhood and taking into account the underlying surface.
2. We keep track of a global 3D map that is continuously updated by means of a surface reconstruction technique that allows us to resample the point cloud. This way, the global 3D map is continuously improved, i.e., the noise is reduced, and the alignment of the following point clouds can be conducted more accurately.
3. The topological space is used to compute low-level features, which will be incorporated in an adapted version of the ICP algorithm to make this latter more robust. The alignment of consecutive local point clouds, as well as the alignment of a local point cloud with the global 3D map will be conducted using this improved ICP algorithm.
4. We incorporate the residual of the ICP process in the loop closure process. Based on this residual, we can predict the share of each pose estimation in the final pose error when a loop has been detected.

2. Related Work

Regarding LiDAR odometry and mapping, existing solutions can roughly be divided in four main classes: based on ICP, normal distribution transform (NDT), features and planar surfaces. The majority of the approaches presented in the literature are based on the iterative closest point (ICP) algorithm in order to perform scan matching and to identify the transformation between sensor poses [6–8]. However, the standard ICP algorithm has many limitations, and therefore, many improvements have been presented in literature. One of the main drawbacks lies in the fact that it is often difficult to find good point to point correspondences between two point clouds, especially in the case of a sparse point density or when the speed of the moving platform is too high. The authors of velocity ICP [9] therefore propose to perform velocity estimation over the ICP iterations. Doing so, the distortion of a scan due to motion can be compensated by re-projecting all points acquired during one sweep to the position of the sensor at the beginning of the sweep. In [10], the authors tackle the correspondence problem by defining low-level features that are used to guide the ICP process. The authors focus on

pair-wise registration of datasets that do not exhibit large changes. They do not investigate a full sequential 3D reconstruction approach that incorporates a global 3D map or loop closure. The authors of Velodyne SLAM [11], on the other hand, incrementally build a map of the environment by aligning each newly arrived Velodyne scan with the whole map using ICP. Their global map is improved over time by adapting incoming measurements according to already existing adapted neighbors. However, the refinement of the global map is limited as they do not perform a surface analysis or re-sampling of the point cloud. Moreover, they do not incorporate a loop closure mechanism to cope with drift. Finally, some studies focus on combining the ICP-based scan matching with (stereo) visual odometry. For example, in [12], the authors propose a solution that uses the output of visual odometry to obtain a rough estimate of the ego motion upon which consecutive point clouds are registered. The generalized ICP algorithm [13] is then used to refine the motion estimation, and finally, the output is combined with reduced IMU outputs. Although these latter solutions seem effective, these studies, in contrast to ours, do not investigate how LiDAR odometry itself can be improved.

Aside from ICP, the normal distribution transform (NDT) [14] is another common technique that is often used for laser scan matching [15–17]. It is based on an alternative representation of a point cloud similar to an occupancy grid. In each 3D cell, a normal distribution is stored, and both scans are matched using Newton's method. Compared to ICP-based methods, the main advantage of this method is that there is no need for correspondence estimation. The main limitation is however that the accuracy of NDT is strongly related to the cell size, which is difficult to define in the case of inhomogeneous point clouds.

Another way to cope with correspondence estimation is to introduce the use of features. Feature-based methods have significant advantages as they concentrate on strong cues, such as corners and lines and filter out irregular points, such as vegetation or tree leaves. One very interesting feature-based system has been developed by Zhang et al. and described in [18,19]. Their technique is based on the extraction of only two different shape features, representing sharp edges and planar surface patches. Edge points of the source cloud are associated with edge lines, while planar patches are matched with other planar patches in the target cloud. In [19], the same authors elaborated on their method by incorporating data from a regular camera to perform visual odometry.

Finally, in [20–22], the authors use yet another approach based on the registration of planar surfaces. These planes can lead to very accurate alignments as they serve as strong cues and can be estimated by an accumulation of data, hence reducing the noise. However, although these methods are very effective, the lack of planes in the scene can cause them to fail and, hence, make them less generic.

In this work, we will adopt a hybrid approach in the sense that we combine ICP-based registration with feature extraction and surface reconstruction. To this end, we adopt the strategy of [10] by incorporating low-level features that describe the underlying surface in a simple way, i.e., using three main classes, linear, planar and volumetric. We exploit these properties to identify for each point an ideal neighborhood and, hence, a topological space representing the captured scene. The low-level features will further be used to guide the ICP process in such a way that the alignment concentrates itself near the most reliable regions, i.e., lines and corners instead of irregular points or clutter. Our approach can benefit from large planar surfaces while at the same time remaining generic. It is thus more widely applicable. In addition, we keep track of a global map, which will continuously be improved and refined with newly-captured points. More specifically, we use a surface reconstruction technique after which we resample the point cloud and improve its topological space. Finally, we incorporate loop closure to cope with the remaining drift.

3. System

3.1. Acquisition Platform

Our acquisition platform consists of a Velodyne High Definition LiDAR (HDL-32e) scanner combined with a Ladybug panoramic camera system. Because of this combination, we named it the Vellady platform. As can be seen in Figure 1, the Ladybug camera is mounted perpendicular to the

ground plane, whereas the Velodyne is tilted on its head, making an angle of approximately 66° with the ground plane. The Velodyne LiDAR scanner is equipped with 32 lasers mounted collinear and covering a vertical FOV of 41.3° , hence resulting in a vertical resolution of 1.29° . The vertical FOV covers 30.67° below the middle point of the Velodyne and 10.67° above it. The head is continuously spinning at approximately 10 Hz, resulting in a horizontal FOV of 360° . Figure 2 shows an example of a point cloud obtained by this Velodyne sensor. The Ladybug on the other hand is a fixed system that comes in the form of a pentagonal prism consisting of five vertical-oriented sides, each one incorporating a camera. A sixth camera is mounted on top pointing upwards. It is shooting images at one frame per second. A stitched image obtained by this Ladybug camera system, corresponding with the point cloud of Figure 2, is depicted in Figure 3. We mounted this Vellady platform on a moving vehicle, i.e., a kitchen cart or four-wheel trolley (see Figure 1). Our platform is hence not able to rotate around its roll angle. A rotation about its pitch angle is only possible in the case that the ground plane has a slope, but these situations were not encountered during the experiments. However, we did not exploit these facts as we want to retain the option to mount the platform on a drone in the future. Our system thus operates as a full six degrees of freedom SLAM algorithm incorporating three unknown position coordinates x, y, z and three rotation angles $\theta_x, \theta_y, \theta_z$. The main features of our platform are summarized in Table 1.

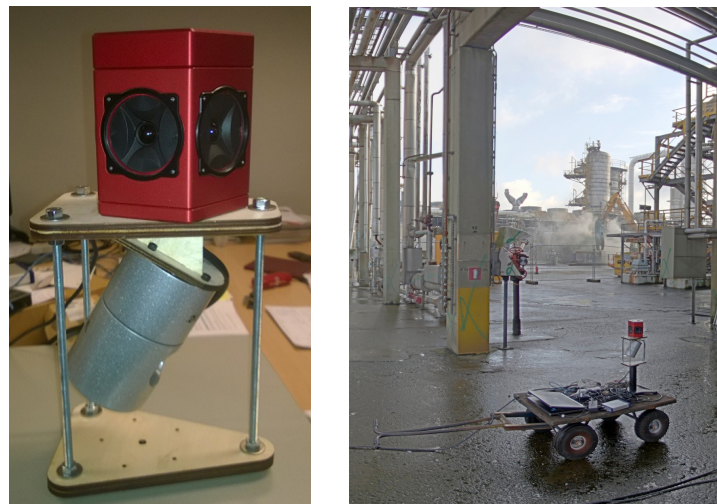


Figure 1. Two pictures of our mobile acquisition platform Vellady mounted on a kitchen cart (**right**) and close-up (**left**). The platform consists of a Ladybug panoramic camera (mounted at the top) and a Velodyne HDL32-e LiDAR scanner (mounted at the bottom).

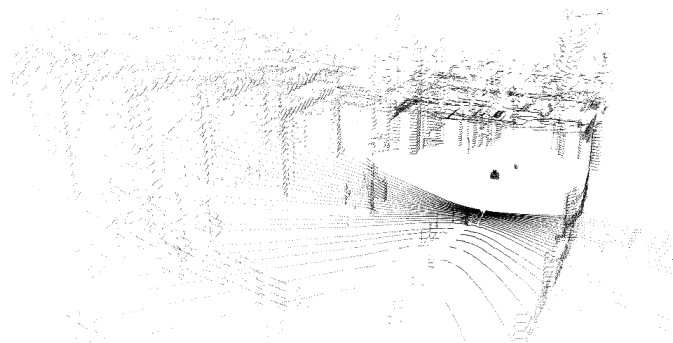


Figure 2. A 360° point cloud acquired by the Velodyne HDL-32e LiDAR scanner, associated with the stitched lady bug image of Figure 3, captured at the Dow chemical company.

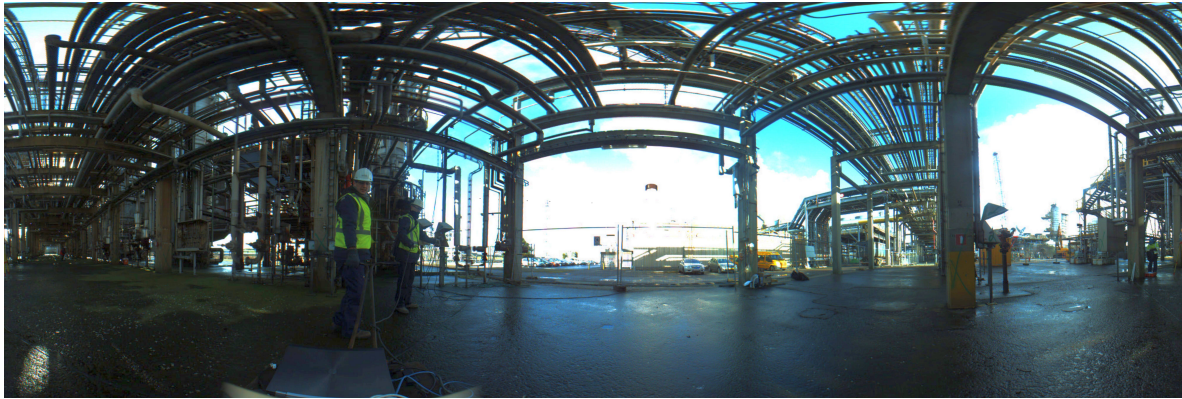




Figure 3. An image acquired by stitching the six images of the Ladybug together. The image shows the starting point of a video sequence captured at the Dow chemical company.

Table 1. A summary containing the main features of the acquisition platform.

