GND-LO: Ground Decoupled 3D Lidar Odometry Based on Planar Patches

Andres Galeote-Luque, Jose-Raul Ruiz-Sarmiento, and Javier Gonzalez-Jimenez

***Abstract*—We present a new fast Ground Decoupled 3D Lidar Odometry (GND-LO) method. The particularity of GND-LO is that it takes advantage of the distinct spatial layout found in urban settings to efficiently recover the lidar movement in a decoupled manner. For that, the input scans are reduced to a set of planar patches extracted from the flat surfaces of the scene, found aplenty in these scenarios. These patches can be labeled as either belonging to the ground or walls, decoupling the estimation into two steps. First, the ground planes from each scan, clustered from the ground patches, are registered. Then, the motion estimation is completed by minimizing the distance between the wall patches and their corresponding points from the other scan, whose pairing is iteratively updated. GND-LO has demonstrated to perform both precisely and efficiently beating state-of-the-art approaches. Concretely, experiments on the popular KITTI dataset show that our proposal outperforms its competitors by reducing the average drift by 19% in translation and 4% in rotation. This is achieved by running in real-time without needing GPU or optimized multi-threading, as it is commonplace in the literature.**

***Index Terms*—Localization, range sensing, autonomous vehicle navigation, range odometry, 3D lidar.**

I. Introduction

3D *Lidar Odometry* consists of estimating the relative pose of a moving vehicle by registering consecutive *scans* provided by a 3D lidar sensor. These sensors stand out for yielding accurate geometric information from the scene while performing reliably under changes in lighting conditions, which frequently occur in real-world scenarios [1]. Given the advantages 3D lidars offer, further amplified by the continued development of more advanced sensors, their popularity among autonomous vehicles has been on the rise in recent years. Examples of this are the autonomous driving platforms found in state-of-the-art projects such as Waymo [2], [3], and Argoverse [4], [5], as well as in datasets such as KITTI [6] and KAIST Urban [7].

Among the diversity of environments in which autonomous vehicles are designed to navigate, it is worth emphasizing the relevance of cities, which are characterized by having an abundance of planar surfaces, mainly from the ground and the walls of surrounding buildings or other structures. This raises a distinctive layout of orthogonal planes that can be exploited for motion estimation.

In this letter, we propose GND-LO, a 3D lidar odometry method that takes advantage of the planar surfaces commonly found in urban environments. This allows us to decouple the movement estimation into two steps which considerably simplifies the optimization problem and the associated computational burden. In turn, this simplification leads to a smaller drift in the trajectory when complying with real-time realization since now the optimization can be upscaled with more input data. For achieving such decoupling, the method extracts planar features, or *patches*, by applying a quadtree segmentation to the range image representation of each input scan, taking into consideration the curvature around every point. The directional dependence of the set of selected patches is analyzed, and low-quality redundant patches are discarded (*culled*). Since those 3D lidars are usually mounted such that their field of view is angled down, it is safe to assume that most of the resulting patches belong to the ground. This allows for a subset of the patches to be clustered together to parametrize the ground plane, which can then be registered with the ground plane extracted from the previous scan. Hence, the pitch and roll w.r.t. the previous ground plane, as well as the translation perpendicular to it can be recovered in a very efficient way. The rest of the patches are used to recover the remaining motion variables, namely the translation along the two axes of the ground plane and the rotation around its normal vector. This is performed by minimizing the point-to-plane distance between each patch and its corresponding point, whose pairing is obtained by reprojecting the center of the patch onto the opposite range image and iteratively updated based on the current motion estimation [8]. Note that this avoids the need to solve for the correspondences as is common in the literature [9]. Given the efficiency introduced by decoupling, patches are extracted from both input scans and leveraged in the optimization algorithm: every patch from the latest scan is matched to a point from the previous scan, and vice versa. This effectively doubles the information available in the scene, improving accuracy while keeping real-time execution. Fig. 1 displays the different stages in the pipeline of GND-LO.

The main contributions of our proposal are:

- Detection of the planar patches of the scene by applying a quadtree segmentation to the range image representing

Manuscript received 25 April 2023; accepted 1 September 2023. Date of publication 7 September 2023; date of current version 19 September 2023. This letter was recommended for publication by Associate Editor S. Scherer and Editor S. Behnke upon evaluation of the reviewers' comments. This work was supported in part by Spanish Government the under Grant PRE2018-085026 and in part by Spanish Government through Research Project ARPEGGIO under Grant PID2020-117057GB-I00. *(Corresponding author: Andres Galeote-Luque.)*

The authors are with the Machine Perception and Intelligent Robotics (MAPIR) Group, Malaga Institute for Mechatronics Engineering and Cyber-Physical Systems (IMECH.UMA), University of Malaga, 29071 Malaga, Spain (e-mail: andresgalu@uma.es; jotaraul@uma.es; javiergonzalez@uma.es).

Digital Object Identifier 10.1109/LRA.2023.3313057

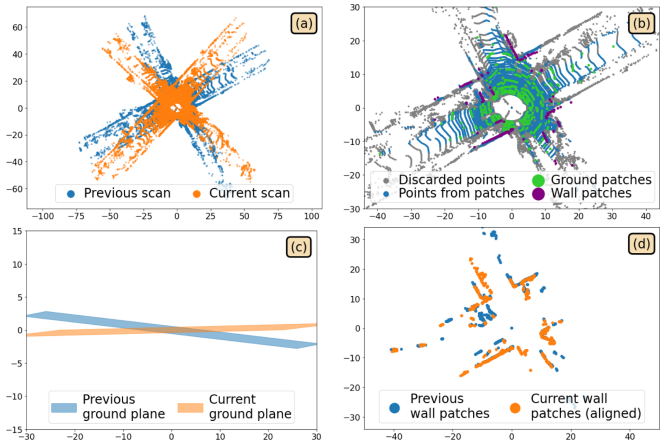

Fig. 1. Different stages of the pipeline of GND-LO: (a) shows the two input scans, before registration; (b) ground and wall patches are extracted from each scan; (c) the ground patches of each scan are compacted into a single plane, and (d) both are then registered, leaving a 2D motion problem, which is solved by minimizing the point-to-plane distance between wall patches and their corresponding points, thus completing the scan registration. The code has been published and can be found at https://github.com/MAPIRlab/GND-LO.

the scan, taking into consideration the flatness around each point.

