2016 IEEE International Conference on Robotics and Automation (ICRA)
Stockholm, Sweden, May 16-21, 2016

Collar Line Segments for Fast Odometry Estimation from Velodyne Point Clouds

Martin Velas[1], Michal Spanel[1] and Adam Herout[1]

***Abstract*— We present a novel way of odometry estimation from Velodyne LiDAR point cloud scans. The aim of our work is to overcome the most painful issues of Velodyne data – the sparsity and the quantity of data points – in an efficient way, enabling more precise registration. Alignment of the point clouds which yields the final odometry is based on random sampling of the clouds using *Collar Line Segments (CLS)*. The closest line segment pairs are identified in two sets of line segments obtained from two consequent Velodyne scans. From each pair of correspondences, a transformation aligning the matched line segments into a 3D plane is estimated. By this, significant planes (ground, walls, ...) are preserved among aligned point clouds. Evaluation using the KITTI dataset shows that our method outperforms publicly available and commonly used state-of-the-art method GICP for point cloud registration in both accuracy and speed, especially in cases where the scene lacks significant landmarks or in typical urban elements. For such environments, the registration error of our method is reduced by 75% compared to the original GICP error.**

I. INTRODUCTION

Exploration and 3D mapping of the environment surrounding a mobile robot plays a key role in robot's perception systems. Nowadays, the mapping becomes even more interesting as it is an integral part of many systems for semantic querying [1], semantic segmentation of scenes [2], change detection, or monitoring [3]. The source of 3D data ranges from traditional stereo cameras, RGB-D cameras (i.e. cameras enhanced by a depth sensor) extending the 2D data to 2.5D data including the spatial information as well.

Recently, numerous laser sensors – *LiDARs* (Light Detection And Ranging) – have also become popular in robotic systems. Besides the simple range finders providing only information about occupancy in a certain height around the robotic platform, sensors capturing precise 3D information of the environment, covering large horizontal and vertical field of view became available. These sensors enable modeling of the environment by precise and rich maps (Figure 1, top).

Since 2007, the *Velodyne LiDAR* sensor has become a valuable asset of vehicles attending DARPA Urban Challenge[2]. This type of sensor captures the full 3D information of the environment around the LiDAR. Currently the most powerful model HDL-64E covers full 360° horizontal field and 26.8° vertical field of view and with up to 15 Hz frame rate captures over 1.3 M of points per second. Example point clouds obtained by this sensor can be found in Figure 2.

The research was funded by the TACR project V3C (no. TE01020415) and the IT4IXS – IT4Innovations Excellence in Science project (LQ1602).

[1]Department of Computer Graphics and Multimedia, Faculty of Information Technology, Brno University of Technology, Czech Republic {ivelas|spanel|herout} at fit.vutbr.cz

[2]http://velodynelidar.com

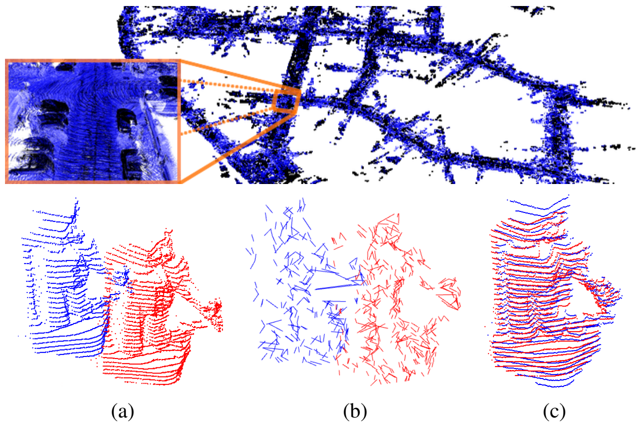

Fig. 1: Top: The environment map created by merging previously registered Velodyne point clouds. Bottom row: An example of the registration process. Two unaligned scans (a), sampled by line segments to produce *line clouds* (b) which are further used to estimate resulting alignment (c).

To be used for environment mapping, Velodyne point clouds must be registered and odometry of the mobile platform computed, in order to estimate the pose of the sensor at the time of scanning. Traditional approaches of the point cloud registration like Iterative Closest Point (ICP) [4] or feature based methods (e.g. based on surface normals derived from the neighborhood of a point) fail for such type of data because of its *vertical sparsity* and *ring structure* as shown in Figure 2. Since the original ICP approach looks for a transformation by minimizing the distance of the closest points, the unaligned data in Figure 2 (left) would be the optimal solution due to large amount of points in the floor rings perfectly fitting to each other. Also note in Figure 2 (left) that because of data sparsity, a lot of points from the source cloud (blue) miss their spatially corresponding point in the target cloud (red).