Sensor	Feature
 Ladybug system	<ul style="list-style-type: none"> - 6 camera's in pentagonal prism, one pointing upwards - resolution of 1600×1200 per image - 1 frame (6 images) per second - mounted perpendicular w.r.t. the ground plane
 Velodyne HDL-32e	<ul style="list-style-type: none"> - 360° horizontal FOV, 41.3° vertical FOV - 32 lasers spinning at 10 sweeps per second - $\pm 700,000$ points per sweep - mounted with an angle of 66° w.r.t. the ground plane

3.2. Terminology

As mentioned in the previous section, the Velodyne is spinning its head, thereby producing 360° point clouds. A full rotation of the head is also referred to as a sweep. Throughout this paper, we use right subscript k , $k \in \mathbb{Z}^+$ to indicate the sweep number and \mathcal{P}_k to indicate the point cloud perceived during sweep k . This point cloud is expressed using the local coordinate system of the Velodyne V_k at the time that sweep k has started. We will use right superscript i to denote a single point \mathbf{p}_k^i . Finally, we define \mathcal{I}_j to denote the j -th set of images. Recall that the Ladybug only outputs six images simultaneously each second, and hence, some point clouds will have no accompanying visual information. However, both sensors were synchronized, and as a result, the following condition is always warranted: $\forall j \in \mathbb{Z}^+, \exists k \in \mathbb{Z}^+ : t(\mathcal{I}_j) = t(\mathcal{P}_k)$. In this expression, t is a function that returns the timestamp at which respectively the j -th set of images was captured and sweep k was started. As the Velodyne is spinning at a frequency of 10 Hz, there will only be visual information available for every tenth point cloud \mathcal{P}_k .

3.3. Reference Coordinate System

As both sensor outputs \mathcal{P}_k and \mathcal{I}_k are expressed in their own coordinate system, we need to define a reference coordinate system to harmonize both. To ease this task, the platform was designed in a way that the origin of both sensors is collinear. Since the Ladybug was put perpendicular to the ground plane, it is sufficient to determine the orientation of the Velodyne sensor with respect to the ground plane and nullify the offset of the origins of both sensors. As the Velodyne scanner incorporates three gyroscopes and three associated two-axis accelerometers, the transformation can be derived from one of its accelerometers. Specifically, given the accelerometer output $\mathbf{a} = (a_y, a_z)$ (cf. Figure 4), the angle about the x-axis, i.e., the pitch angle, can be computed as $\alpha = \text{atan}(a_z/a_y)$. The final transformation

$T^{(vel2lady)} \in \mathbb{R}^{4 \times 4}$ from the Velodyne coordinate system V to the Ladybug coordinate system L is then given by Equation (1):

$$T^{(vel2lady)} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}_3^\top & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & \cos \alpha & \sin \alpha & t_y \\ 0 & -\sin \alpha & \cos \alpha & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (1)$$

Herein, the values of $\mathbf{t} = (t_x, t_y, t_z)$ were determined by subtracting the two vectors that project both origins to the ground plane. In the case of the Velodyne coordinate system V, the projection of the origin is performed after applying R. Next, the coordinate system W denotes the global coordinate system in which the final 3D point cloud is expressed. We decided to set the coordinate system of the Ladybug at the start of the first sweep L_0 as the global world coordinate system W.

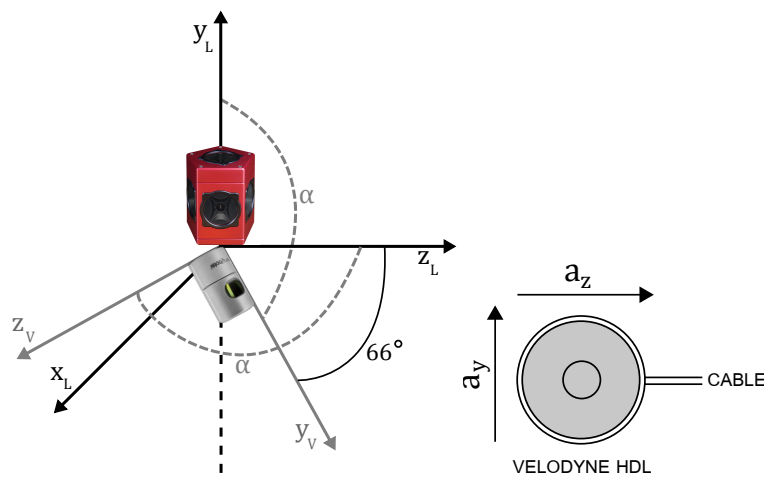


Figure 4. Schematic drawing of the rotation performed after nullifying the offset of the sensor origins. The angle about the x-axis, i.e., the pitch angle, was computed using the output of the two-axis accelerometer $\mathbf{a} = (a_y, a_z)$ from the Velodyne HDL. In this figure, the subscript L denotes the coordinate system of the Ladybug, whereas V denotes the coordinate system of the Velodyne.

3.4. Problem Statement

The problem can be formulated as finding the trajectory, i.e., the sequence of sensor poses $\mathbf{S} = \{S_1, \dots, S_k, \dots, S_N\}$, that transforms each point cloud \mathcal{P}_k in the world coordinate system W. The poses S_k are defined as in Equation (2):

$$S_k = \begin{bmatrix} \mathbf{R}_k & \mathbf{t}_k \\ \mathbf{0}_3^\top & 1 \end{bmatrix} = T_{k,k-1} S_{k-1}, \quad (2)$$

$$S_0 = \begin{bmatrix} \mathbf{R}_0 & \mathbf{t}_0 \\ \mathbf{0}_3^\top & 1 \end{bmatrix} = T^{(vel2lady)}. \quad (3)$$

Herein, $T_{k,k-1}$ denotes the transformation of the point clouds \mathcal{P}_k and \mathcal{P}_{k-1} acquired in two consecutive sweeps. The reconstructed 3D point cloud \mathcal{W}_k after k sweeps have been processed is given by $\mathcal{W}_k = \{\mathcal{P}_0, \dots, S_k \mathcal{P}_k\}$. The reconstructed point cloud of the entire sequence containing N sweeps is finally given by $\mathcal{W} = \{\mathcal{P}_0, \dots, S_k \mathcal{P}_k, \dots, S_N \mathcal{P}_N\}$. Figure 5 depicts a schematic drawing of this problem statement.

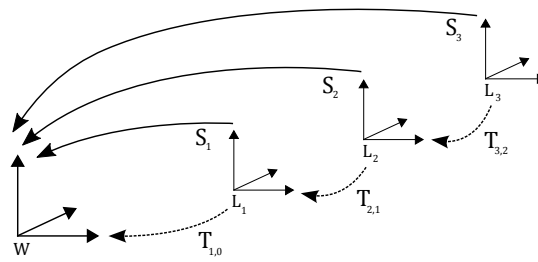


Figure 5. Overview of the problem statement. Each time a new point cloud \mathcal{P}_k has arrived, its sensor pose S_k in relation to the world reference coordinate system $W = L_0$ is determined using the concatenation of the previous pose S_{k-1} and the transformation $T_{k,k-1}$.

4. Approach

A schematic overview of our approach is depicted in Figure 6. As can be seen, it is implemented as a sequential, i.e., incremental, process in which every newly-arrived point cloud \mathcal{P}_k is first processed and subsequently added to the current world model \mathcal{W}_k . The process repeated for every point cloud is conducted in five steps. First, we project the generated point clouds on a 2D grid (Step 1). Subsequently, we conduct a surface analysis (Step 2), after which we perform pairwise alignment of two consecutive point clouds (Step 3), which serves as an initial guess for the current pose. Next, we register the aligned point cloud with a global 3D map (Step 4) and fuse the new points with this 3D map (Step 5). This fusion consists of re-sampling the point cloud by means of a surface reconstruction technique. Optionally, when a loop has been detected, we adopt loop closure to preserve global consistency (Step 6). This is done by means of pose graph optimization, which propagates the estimated error back in the pose graph. In the following sections, we will further clarify each of these steps.

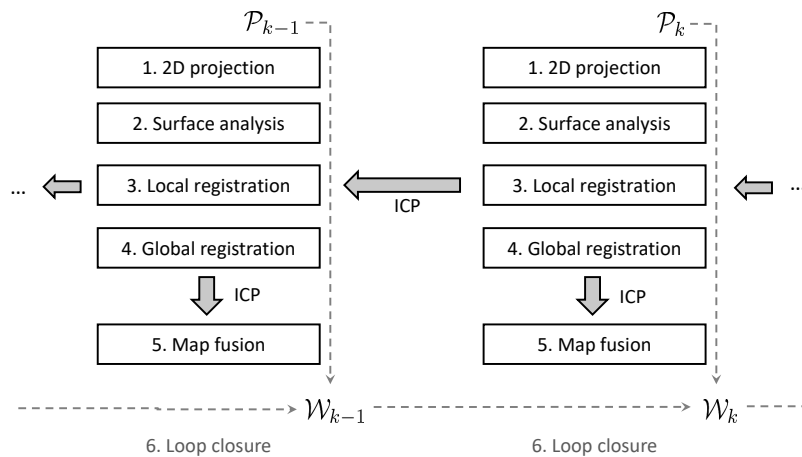


Figure 6. Overview of the system, which is implemented as an incremental process. Every time a consecutive point cloud \mathcal{P}_k has arrived, it is first projected on a 2D grid (Step 1). Next, it is analyzed to obtain some low-level surface features (Step 2), after which it is aligned (registered) with the point cloud \mathcal{P}_{k-1} acquired during the previous sweep $k-1$ (Step 3). The obtained transformation serves as an initial guess to subsequently align \mathcal{P}_k with the current world model \mathcal{W}_{k-1} (Step 4) and to fuse them together to form \mathcal{W}_k (Step 5). This fusion consists of re-sampling the point cloud by means of a surface reconstruction technique. Optionally, when a loop has been detected, loop closure is performed to cope with the drift and to preserve global consistency (Step 6).

4.1. 2D Projection

The 3D points captured by the Velodyne are organized since the 32 lasers are placed collinear in the vertical direction. Because of that, we can project the 3D points onto a two-dimensional spherical

grid; cf. Figure 7. Figure 8 shows an example of a 360° range image corresponding to the point cloud of Figure 2. Using this 2D projection, we can exploit the adjacency in the pixel domain, e.g., to quickly find neighboring points in 3D. We will exploit this knowledge to perform a surface analysis as explained in Section 4.2.

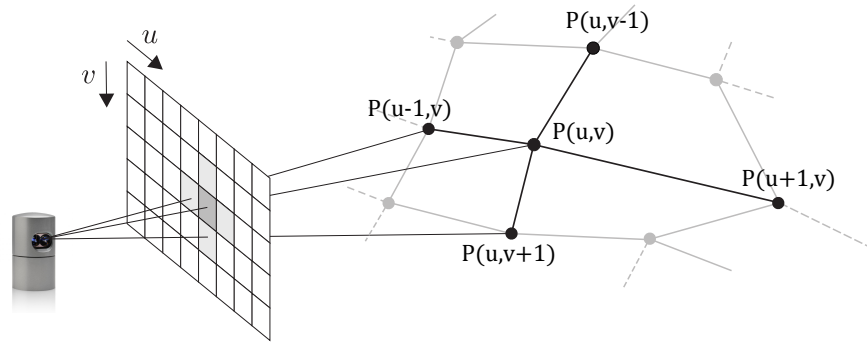


Figure 7. The 3D points generated by the Velodyne scanner are projected onto a 2D spherical grid. This way, we can exploit the adjacency in the 2D domain, e.g., to quickly find neighboring points in 3D.