- A direction-based downsampling of the planar patches, removing low-quality patches from populated directions to reduce noise, referred from now on as *culling*.
- A decoupled motion estimation that exploits the local planarity of the ground to greatly simplify the registration between the ground planes of consecutive scans. This leaves extra computational resources to the later optimization problem, which furthermore needs to solve only 3 out of the original 6 degrees-of-freedom (DOFs).
- An extensive comparison of the proposed method with state-of-the-art methods on urban sequences of the KITTI dataset [6], showing better performance than them in terms of drift.

II. Related Work

3D lidar odometry has been traditionally treated as an extension of 3D registration, since the relative pose between two consecutive scans can be estimated by registering their point cloud representations. In this regard, the Iterative Closest Point (ICP) algorithm [10], characterized for being a simple but effective method, has dominated the research. Over the years, different variations of ICP have claimed a place in the literature by improving the basic algorithm in different ways. Examples of this are point-to-plane ICP [11], generalized ICP (GICP) [12], and multi-channel GICP [13]. It is only normal that numerous state-of-the-art 3D lidar odometry methods are built upon ICP and its variants, each improving on the original with different contributions, like ELO [14], LOCUS 2.0 [15], SGLO [16], and Reinke et al. [17], for instance. Despite its clear popularity, ICP exhibits two main limitations, namely its computational cost and its tendency to fall into local minima if the problem is not correctly initialized.

Biber and Strasser introduced in 2003 the normal distributions transform (NDT) [18], which allows matching two scans from 2D lidars without finding correspondences between points, unlike ICP. In 2007, Magnusson et al. expanded this concept for 3D lidars [19], offering an alternative to the predominant ICP [20]. Although the grid representation introduced by these NDT-based methods allows for a memory- and time-efficient optimization, it also comes at the cost of having discontinuous cost functions that can hinder the optimization.

Another category of 3D lidar odometry algorithms inherits its core concept from Visual Odometry (VO) [21] literature: *feature-based* methods consist of reducing the input scans to a set of features (like points, lines or planes) and, after correspondences between the features in both scans have been computed, the motion is recovered by minimizing the distance of each pair. Notable works in this category include Velas et al. [22] for choosing line features, and DiLO [23], which performs bundle adjustment with keypoints located on edges and planes along multiple consecutive scans. The right operation of these methods depends on a correct selection and matching of features, and thus solving correspondences becomes essential. Instead, in our proposal the planar patches are paired with points iteratively according to the current motion estimation, reducing the computation time and allowing room for error in the matching.

In recent years, the rise of Deep Learning (DL) in general and Convolutional Neural Networks (CNNs) in particular has resulted in the advancement of various research topics that benefit from said technology. 3D lidar odometry is one of those topics, and works like [1], [24], [25], [26] are proof that this type of methods can achieve high performance. These approaches are reliant on high-performance computers, even requiring a GPU since both the training and the normal operation (inference) are computationally expensive. In contrast, the algorithm proposed in this letter still performs in real-time when running on mobile platforms with limited computational resources.

It is worth mentioning the relevance of 3D lidar odometry methods as the front end of *Lidar Odometry and Mapping* (LOAM) and *Simultaneous Localization and Mapping* (SLAM), which create a map of the surrounding while providing the ego-location. By employing said map, for example performing scan-to-map registration, the drift over time is greatly reduced. Since these algorithms usually depend on an odometry method to provide the scan-to-scan motion, they can be categorized in a similar manner. Two main groups stand out in the literature, those that leverage feature-based algorithms (e.g. see [27], [28], [29], [30], [31], [32], [33]), and those whose localization algorithm is a variation of ICP (e.g. see [34], [35], [36]). Given the number of parallel processes running in a LOAM or SLAM system, it becomes highly important that the front-end odometry is computationally inexpensive. The proposed method GND-LO recovers the movement between consecutive scans employing minimal resources, thus becoming a suitable candidate as the front-end in this kind of algorithms.

Various methods in the literature achieve a better accuracy and efficiency by performing a ground segmentation on the input scans. For example, in [29], [33] ground features are separated from the rest to decouple the motion estimation in two

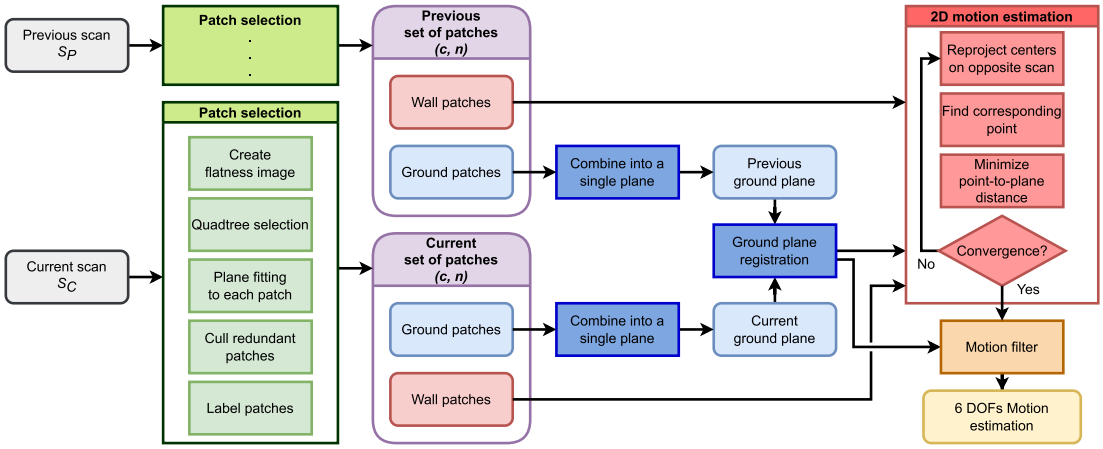

Fig. 2. Workflow of GND-LO. Data blocks have round corners to distinguish them from process blocks.

