Review on 3D Lidar Localization for Autonomous Driving Cars.

Mahdi Elhousni and Xinming Huang

Abstract— **LIDAR sensors are bound to become one the core sensors in achieving full autonomy for self driving cars. LIDARs are able to produce rich, dense and precise spatial data, which can tremendously help in localizing and tracking a moving vehicle. In this paper, we review the latest finding in 3D LIDAR localization for autonomous driving cars, and analyze the results obtained by each method, in an effort to guide the research community towards the path that seems to be the most promising.**

I. INTRODUCTION

Lately, autonomous driving has become one of the most researched topics in the scientific community. The race toward full autonomous driving cars, or level 6 autonomy such as it categorized by SAE International, is mainly fueled by the hope of eliminating the human errors when it comes to driving. This could have a huge impact on our daily lives : According to a report by The Department of Transportation of the USA, self-driving cars could reduce traffic fatalities by up to 94%.

In order for a car to drive autonomously, the first challenge that a traditional pipeline would try to solve is to localize the car. Localization in this context means : finding the position and orientation of the vehicle inside of a map.

Defining what a map means is also important : In this paper, we focus on Autonomous Driving Cars (ADC) with a perception system consistent only of a LIght Detection and Ranging (LIDAR) sensor : a sensor that uses laser light in order to measure distances and is capable of producing PointClouds, which are a 3D representation of space where each point contains the (x, y, z) coordinates of the surface that reflected the laser beam originating from the LIDAR sensor. The maps used to localize the ADC have to match the available sensor data, which has given rise to PointCoud based maps. These maps can either be constructed beforehand by concatenating successive LIDAR scans offline, or during the navigation process by optimizing and taking advantage of the odometry that could be generated using the LIDAR data and combining that with a loop closure mechanism. The later is usually known as Simultaneous Localization And Mapping (SLAM). Both approaches have advantages and disadvantages that will be discussed in details.

We decided to focus on localization using 3D LiDAR for the following reasons : LIDAR data, compared to other perception sensors, is the richest and most detailed in term of spacial information. This results in the LIDAR sensor being practically always more accurate when it comes to solving spatial based challenges such as localizing a vehicle. We also observed that the price of this sensor is going down constantly (from $75000 a few years ago to less than $5000 today), making this sensor more accessible to the public and more affordable for car manufacturers. However, localization with 3D LIDARs can face multiple issues, which usually revolve around efficiency and real-time execution : Since LIDAR data tends to be heavy in size (containing sometimes up to 15000 individual points in each scan at a frequency of 10Hz), processing the data in a quick fashion to guarantee the real-time necessities of ADCs can be challenging and usually demands an efficient processing pipeline with some sort of downsampling or feature extraction method.

Generating an odometry measurement when localizing a vehicle is an essential step. Multiple approches have been proposed over the year to utilize the 3D data from LIDARs to calculate the displacement of a subject or robot, which we decided to separate in three distinct categories :

3D Registration Based Methods : Usually combined with a map that was built offline, these methods take advantage of the advances that were achieved in 3D pointclouds registration [1]. While being very accurate, these types of approaches tend to be too slow to achieve real-time processing when relying on LIDAR data only. These methods can be seen as "dense" methods, since they take advantage of all the points present in the LIDAR data.

3D Features Based Methods : Inspired by the popular methods relying on 2D feature extraction and matching [2], [3], [4], these approaches design relevant features in the 3D space that are then used to calculate the displacements between successive scans. The accuracy and real-time processing of these methods are satisfactory, however, poor results are to be expected when dealing with rough maneuvers or high speed movements. These methods can be seen as "sparse" method, since they only use a select number of points in the LIDAR data.

3D Deep Learning Based Methods : The use of Deep Learning for solving localization challenges has been gaining more and more popularity lately. 2D camera images were first used to try and predict the odometry between a pair of images [5], [6], [7], [8] with results that were more or less acceptable, but still not outperforming the state of the art. Lately, more works have been exploring the use of LIDAR data instead, with results that seem more promising.

Results reported by the different publications in these three categories will be compared on the KITTI Odometry Dataset [9], being the most popular benchmark in the field. The goal of this survey is to review and present the most relevant work related to 3D LIDAR localization, compare the different results reported in the literature and discuss the strengths and weaknesses of each of them.

II. 3D Lidar Localization for Autonomous Driving Cars

We will review and discuss all the methods available in the literature where the 3D localization of a moving vehicle was achieved using a 3D LIDAR sensor only. We separated the available methods in three categories (Registration, 3D features and Deep learning), and listed them in Table 1.

TABLE I
3D LIDAR LOCALIZATION METHODS.