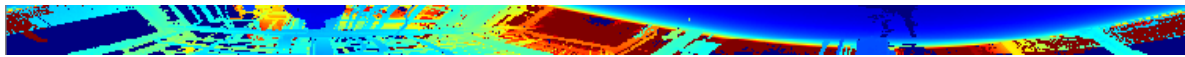


Figure 8. Example of a 360° range image obtained by projecting the point cloud of Figure 2 onto a 2D grid. The blue color means that points are close-by, whereas the red color denotes that points are located further away.

4.2. Surface Analysis

The surface analysis step is performed twice each iteration, once on the point cloud \mathcal{P}_k perceived during sweep k and once on the point cloud map \mathcal{W}_{k-1} after the fusion of the previous point cloud \mathcal{P}_{k-1} . Our proposed algorithm is an extension of the one presented in [10]. It aims to find for each 3D point the optimal neighborhood size or the most suitable local point set that describes the underlying geometry. This is an interdependence problem, as geometrical features largely depend on the choice of the neighborhood, whereas a good neighborhood definition should rely on the local geometry and, thus, on geometrical features. Let us first define a neighborhood function n as $n^r(\mathbf{p}) : (x, y, z) \mapsto (x, y, z)^N$. In this equation, r denotes the search radius for neighboring points, and N denotes the number of points belonging to the neighborhood. Using this neighborhood, we determine the principal components of the covariance matrix of the points belonging to it. These principal components are used to decide whether the underlying surface is linear (1D), planar (2D) or volumetric (3D), cfr. Figure 9. Hence, we first compute the three eigenvalues λ_1, λ_2 and λ_3 and their corresponding eigenvectors $\mathbf{v}_1, \mathbf{v}_2$ and \mathbf{v}_3 . Next, we define the standard deviation along an eigenvector as $\sigma_i = \sqrt{\lambda_i}$ for $i \in 1, 2, 3$. Using these three values, we can obtain a measure on how linear, planar or scattered the underlying surface is. To this end, we define the three dimensionality values as follows:

$$\psi_1 = \frac{\sigma_1 - \sigma_2}{\sigma_1}, \quad \psi_2 = \frac{\sigma_2 - \sigma_3}{\sigma_1}, \quad \psi_3 = \frac{\sigma_3}{\sigma_1}. \quad (4)$$

These three values are normalized such that $\psi_1 + \psi_2 + \psi_3 = 1$; hence, they represent a partition of unity Ψ . Finally, the dimensionality label l of each point is given by Equation (5):

$$l = \operatorname{argmax}_{i \in [1,3]}(\psi_i). \quad (5)$$

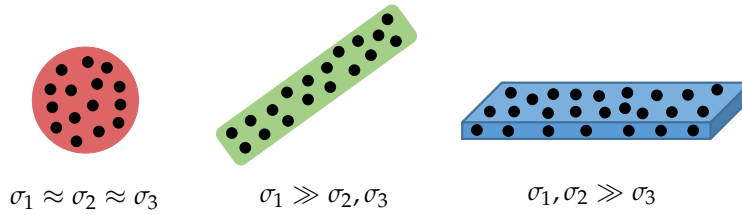


Figure 9. Visual representation of the low-level surface features. The values $\sigma_i = \sqrt{\lambda_i}$ represent the standard deviation along the eigenvectors. When $\sigma_1 \approx \sigma_2 \approx \sigma_3$, the points are volumetric, often times representing a scatter, such as vegetation. When $\sigma_1 \gg \sigma_2, \sigma_3$, the respective points are lying on a line between planar surfaces or are part of thin structures, such as pipelines. Finally, when $\sigma_1, \sigma_2 \gg \sigma_3$, the points are lying on a planar surface.

When $\sigma_1 \gg \sigma_2, \sigma_3$, then ψ_1 will be larger than the two other features. This corresponds to lines between planar surfaces or thin structures, such as pipelines. On the other hand, when $\sigma_1, \sigma_2 \gg \sigma_3$, then ψ_2 will be larger, corresponding to planar surfaces. Finally, when $\sigma_1 \approx \sigma_2 \approx \sigma_3$, then ψ_3 will be larger often times representing a scatter, such as bushes or tree leaves. To determine the optimal radius r^* , we use the concept of Shannon entropy. To this end, we compute the geometrical features for a growing radius size $r \in [r_{min}, r_{max}]$. The Shannon entropy for the partition $\Psi^r = \{\psi_1, \psi_2, \psi_3\}_r$ is given by $E(\Psi^r) = -\psi_1 \ln(\psi_1) - \psi_2 \ln(\psi_2) - \psi_3 \ln(\psi_3)$. This value gives a measure for the uncertainty about the dimensionality label. The optimal neighborhood radius r^* is then defined as the minimum of the entropy function E :

$$r^* = \underset{r \in [r_{min}, r_{max}]}{\operatorname{argmin}} E(\Psi^r). \quad (6)$$

This radius r^* leads to an initial guess of the optimal neighborhood $\mathcal{V}_{\mathbf{p}}^*$ of a point \mathbf{p} . As points within the optimal radius r^* can still belong to different surfaces and can have different dimensionality labels, we define a similarity measure S denoting the homogeneity within the neighborhood of each point:

$$S(n^r(\mathbf{p})) = \frac{1}{N} \sum_{\mathbf{p}_i \in n^r(\mathbf{p})} \mathbf{1}_{l(\mathbf{p})=l(\mathbf{p}_i)}. \quad (7)$$

In this equation, $\mathbf{1}$ represents the indicator function, and N is the cardinality of $n^r(\mathbf{p})$. If this value S is smaller than 0.5, we reconsider the optimal radius r^* using Equation (8):

$$r^* = \underset{r \in [r_{min}, r_{max}]}{\operatorname{argmax}} S(n^r(\mathbf{p})). \quad (8)$$

In the other case, we remove the points belonging to the neighborhood of \mathbf{p} that have another dimensionality label. Hence, we define the optimal neighborhood $\mathcal{V}_{\mathbf{p}}^*$ of a point \mathbf{p} as in Equation (9):

$$\mathcal{V}_{\mathbf{p}}^* = \{\mathbf{p}_i : \mathbf{p}_i \in n^{r^*}(\mathbf{p}) \wedge l(\mathbf{p}) = l(\mathbf{p}_i)\}. \quad (9)$$

In summary, the proposed method tries to find the ideal topological space \mathcal{T}^* representing the underlying surface of the point cloud. For each node (or point) in \mathcal{T}^* , we store a 19-dimensional feature vector, which is the combination of the eigen values $\lambda = \{\lambda_1, \lambda_2, \lambda_3\}$, the eigen vectors $V = \{\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3\}$, the dimensionality values $\Psi = \{\psi_1, \psi_2, \psi_3\}$, the dimensionality label l and the values r^* and E^* . The last feature we define represents the omnivariance and is given by $O = \prod_{i \in [1,3]} \sigma_i$. One of the most important features in this vector is the local surface normal \mathbf{n}_i of a point \mathbf{p}_i , which can be approximated by the eigenvector \mathbf{v}_3 corresponding to the smallest eigenvalue λ_3 of the principal components, which was computed using the optimal neighborhood.

4.3. Local Registration

The majority of the solutions presented in the literature to find the alignment of consecutive scans or point clouds are based on the iterative closest point (ICP) algorithm. As the name suggests, it is an iterative method that consists of four main steps in each iteration: (1) point selection; (2) correspondence estimation; (3) weighting; and (4) transformation estimation. In every iteration, the transformation is updated until convergence has been reached. In Figure 10, a graphical representation of the algorithm is depicted. However, the standard ICP approach has some severe limitations. One of the main issues originates from the fact that the point clouds generated by the Velodyne scanner have a sparse and inhomogeneous density. As a result, it is hard to find good point correspondences between two consecutive point clouds (referred to as source and target point cloud), as most of the time, the determined corresponding pairs will not represent the same physical point in space. This leads to point clouds that are not aligned accurately, even after a large number of iterations and, hence, to wrongly estimated sensor poses. In addition, the convergence process will be very slow. Motivated by this shortcoming, we propose to incorporate the low-level features that we derived in the surface analysis step to guide the ICP process. Doing so, we will not use the closest points in 3D as target points directly as is done in traditional ICP. Instead, we present an approach in which we use the local surface normals that were computed using the optimal neighborhood (cf. Section 4.1). We then minimize the distance from each source point to the tangent plane of its corresponding surface patch in the target point cloud. In the following, we will briefly discuss the four main sub-parts of the process.

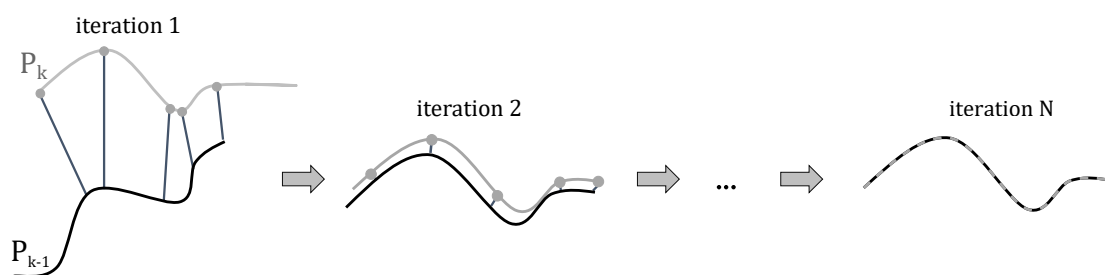


Figure 10. Graphical representation of the ICP algorithm. In every iteration, four main steps are conducted: (1) point selection; (2) correspondence estimation; (3) weighting; and (4) transformation estimation. In every iteration, the transformation is updated until convergence has been reached and the two scans are perfectly aligned.

(1) Point Selection

The goal of the point selection step is to focus on the most reliable areas for accurate registration. An area is considered as unreliable if it is located near the border between several objects or surfaces or if it belongs to a geometrically complex object. As the points corresponding with a volumetric label ($l = 3$) are mainly corresponding to scattered points, e.g., bushes or tree leaves, we will exclude them from the estimation process, as we consider them less reliable. This is mainly due to the fact that it is hard to estimate accurate surface normals for these points, as they represent complex structures and are too sparsely sampled. In addition, we will exclude planar points for which the uncertainty, i.e., the entropy, is too high. In the experiments, the threshold for the entropy was set to 0.8.