This paper presents a novel method of Velodyne point cloud registration in order to estimate odometry of the mobile platform. The main contributions of our work can be summarized in two steps of Velodyne point cloud processing. First, the typical point cloud representation is transformed into a *line cloud* by random generation of *Collar Line Segments (CLS)*. This step overcomes both the quantity and the sparsity of data. Second, we introduce an algorithm for *registration* of this line cloud representation. Our method achieves better results than publicly available state-of-the-art method GICP especially in cases when the scene lacks significant landmarks or typical "Manhattan" urban elements. Also the third contribution is making the implementation of

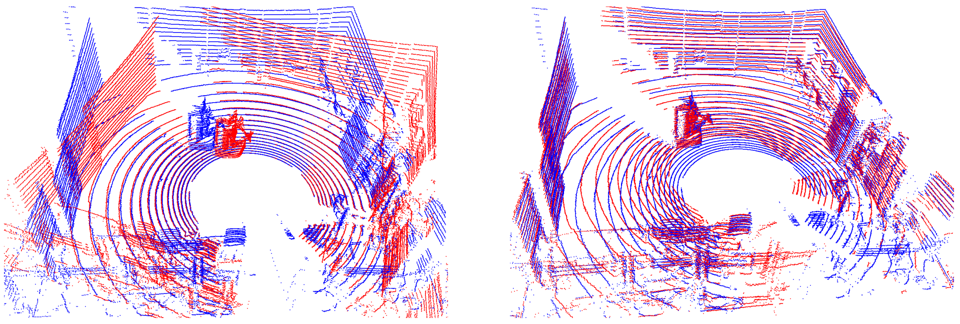
Fig. 2: Point clouds captured by the Velodyne LiDAR scanner and associated issues. The ring structures fit to each other for unregistered data (left) which disables the convergence of typical ICP approach to proper registration. The data is also sparse (large "gaps" between rings), causing lack of spatial correspondences between scans. See the well registered scans (right) - most of the points on floor in blue scan have no proper correspondence in red scan

.

our method and evaluation scripts publicly available[3].

II. RELATED WORK

In recent years, couple of algorithms addressing the point cloud registration problem have been published. Although they are able to register Velodyne scans, the lack of accuracy generally occurs.

Grant et al. [5] introduced a plane detection algorithm for Velodyne scans. Their method is based on the rings analysis and voting in a modified Hough space. For plane matching and computation of the final transformation, existing approach by Pathak et al. [6] is used. Their method was evaluated in indoor office environment and the error of estimated odometry exceeded 1 m after only ≈15 m run. Segmentation of the Velodyne point cloud for the registration was exploited by Douillard et al. [7]. First, the ground plane is removed from the scan by using scan voxelization. Then, the separated clusters of points are used as individual segments. The segments found in the previous step are then matched and a modified version of ICP computes the transformation by a segment-to-segment strategy. This method uses a very coarse voxel grid (20 cm resolution in experiments) which compromises the accuracy.

Accurate and effective registration of sweeping LiDAR scans has been achieved by the LOAM method [8] which was further improved by fusion with data provided by a RGB-D camera [9]. Both methods detect edges and planar points in the LiDAR scans for which a set of nonlinear equations constraining the odometry is generated. The final transformation is the result of a non-linear optimization. So far, these methods achieved the best results in the KITTI evaluation benchmark [10], but the algorithm specifics for processing the Velodyne scans have never been published nor the source codes are publicly available anymore.

Segal et al. [11] introduced a modification of the original ICP algorithm – the Generalized ICP (GICP). They replaced the standard point-to-point matching by a plane-to-plane strategy. The matching is based on covariance matrices of the local surfaces. For Velodyne data used in their evaluation, GICP reaches ±20 cm accuracy in registration of pairwise scans. This approach also assumes that for each local group of points in the source point cloud, there is a corresponding group in the target cloud. As shown in Figure 2, this is not always true in the case of sparse Velodyne data. Our method drops such assumption and approximates the local surfaces in a different way – by randomly generated collar lines. We will demonstrate that this strategy yields better results in terms of average accuracy, speed, and stability for natural and non-urban scenes than GICP.