steps, each providing half of the DOFs. Similarly, the previously mentioned ELO [14] performs ICP to ground and non-ground features independently. In [37], the ground segmentation is used to reject non-planar features on the ground. In our proposal, not only the motion estimation is decoupled into two steps, but the ground registration is performed faster and more efficiently by simply clustering the ground planar patches into a single plane, removing the need of employing an optimization approach.

III. Method Overview

In this section, we will go over the pipeline followed by our proposal, which is shown in Fig. 2. Basically, it consists of leveraging selected patches from the scene to estimate the relative motion between two consecutive input scans. It becomes clear that the quality of the selected sets of patches will have a great impact on the movement estimation procedure. Therefore, special consideration is taken in the selection process in order to reach high accuracy.

The input to the algorithm is a pair of consecutive scans, referred to as *previous* (S_P) and *current* (S_C) from now on. These scans consist of an ordered point cloud, and so they can also be represented as a range image $S_P, S_C : (\Omega \subset \mathbb{R}^2) \rightarrow \mathbb{R}$. Taking advantage of this fact, the first step is to create a flatness (low-curvature) image from each scan to help discern which points from the scene belong to planar surfaces. A quadtree segmentation is applied to said images to extract the regions located on flat surfaces of the scene. The group of points included in each selected region of the scans is fit into a planar patch defined by its center c and normal vector n (see Section III-A). Once the initial set of patches has been extracted from each scan, denoted by $\mathbf{P}_P$ and $\mathbf{P}_C$ respectively, their directional-independence is analyzed. For the motion to be fully observable, each set needs to include patches oriented in three independent directions. The distribution of normal vectors of the patches can be checked via PCA, and then patches that provide redundant information can be culled to reduce the computational cost, while keeping patches in less populated directions (see Section III-B). As the final step in the patches' selection, they are labeled as either belonging to the ground, walls, or outliers (Section III-C). Fig. 3 shows the different stages of the selection process, from the original range image to the final selection of patches, labeled as ground or wall. Of course, once the first two scans in a sequence have been processed, this selection process is only done for each new input scan.

Once each input scan has been reduced to a set of patches with enough information in three independent directions, the motion between the two scans can be estimated. Since the patches are labeled, the motion can be decoupled into two steps, each using the ground and wall patches respectively. In the first stage, the ground patches from each set are combined into a single ground plane. Three of the total six DOFs of the 3D motion can be recovered by registering the obtained ground planes (see Section III-D). The rest of the motion corresponds to the 2D motion that "slides" over the ground plane. To retrieve it, each wall patch is reprojected onto the opposite image to find its matching point, and the transformation is estimated by minimizing point-to-plane distances between pairs. The correspondence of each patch is calculated as an iterative procedure that makes use of the latest estimation of the motion (Section III-E). Finally, a motion filter is applied to the resulting estimation, leveraging previous instances of movement to reduce the uncertainty of the solution. For that, a similar approach as in [38] and [8] has been implemented.

A. Patches Selection Via Quadtrees

This section details the process to obtain the initial set of patches ($\mathbf{P}_P$ and $\mathbf{P}_C$) from each input scan (S_P and S_C), which is portrayed in Fig. 3. The objective is for each set of patches to contain the most informative flat areas of the scene, and to this end, the first step is estimating how flat the area around each point of the scan is. This is accomplished by analyzing the curvature around each pixel in the range image representation of the scan. Since the curvature of a function can be approximated by its second derivative [39], the pixel-wise flatness $\mathcal{F}(u, v)$ can be calculated as the sum of the squared second-order derivatives in each direction, which can be efficiently obtained by applying

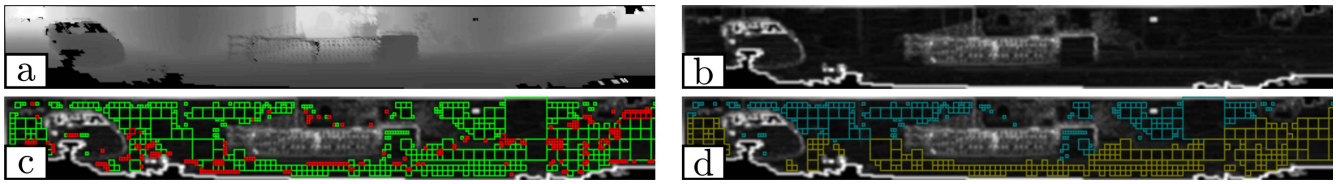

Fig. 3. Different stages of the extraction and classification of planar patches from a lidar scan. From the original range image (a), the flatness image is created (b). Quadtree segmentation is applied to it and an initial set of patches is selected (c). Based on the directional independence of the set, the redundant patches are culled (red patches). Finally, patches are labeled as either ground or wall (d), shown as yellow and blue respectively. Note that only half of the scan is shown for clarity. The scene captured in these images is a road (visible in both extremes of the image) going over a bridge (identified by its fence), with a couple of building corners on each side of it and a parked car located on the left side.

the Sobel operator.

$$\mathcal{F}(u,v) = \left[\frac{\partial^2 S}{\partial u^2}(u,v)\right]^2 + \left[\frac{\partial^2 S}{\partial v^2}(u,v)\right]^2 \tag{1}$$

We call this the flatness image, as pixels with a low value in this image represent points whose surroundings exhibit a low curvature and therefore are more likely located on a flat surface of the scene. See Fig. 3(b) as an example.