REGISTRATION	3D FEATURES	DEEP LEARNING
ICP [10], [11]	LSF [12]	DL-LBO [13]
NDT [14], [15]	PoseMap [16]	DeepPCO [17]
SALO [18]	CPFG-SLAM [19]	DeepLO [18]
IMLS-SLAM [20]	PLANES-SLAM [21], [22]	LocNet [23]
SuMa [24]	LOAM [25], [26], [27], [28]	LORAX [29]
SuMa++ [30]		3D-SegMap [31], [32], [33], [34], [35]
Proba-SURFEL [36]		LO-Net [37]
CLS-SLAM [38]		BIAS-COR [39]
DLO [40]		L3-Net [41]
		DeepICP [42], [43]
		CAE-LO [44]

A. 3D Registration based methods

In this section, we review the 3D localization methods based on the 3D point clouds registration approaches. Registration transforms a pair of point clouds in order to align them in the same coordinates frame, making it possible to deduct the transformation between both scans. In the context of ADC localization, registration can be used in two ways : (1) By combining the incoming scans with portions of a pre-built point cloud map in order to localize the vehicle, or (2) By combining successive LIDAR scans in order to calculate the odometry of the vehicle.

3D point cloud registration is mostly used in the areas of shape alignment and scene reconstruction, with the Iterative Closest Point (ICP) algorithm [45], [46] being one of the most popular. In the ICP algorithm, a transformation between a source and target point cloud is iteratively optimized by minimizing an error metric between the points of both point clouds. Multiple variants of the algorithm were developed [47], such as : point-to-line ICP [48], point-to-plane ICP [49] and Generalized ICP [10]. The ICP algorithm was the standard for many years to solve point cloud registration tasks, and methods that incorporate it in the localization pipeline were designed, such as in [11], where it is combined with a loop closure mechanism and a pose graph building process in order to reduce the accumulated errors from the consecutive registrations. More recently in [50], an odometry pipeline is proposed which integrates the knowledge about the Lidar sensor physics and improved the ICP algorithm with a novel downsampling and point matching rejection methods : The downsampling of the Lidar scans is done using a Normal Covariance Filter (NCF) which keeps only the points with precises normals. On the other hand, the outlier rejection when matching points is achieved with a Geometric Correspondance Rejector : By taking advantage of the rings structure of the Lidar scans and the normals that were previously calculated, the author defines a threshold named the highest neighbor beam distance, which is used

as matching rejection criteria. When plugging both of these method to the ICP algorithm, the author reports a 27% drop in the drift of the odometry on the KITTI dataset.

However, the ICP algorithm was eventually surpassed by the 3D Normal Distribution Transform (NDT) algorithm [14], [51]. First developed to assist autonomous mining vehicles, the 3D NDT is a point cloud registration algorithm that extend the 2D NDT algorithm to the 3D space. Similarly to the ICP algorithm, a transformation between a source and target point cloud is iteratively optimized. But in this case, the error that is being minimized is not between pairs of points, instead, the point clouds are first transformed into a probability density function (PDF), based on the mean and covariance of the points present in pre-computed voxels. This PDF can then be used with Newton's algorithm to find the spatial transformation between them. An extension to the 3D NDT algorithm was proposed in [52] and named Probabilistic NDT, which attempts to deal with the sparsity of the classical NDT representation, resulting from its hard threshold on the number of points in a voxel necessary for it to be considered. This is done by computing the mean and covariance usually needed in the 3D NDT, not based on the number of points, but on their probability, which generates a much denser distribution and results in a bump in accuracy in both translation and rotation.

Methods such as the ICP algorithm or the 3D NDT algorithm can produce very accurate transformations between LIDAR pairs, however, in the context of ADC, these methods rarely meet the real-time execution criteria. Also, for these methods to be accurate, an initial guess is usually needed to start the optimization process and avoid local minimas. This usually means having to use an additional sensor (such as an IMU [15], [53]) to produce an odometry that could be used as an initial guess, and thus does not fit our LIDAR only setup.

In a more classical SLAM approach, the authors in [20] propose a 3 steps algorithm named IMLS-SLAM : First is the dynamic object removal which is simplified to clustering of the scans and removal of small clusters. Second step is to apply a sampling strategy based on the observability of each point in order to down-sample the scan, then finally the matching step where the transformation is optimized by following a scan-to-model matching strategy, using the Implicit Moving Least Square (IMLS) representation.