Another modification of ICP by Pandey et al. [12] benefits from fusion of omni-directional RGB camera images with the Velodyne LiDAR scans. The prerequisite of this approach is known calibration of these two sensors. Then, the image features can be used for visual bootstrapping of generalized ICP algorithm in order to increase its robustness in case of large distances between scans (> 6m).

Badino et al. [13] solve visual odometry estimation by using stereo images. Their approach outperformed previously published image based methods on the KITTI benchmark. Instead of the traditional approach they propose a technique that uses the history of the tracked feature points for multi-frame feature integration into a single estimate.

This paper proceeds by introducing our novel method of Velodyne data registration using Collar Line Segments, and then introducing its multi-scan modification capable to increase the resulting accuracy.

III. VELODYNE POINT CLOUD REGISTRATION

As we showed in the introduction by Figure 2, the sparsity and ring structure of Velodyne data are serious issues. In our approach, we overcome these problems by generating a set of collar line segments as shown in Figure 3, which naturally fill the "gaps" between rings. They also drag corresponding planes between point clouds (floor, walls, ...) together.

The proposed registration method of Velodyne point clouds consists of two main parts. Both parts are described as a general registration framework together with notes about our implementation used in the experiments. First, the point cloud to be registered is sampled into a set of line segments – a line cloud (Figure 1b). Second, the two line clouds are registered and 6 DoF parameters are estimated by a strategy similar to the ICP [4], using line correspondences between

[3]https://github.com/robofit/but_velodyne_lib

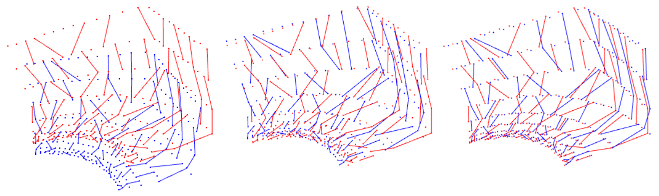
Fig. 3: Artificially generated Velodyne point cloud of a room corner iteratively registered by our method. Initial pose (left), in progress (middle) and final alignment (right).

the clouds. The steps are described as distinct, but they can be integrated and the sampling can be done on demand from the matching and registration steps.

A. *Sampling by Line Segments*

Each 3D point $[x, y, z]$ of the original point cloud is transformed into the *ring coordinates* $P_{r,\alpha} = [r, \alpha]$ where $r \in \{1, \ldots, N\}$ is the index of the ring the point belongs to and α is the angle within the ground plane xy:

$$\alpha = \mathrm{atan}(y/x) \qquad \alpha \in [0, 360)\,. \tag{1}$$

Vertical axis z is not used due to horizontal ring layout of Velodyne LiDAR scans.

For the points of cloud $\mathcal{P}$ in the ring coordinate system, the set of *collar line segments – line cloud* $\mathcal{L}_g$ is generated. Eq. (2) describes the *generator* of line segments $l_{r,\alpha,\alpha'} = [P_{r,\alpha}, P_{r+1,\alpha'}]$ between points of consequent rings r, $r+1$:

$$G : P_{r,\alpha} \rightarrow P_{r+1,\alpha'} \cup \{\perp\} \qquad P_{r,\alpha}, P_{r+1,\alpha'} \in \mathcal{P} \tag{2}$$

This function is required to join the points of similar angle (α is close to α') from subsequent rings so that the generated lines capture local surface properties ($\perp$ indicates that no matching point of interest was found). Line segments are not generated for every point, but the point cloud is randomly sampled. In order to select promising line segments, this line cloud $\mathcal{L}_g$ is further filtered by a *filter function*

$$F : \mathcal{L}_g \rightarrow \mathcal{L}, \quad \text{where } \mathcal{L} \subset \mathcal{L}_g. \tag{3}$$

The purpose of this function is to produce an as small as possible set of most descriptive lines.

A practical implementation of functions G and F (Eq. (2) and (3)) is depicted by Figure 4. Segments are generated only within one polar bin (sized φ). This function could alternatively be implemented with higher computational complexity by using a sliding window (rectangular or smooth Gaussian). Within each polar bin, a given number of line segments is randomly generated by G. Filtration (3) is implemented as preserving only the shortest of them. Preserving only the shortest lines discards lines formed where the rings cross an object edge as shown in Figure 4b. In this case, the single bin where lines were generated is split between multiple planes (object plane and the ground plane).

In our experiments on the KITTI dataset, the best results were achieved when the space was divided into 36 polar bins (per 10°), 20 lines within each polar cell were generated and 5 shortest of them were preserved. Approximately 11k collar line segments are generated for each point-cloud consisting of 64 rings of points (originally 130k of points in total).