The resulting flatness image is then leveraged to select groups of points from the scan located on flat surfaces, from which the patches will be created. There are two main advantages associated with selecting patches with a big supporting group of points: reducing the impact of noise, and improving the matching in the future motion estimation process. The former is rather self-explanatory, as a bigger support region means the error of individual points (due to noise, occlusions, or similar) has less impact on the planar patch estimation. The latter is linked to the iterative projection-based matching employed in the 2D motion estimation, where patches with a bigger area are more likely paired with a point that belongs to the same planar surface of the scene, even after big movements.

For this reason, a quadtree segmentation is applied to the flatness image to find the most relevant and flat regions distributed along the scan, which are selected to create patches. Quadtree segmentation is a popular technique employed in image processing to efficiently decompose an image into rectangular subsets of similar pixels [40]. The general operation of quadtrees consists of dividing the input image into blocks of a certain size. The variation of the pixel intensities inside each block is then analyzed, and it is divided into four if the difference is over a threshold. This procedure is repeated for the smaller blocks until the image has been divided into rectangular regions containing groups of pixels with similar values. This procedure is seen in Algorithm 1, and the results are shown in Fig. 3(c). By applying it to the flatness image, neighboring pixels with a similar curvature get grouped into blocks of varying sizes, each one representing a plane from the scene. The main advantage of quadtree segmentation is grouping large flat areas from the scene into a single block, in contrast to the fixed-size block segmentation in [8], while keeping a low execution time compared to more advanced segmentation algorithms. The 3D points belonging to the blocks that present an average flatness over a threshold are then grouped into a planar patch, with its center c being the centroid of the points. The normal vector n of the patch is calculated as the eigenvector associated with the smallest eigenvalue of the covariance matrix of the group.

Algorithm 1: Proposed Quadtree Selection Algorithm.

Input: Flatness image $\mathcal{F}$
Output: Set of initial patches $\mathbf{P}$
$sz \leftarrow$ initial block size (32);
$B \leftarrow$ division of $\mathcal{F}$ into blocks of size sz;
repeat
  foreach *block* $B_i \in B$ *of size* sz **do**
    if $std(B_i) > max\_std$ **then**
      $B \leftarrow$ division of B_i into blocks of size $sz/2$;
    else if $avg(B_i) < max\_avg$ **then**
      $\mathbf{P} \leftarrow$ patch from points in block B_i;
  end
  $sz \leftarrow sz/2$
until $sz <= min\_size$;

B. Direction-Based Culling

The initial set of patches after applying the selection algorithm from the previous section is composed of patches of different sizes, flatness, and orientations. Previously, the significance of working with big patches has been explained. On the other hand, small patches are noisy and usually belong to surfaces that can only be considered flat in a small neighborhood. In this section, we describe the culling process implemented to discard low-quality patches while maintaining enough information to keep the vehicle movement observable.

For the motion to be completely determined, the selected set must include patches oriented in three independent directions [41]. This can be analyzed through PCA of the matrix M built with the normal vectors:

$$M_{3\times 3} = \sum_{i}^{N} n_i n_i^T = \overline{n}\,\overline{n}^T, \tag{2}$$

where n_i is the normal vector of patch i, N is the total number of patches, and $\overline{n}$ is the $(3 \times N)$ matrix of normal vectors.

The eigenvectors of M represent the principal directions of the patch orientations, and the eigenvalues provide an approximation of how many patches are oriented in (or more accurately, how many "contribute" to) each of these directions. This lets us identify those directions with redundant information and

those where every patch provides significant information. Small patches are removed if they provide redundant information in highly populated directions, and kept otherwise. An example of the culled patches can be seen in Fig. 3(c).

C. Patch Labeling

As a final step before starting the decoupled motion estimation, the patches are labeled as either ground, wall, or outliers. Although there are several techniques in the literature to segment the ground plane from a point cloud, they tend to be computationally expensive. In GND-LO, we propose a fast and simple yet effective way of patch classification. A patch is considered to belong to the ground if the angle difference between the normal vectors of the patch and the ground plane from the previous scan is below a threshold. This simple process works properly as long as the ground normal vector does not change abruptly along the trajectory. To initialize the ground plane, it is first considered to be perfectly parallel to the XY plane.

As for the patches located in walls, a similar strategy is applied. A wall patch must satisfy that its angular difference to the previous ground plane is $90°\pm$ a threshold. The patches that are not labeled as ground or wall are considered outliers and rejected. Refer to Fig. 3(d) for a representation of the final set of labeled patches.

D. Ground Clustering and Registration

Now we elaborate on the first stage of the decoupled motion estimation, in which the ground planes extracted from each input scan are registered. Although the patches have already been labeled, they still need to be combined into a single ground plane. Different approaches can be applied to that end, but we found that the best strategy is to obtain the average of the centers and normal vectors weighted by the number of points contained within each patch.