(2) Correspondence Estimation

In contrast with existing systems, we will select the corresponding target point \mathbf{p}_{k-1}^j as the point having the most similar neighborhood as the source point \mathbf{p}_k^i in terms of the geometry of the underlying surface. To this end, we compute the distance from the source point to the target point in feature space. However, as it is too infeasible to compare every point in the target point cloud with the source point, we first determine corresponding candidates by selecting the closest points in Euclidean space using

the 2D projection of both point clouds. More specifically, we will look for the points that are only a few pixels away in the 2D domain. Finally, the target point that has the smallest distance in feature space is chosen as the corresponding point.

(3) Weighting

In order to make the procedure more robust against correspondence outliers computed in the previous section, we incorporate robust M-estimators in the iteration process. Thus, instead of using fixed weights, we adapt them through the iterations resulting in an iteratively reweighted least squares (IRLS) ICP optimization. Compared to traditional solutions, this weighting is carried out using the distance of the corresponding pairs in feature space instead of Euclidean 3D space. This is important as the point clouds are inhomogeneous, and the distance between correct correspondence pairs, i.e., representing the same physical point, will vary greatly based on the distance of the points to the origin of the sensor. We want to emphasize that the feature vector of a point itself is not changing during consecutive ICP iterations, but its corresponding point can be chosen differently, as the matching process also depends on their Euclidean distance (cf. Section 4.3). Thus, the distance in feature space can be varying during the iterations.

(4) Transformation Estimation

In traditional ICP approaches, the transformation between point clouds is estimated by minimizing the sum of the Euclidean distances between corresponding points, known as the point-to-point distance. In our approach, we will not match points from the source point cloud \mathcal{P}_k with points in the target point cloud \mathcal{P}_{k-1} , but rather match points of \mathcal{P}_k with surface patches of \mathcal{P}_{k-1} . The goal is to minimize the distance between the points in \mathcal{P}_k with the tangent plane of the corresponding surface patch in \mathcal{P}_{k-1} , known as the point-to-plane distance and given by the following error metric:

$$E(\mathcal{P}_k, \mathcal{P}_{k-1}; \mathbf{T}_{k,k-1}) = \sum_{i=1}^N \mathbf{w}^i ((\mathbf{T}_{k,k-1} \mathbf{p}_k^i - \mathbf{p}_{k-1}^{\mathbf{c}(i)}) \cdot \mathbf{n}_{k-1}^{\mathbf{c}(i)})^2. \quad (10)$$

Herein, $\mathbf{T}_{k,k-1}$ is the estimated transformation matrix; \mathcal{P}_k is the source point cloud; \mathcal{P}_{k-1} is the target point cloud; \mathbf{n}_{k-1}^i is the surface normal according to target point; \mathbf{p}_{k-1}^i , \mathbf{w}^i is the weight vector; and \mathbf{c} is the vector containing the indices of the N corresponding points. Equation (11) gives the expression to derive the final transformation matrix $\mathbf{T}_{k,k-1}$:

$$\mathbf{T}_{k,k-1} = \begin{bmatrix} \mathbf{R}_{k,k-1} & \mathbf{t}_{k,k-1} \\ \mathbf{0}_3^\top & 1 \end{bmatrix} = \underset{\mathbf{T}_{k,k-1}}{\operatorname{argmin}} E(\mathcal{P}_k, \mathcal{P}_{k-1}; \mathbf{T}_{k,k-1}). \quad (11)$$

In order to solve this optimization problem, we adopt the method proposed by Low et al. in [23]. In that paper, a method is derived to approximate the nonlinear optimization problem with a linear least squares one in the case that the relative orientation between the two input point clouds is small. Since we consider point clouds acquired in two consecutive sweeps, this assumption is guaranteed in our case.

4.4. Global Registration

Once we have found the initial transformation $\mathbf{T}_{k,k-1}$ between point clouds \mathcal{P}_k and \mathcal{P}_{k-1} , the next step consists of fusing \mathcal{P}_k with the current registered point cloud \mathcal{W}_{k-1} . To this end, we first transform \mathcal{P}_k in the world coordinate system using the estimate of the current pose, which is given by $\hat{\mathbf{S}}_k = \mathbf{T}_{k,k-1} \mathbf{S}_{k-1}$ resulting in $\hat{\mathcal{P}}_k = \hat{\mathbf{S}}_k \mathcal{P}_k$. As the point cloud $\hat{\mathcal{P}}_k$ is still not accurately aligned with \mathcal{W}_{k-1} , we perform a second registration step based on ICP as explained in the previous section. However, regarding the transformation estimation, the cost function that we want to minimize is now given by Equation (12):

$$\mathbf{T}_{k,w} = \begin{bmatrix} \mathbf{R}_{k,w} & \mathbf{t}_{k,w} \\ \mathbf{0}_3^\top & 1 \end{bmatrix} = \underset{\mathbf{T}_{k,w}}{\operatorname{argmin}} E(\hat{\mathcal{P}}_k, \mathcal{W}_{k-1}; \mathbf{T}_{k,w}). \quad (12)$$

To find the correspondence points of $\hat{\mathcal{P}}_k$ in \mathcal{W}_{k-1} , we can still use the criterion explained in Section 4.3. However, as the world map is growing, it is practically infeasible to look in the entire map for correspondences. Fortunately, this is not necessary as corresponding points in the map will be located close to the newly-added point cloud $\hat{\mathcal{P}}_k$, as this latter is already transformed with an estimate of the current pose $\hat{\mathbf{S}}_k$. Therefore, we can first filter the map by means of frustum culling using the oriented bounding box of the current sweep as a box filter extended with a small offset. The principal axes of this oriented bounding box are already known as these are previously determined by the accelerometer output. This box filtering will lower the processing speed without harming the guarantee to find good point correspondences. As we cannot use the organized structure of the separated point clouds any more after they have been fused with the point cloud map, we first create a kd-tree of the points remaining after the box filtering. After the registration of $\hat{\mathcal{P}}_k$ with \mathcal{W}_{k-1} , we define $\mathbf{S}_k = \mathbf{T}_{k,w} \hat{\mathbf{S}}_k$.

4.5. Map Fusion

After adding \mathcal{P}_k to the former world model \mathcal{W}_{k-1} , the point density in this world model will increase. We can now refine the ideal topological space \mathcal{T}^* and hence the local surface normals in the global 3D map by re-estimating them using their new neighborhood. However, as this point cloud contains noise and has an inhomogeneous point density, we will resample it using the moving least squares (MLS) surface reconstruction algorithm [24]. In summary, this method will try to locally approximate the underlying surface by higher order polynomial interpolations between surrounding data points. Using these polynomials one can resample the point cloud and obtain more accurate estimates for the surface normals. The procedure can be described as follows. Using the ideal topological space \mathcal{T}^* , we have for each point \mathbf{p} an optimal neighborhood $\mathcal{V}_{\mathbf{p}}^*$, as well as the tangent plane to \mathbf{p} defined as $H_{\mathbf{p}} \triangleq [\mathbf{n}, d]$. For all points lying within this neighborhood, we can compute the distance to this tangent plane. Subsequently, we fit a polynomial in the set of distances from these points to the surface. To this end, we define a local approximation of degree m by a polynomial $\tilde{p} \in \Pi_m$ minimizing, among all $p \in \Pi_m$, the weighted least-squares error of Equation (13):

$$\sum_{i \in I} (p(\mathbf{x}_i) - f_i)^2 \theta(\|\mathbf{p}_i - \mathbf{p}\|). \quad (13)$$

In this equation, I is the vector of indices representing the points in $\mathcal{V}_{\mathbf{p}}^*$; $\{\mathbf{x}_i\}_{i \in I}$ are the orthogonal projections of the points $\{\mathbf{p}_i\}_{i \in I}$ onto $H_{\mathbf{p}}$; and $f_i \triangleq \langle \mathbf{p}_i, \mathbf{n} \rangle - d$ is the distance of \mathbf{p}_i to the tangent plane $H_{\mathbf{p}}$. Finally, $\theta(x) = e^{-\left(\frac{x}{\sigma_r}\right)^2}$ represents the weighting function that is based on the distances to the tangent plane and the average separation σ_r of the 3D points.

Once the parameters of the polynomials are known, we finally project the data points back on the moving least squares surface. This procedure will hence manipulate the sampled data points in such a way that they represent the underlying surface in a better way. In addition, we upsample the point cloud using voxel grid dilation in order to fill small gaps. This latter procedure will first dilate a voxel grid representation of the world model built using a predefined voxel size. After that, the resulting new points are projected to the MLS surface of the closest point in the world point cloud. With time, \mathcal{W}_{k-1} is becoming larger and larger, and as a result, the search for nearest neighbors becomes quite intractable. For that reason, we store the map as an octree data structure. Using this octree, we can work on a downsampled version of the point cloud map \mathcal{W}_{k-1} . The main benefit of this octree representation is that we can keep all points, but at the same time, we can freely choose the level of density we want to use.

4.6. Loop Closure

Detecting loops using the acquired point clouds would be too cumbersome, as it would take ages to compare every newly-generated point cloud with the entire 3D map. For this reason, we perform loop detection on the images of the Ladybug. We adopt the same strategy presented in [25] (DBOW2) and improved by [1] (ORB-SLAM). More specifically, we adopt a bag of words (BOW) implemented as a hierarchical tree that uses a visual vocabulary. This vocabulary is built offline using ORB descriptors and converts an image into a sparse numerical vector. Each image is thus converted into a bag of word vector that is used to compare it with other images and to measure the similarity. This latter is conducted in the same way as described in [25]. In addition, an index is maintained that stores for each word in the vocabulary the list of images where it is present. Doing so, we are able to perform comparisons only against images that have some word in common with the query image.

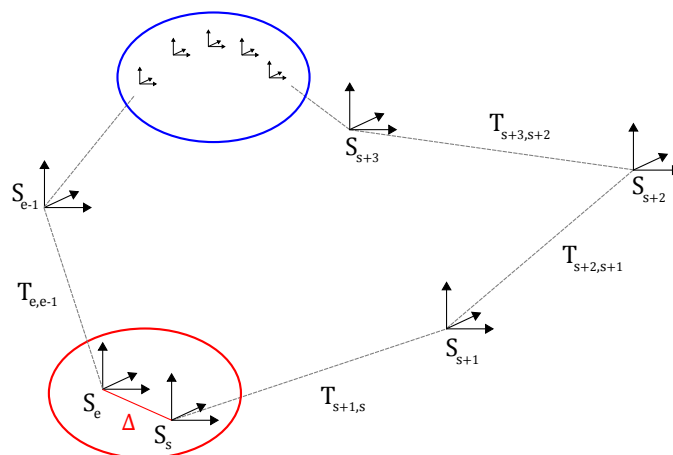


Figure 11. Overview of the loop closure procedure. When a loop has been detected, its loop transform Δ is considered as an error and is propagated back in the pose graph. To this end, we use the residual of the minimization step (cf. Section 4.3) to assign a weight to each link in the pose graph. We assign a higher weight for those transformations that had a high residual in the previous minimization step. The idea is that a high residual indicates that two consecutive point clouds were potentially inaccurately aligned.