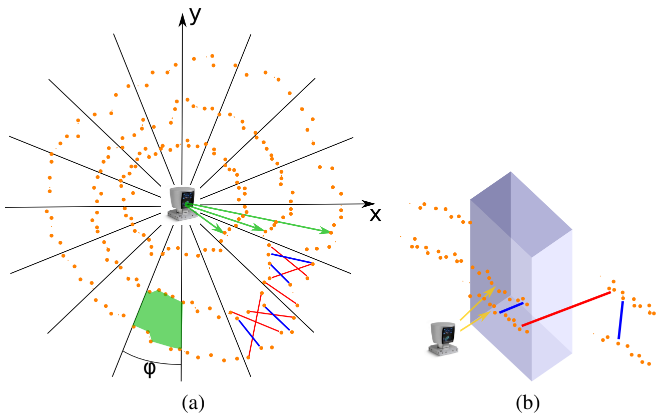

Fig. 4: Sampling the Velodyne point cloud (orange dots) by CLS (a). The Velodyne casts rays (green arrows) each capturing one “ring” of points. The space around the sensor is divided into a polar grid of bins (one bin is green colored). Within each bin, line segments are generated by randomly generating joints between the points of consequent rings. The shortest ones (blue lines) are preserved, the others (red lines) are discarded. Demonstration of the problem when particular laser scans (i.e. measured “rings” of points) cross an object boundary (b). Preserved shorter line segments are usually generated within real 3D planes (blue lines) and rejected longer segments typically connect different planes (red one).

B. *Registration of line clouds*

Once the Velodyne point clouds are sampled into the set of line segments, these line clouds can be registered by an iterative approach. Alternatively, the original point clouds can be repeatedly resampled.

In the target line cloud $\mathcal{L}_t$, the matching line segment for each element in source line cloud $\mathcal{L}_s$ is found:

$$M : (l_s, \mathcal{L}_t) \rightarrow l_t \qquad l_s \in \mathcal{L}_s,\ l_t \in \mathcal{L}_t \cup \{\perp\} \tag{4}$$

In our implementation used in the experiments, function M finds the line in $\mathcal{L}_t$ whose center is closest to l_s (euclidean distance), Figure 5b. If its distance is above a computed threshold (mean distance), the match is not found ($\perp$ is returned). Finding the closest line is accelerated by kD-tree.

Matching of lines by finding middle points and the building of kD-tree is done only once during the initialization. To eliminate effect of the incorrect matches, those with euclidean distance of line centers bigger than the mean distance of all found matches are discarded. We have also experimented with re-sampling the point cloud by line segments after every few iterations and continuing in registration using the last estimated transformation. This has not brought any more improvement in the registration accuracy.

In order to find the optimal transformation in the same manner as ICP approaches do using SVD (Singular Value Decomposition) [14], the corresponding 3D points P_s, P_t have to be derived for each previously matched pair l_t, l_s:

$$C : (l_s, l_t) \rightarrow (P_s, P_t)\,, \tag{5}$$

where points P_s, P_t do not necessarily come from the original point clouds. The computed transformation minimizes the

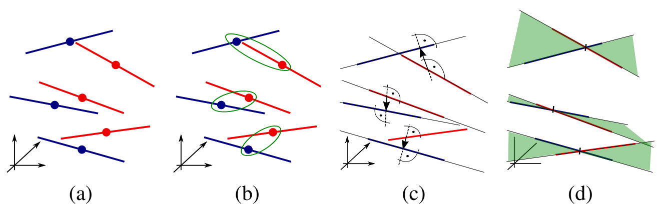

Fig. 5: Registration of "line clouds" shown on three pairs of matching lines. The middle points of segments are found (a) and used for matching the lines by closest centers (b). The segments are extended into infinite lines and closest points of the matching line pairs are found (arrows in (c)). These correspondences define the transformation which "pushes" the two matched lines into a single (green) plane (d).

distance of these corresponding points. This process can be perceived that our proposed registration method resamples the original point cloud to a new point cloud in the each iteration of the original ICP algorithm. Resampling is done so that it overcomes the problem of data sparsity capturing the properties of local surface by collar line segments.

The corresponding points (5) are found such that the estimated transformation causes matching lines to cross. This simulates fitting the corresponding planes between point clouds as has been previously shown in Figure 3. The line segments are extended to infinite lines, and closest points – pair (P_s, P_t) – are found as follows.