Once the ground planes are defined by their center c_G and normal vector n_G, the transformation that registers them comes from the following two equations. The rotation r_G, expressed in axis-angle $(a,\ \alpha)$ notation, is defined as the cross product between their normal vectors:

$$r_G = n_G^C \times n_G^P, \qquad (a = r_G/\alpha, \quad \alpha = ||r_G||)\,. \tag{3}$$

Note that superscript is used to indicate belonging to previous $(^P)$ or current $(^C)$ scan. The translation t_G can be obtained as the distance between the ground planes after applying the rotation r_G. The distance d between two parallel planes can be calculated as the difference between their distances to the origin. Since the rotation center is the origin, the distance to the origin of each plane does not change after applying the rotation r_G. Hence, the translation can be calculated independently of the rotation as the distance d along the axis n_G^P:

$$t_G = d n_G^P = \left(n_G^P \cdot c_G^P - n_G^C \cdot c_G^C\right) n_G^P. \tag{4}$$

Note that the translation is only determined along the axis of n_G^P. Similarly, the rotation r_G represents the minimal rotation to align the normal vectors of the ground planes, but the rotation around n_G^P remains undetermined.

E. 2D Motion Estimation

In this section, we cover how the wall patches are exploited to recover the remaining three DOFs, which corresponds to a 2D movement on the ground plane. This is achieved by minimizing the point-to-plane distance between each patch (c, n) and its corresponding point p from the opposite scan, whose pairing is found by reprojecting the center of the patch onto the opposite range image. Note that reprojection depends on the relative pose between the scans, thus the matches are iteratively updated based on the latest motion estimation. The optimization problem is solved using the Ceres [42] implementation of Levenberg-Marquardt. To further explain the algorithm, the following notation will be used:

- N_P, N_C: total number of patches included in $\mathbf{P}_P$ and $\mathbf{P}_C$ respectively. Note that in this section $\mathbf{P}$ will refer only to patches labeled as walls since the ground patches have already fulfilled their purpose.
- $i \in [1, N_P], j \in [1, N_C]$: iterators used to go through each set of patches.

The step-by-step procedure is now described:

1) Initialize the rigid transformation T as the one obtained by registering the ground planes: $T = T_G = T(r_G, t_G)$.
2) Reproject the center c of each patch, after applying the corresponding transformation, onto the opposite image making use of the projection function $\pi : \mathbb{R}^3 \to \Omega$, to obtain its pixel coordinates. This function depends on the sensor used, but generally consists of converting a 3D point (first argument) to spherical coordinates, which can then be employed to locate the corresponding pixel location on the range image (second argument):
$$\text{px}_i^C = \pi\left(T^{-1} c_i^P,\ S_C\right), \qquad \text{px}_j^P = \pi\left(T c_j^C, S_P\right) \tag{5}$$
3) The point p associated to each patch is the one located on the pixel coordinates px of the opposite range image. The inverse projection function $\pi^{-1} : \Omega \to \mathbb{R}^3$ is used.
$$p_i^C = \pi^{-1}\left(\text{px}_i^C,\ S_C\right), \qquad p_j^P = \pi^{-1}\left(\text{px}_j^P,\ S_P\right) \tag{6}$$
4) Once matching is completed, the point-to-plane distance of each pair is minimized to obtain the optimal transformation. In the following equation, ρ is the Huber loss function, which helps with convergence when there are outliers within the data; and T is not a full 6 DOFs 3D transformation, but is instead generated from one rotation (around n_G^P) and two translations (along two orthogonal axes perpendicular to n_G^P).
$$T^* = \arg\min_T \rho\left(\sum_i^{N_P}\left[n_i^P \cdot \left(c_i^P - T p_i^C\right)\right] + \sum_j^{N_C}\left[n_j^C \cdot \left(c_j^C - T^{-1} p_j^P\right)\right]\right) \tag{7}$$
5) Update T by composition: $T = T T^*$.
6) Check for convergence: if the sum of the differences between the pixel coordinates px after updating the transformation (δ) is below a threshold, the estimation has

converged. Otherwise, repeat from step 2. In the following equation, k represents the iteration number.

$$\delta = \sum_{i}^{N_P} \left| px^C_{i,k-1} - px^C_{i,k} \right| + \sum_{j}^{N_C} \left| px^P_{j,k-1} - px^P_{j,k} \right| \quad (8)$$

Finally, the relative pose between the two input scans, a full 3D transformation with 6 DOFs, has been estimated and the pose of the vehicle can be updated.

IV. Experiments and Results

In this section, we evaluate the performance of our method GND-LO on sequences of 3D lidar scans from the widely-used KITTI dataset [6] (Section IV-A). The obtained results are compared with different state-of-the-art algorithms, including ELO [14], MULLS [32], SuMa [35], CLS [22], GICP and P2P-ICP [17] (Sections IV-B and IV-C). For all the experiments, the test platform used is a standard PC with 8 GB of RAM, running Ubuntu 20.04 with an Intel Core i7-7700HQ CPU with four cores and eight execution threads at 2.8 GHz. We also present an ablation study exploring different configurations of the proposed method (Section IV-D).

A. Testbed

The proposed method was evaluated on the urban sequences included in the KITTI dataset [6]. The Velodyne HDL-64E sensor, running at 10 Hz, provides point clouds that are converted to 1000 $\times$ 64 range images. The error metric employed to compare the different localization algorithms is the one proposed by the dataset, in which the trajectory error is evaluated in all subsequences of different lengths (100, 200,..., 800). The error is calculated as the translational drift [m] and angular error [°] accumulated every 100 m of trajectory. Hence the measurement units are percentage in the case of translational error (m/100 m = %), and °/100 m in the case of rotational error.

B. Execution Time Performance

3D lidar sensors usually work at 10 Hz, with some devices reaching up to 20 Hz. Thus, for an odometry method to perform in real-time, it has to estimate the motion from the input scans in 100 ms or less. It is desirable for odometry algorithms to operate even at a higher frequency since the computational resources of an autonomous vehicle have to be shared between the different processes running simultaneously.

The methods found in the literature are often parallelized and rely on a GPU to perform in real-time, especially deep-learning odometry algorithms. The proposed approach GND-LO, without any special parallelization or optimization, can run at 14 Hz on average, allowing for real-time operation even on resource-constrained platforms.