Let us assume that a loop has been detected between image set \mathcal{I}_j and \mathcal{I}_i . We now consider all point clouds \mathcal{P}_l captured in the interval $[t(\mathcal{I}_j), t(\mathcal{I}_{j+1})[$ (recall that there are ten sweeps in the interval) and try to find the most similar point cloud \mathcal{P}_k captured in the interval $[t(\mathcal{I}_i), t(\mathcal{I}_{i+1})[$. As the sensor could have visited the same location from different directions, it can happen that both ends of the loop have different orientations. Therefore, we first determine the matrix $R_{k,l}$ that describes the rotation between \mathcal{P}_l and \mathcal{P}_k . It is important to note that we compute the transformation on the local point clouds, which are expressed in their own coordinate system. The rotation estimation is a two-step process, consisting of an initial rough alignment and a refinement step using the ICP algorithm. The two point clouds \mathcal{P}_l and \mathcal{P}_k that have the lowest residual after the ICP step are considered the most similar point clouds and are denoted by \mathcal{P}_e and \mathcal{P}_s , representing both ends of the loop. Their poses are respectively S_e and S_s . Next, the difference in pose, i.e., the loop transform, is given by $\Delta = (R_{s,e}S_e)^{-1}S_s$. This loop transform is considered as an error as both poses $R_{s,e}S_e$ and S_s should be equal. The next step consists of propagating the error Δ back in the pose graph. To this end, we use the residual of the minimization step (cf. Section 4.3) to assign a weight $c_{i,j}$ to each link in the pose graph. We assign a higher weight for those transformations that had a high residual in the previous minimization step. The idea is that a high residual indicates that two consecutive point clouds were potentially inaccurately aligned. On the other hand, the ICP process will have already been converging in the right direction. Next, we define the distance between two poses S_k and S_l as $d(S_k, S_l) = \sum_{i,j} c_{i,j}$. Herein, $\{i, j\}$ denotes the set of all

edges in the path from S_k to S_l . Finally, we define a weight $w_i = \frac{d(S_s, S_i)}{d(S_s, S_e)}$ for each pose in the graph that specifies the fraction of the matrix Δ by which the pose has to be transformed. The poses S_k are then updated replacing \mathbf{t}_k by $\mathbf{t}_k w_k \Delta$ and \mathbf{R}_k by $\text{slerp}(\mathbf{R}_k, w_k \Delta)$. In this latter assignment, slerp denotes the spherical linear interpolation function as described in [26]. A graphical representation of this loop closure procedure is depicted in Figure 11.

5. Evaluation

To evaluate the system, we considered two different datasets. The first dataset we have captured ourselves. It covers different environments, including a university campus and an industrial site. The latter was captured at a chemical site of the Dow company in Terneuzen in the Netherlands. In order to be able to compare our system with the state of the art, we also performed some experiments on the well-known Kitti vision benchmark that was presented in [27].

5.1. Our Dataset

The first set of sequences was captured at a chemical site of the Dow company in Terneuzen. This environment is part of a disused area that is planned to be demolished. It consists of many pipelines that were formerly used to carry liquids or gases, as can be seen in Figure 3. Although this environment seems outdoors, the GPS signal is far too unreliable due to the abundance of pipelines. The acquired data consists of video sequences recorded with our Vellady platform. The speed of the platform approximated walking speed, i.e., 4 km/h. Furthermore, accurate ground truth information by means of terrestrial laser scanning was captured using a Leica system. (www.leica-geosystems.be) An image of this ground truth point cloud is depicted in Figure 12. As we do not have ground truth of the actual trajectory, we have to compare the reconstructed point cloud with the ground truth point cloud. For this comparison, it is necessary that both point clouds are aligned. Hence, we will first perform a rough alignment based on key points that we manually selected from both point clouds. Subsequently, we perform ICP to align them in a fine way. The final residual of the ICP process, i.e., the Euclidean distance between all closest point pairs, can be considered as a measure for the accuracy of the reconstruction. The experiments demonstrated that the final residual of this ICP process is approximately 2.1 cm for our reconstruction. However, as the closest point pairs do not necessarily represent the same physical points in space, the ICP residual is not a perfect measure for the evaluation of the accuracy. For that reason, we conducted another experiment in which we use the dominant planes in the scene to make the comparison. More specifically, in both point clouds, we estimate the parameters of the most dominant planes after which we determine the corresponding planes. Subsequently, we compute for all inliers of the planes from the source point cloud the distance to its corresponding plane in the target point cloud. We hence define the average distance of two corresponding planes $H_s \triangleq [\mathbf{n}_s, d_s]$ and $H_t \triangleq [\mathbf{n}_t, d_t]$ as $d(H_s, H_t) \triangleq \frac{1}{|H_t|} \sum_{\mathbf{p}_t \in H_t} \langle \mathbf{p}_t, \mathbf{n}_s \rangle - d_s$. In addition, we also define the difference in angle of the two corresponding planes as $\phi(H_s, H_t) \triangleq \text{acos}(\mathbf{n}_s \cdot \mathbf{n}_t)$. The results are summarized in Table 2 for the eight most dominant planes in the scene. The table shows that on average, the deviation in angle of two corresponding planes is approximately 0.84° , whereas the average distance between two planes is approximately 1 cm. In Figure 13, an image is depicted in which both the ground truth and estimated point cloud are shown after alignment seen from a bird's eye view. In Figure 14, the same models are depicted, but this time after they have been projected onto the ground plane to make it easier to evaluate the accuracy. Visually, there are little to no discrepancies noticeable between the two 3D point cloud models.

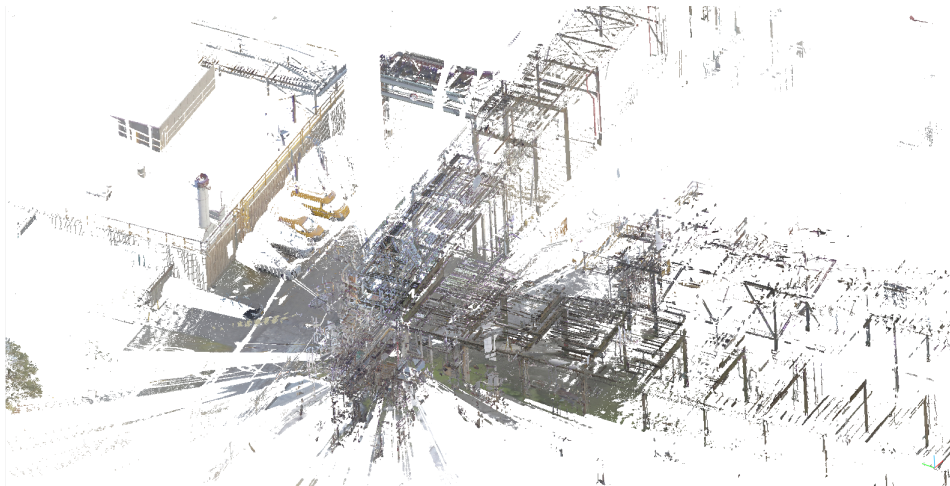


Figure 12. Part of the ground truth 3D point cloud of the Dow chemical site acquired using a Leica static terrestrial LiDAR scanning system.

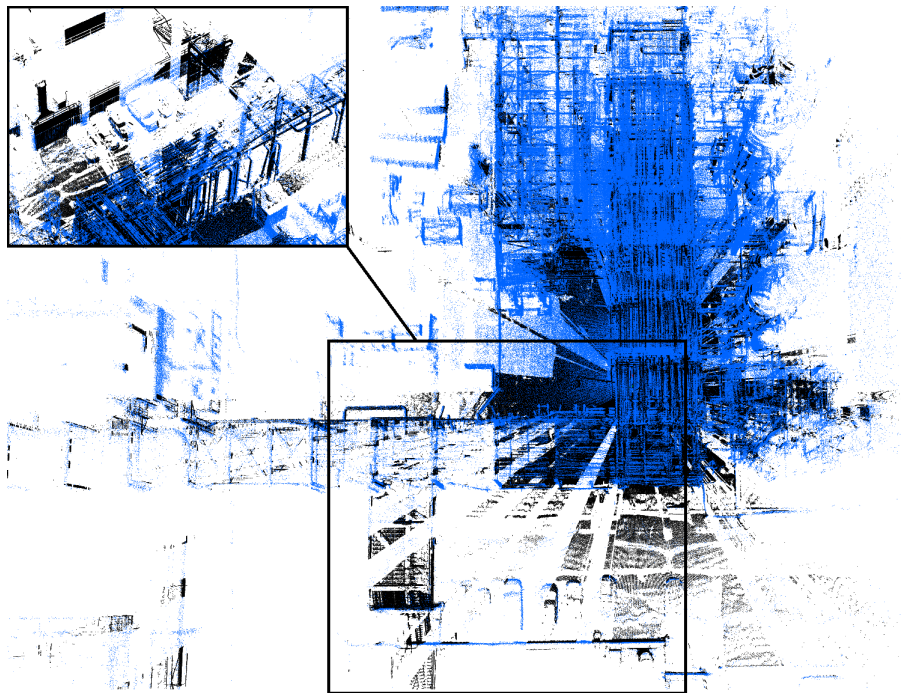


Figure 13. Bird's-eye view of the ground truth (black) and reconstructed (blue) point cloud aligned with each other. Visually, one can see that the walls and edges of both point clouds are overlapping. This is best visible in areas for which the ground truth point cloud has a LiDAR shadow due to occlusion. The average distance of the corresponding dominant planes (cf. Table 2) is approximately 1 cm, whereas the average deviation in angle is 0.84° .

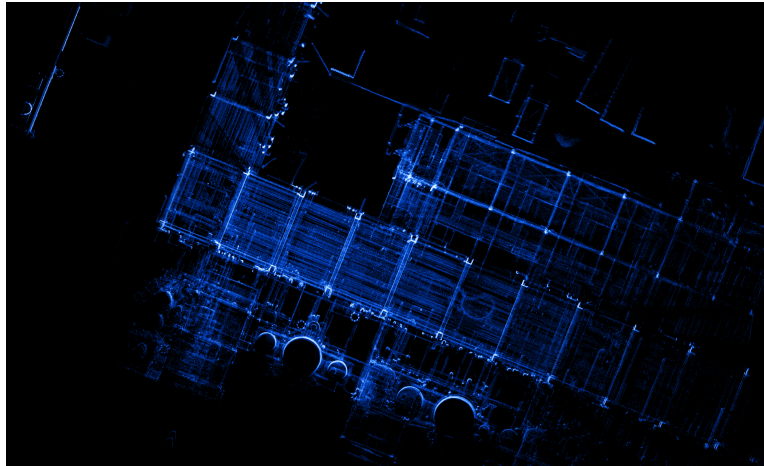


Figure 14. 2D projection on the ground plane of both the reconstructed and the ground truth 3D model in order to evaluate the accuracy qualitatively. To obtain the image, the ground plane was removed from both point clouds, and a quantization in 2D cells was conducted. Brighter areas denote a higher point density in the cell and, hence, emphasize the overlap of both point clouds. Visually, one can notice little to no discrepancies as the walls and edges of both models are overlapping.