Assuming the line of the source line cloud l_s and the line of the target cloud l_t is given by 3D point $\hat{P}_s$ and vector $\mathbf{u_s}$ ($\hat{P}_t$ and $\mathbf{u_t}$ for the target line, respectively):

$$l_s: \quad X = \hat{P}_s + \mathbf{u_s}.t_s; \quad t_s \in (-\infty, \infty) \tag{6}$$
$$l_t: \quad X = \hat{P}_t + \mathbf{u_t}.t_t; \quad t_t \in (-\infty, \infty), \tag{7}$$

the closest points P_s, P_t between these two lines are [15]:

$$P_s = \hat{P}_s + \mathbf{u_s}.t_s^c \qquad P_t = \hat{P}_t + \mathbf{u_t}.t_t^c \tag{8}$$
$$\text{where} \qquad t_s^c = \frac{be - cd}{ac - b^2} \qquad t_t^c = \frac{ae - bd}{ac - b^2} \tag{9}$$
$$a = \mathbf{u_s} \cdot \mathbf{u_s} \qquad b = \mathbf{u_s} \cdot \mathbf{u_t} \qquad c = \mathbf{u_t} \cdot \mathbf{u_t} \tag{10}$$
$$d = \mathbf{u_s} \cdot \mathbf{w} \qquad e = \mathbf{u_t} \cdot \mathbf{w} \qquad \mathbf{w} = \hat{P}_s - \hat{P}_t \tag{11}$$

Operation $\mathbf{u} \cdot \mathbf{v}$ represents the dot product of two vectors and auxiliary variables a, b, c, d, e are scalars, $\mathbf{w}$ is a vector.

C. Prediction of Transformation from Previous Frames

Since the LiDAR scanner is commonly mounted on a moving vehicle, the physical constraints like momentum are bound to the vehicle odometry. Thus the previously computed transformation can be used for prediction and initialization of the next pose estimation.

Traditional solutions of this problem use non-linear predictors like Extended Kalman Filter (EKF) [16]. In our experiments, linear prediction using last N sets of transformation parameters (6DoF) brings significant improvement in prediction of odometry. We have also experimentally compared it with EKF while obtaining almost the same results. So finally, for the sake of speed, we decided to use the simple linear prediction.

Let T_i be the transformation between two consequent scans P_i and P_{i+1} taken by the LiDAR scanner at times i and $(i + 1)$. The transformation is a 6DoF vector $[t_x, t_y, t_z, r_r, r_p, r_y]$. Then, the initial prediction T_{init} of the transformation at time i can be computed as:

$$T_{init} = \frac{2}{N(N+1)} \sum_{j=1}^{N} (N - j + 1)\, T_{i-j}. \tag{12}$$

D. Processing of Multiple Scans

Badino et al. [13] improved the estimated transformations by multi-frame feature integration. Similarly, in our approach, the history of LiDAR scans and computed transformations are used to improve the odometry precision. New scan P_{i+1} is additionally registered against H previous historical records $P_{i-1}, P_{i-2}, \ldots, P_{i-H}$ (Figure 6).

In the first experiments, these multiple transformations estimated for each Velodyne scan of data sequence were used to build pose graph further optimized by a nonlinear solver (SLAM++ [17] was used). These experiments resulted in inaccurate results due to sensitivity of the optimizer to noise so we proposed another finally more robust solution described below.

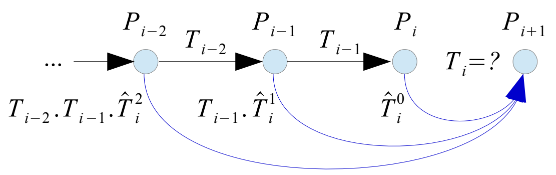

Fig. 6: New LiDAR scan is registered against multiple previous records $P_{i-1}, P_{i-2}, \ldots, P_{i-H}$ and multiple transformations (blue edges) are estimated.

As the previous transformations $T_{i-1}, T_{i-2}, \ldots, T_{i-H}$ are known, using their inverse, transformations $\hat{T}_i^1, \ldots, \hat{T}_i^H$ (see Figure 6) can be derived. Assuming the normal Gaussian distribution of the noise they suffer from, the resulting transformation T_i can be obtained as a mean of these values. More details can be found in Algorithm 1.