Another popular pre-processing approach that could be applied to the scans before attempting to register them is to compute the surfel (SURFace ELement) representation of the point clouds. In [24], a surfel map is being built while the incoming scan are being converted into vertex and normal maps that are used to calculate the odometry of the vehicle with a so-called frame-to-model ICP algorithm. The surfel map is then used to find loop closure candidates in order to optimize the trajectory of the vehicle and minimize the drift. An extension to this method was proposed in [30], where a semantic segmentation of a spherical projection of the LI-DAR data is used to remove dynamic objects and improve the frame-to-frame matching by enforcing semantic constraints

on the frame-to-model ICP algorithm. In [36], two different surfel representations are computed : an ellipsoid surfel map (ESM) and a disk surfel map (DSM). The ESM, due to it's sparsity, is only used for localization. On the other, the DSM, which is much denser that the ESM is used to reconstruct the surrounding environment.

In the same spirit, Collar Line Segments (CLS) construction is a useful pre-processing method that makes it possible to achieve a good level of accuracy when aligning point clouds : In [38], the Lidar scans are transformed into line clouds, by sampling line segments between neighbouring points from neighbouring rings. These line clouds are then aligned using an iterative approach : First, the center points of the generated lines are calculated. These points are then used to find the transformation between successive scans by finding the lines in the target pointcloud whose center is closest to the lines in the source pointcloud. Additional post processing tricks are then used to boost the accuracy, such as using previous transformations to predict and initialize the next pose estimation step.

Sometimes, reducing the dimensionality of the LIDAR data can also yield reasonable results, such as in [40] where the incoming scans are projected onto a 2.5D grid map with occupancy and height. This grid map is equivalent to gray scale image which is used to register the scans based on the photo-metric errors as it is usually done with camera data [54].

B. 3D Features based methods

In this section, we tackle the 3D localization methods based on 3D features extraction and matching. 3D features [55], [56], [57], [58] are interest points that represent recognizable areas that are consistent in time and space, such as corners and planes. Commonly used for 3D object detection tasks, these features are usually represented using a unique vector called feature descriptor, which can be used to match features in two different point clouds. By finding sufficient and consistent matches, we can calculate the transform between scans using an optimization method and thus construct an odometry measurement.

In [12], the authors propose a study that focuses on finding what type of data and features should be observed when trying to achieve accurate localization of an ADC. The authors here argue that features have to be built and extracted based on the distribution of clusters of points. However, their experiments show that the distribution of points changes drastically from one scene to another, making this method very instable.

In an approach named PoseMap, which was proposed in [16], the authors argue that a 'coherent' map representation of the environment is not necessary to achieve high quality localization : The method takes advantages of a pre-build pointcloud map using [59], which is subsampled based on an overlap threshold in order to produce a simple, sparse and light representation of the environment where key poses are maintained. This map representation can be seen as a collection of submaps which can be updated independently

from each other, at different points in time. The localization is then solved using a sliding window approach by simply using the two closest submaps to the current vehicle position and minimizing the distance between the old features and the new ones.

Geared toward off-road environment, the method proposed in [19] and named CPFG-SLAM is inspired by the ICP and NDT algorithms and relies on 3D features and a probability grid map. By taking advantage of the nearest neighbor in the grid instead of the nearest neighbor point, the authors are able to matches and registers point cloud onto the grid map more efficiently. The Expectation-maximization (EM) algorithm is used to estimate the pose, while the final optimization problem is solved using the Levenberg-Marquardt algorithm.

Other localization methods try take advantage of predominant geometries that are present in the environment where the ADC will be moving : In [21] and [22], plane extraction algorithms are combined with frame-to-frame techniques in order to produce a pose estimation for the vehicle. When compared with the results obtained by the ICP algorithm, plane extraction and alignment methods show great improvement in both accuracy and speed.

While being a pure engineering solution that sometimes requires to tweak and adapt multiple parameters, 3D features based localization methods have been known to generate impressive results in both accuracy and speed : Currently holding the first spot in the KITTI odometry leaderboard, the method proposed in [25] starts by extracting planar and corner features based on the smoothness and occlusion of the points. These features are matched with patches of points in the following scan and the Levenberg-Marquardt method is then used to solve the LIDAR motion. As it is usually done in most SLAM pipelines, a map is also being built in the background at a slower frequency than the odometry estimation, which helps with improving the final localization results. An extension to this method was proposed in [26] in order to improve its speed and guarantee real-time aspect of the odometry calculation. The main improvements reside in taking advantage of the presence of the ground by removing unreliable features and using a two-step Levenberg-Marquardt method to speed up the optimization step. Still, one of the main remaining issues of the LOAM pipeline is the odometry drift due to the accumulated errors. However, plugging in a loop closure mechanism to the pipeline can solve this issue as it was shown in [28] or [27].