Table 2. The difference in angle between two corresponding planes $\phi(H_s, H_t) \triangleq \arccos(\mathbf{n}_s \cdot \mathbf{n}_t)$ (in degrees) and the average distance of two corresponding planes $d(H_s, H_t) \triangleq \frac{1}{|H_t|} \sum_{\mathbf{p}_t \in H_t} \langle \mathbf{p}_t, \mathbf{n}_s \rangle - d_s$ (in meter) for the eight most dominant planes in the scene.

Plane	$d(\phi_g, \phi_r)$	$d_\mu(H_g, H_r)$
1	0.759761	0.00818437
2	0.276246	0.00724192
3	1.12969	0.008299
4	0.286674	0.0115731
5	0.292083	0.0164202
6	0.492578	0.0106246
7	1.80667	0.0118678
8	1.68458	0.00993411
	0.841036	0.0105181

By means of comparison, the trajectory of the mobile observer is plotted in Figure 15. Herein, six plots are depicted representing the trajectories obtained by the visual structure from motion (SfM) approach using SIFT features, presented in [28] (in red) and the trajectory obtained by our proposed LiDAR mapping system (in blue). Each red graph is the result of the visual SfM method run on the output of one camera of the Ladybug system. As can be clearly seen, many poses are missing for this approach. This is due to the fact that sometimes too few good matches could be found between one image and the others. As a consequence, the resulting point cloud of the visual SfM is far more sparse than the one obtained by the LiDAR system. In addition, many outlier poses are present. The graphs show that the performance of visual SfM is less compared to odometry and mapping using LiDAR data, and hence, the former is less suited to obtain an accurate 3D model of the environment using a mobile observer. Finally, Figure 16 shows an image of the estimated trajectory, as well as the obtained 3D reconstruction using our proposed approach using the LiDAR data acquired by our Vellady platform.

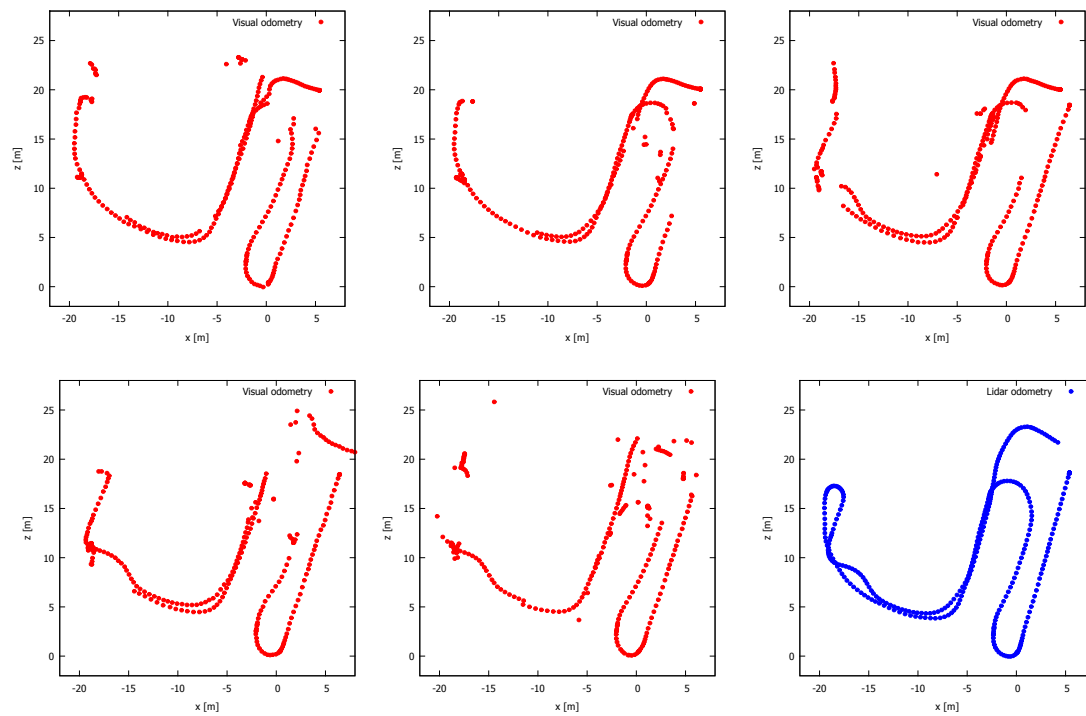


Figure 15. Five plots of the trajectories estimated by the visual structure from motion (SfM) framework presented in [28] using the images of each Ladybug camera separately (red color). The result of the camera pointing upwards is omitted. The blue plot represents the trajectory estimated using our method. For the visual SfM approach, many poses are missing due to the lack of good feature matches. Moreover, many outlier poses are present.

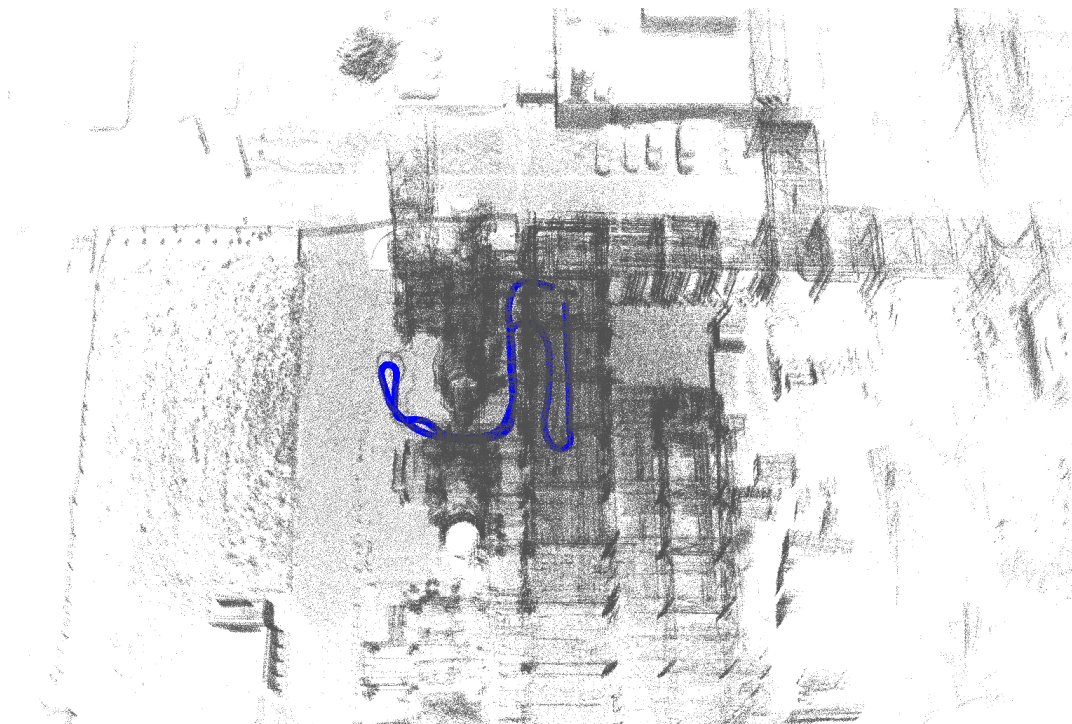


Figure 16. Bird's eye view of the obtained 3D reconstruction and trajectory of a data sequence captured at the chemical site of Dow company using LiDAR data acquired by the Vellady platform. The blue line is representing the estimated trajectory of the mobile observer.

By means of further evaluation, a second set of sequences was captured at a campus of Ghent University. Some example images of 3D reconstructions are shown in Figure 17. Unfortunately, there is no ground truth available for these environments. Visually, we can see that the reconstructions are adequate. The reconstructions of the indoor environment (images at the bottom) however have a mirror effect due to the reflectance of the windows.

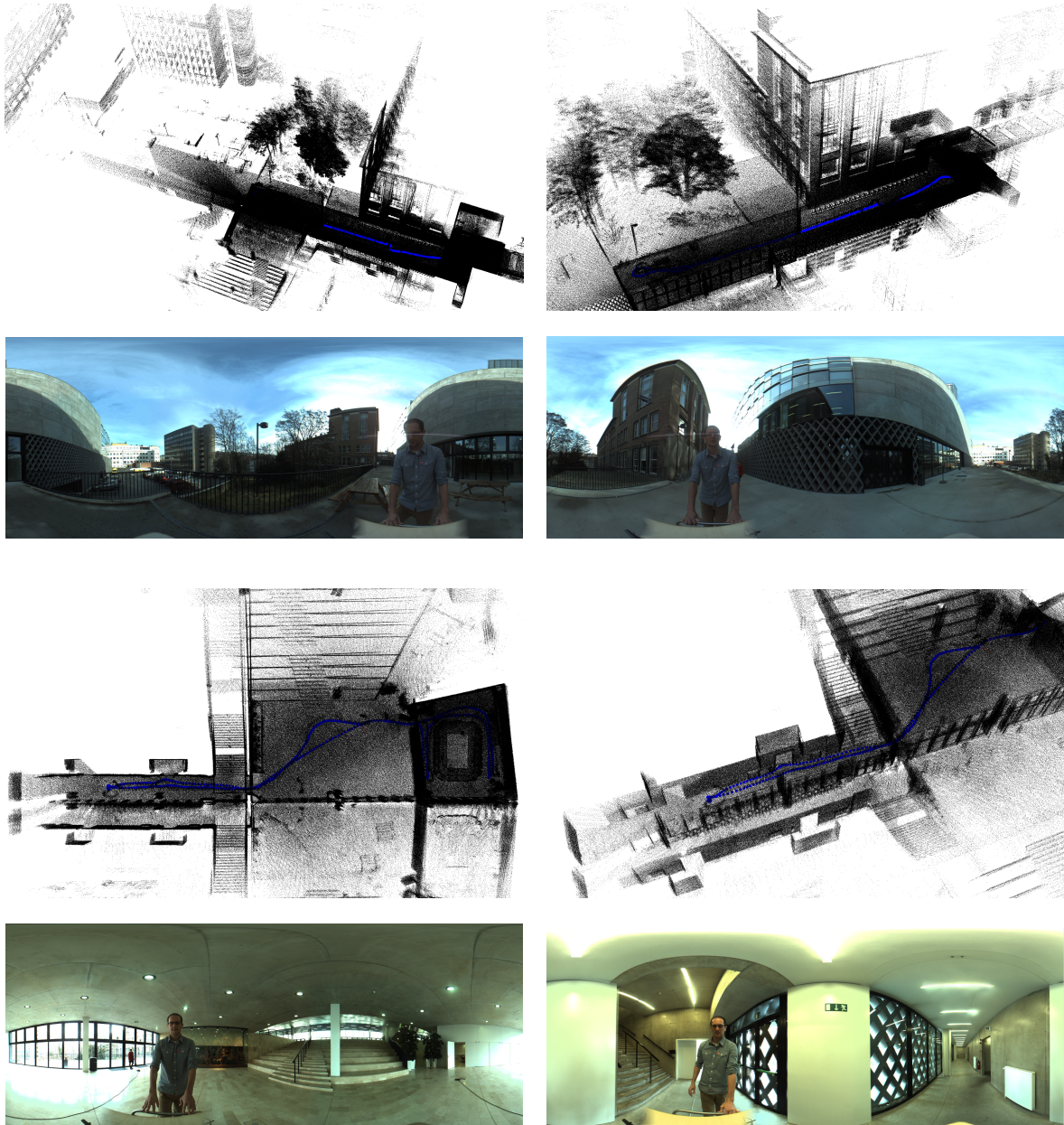


Figure 17. Some example images of the 3D reconstructions obtained by applying our algorithm on the recordings of a campus of Ghent University.