Algorithm 1 Registration against H previous scans for noise reduction

```
1: $\hat{T}_i^0$ = REGISTER($P_i$, $P_{i+1}$, $T_{init}$)
2: $T_{inv}$ = Identity
3: for $j = 1$ to $H$ do
4:     $T_{inv}$ = $T_{inv}$ * INVERT($T_{i-j}$)
5:     $S$ = $\{T_{inv} * p \,|\, p \in P_{i-j}\}$
6:     $\hat{T}_i^j$ = REGISTER($S$, $P_{i+1}$, $\hat{T}_i^{j-1}$)
7: end for
8: $T_i = \frac{1}{H} \sum_{j=0}^{H} \hat{T}_i^j$
```

IV. Experimental Evaluation

For evaluation of the odometry estimation, we used the publicly available KITTI odometry dataset [10]. It consists of 22 independent sequences captured during driving in and outside the city of Karlsruhe. The following data sequences are available for each run: stereo gray-scale and color camera images, point clouds captured by Velodyne LiDAR, mutual calibration of sensors, and ground truth data (for the first 11 runs only) obtained by GPS/OXTS. For the odometry estimation, only the Velodyne data was used in our case.

Since our work presents a single standalone component for point cloud registration rather than a complex system, we have chosen GICP method of similar complexity for comparison. Until the other modules (e.g. dynamic objects filter, visual loop closure, fusion with RGB data) are not involved, we do not find KITTI benchmark suitable for objective evaluation.

A. *Evaluation metric*

The quality of point cloud registration and the odometry estimation was evaluated by using the first 11 sequences of the KITTI dataset for which the ground truth data are publicly available. Since this data was obtained by a GPS sensor which yields significant imprecision in the vertical position estimation (z axis), only horizontal coordinates (xy plane) are used for the error estimation. The error of a single point cloud registration is then defined as

$$e_i = \sqrt{(t_i.x - g_i.x)^2 + (t_i.y - g_i.y)^2}, \tag{13}$$

where t_i is estimated position of i^{th} LiDAR frame with respect to the previous frame $i-1$ and g_i is the ground truth data. The error of whole sequence (N frames) is defined as

$$e = \frac{1}{N}\sum_{i=1}^{N} e_i. \tag{14}$$

B. *Results: Precision of Registration on the KITTI Dataset*

In this section, we compare our method with publicly available state of the art method for point cloud registration GICP [11]. Since our registration process was improved by prediction described in Section III-C, the same prediction was used also for GICP to keep comparison fair. Apart of this, the default parameters of the test application[4] were used.

The sum of registration error e yielded by GICP and our method can be found in Table I. Comparison of the 3rd and 4th column of Table I shows that our method preserves stability among different KITTI data sequences captured in different environments. Our method outperforms GICP especially in challenging sequences of non-Manhattan environment outside of the city – highway (seq. #1) and rural area (seq. #2) – where GICP approach fails. For other sequences, our method reaches comparable results. In average (weighted by sequence length), our method yields better accuracy of the estimated odometry.

The last column of Table I (CLS-M) contains the results of our method further improved by processing of multiple (i.e. 10) scans as described in Section III-D. Since this modification requires the registration to be repeated multiple times and each single GICP registration is quite time consuming, each scan is registered only with a single predecessor.

[4]http://www.robots.ox.ac.uk/∼avsegal/generalized_icp.html

Seq. #	Length [frames]	GICP	CLS	CLS-M
0	4540	0.0315	0.0622	0.0529
1	1100	0.4215	0.0960	0.0685
2	4660	0.3347	0.0858	0.1144
3	800	0.0218	0.0275	0.0239
4	270	0.0497	0.0316	0.0394
5	2760	0.0228	0.0726	0.0413
6	1100	0.0362	0.0327	0.0383
7	1100	0.0132	0.0222	0.0117
8	4070	0.0626	0.1001	0.0643
9	1590	0.0530	0.0688	0.0583
10	1200	0.0177	0.0464	0.0369
weighted avg	2108	**0.1153**	**0.0712**	**0.0624**

TABLE I: Odometry estimation error for data sequences in the KITTI dataset for which the ground truth data are publicly available. The average error is the weighted average of sequence errors where the weight is the length of sequence. The results referred as CLS-M were obtained while processing multiple frames as described in Section III-D.