C. 3D Deep Learning based methods

In this section, we review the 3D localization methods based on deep learning. While still being a very young approach to odometry and localization estimation, the use of deep learning has been gaining more popularity lately after it proved being very promising in the camera domain, and after methods such as PointNet [60] and PoinetNet++ [60] showed how efficient deep neural networsk can be when attempting to solve 3D pointclouds related challenges. Usually formulated as a regression problem, methods involving deep learning can either try to solve this task in an end-

to-end fashion by using the raw pointclouds as inputs and directly predicting the displacements of the vehicle using a single network, or by trying to substitute certain parts of the pre-established classical pipelines that could benefit for the generalizations possible with deep learning networks.

One of the first method that proposed to solve this task using a deep learning approach is [13] : The idea here is to try and take this challenge back to the image domain instead of attempting to solve it directly in the 3D pointcloud one, in order to simplify the data input of the network. Incoming LIDAR frames are first projected onto the 2D space to produce panoramic depth images which are then fed into a simple 2 branch convolution network, in an attempt to regress the values of the displacement and change in orientation of the vehicle between the two input frames. The results obtained by the authors where subpar when compared with the state of the art. However, they were able to prove that exploring the use of deep learning to solve this task could eventually lead to better results.

Panoramic depth images are a popular representation of the LIDAR data, and another method that makes use of them is DeepPCO [17]. The projected LIDAR frames are fed to a 2 branch network where the first branch predict the translation of the vehicle, while the second one predicts the rotation.

Another method that attempts to simplify the input data by projecting onto the 2D space is the one presented in [18]. Here, the LIDAR frames are projected using the spherical coordinate system to generate two new 2D representations : a vertex map (representing the location (x, y, z) of each point) and a normal map (representing the values of the normals of each point). The proposed network is mainly composed residual blocks and has two major branches : First one, named VertexNet, takes as input the vertex maps and is used to predict the translation between subsequent frames. The second branch, named NormalNet, takes as input the normal maps and is designed to predict the rotation between two subsequent frames. The output of both branches is then combined in order to construct the full transformation between the two LIDAR frames. In order to train the full network in and end-to-end fashion, the authors propose two different training schemes with two different loss function, based on the availability of labeled data : First is a classical supervised loss, where the labeled data is compared with the network prediction in order to optimize the weights of the network, and second is the unsupervised loss, where no labeled data is needed and the ICP algorithm is used to guide the network toward the correct motion predictions.

More recently, a solution coined CAE-LO was proposed in [44] where an unsupervised convolution auto-encoder was used to extract features from spherical projections of LIDAR data in a multi-scale fashion. An aditional Auto-Encoder is used to generate the feature descriptors which are then used to match points using RANSAC based frame to frame matching. Finally, the ICP algorithm is used to refine the odometry results.

In [23], and in an effort to simply the input data again, a handcrafted rotational invariant representation (RIR) based on the rings distribution of the point clouds is presented. The authors claim that thanks to this representation, the global localization problem is reformulated to an identity verification one. This was solved by using a siamese network named LocNet, which takes as input 2 subsequent RIR and aims to optimize a contrastive loss function [61]. The output of LocNet is a dimension reduced feature vector that is used later in complete SLAM pipeline, where the MCL [62] and ICP algorithm are used to generate the final transformation in a coarse to fine manner.

In [29], the LORAX algorithm was proposed. This approach introduces the notion of super-points, a subset of points located inside of a sphere and describing a local surface, which are projected onto the 2D space to form 2D depth maps. These depth maps are then filtered using a series of test to leave only the relevant super-points and encoded using a PCA and Deep Auto-Encoder. Candidates for matching are then selected based on the euclidien distance between features before engaging in a coarse registration step where an iterative approach involving the RANSAC algorithm is used. As a final step, and in order to fine tune the results of the registration step, the ICP algorithm is used to improve the accuracy of the whole pipeline.

In a series of papers [32], [31], [33], [34] that eventually led to the final 3D SegMap method [35], the authors explore how to efficiently extract and encode segments from pointclouds using simple convolution networks, with the hope of solving localization and mapping related tasks. The main contribution of this approach is its data driven 3D segment descriptor which is extracted using a network composed of a series of convolutional and fully connected layers. The descriptor extractor network is trained using a loss function composed of two parts : a classification loss and reconstruction one. Finally the extracted segments and their candidate correspondences are found using the kNearest Neighbors (k-NN) algorithm, which makes it possible to solve the localization task. Note that the 3D SegMap descriptor is a versatile descriptor which can also be used to solve other tasks such object classification.