5.2. Kitti Vision Benchmark

To further evaluate our system, we performed experiments on the Kitti vision benchmark presented by A. Geiger in [27], currently one of the main benchmarks related to (visual) odometry. For that benchmark, a van was driving in the streets of the German city Karlsruhe, thereby recording data from different modalities, among them a Velodyne HDL-64e LiDAR scanner and a stereo camera rig. To evaluate our odometry system, we ran our algorithm on the eleven test sequences that were

recorded. Important to note is that the sensor set-up by which the sequences were captured differs from our set-up in the sense that the LiDAR scanner was placed perpendicularly. In order to evaluate our results quantitatively, we projected the ground truth and estimated poses on the ground plane as is suggested by the Kitti vision benchmark. The plots of these trajectories are depicted in Figure 18. Visually, the graphs show that our method is capable of approximating the ground truth in most cases. For the very long sequences, the system suffers from drift as is inherent to the SLAM problem. These results were also obtained without the loop detection and loop closure as described in Section 4.6. Next, we derived the translational error e_t and rotational error e_r from the trajectories. In order to evaluate both errors, the trajectories were quantized in intervals of length Δ . The sensor pose S_f corresponding with sweep f represents the first sensor pose of the interval, whereas S_l represents the last sensor pose of the interval. For S_f and S_l , the following condition is valid: $\|\mathbf{t}_f - \mathbf{t}_l\|_2 \approx \Delta$. We now define the difference of the two ground truth poses S_f and S_l and estimated poses \hat{S}_f and \hat{S}_l as $S_\Delta = S_f^{-1}S_l$ and $\hat{S}_\Delta = \hat{S}_f^{-1}\hat{S}_l$, respectively. The final pose error E_Δ is then given by Equation (14):

$$E_\Delta = \begin{bmatrix} \mathbf{R}_e & \mathbf{t}_e \\ \mathbf{0}_3^\top & 1 \end{bmatrix} = \hat{S}_\Delta^{-1}S_\Delta. \quad (14)$$

Finally, the rotational error e_r and translational error e_t are given by Equations (16) and (17):

$$d = \frac{1}{2}(\text{tr}(\mathbf{R}_e) - 1), \quad (15)$$

$$e_r = \frac{1}{\Delta} \text{acos}(\max(\min(d, 1), -1)), \quad (16)$$

$$e_t = \frac{1}{\Delta} \|\mathbf{t}_e\|_2. \quad (17)$$

The graphs that summarize the translational and rotational errors for $\Delta \in \{100, \dots, 800\}$ are depicted in Figures 19 and 20. These graphs are the average numbers for the eleven different sequences of the Kitti dataset. As can be seen, the rotational error decreases from 0.016° per meter to 0.006° per meter when the path length Δ increases from 100 to 800 meter. The translational error on the other hand increases from 1% to 1.75% per meter for the path length increasing from 100 to 500. After that, it stabilizes more or less. Regarding the translational error, our algorithm competes with the methods ranked seventh to 20th in the Kitti vision benchmark. Figure 20 also shows the translational and rotational errors for an increasing speed. The rotational error varies only slightly when the speed increases. Regarding the translational error, we can see that it increases for higher speeds, which is expected. For speeds higher than 70 km/h, the error increases drastically. The reason for this failure is clear, as for higher speeds, consecutive point clouds begin to differ greatly from each other and no good correspondences can be found. However, we want to point up that our 3D mapping system is meant to operate in industrial plants or indoor environments where the speed of a mobile observer or a drone is always limited. Our system can perfectly cope with speeds up to 30 km/h.

Moreover, our method is more generic, as it does not pose any restriction on the sensor set-up nor any prior knowledge of the type of the scene. Furthermore, it does not rely on the detection of the ground plane, and as a result, it does not need to be present in the LiDAR image. Although the latter assumption could improve the accuracy of the system, we want it to be generic and applicable in any environment and in combination with any sensor set-up. Finally, some example images of 3D reconstructions using the Kitti benchmark are presented in Figure 21. These reconstructions are part of the validation sequences for which no ground truth is provided.

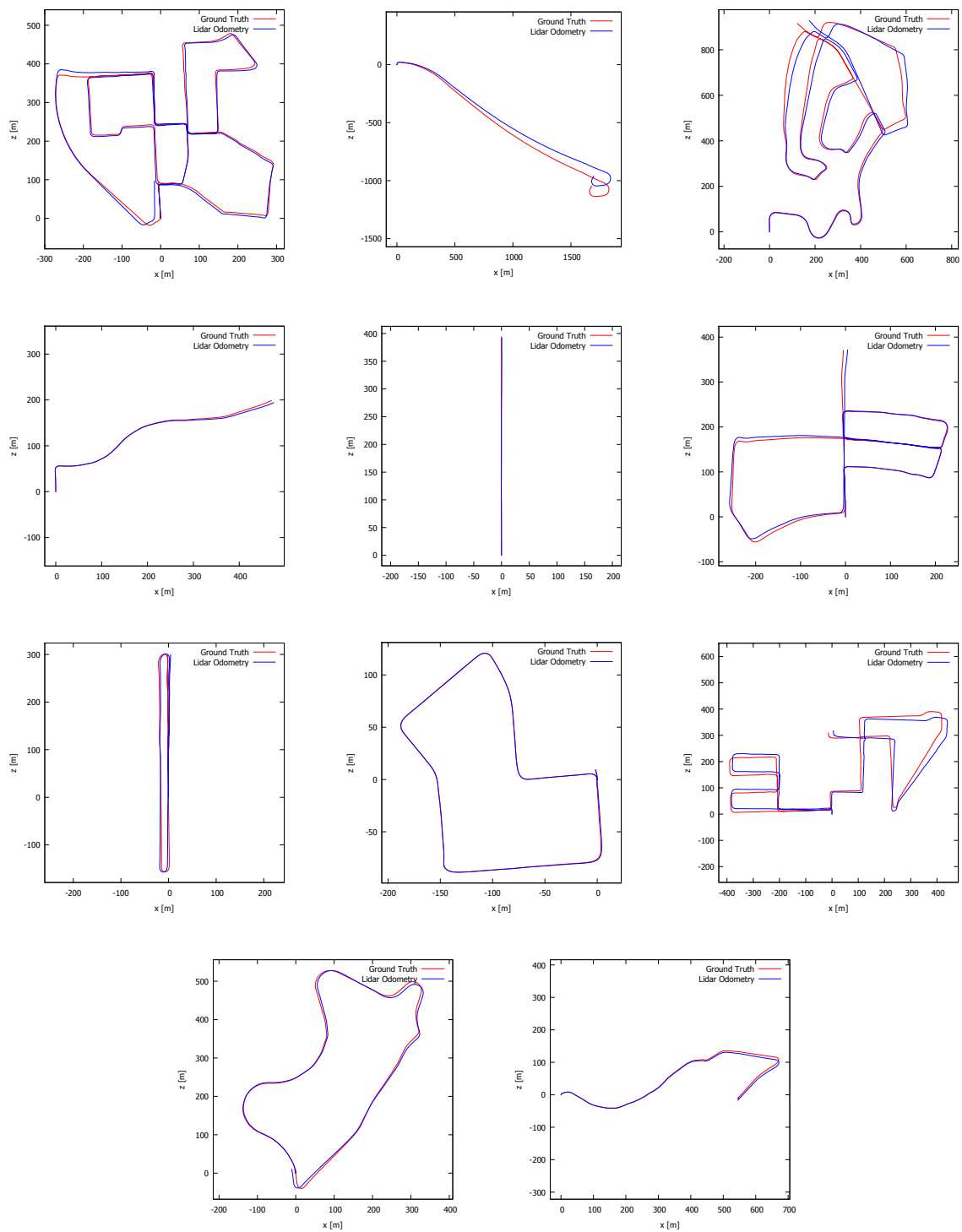


Figure 18. Eleven plots of the estimated (blue) and ground truth (red) trajectories of the Kitti benchmark presented in [27]. The sequences “00” to “10” are presented from left to right and from top to bottom. The algorithm was run without the loop detection and closure algorithm. The majority of the results are satisfying.

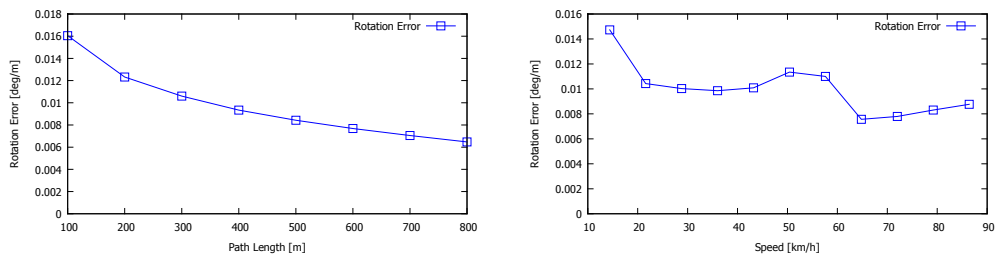


Figure 19. Results of the Kitti vision benchmark showing the average rotation error of all test sequences. Left: the rotation error is expressed as a function of the path length. Right: the rotation error is expressed as a function of the speed.