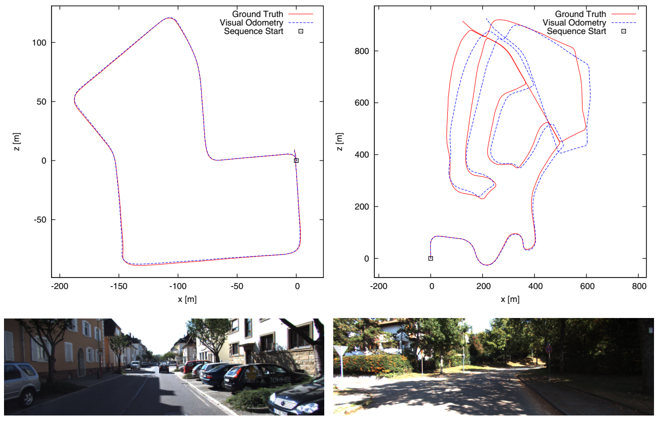

Fig. 7: The best (left, seq. #7, error 0.0117) and the worst (right, seq. #2, error 0.1144) estimated odometry using our method. Bottom row: Typical images from the sequence (images themselves are not used for processing).

The best results were obtained when processing sequence #7 which was recorded in a Manhattan-like urban environment as is shown in Figure 7, left. On the other hand, data sequence #2 yields the worst results. It was captured outside the city center and besides the road, it captures mostly natural phenomena (trees, bushes, etc.) as shown in Figure 7, right. Compared to our approach, GICP is not able to handle these natural objects in sequence #2 and estimate the vehicle odometry with a reasonable error.

The GICP method is also failing (42cm error) on data sequence #1, which was recorded outside the city on the open highway (Figure 8) and the LiDAR captures only the road, without any other significant landmarks. The largest drift appears when the tested car (sensor platform) is overtaken by another vehicle. Our method is able to handle these situations and it estimates odometry of feasible accuracy (error below 7cm for CLS-M).

Evaluation for different driving speeds (Figure 9) shows that both methods preserve equal precision for speeds below 30 km/h. For higher speeds (especially over 70 km/h), our method outperforms GICP.

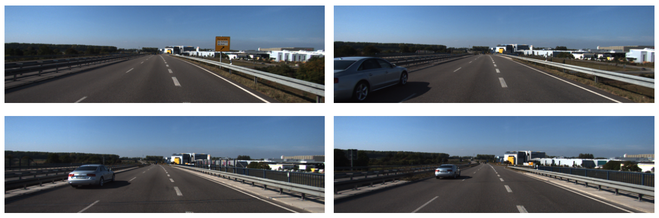
Fig. 8: Images from KITTI data sequence #1 where GICP approach fails but our method preserves accuracy. Images show the challenging situation when overturning car appear as confusing landmarks on otherwise empty highway.

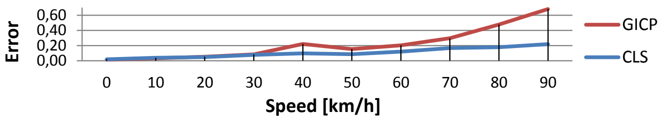

Fig. 9: Average error for different driving speed. High speed (over 50km/h) was simulated by omitting every 2$^{\text{nd}}$ frame.

C. Finding Optimal Parameters

Graphs in Figure 10 show the results for different parameters setup of the sampling and registration process. The best setup regarding the registration accuracy, used also for the evaluation above, generates 20 line segments per angular bin, preserves 5 shortest ones of them, uses 3 latest measurements for prediction and 10 registrations against previous scans for noise reduction.

D. Speed

As shown in Table II, for the registration of a pair of Velodyne scans (no special optimization or parallelization), achieves approximately 10× better frame rate comparing to the publicly available implementation of GICP.

Table II also shows, that frame rate of our multi-scan modification falls proportionally to the number of previous scans used due to the multiple registrations against the previous records. This modification is generally applicable to improve the registration accuracy and so it would also lower the frame rate when applied to the GICP approach.

V. CONCLUSION

This paper introduces a novel way of Velodyne point cloud representation using the Collar Line Segments (CLS), the algorithm of "line clouds" registration, and its further improvement by processing multiple preceding scans. These algorithms were used for Velodyne LiDAR scans registration of the KITTI dataset and compared to the state-of-the-art technique Generalized ICP. The new method achieves better results in terms of registration accuracy, especially for challenging situations like natural scenes or lack of relevant landmarks. Considering the time consumption, our approach is approximately 10× faster. Using further proposed improvements, the registration reaches 6 cm weighted average registration error on the KITTI evaluation data sequences.

In the future, visual loop detection, its closure as well as detection and exclusion of disturbing moving objects can be valuable assets in further improvement of the accuracy of the estimated odometry.