Most of the method discussed previously will inevitably suffer from the presence of dynamic objects (cars, pedestrians ...etc.) in the scene when trying to regress the motion between two frames. Removing dynamic object in the scene has been known to improve the odometry results in most SLAM pipelines. However, detecting then deleting the dynamic objects from the scene in a supervised manner introduces an extra level of complexity which could lead to higher processing times and unstable results. In order to solve this issues in an unsupervised manner, the authors in [37] have proposed to train an encoder-decoder branch for the task of dynamic mask prediction. This is done by optimizing geometric consistency loss function, which indicates areas where geometric consistency can be modeled thanks to the normals of the point cloud data. The full network (named LO-Net) can be trained in an end-to-end fashion by combining the geometric consistency loss, the odometry regression loss and a cross-entropy loss for regularization purposes.

Method	Sequences											
	00	01	02	03	04	05	06	07	08	09	10	Avg
G-ICP [10]	1.29/0.64	4.39/0.91	2.53/0.77	1.68/1.08	3.76/1.07	1.02/0.54	0.92/0.46	0.64/0.45	1.58/0.75	1.97/0.77	1.31/0.62	1.91/0.73
SALO [50]	0.91/0.72	1.13/0.37	0.98/0.45	1.76/0.50	0.51/0.17	0.56/0.29	0.48/0.13	0.83/0.51	1.33/1.43	**0.64/0.30**	0.97/0.41	0.95/0.80
CLS-SLAM [38]	2.11/0.95	4.22/1.05	2.29/0.86	1.63/1.09	1.59/0.71	1.98/0.92	0.92/0.46	1.04/0.73	2.14/1.05	1.95/0.92	3.46/1.28	2.13/0.82
SuMa [24]	0.3/0.7	0.5/1.7	0.4/1.1	0.5/0.7	0.3/0.4	0.2/0.5	0.2/0.4	0.4/1.0	0.3/0.5	0.3/0.7	0.3/0.7	0.3/0.7
SUMA++ [30]	**0.22**/0.64	0.46/1.60	0.37/1.00	0.46/0.67	0.26/0.37	0.20/0.40	0.21/0.46	0.19/0.34	0.35/1.10	0.23/0.47	0.28/0.66	0.29/0.70
IMLS-SLAM [20]	-/0.50	-/0.82	-/0.53	-/0.68	-/0.33	-/0.32	-/0.33	-/0.33	-/0.80	-/0.55	-/0.53	-/0.55
LOAM [25]	0.78/0.53	1.43/0.55	0.92/0.55	0.86/0.65	0.71/0.50	0.57/0.38	0.65/0.39	0.63/0.50	1.12/0.44	0.77/0.48	0.79/0.57	0.85/0.51
LO-Net [37]	1.47/0.72	1.36/0.47	1.52/0.71	1.03/0.66	0.51/0.65	1.04/0.69	0.71/0.50	1.70/0.89	2.12/0.77	1.37/0.58	1.80/0.93	1.09/0.63
DeepLO [18]	0.32/**0.12**	**0.16/0.05**	**0.15/0.05**	**0.04/0.01**	**0.01/0.01**	**0.11/0.07**	**0.03/0.07**	**0.08/0.05**	**0.09/0.04**	13.35/4.45	5.83/3.53	1.83/0.76
DeepPCO [17]	-/-	-/-	-/-	-/-	0.02/0.03	-/-	-/-	-/-	-/-	-/-	**0.02/0.06**	-/-
DeepICP [42]	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-	**0.071/0.164**
BIAS-COR [39]	-/1.42	-/1.96	-/0.86	-/0.83	-/0.48	-/0.53	-/0.41	-/0.75	-/1.00	-/1.00	-/1.35	-/1.02

Method	Translation Error (m)	Rotation Error (deg/m)	Runtime (s)
IMLS-SLAM [20]	0.69	0.0018	1.25s
CPFG-SLAM [19]	0.87	0.0025	**0.03**
SuMa++ [30]	1.06	0.0034	0.1
SuMa [24]	1.39	0.0034	0.1
LOAM [25]	**0.55**	**0.0013**	0.1
CAE-LO [44]	0.86	0.0025	2

Rather that learning how to localize a vehicle using the LIDAR frames directly, other methods attempt to learn the error model of a classical pipeline. In other words, deep learning can be used to correct the odometry measurements already available, resulting in a powerful and flexible plug in module. The authors in [39] have proposed to learn a bias correction term, aimed to improve the results of a classical state estimator that takes LIDAR data as input. The Gaussian Process Model was used to model the 6 odometry errors independently from each other with carefully selected input features that concentrated on the 3 DoF that are most affected by the errors.