Table I shows a breakdown of the time taken by the different processes in the proposed workflow. It is evident how the matching and 2D motion estimation are the two most computationally demanding tasks, which is expected given they are part of an iterative process that takes 7.5 iterations on average before convergence.

TABLE I
Time Breakdown of the Proposed Method

Flatness image creation	6
Patch selection with quadtrees	5.5
Plane fitting to each patch	13
Patch labeling	0.14
Culling set of patches	7.5
Ground plane clustering	0.08
Ground planes registration	0.03
Patch-point Matching	10.5
2D motion estimation	28
Motion filter	0.10
Total	70.85 ms

TABLE II
Average Amount of Extracted Planar Patches Per Frame, Based on Their Size (pixels) and Label

Size		2	4	8	16	32	**Total**
Outlier	[%]	95	0	0	0	0	55.67
Ground	[%]	0	74	81	96	86	31.32
Wall	[%]	5	26	19	4	14	13.01
All	[n°]	782.00	434.39	104.46	11.89	1.03	1333.77

Culled patches are included as outliers.

Table II includes the average number of planar patches found in each scan, and their distribution based on size and label, to better establish the scale of the problem. Note that only wall patches are employed in the 2D motion optimization, the most time consuming part of the pipeline. Ground patches are usually big and abundant, in contrast to wall patches. The effect of the culling is very noticeable, as 95% of the small patches are discarded, keeping only those that are needed to completely constrain the movement.

C. Accuracy Results

Most of the localization algorithms that evaluate themselves via the KITTI dataset are LOAM or SLAM, which clearly outperform simple odometry methods by exploiting information from a map or even performing loop closure. Additionally, deep learning algorithms are usually trained with the same sequences that serve as evaluation, which influences the measured accuracy.

In favor of a fair comparison, and although GND-LO achieves competitive results with those methods in some cases, only non-DL odometry methods that estimate the motion in a frame-to-frame fashion have been considered in the study. This includes MULLS [32], CLS [22], SuMa [35], ELO [14], GICP, and P2P-ICP [17]. Note that while both MULLS and SuMa are SLAM methods, they also report the accuracy of their frame-to-frame odometry module, which is added to this comparison. Table III shows the error obtained from each method by applying the error metric mentioned before. It should be noted that the "Avg." column from the table is the result of averaging the different sequences weighted by their length, given by their number of scans.

It can be seen how the proposed method exhibits a higher accuracy in three of the four test sequences, and clearly outperforms the competing methods in the total average. As for a qualitative

TABLE III
ACCURACY COMPARISON BETWEEN THE DIFFERENT METHODS EVALUATED ON THE KITTI DATASET

	Sequence n° scans		00 4540	05 2760	06 1100	07 1100	Avg.
Method	MULLS	[%]	2.36	2.19	1.12	1.65	2.085
		[°/100m]	1.06	1.01	0.51	1.27	1.006
	CLS	[%]	2.11	1.98	0.92	1.04	1.811
		[°/100m]	0.95	0.92	**0.46**	0.73	0.859
	SuMa	[%]	2.1	1.5	1.0	1.8	1.764
		[°/100m]	0.9	0.8	0.6	1.2	0.871
	ELO	[%]	1.14	0.75	0.76	**0.60**	0.920
		[°/100m]	0.52	0.43	0.51	**0.48**	0.488
	GICP	[%]	1.28	1.18	0.87	0.87	1.156
		[°/100m]	0.61	0.60	0.53	0.57	0.593
	P2P-ICP	[%]	1.57	1.32	1.12	0.95	1.373
		[°/100m]	0.77	0.76	0.76	0.62	0.749
	GND-LO	[%]	**0.85**	**0.59**	**0.71**	0.71	**0.744**
		[°/100m]	**0.49**	**0.36**	0.47	0.66	**0.468**

The bold values represent minimum values for each sequence.

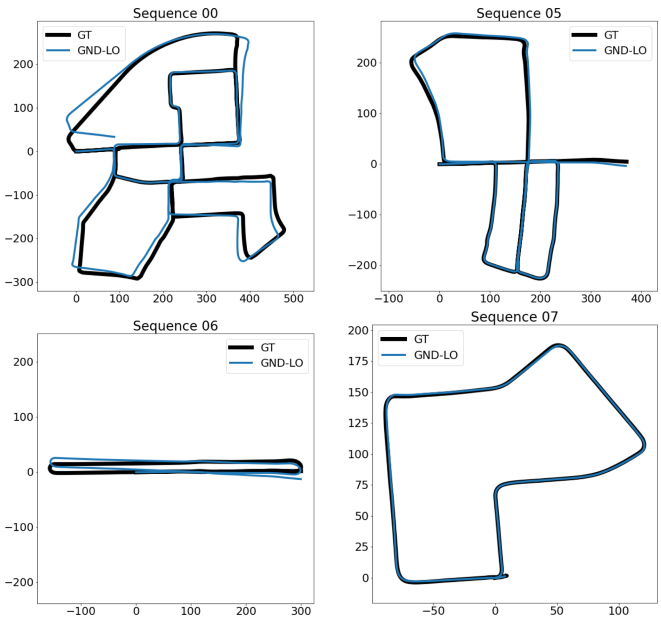

Fig. 4. Top view of the resulting trajectories over each urban sequence of the dataset.

evaluation of the proposal, Fig. 4 includes the trajectories of each sequence. Sequence 06 can be singled out for its higher drift, since less planar information is present along its trajectory. For short and well determined scenarios, as sequences 05 and 07 confirm, the drift is almost unnoticeable for an odometry method. Sequence 00 is the longest of the group, and thus significant drift can be observed near the end of the trajectory, but it is still a good result for odometry.