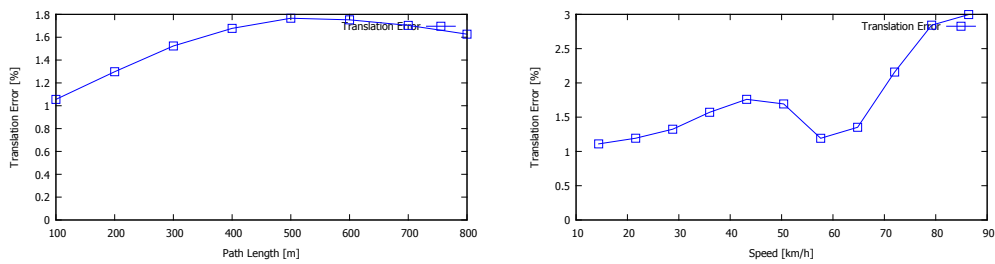


Figure 20. Results of the Kitti vision benchmark showing the average translation error of all test sequences. Left: the translation error is expressed as a function of the path length. Right: the translation error is expressed as a function of the speed.

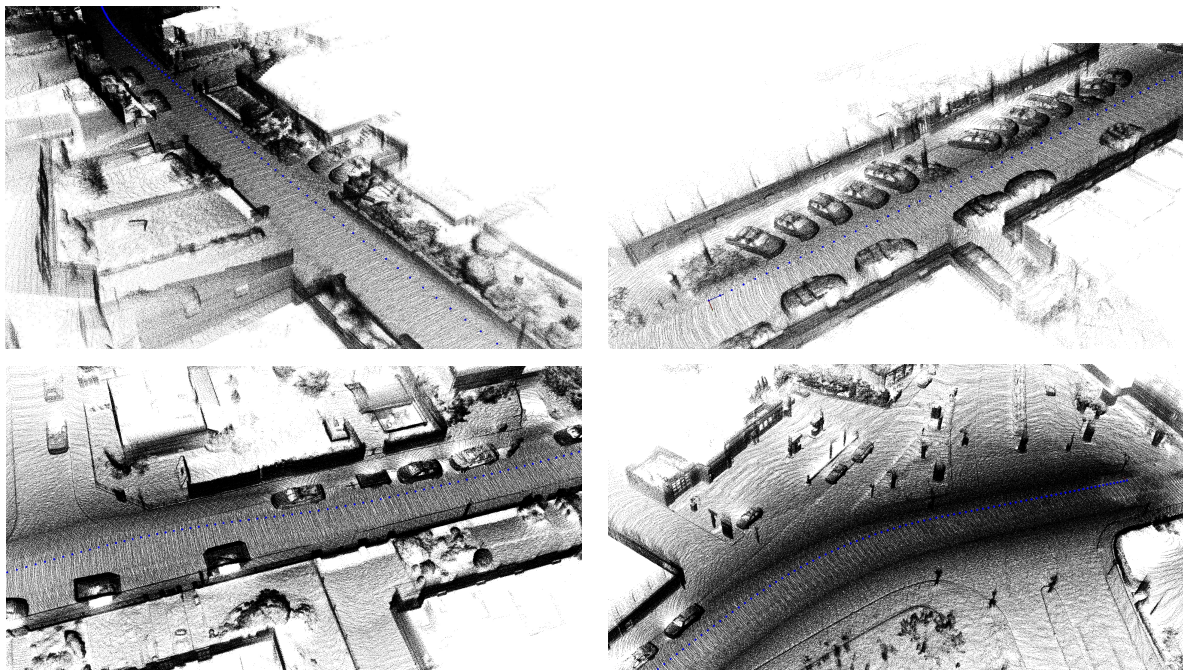


Figure 21. Results of the Kitti vision benchmark showing 3D reconstructions of the validation sequences “11”, “13”, “15” and “18” (left to right, top to bottom) for which no ground truth is provided.

6. Conclusions

In this paper, a novel mobile mapping system using a multi-modal sensor set-up was presented. The platform is unique in the sense that it copes with the inhomogeneity of the point clouds produced by LiDAR scanners. To this end, it integrates both an intensive surface analysis, as well as a global map

that is continuously updated and improved. Moreover, loop detection is conducted using the output of the Ladybug images, and a loop closure technique was proposed based on the output of the LiDAR odometry. Experiments demonstrated that our system is able to reconstruct challenging environments, such as chemical plants, and can compete with state of the art methods concerning LiDAR mapping in the field of autonomous vehicles; cf. the Kitti vision benchmark.

Acknowledgments: This research is part of the project entitled *Generic Interoperability Platform for Augmented Reality Applications* (www.iminds.be/en/projects/gipa) (GiPA), an ICON (www.iminds.be/en/calls/icon) project co-funded by iMinds, a digital research institute founded by the Flemish Government. The project partner is Grontmij, with project support from IWT (www.iwt.be/). The work was made possible by a cooperative agreement that was administered by engineer Pieter Raes, project manager at Dow Benelux. The ground truth was provided by the team of Steven Couwels, co-founder of Real to Desk (R2D).

Author Contributions: The research in this work was mainly conducted by Michiel Vlamincx, M.Sc. in Computer Science and PhD candidate. The research is supervised by Hiep Luong and Wilfried Philips. They provided ideas, inputs and feedback to improve the work. The development of the acquisition platform, i.e., the construction and synchronisation of the sensors, was done by Werner Goeman.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Mur-Artal, R.; Montiel, J.M.M.; Tardós, J.D. ORB-SLAM: A Versatile and Accurate Monocular SLAM System. *IEEE Trans. Robot.* **2015**, *31*, 1147–1163.
2. Caruso, D.; Engel, J.; Cremers, D. Large-Scale direct SLAM for omnidirectional cameras. In Proceedings of the 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Hamburg, Germany, 28 September–2 October 2015; pp. 141–148.
3. Billings, S.D.; Boctor, E.M.; Taylor, R.H. Iterative Most-Likely Point Registration (IMLP): A Robust Algorithm for Computing Optimal Shape Alignment. *PLoS ONE* **2015**, *10*, e0117688.
4. Pomerleau, F.; Colas, F.; Siegwart, R.; Magnenat, S. Comparing ICP Variants on Real-world Data Sets. *Auton. Robot.* **2013**, *34*, 133–148.
5. Han, J.; Yin, P.; He, Y.; Gu, F. Enhanced ICP for the Registration of Large-Scale 3D Environment Models: An Experimental Study. *Sensors* **2016**, *16*, 228, doi:10.3390/s16020228.
6. Zlot, R.; Bosse, M. Efficient Large-Scale 3D Mobile Mapping and Surface Reconstruction of an Underground Mine. *Field Serv. Robot.* **2012**, *92*, 479–493.
7. Bosse, M.; Zlot, R.; Flick, P. Zebedee: Design of a Spring-Mounted 3-D Range Sensor with Application to Mobile Mapping. *IEEE Trans. Robot.* **2012**, *28*, 1104–1119.
8. Nüchter, A.; Lingemann, K.; Hertzberg, J.; Surmann, H. 6D SLAM—3D Mapping Outdoor Environments: Research Articles. *J. Field Robot.* **2007**, *24*, 699–722.
9. Hong, S.; Ko, H.; Kim, J. VICP: Velocity updating iterative closest point algorithm. In Proceedings of the 2010 IEEE International Conference on Robotics and Automation (ICRA), Anchorage, Alaska, 3–8 May 2010; pp. 1893–1898.
10. Gressin, A.; Mallet, C.; Demantké, J.; David, N. Towards 3D LiDAR point cloud registration improvement using optimal neighborhood knowledge. *ISPRS J. Photogramm. Remote Sens.* **2013**, *79*, 240–251.
11. Moosmann, F.; Stiller, C. Velodyne SLAM. In Proceedings of the IEEE Intelligent Vehicles Symposium, Baden-Baden, Germany, 5–9 June 2011; pp. 393–398.
12. Sarvrood, Y.B.; Hosseinyalamdary, S.; Gao, Y. Visual-LiDAR Odometry Aided by Reduced IMU. *ISPRS Int. J. Geo Inf.* **2016**, *5*, doi:10.3390/ijgi5010003.
13. Segal, A.; Hähnel, D.; Thrun, S. Generalized-ICP. In *Robotics: Science and Systems*; Trinkle, J., Matsuoka, Y., Castellanos, J.A., Eds.; The MIT Press: Cambridge, MA, USA, 2009.
14. Biber, P.; Straßer, W. The normal distributions transform: A new approach to laser scan matching. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, 27 October–1 November 2003; pp. 2743–2748.
15. Sun, Y.X.; Li, J.L. Mapping of Rescue Environment Based on NDT Scan Matching. In Proceedings of the 2nd International Conference on Computer Science and Electronics Engineering (ICCSEE 2013), Hangzhou, China, 22–23 March 2013; Volume 760, pp. 928–933.

16. Magnusson, M.; Lilienthal, A.; Duckett, T. Scan registration for autonomous mining vehicles using 3D-NDT. *J. Field Robot.* **2007**, *24*, 803–827.
17. Einhorn, E.; Gross, H.M. Generic NDT Mapping in Dynamic Environments and Its Application for Lifelong SLAM. *Robot. Auton. Syst.* **2015**, *69*, 28–39.
18. Zhang, J.; Singh, S. LOAM: LiDAR Odometry and Mapping in Real-time. In Proceedings of the Robotics: Science and Systems Conference (RSS), Berkeley, CA, USA, 13–15 July 2014.
19. Zhang, J.; Singh, S. Visual-LiDAR Odometry and Mapping: Low-drift, Robust, and Fast. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Seattle, Washington, DC, USA, 26–30 May 2015.
20. Pathak, K.; Birk, A.; Vaskevicius, N.; Poppinga, J. Fast Registration Based on Noisy Planes with Unknown Correspondences for 3D Mapping. *IEEE Trans. Robot.* **2010**, *26*, 424–441.
21. Grant, W.; Voorhies, R.; Itti, L. Finding Planes in LiDAR Point Clouds for Real-Time Registration. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Tokyo, Japan, 3–7 November 2013; pp. 4347–4354.
22. Xiao, J.; Adler, B.; Zhang, J.; Zhang, H. Planar Segment Based Three-dimensional Point Cloud Registration in Outdoor Environments. *J. Field Robot.* **2013**, *30*, 552–582.
23. Low, K.L. *Linear Least-Squares Optimization for Point-to Plane ICP Surface Registration*; Technical Report 4; University of North Carolina: Chapel Hill, NC, USA, 2004.
24. Levin, D. The Approximation Power of Moving Least-squares. *Math. Comput.* **1998**, *67*, 1517–1531.
25. Galvez-Lopez, D.; Tardos, J.D. Bags of Binary Words for Fast Place Recognition in Image Sequences. *IEEE Trans. Robot.* **2012**, *28*, 1188–1197.
26. Shoemake, K. Animating Rotation with Quaternion Curves. *SIGGRAPH Comput. Graph.* **1985**, *19*, 245–254.
27. Geiger, A.; Lenz, P.; Urtasun, R. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR), Providence, RI, USA, 16–21 June 2012.
28. Wu, C. Towards Linear-Time Incremental Structure from Motion. In Proceedings of the 2013 International Conference on 3D Vision, Sydney, Australia, 1–8 December 2013; pp. 127–134.



© 2016 by the authors; licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC-BY) license (<http://creativecommons.org/licenses/by/4.0/>).