In [41], a more advanced method named L3-Net was proposed, which can be linked to the bias correction theme, since instead of predicting the full transformation between frames, the authors here are proposing a network that attempts to learn the residual value between their traditional localization system and the ground truth. Relevant features are first being extracted and fed into a miniPointNet to generate their corresponding feature descriptors. A cost volume is then constructed in the solution space (x, y, z) and regularized with 3D convolutional neural networks. Additionally, an RNN branch is added to the network structured to guarantee the temporal smoothness of the displacements predictions.

A more complete and general variant of the L3-Net was proposed by the same authors in [42], [43] and named DeepICP. Here, the features are being extracted using Point-Net++, then filterd using a weighting layer that only keeps the most relevant ones. Similarly to the previous method, the features descriptors are computed using a miniPointNet structure then fed into a corresponding point generation layer, which generate the corresponding key points in the target point cloud. In order to regress the final value of the transformation, two loss function are combined, hoping to encode both the local similarities and global geometric constraints.

III. EVALUATION AND DISCUSSION

We compare the previously cited methods based on their reported results on the KITTI odometry benchmark [9], which is one of the most popular large scale dataset for outdoor odometry evaluation : It contains 22 sequences recorded using a Velodyne HDL-64E that was mounted on top of a car, with LIDAR scans that were already preprocessed to compensate for the motion of the vehicle. Ground truth is available for the 11 first sequences and was obtained using an advanced GPS/INS system.

Table II lists all the reported results on the training dataset, while Table III lists all the results on the test dataset that were reported on the KITTI official leaderbord. Note that we only consider the results that do not involve any loop closure mechanims. While LOAM still occupies the first position of the KITTI's leaderboard, it is clear from that the methods involving deep learning are becoming more and more accurate. As an exemple, DeepICP's reported average result outperform any other proposed method on the training dataset. However, It is hard for us to qualify them as "state of the art" methods for two main reason : (1) DeepICP is reporting that it takes around 2 seconds to register each pair of frames. This is too slow to be deloyed on a real autonomous driving car operating in real life situations, (2) The results of these approaches on the test dataset have not yet been reported. Good results on the test dataset would prove that these method are capable of being used in real scenarios, and not only on data that the deep neural networks have already seen. Until then, LOAM and its variant remain the best option and most trustworthy for real autonomous driving deployment.

IV. CONCLUSION

In this paper, we reviewed, analyzed, compared and discussed most of the recent advances and findings in the area of 3D LIDAR localization for autonomous driving cars. We considered systems where the only sensor used was a 3D LIDAR due to the increasing importance of this sensor in most accurate perception and localization systems nowdays,

in addition to the increase in its availability to the general public and manufacturers. The results on the KITTI odometry dataset reported by the cited papers where compiled and compared, leading us to the following statement: While deep learning based methods are shown to be producing very promising results and seem to represent the right path to follow in order to solve this challenge in the future, methods based on 3D feature detection and matching are still considered as state of the art due to their proven stability when deployed in real life scenarios.

REFERENCES

[1] G. K. Tam, Z.-Q. Cheng, Y.-K. Lai, F. C. Langbein, Y. Liu, D. Marshall, R. R. Martin, X.-F. Sun, and P. L. Rosin, "Registration of 3d point clouds and meshes: A survey from rigid to nonrigid," *IEEE transactions on visualization and computer graphics*, vol. 19, no. 7, pp. 1199–1217, 2012.

[2] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "Orb-slam: a versatile and accurate monocular slam system," *IEEE transactions on robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[3] R. Mur-Artal and J. D. Tardós, "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[4] D. Scaramuzza and F. Fraundorfer, "Visual odometry [tutorial]," *IEEE robotics & automation magazine*, vol. 18, no. 4, pp. 80–92, 2011.

[5] K. R. Konda and R. Memisevic, "Learning visual odometry with a convolutional network." in *VISAPP (1)*, 2015, pp. 486–490.

[6] S. Wang, R. Clark, H. Wen, and N. Trigoni, "Deepvo: Towards end-to-end visual odometry with deep recurrent convolutional neural networks," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 2043–2050.

[7] R. Clark, S. Wang, H. Wen, A. Markham, and N. Trigoni, "Vinet: Visual-inertial odometry as a sequence-to-sequence learning problem," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[8] N. Yang, R. Wang, J. Stuckler, and D. Cremers, "Deep virtual stereo odometry: Leveraging deep depth prediction for monocular direct sparse odometry," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 817–833.

[9] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.

[10] A. Segal, D. Haehnel, and S. Thrun, "Generalized-icp." in *Robotics: science and systems*, vol. 2, no. 4. Seattle, WA, 2009, p. 435.

[11] E. Mendes, P. Koch, and S. Lacroix, "Icp-based pose-graph slam," *2016 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pp. 195–200, 2016.