D. Ablation Study

In this section we study the impact on the accuracy of GND-LO of the different modules of the pipeline. Specifically, four major components are evaluated: i) the motion filter, ii) the culling algorithm, iii) the usage of planar patches from both input scans, and iv) the decoupling of the motion estimation.

TABLE IV
ABLATION STUDY OF THE PROPOSED METHOD GND-LO

	Sequence n° scans		00 4540	05 2760	06 1100	07 1100	Avg.
Method	GND-LO (original)	[%]	**0.85**	**0.59**	**0.71**	0.71	**0.744**
		[°/100m]	0.49	**0.36**	**0.47**	0.66	**0.468**
	No motion filter	[%]	0.86	**0.59**	0.77	0.70	0.752
		[°/100m]	**0.47**	0.37	0.48	0.67	**0.468**
	No culling	[%]	**0.85**	0.65	0.96	0.67	0.786
		[°/100m]	**0.47**	0.42	0.67	0.62	0.498
	Patches from single scan	[%]	1.68	0.88	7.17	1.83	2.097
		[°/100m]	0.86	0.45	2.93	0.79	0.975
	No decoupling	[%]	1.06	0.87	1.18	**0.57**	0.961
		[°/100m]	0.59	0.52	0.67	**0.53**	0.575

The bold values represent minimum values for each sequence.

The results of this ablation study are displayed in Table IV. Although the unaltered method presents the lowest average error, some of the evaluated techniques slightly worsen the accuracy on some sequences. There is a pattern where the evaluated components produce the most improvement on sequence 06. This sequence, despite being in an urban environment, presents the least flat structures along its trajectory, and thus estimating the motion becomes increasingly challenging. Every part of the algorithm helps dealing with these low-information scenarios, and thus are more noticeable on sequence 06. Adding information from both scans is the most noticeable improvement, more so on this sequence, where every little piece of information becomes crucial to correctly determine the relative pose.

Without decoupling, the method estimates the full 3D motion by minimizing the distance between every patch (ground and wall) and its corresponding point. Although its higher accuracy on sequence 07, on average it is still worth decoupling. In addition, the execution time of this "no decoupling" implementation is 106 ms, which exceeds the limit for real-time operation. To keep its frequency above 10 Hz, only patches from a single scan must be employed, which drives the error up 27% and 53% on translation and rotation respectively. This evidences how the motion decoupling not only reduces the drift but also makes the method more efficient.

V. CONCLUSION

In this letter, we have introduced GND-LO, a novel method for 3D lidar odometry that exploits the information found in the flat surfaces abundant in urban settings. After a careful selection of planar patches, the localization can be calculated efficiently and with low drift by decoupling the motion estimation into two stages. First, the ground planes of consecutive scans are registered; then, the 2D motion on the ground plane is recovered by minimizing the point-to-plane distance between pairs of features whose matching is iteratively updated.

To test its suitability, our approach has been evaluated in the KITTI dataset and compared to odometry methods from the literature. Not only does our proposal achieve a lower drift over the trajectory, but also runs in real-time without the need of a

GPU or multi-threading like state-of-the-art competitors. This allows for accurate localization while sharing computational resources with other processes running simultaneously, as is desirable for odometry methods.

Based on the results obtained in this work, it is our opinion that this method would benefit from implementing techniques inherited from LOAM or SLAM algorithms, like scan-to-map registration or loop closure, to further reduce the drift over long sequences.

Even though the use of planar features has proven to be worthwhile, it comes with the limitation of being unable to recover the motion in scenes lacking geometrical information from 3 independent directions. This will be solved in future works by also exploiting a different type of sensor, like an IMU or an RGB camera.

GND-LO code has been made public and can be found in https://github.com/MAPIRlab/GND-LO.

REFERENCES

[1] P. Adis, N. Horst, and M. Wien, "D3DLO: Deep 3D LiDAR odometry," in *Proc. IEEE Int. Conf. Image Process.*, 2021, pp. 3128–3132.

[2] P. Sun et al., "Scalability in perception for autonomous driving: Waymo open dataset," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 2443–2451.

[3] S. Ettinger et al., "Large scale interactive motion forecasting for autonomous driving: The waymo open motion dataset," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 9690–9699.

[4] M.-F. Chang et al., "Argoverse: 3D tracking and forecasting with rich maps," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 8740–8749.

[5] B. Wilson et al., "Argoverse 2: Next generation datasets for self-driving perception and forecasting," 2021, *arXiv:2301.00493*.

[6] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1231–1237, 2013.

[7] J. Jeong, Y. Cho, Y.-S. Shin, H. Roh, and A. Kim, "Complex urban dataset with multi-level sensors from highly diverse urban environments," *Int. J. Robot. Res.*, vol. 38, no. 6, pp. 642–657, 2019.

[8] A. Galeote-Luque, J.-R. Ruiz-Sarmiento, and J. Gonzalez-Jimenez, "Efficient 3D lidar odometry based on planar patches," *Sensors*, vol. 22, no. 18, 2022, Art. no. 6976.

[9] M. Elhousni and X. Huang, "A survey on 3D LiDAR localization for autonomous vehicles," in *Proc. IEEE Intell. Veh. Symp.*, 2020, pp. 1879–1884.

[10] P. J. Besl and N. D. McKay, "Method for registration of 3-D shapes," in *Sensor Fusion IV: Control Paradigms and Data Structures*, vol. 1611, Bellingham, WA, USA: Int. Soc. Opt. Photon., 1992, pp. 586–606.

[11] Y. Chen and G. Medioni, "Object modelling by registration of multiple range images," *Image Vis. Comput.*, vol. 10, no. 3, pp. 145–155, 1992.

[12] A. Segal, D. Haehnel, and S. Thrun, "Generalized-ICP," in *Proc. Robot., Sci. Syst.*, Seattle, USA, Jun. 2009, vol. 2, no. 4.