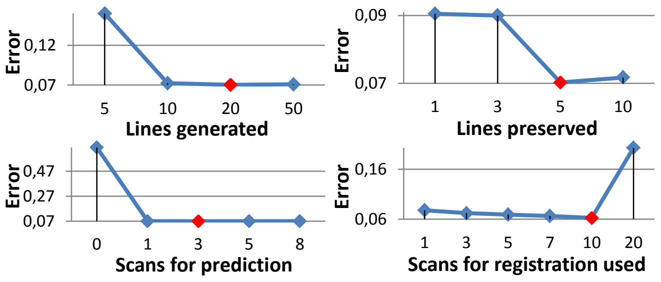

Fig. 10: Tuning of the registration parameters. Values of the lowest error (14) are highlighted in red. Graphs show the error trend based on the number of lines generated and preserved in the each polar bin, the number of previous registrations used for prediction and the number of frames used for multi-scan approach.

	GICP	CLS	CLS-M
Avg. time per frame [s]	25.68	2.36	28.56

TABLE II: Comparison of time consumption. In CLS-M, the registration was performed against 10 previous scans.

REFERENCES

[1] J. Mason and B. Marthi, "An object-based semantic world model for long-term change detection and semantic querying," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ Int. Conference on*, 2012.
[2] A. Hermans, G. Floros, and B. Leibe, "Dense 3d semantic mapping of indoor scenes from rgb-d images," in *Robotics and Automation (ICRA), 2014 IEEE Int. Conference on*, May 2014, pp. 2631–2638.
[3] R. Ambrus, N. Bore, *et al.*, "Meta-rooms: Building and maintaining long term spatial models in a dynamic world," in *Intelligent Robots and Systems (IROS), 2014 IEEE/RSJ Int. Conference on*, 2014.
[4] P. Besl and N. D. McKay, "A method for registration of 3-d shapes," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 14, no. 2, pp. 239–256, Feb 1992.
[5] W. Grant, R. Voorhies, and L. Itti, "Finding planes in lidar point clouds for real-time registration," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ Int. Conference on*, Nov 2013, pp. 4347–4354.
[6] K. Pathak, A. Birk, *et al.*, "Fast registration based on noisy planes with unknown correspondences for 3-d mapping," *Robotics, IEEE Transactions on*, vol. 26, no. 3, pp. 424–441, June 2010.
[7] B. Douillard, A. Quadros, *et al.*, "Scan segments matching for pairwise 3d alignment," in *Robotics and Automation (ICRA), 2012 IEEE Int. Conference on*, May 2012, pp. 3033–3040.
[8] J. Zhang and S. Singh, "Loam: Lidar odometry and mapping in real-time," in *Robotics: Science and Systems Conference (RSS 2014)*, 2014.
[9] J. Zhang and S. Singh, "Visual-lidar odometry and mapping: Low-rift, robust, and fast," in *IEEE ICRA*, Seattle, WA, 2015.
[10] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *Int. Journal of Robotics Research (IJRR)*, 2013.
[11] A. Segal, D. Haehnel, and S. Thrun, "Generalized-icp," in *Proceedings of Robotics: Science and Systems*, Seattle, USA, June 2009.
[12] G. Pandey, J. McBride, S. Savarese, and R. Eustice, "Visually bootstrapped generalized icp," in *Robotics and Automation (ICRA), 2011 IEEE Int. Conference on*, May 2011, pp. 2660–2667.
[13] H. Badino, A. Yamamoto, and T. Kanade, "Visual odometry by multi-frame feature integration," in *Int. Workshop on Computer Vision for Autonomous Driving @ ICCV*, December 2013.
[14] A. Nuchter, K. Lingemann, J. Hertzberg, and H. Surmann, "6d slam with approximate data association," in *Advanced Robotics, 2005. ICAR '05. Proceedings., 12th Int. Conference on*, July 2005, pp. 242–249.
[15] J. Olive, *Maths: A Student's Survival Guide: A Self-Help Workbook for Science and Engineering Students*. Cambridge Uni. Press, 2003.
[16] S. Haykin, *Kalman Filtering and Neural Networks*, ser. Adaptive and Cognitive Dynamic Systems: Signal Processing, Learning, Communications and Control. Wiley, 2004.
[17] S. V. Ila, L. Polok, M. Šolony, P. Zemčík, and P. Smrž, "Fast covariance recovery in incremental nonlinear least square solvers," in *Proceedings of IEEE ICRA*. IEEE Computer Society, 2015.