[12] K. Yoneda, H. T. Niknejad, T. Ogawa, N. Hukuyama, and S. Mita, "Lidar scan feature for localization with highly precise 3-d map," *2014 IEEE Intelligent Vehicles Symposium Proceedings*, pp. 1345–1350, 2014.

[13] A. Nicolai, R. Skeele, C. Eriksen, and G. A. Hollinger, "Deep learning for laser based odometry estimation."

[14] M. Magnusson, A. Lilienthal, and T. Duckett, "Scan registration for autonomous mining vehicles using 3d-ndt," *Journal of Field Robotics*, vol. 24, no. 10, pp. 803–827, 2007.

[15] B. Zhou, Z. Tang, K. Qian, F. Fang, and X. Ma, "A lidar odometry for outdoor mobile robots using ndt based scan matching in gps-denied environments," in *2017 IEEE 7th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER)*. IEEE, 2017, pp. 1230–1235.

[16] P. Egger, P. V. Borges, G. Catt, A. Pfrunder, R. Siegwart, and R. Dubé, "Posemap: Lifelong, multi-environment 3d lidar localization," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 3430–3437.

[17] W. Wang, M. R. U. Saputra, P. Zhao, P. Gusmao, B. Yang, C. Chen, A. Markham, and N. Trigoni, "Deeppco: End-to-end point cloud odometry through deep parallel neural network," *arXiv preprint arXiv:1910.11088*, 2019.

[18] Y. Cho, G. Kim, and A. Kim, "Deeplo: Geometry-aware deep lidar odometry," *arXiv preprint arXiv:1902.10562*, 2019.

[19] K. Ji, H. Chen, H. Di, J. Gong, G. Xiong, J. Qi, and T. Yi, "Cpfg-slam:a robust simultaneous localization and mapping based on lidar in off-road environment," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, June 2018, pp. 650–655.

[20] J. Deschaud, "IMLS-SLAM: scan-to-model matching based on 3d data," *CoRR*, vol. abs/1802.08633, 2018. [Online]. Available: http://arxiv.org/abs/1802.08633

[21] K. Pathak, A. Birk, N. Vaskevicius, M. Pfingsthorn, S. Schwertfeger, and J. Poppinga, "Online three-dimensional slam by registration of large planar surface segments and closed-form pose-graph relaxation," *Journal of Field Robotics*, vol. 27, no. 1, pp. 52–84, 2010.

[22] W. S. Grant, R. C. Voorhies, and L. Itti, "Finding planes in lidar point clouds for real-time registration," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 4347–4354.

[23] H. Yin, L. Tang, X. Ding, Y. Wang, and R. Xiong, "Locnet: Global localization in 3d point clouds for mobile vehicles," in *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018, pp. 728–733.

[24] J. Behley and C. Stachniss, "Efficient surfel-based slam using 3d laser range data in urban environments."

[25] J. Zhang and S. Singh, "Loam: Lidar odometry and mapping in real-time."

[26] T. Shan and B. Englot, "Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 4758–4765.

[27] X. Ji, L. Zuo, C. Zhang, and Y. Liu, "Lloam: Lidar odometry and mapping with loop-closure detection based correction," in *2019 IEEE International Conference on Mechatronics and Automation (ICMA)*, Aug 2019, pp. 2475–2480.

[28] J. Lin and F. Zhang, "A fast, complete, point cloud based loop closure for lidar odometry and mapping," 09 2019.

[29] G. Elbaz, T. Avraham, and A. Fischer, "3d point cloud registration for localization using a deep neural network auto-encoder," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017, pp. 2472–2481.

[30] X. Chen, A. M. E. Palazzolo, P. Giguère, J. Behley, and C. Stachniss, "Suma + + : Efficient lidar-based semantic slam," 2019.

[31] A. Cramariuc, R. Dubé, H. Sommer, R. Siegwart, and I. Gilitschenski, "Learning 3d segment descriptors for place recognition," *arXiv preprint arXiv:1804.09270*, 2018.

[32] R. Dube, D. Dugas, E. Stumm, J. Nieto, R. Siegwart, and C. Cadena, "Segmatch: Segment based place recognition in 3d point clouds," 05 2017, pp. 5266–5272.

[33] R. Dubé, M. G. Gollub, H. Sommer, I. Gilitschenski, R. Siegwart, C. Cadena, and J. Nieto, "Incremental-segment-based localization in 3-d point clouds," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1832–1839, 2018.

[34] R. Dub, A. Gawel, H. Sommer, J. Nieto, R. Siegwart, and C. Cadena, "An online multi-robot slam system for 3d lidars," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2017, pp. 1004–1011.