[13] J. Servos and S. L. Waslander, "Multi-channel Generalized-ICP: A robust framework for multi-channel scan registration," *Robot. Auton. Syst.*, vol. 87, pp. 247–257, 2017.

[14] X. Zheng and J. Zhu, "Efficient LiDAR odometry for autonomous driving," *IEEE Robot. Automat. Lett.*, vol. 6, no. 4, pp. 8458–8465, Oct. 2021.

[15] A. Reinke et al., "Locus 2.0: Robust and computationally efficient LiDAR odometry for real-time underground 3D mapping," *IEEE Robot. Automat. Lett.*, vol. 7, no. 4, pp. 9043–9050, Oct. 2022.

[16] S. Liang, Z. Cao, P. Guan, C. Wang, J. Yu, and S. Wang, "A novel sparse geometric 3-D lidar odometry approach," *IEEE Syst. J.*, vol. 15, no. 1, pp. 1390–1400, Mar. 2021.

[17] A. Reinke, X. Chen, and C. Stachniss, "Simple but effective redundant odometry for autonomous vehicles," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 9631–9637.

[18] P. Biber and W. Strasser, "The normal distributions transform: A new approach to laser scan matching," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2003, pp. 2743–2748.

[19] M. Magnusson, A. Lilienthal, and T. Duckett, "Scan registration for autonomous mining vehicles using 3D-NDT," *J. Field Robot.*, vol. 24, no. 10, pp. 803–827, 2007.

[20] M. Magnusson, A. Nuchter, C. Lorken, A. J. Lilienthal, and J. Hertzberg, "Evaluation of 3D registration reliability and speed-a comparison of ICP and NDT," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2009 pp. 3907–3912.

[21] D. Nistér, O. Naroditsky, and J. Bergen, "Visual odometry," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 1, 2004, pp. I-652–I-659.

[22] M. Velas, M. Spanel, and A. Herout, "Collar line segments for fast odometry estimation from velodyne point clouds," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2016, pp. 4486–4495.

[23] S.-J. Han, J. Kang, K.-W. Min, and J. Choi, "DiLO: Direct light detection and ranging odometry based on spherical range images for autonomous driving," *ETRI J.*, vol. 43, no. 4, pp. 603–616, 2021. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.4218/etrij.2021-0088

[24] Q. Li et al., "Lo-Net: Deep real-time lidar odometry," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 8465–8474.

[25] Z. Li and N. Wang, "DMLO: Deep matching LiDAR odometry," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 6010–6017.

[26] G. Wang, X. Wu, S. Jiang, Z. Liu, and H. Wang, "Efficient 3D deep LiDAR odometry," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 5, pp. 5749–5765, May 2023.

[27] J. Zhang and S. Singh, "LOAM: Lidar odometry and mapping in real-time," in *Proc. Robot., Sci. Syst.*, 2014, pp. 1–9.

[28] J. Zhang and S. Singh, "Visual-lidar odometry and mapping: Low-drift, robust, and fast," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2015, pp. 2174–2181.

[29] T. Shan and B. Englot, "LeGO-LOAM: Lightweight and ground-optimized lidar odometry and mapping on variable terrain," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 4758–4765.

[30] J.-E. Deschaud, "IMLS-SLAM: Scan-to-model matching based on 3D data," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 2480–2485.

[31] W. S. Grant, R. C. Voorhies, and L. Itti, "Efficient velodyne SLAM with point and plane features," *Auton. Robots*, vol. 43, no. 5, pp. 1207–1224, 2019.

[32] Y. Pan, P. Xiao, Y. He, Z. Shao, and Z. Li, "MULLS: Versatile LiDAR SLAM via multi-metric linear least square," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 11633–11640.

[33] D.-U. Seo, H. Lim, S. Lee, and H. Myung, "PaGO-LOAM: Robust ground-optimized LiDAR odometry," in *Proc. IEEE 19th Int. Conf. Ubiquitous Robots*, 2022, pp. 1–7.

[34] F. Moosmann and C. Stiller, "Velodyne SLAM," in *Proc. IEEE Intell. Veh. Symp.*, 2011, pp. 393–398. [Online]. Available: http://www.mrt.kit.edu/z/publ/download/Moosmann_IV11.pdf

[35] J. Behley and C. Stachniss, "Efficient surfel-based SLAM using 3D laser range data in urban environments," in *Proc. Robot.: Sci. Syst.*, Pittsburgh, Pennsylvania, Jun. 2018.

[36] M. Yokozuka, K. Koide, S. Oishi, and A. Banno, "LiTAMIN2: Ultra light LiDAR-based SLAM using geometric approximation applied with KL-divergence," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 11619–11625.

[37] P. Chen, W. Shi, S. Bao, M. Wang, W. Fan, and H. Xiang, "Low-drift odometry, mapping and ground segmentation using a backpack LiDAR system," *IEEE Robot. Automat. Lett.*, vol. 6, no. 4, pp. 7285–7292, Oct. 2021.

[38] M. Jaimez and J. Gonzalez-Jimenez, "Fast visual odometry for 3-D range sensors," *IEEE Trans. Robot.*, vol. 31, no. 4, pp. 809–822, Aug. 2015.

[39] P. J. Flynn and A. K. Jain, "On reliable curvature estimation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 1989, pp. 110–116.

[40] H. Samet, "The quadtree and related hierarchical data structures," *ACM Comput. Surv.*, vol. 16, no. 2, pp. 187–260, 1984.

[41] Y. Taguchi, Y.-D. Jian, S. Ramalingam, and C. Feng, "Point-plane SLAM for hand-held 3D sensors," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2013, pp. 5182–5189.

[42] S. Agarwal et al., "Ceres solver," 2022. [Online]. Available: http://ceres-solver.org