[35] R. Dubé, A. Cramariuc, D. Dugas, J. Nieto, R. Siegwart, and C. Cadena, "Segmap: 3d segment mapping using data-driven descriptors," *arXiv preprint arXiv:1804.09557*, 2018.

[36] C. Park, S. Kim, P. Moghadam, C. Fookes, and S. Sridharan, "Probabilistic surfel fusion for dense lidar mapping," *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, pp. 2418–2426, 2017.

[37] Q. Li, S. Chen, C. Wang, X. Li, C. Wen, M. Cheng, and J. Li, "Lo-net: Deep real-time lidar odometry," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8473–8482.

[38] M. Velas, M. Spanel, and A. Herout, "Collar line segments for fast odometry estimation from velodyne point clouds," *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4486–4495, 2016.

[39] T. Tang, D. Yoon, F. Pomerleau, and T. D. Barfoot, "Learning a bias correction for lidar-only motion estimation," in *2018 15th Conference on Computer and Robot Vision (CRV)*. IEEE, 2018, pp. 166–173.

[40] L. Sun, J. Zhao, X. He, and C. Ye, "Dlo: Direct lidar odometry for 2.5 d outdoor environment," in *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018, pp. 1–5.

[41] W. Lu, Y. Zhou, G. Wan, S. Hou, and S. Song, "L3-net: Towards learning based lidar localization for autonomous driving," in *Proceedings*

of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 6389–6398.

[42] W. Lu, G. Wan, Y. Zhou, X. Fu, P. Yuan, and S. Song, "Deepicp: An end-to-end deep neural network for 3d point cloud registration," *arXiv preprint arXiv:1905.04153*, 2019.

[43] ——, "Deepvcp: An end-to-end deep neural network for point cloud registration," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 12–21.

[44] D. Yin, Q. Zhang, J. Liu, X. Liang, Y. Wang, J. Maanpää, H. Ma, J. Hyyppä, and R. Chen, "Cae-lo: Lidar odometry leveraging fully unsupervised convolutional auto-encoder for interest point detection and feature description," *arXiv preprint arXiv:2001.01354*, 2020.

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

[46] Z. Zhang, "Iterative point matching for registration of free-form curves and surfaces," 1994.

[47] S. Rusinkiewicz and M. Levoy, "Efficient variants of the icp algorithm." in *3dim*, vol. 1, 2001, pp. 145–152.

[48] A. Censi, "An icp variant using a point-to-line metric," 2008.

[49] K.-L. Low, "Linear least-squares optimization for point-to-plane icp surface registration," *Chapel Hill, University of North Carolina*, vol. 4, no. 10, pp. 1–3, 2004.

[50] D. Kovalenko, M. Korobkin, and A. Minin, "Sensor aware lidar odometry," in *2019 European Conference on Mobile Robots (ECMR)*. IEEE, 2019, pp. 1–6.

[51] M. Magnusson, "The three-dimensional normal-distributions transform: an efficient representation for registration, surface analysis, and loop detection," Ph.D. dissertation, Örebro universitet, 2009.

[52] H. Hong and B. H. Lee, "Probabilistic normal distributions transform representation for accurate 3d point cloud registration," pp. 3333–3338, 2017.

[53] R. Kuramachi, A. Ohsato, Y. Sasaki, and H. Mizoguchi, "G-icp slam: An odometry-free 3d mapping system with robust 6dof pose estimation," *2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 176–181, 2015.

[54] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Mar. 2018.

[55] R. B. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (fpfh) for 3d registration," in *2009 IEEE international conference on robotics and automation*. IEEE, 2009, pp. 3212–3217.

[56] R. B. Rusu, G. Bradski, R. Thibaux, and J. Hsu, "Fast 3d recognition and pose using the viewpoint feature histogram," in *Proceedings of the 23rd IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Taipei, Taiwan, October 2010.

[57] B. Steder, R. B. Rusu, K. Konolige, and W. Burgard, "Narf: 3d range image features for object recognition."

[58] Y. Guo, F. Sohel, M. Bennamoun, M. Lu, and J. Wan, "Rotational projection statistics for 3d local surface description and object recognition," *International journal of computer vision*, vol. 105, no. 1, pp. 63–86, 2013.

[59] M. Bosse and R. Zlot, "Continuous 3d scan-matching with a spinning 2d laser," in *2009 IEEE International Conference on Robotics and Automation*. IEEE, 2009, pp. 4312–4319.

[60] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660.

[61] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 2. IEEE, 2006, pp. 1735–1742.

[62] D. Fox, W. Burgard, F. Dellaert, and S. Thrun, "Monte carlo localization: Efficient position estimation for mobile robots," 1999.