CNN for IMU Assisted Odometry Estimation using Velodyne LiDAR

Martin Velas, Michal Spanel, Michal Hradis, and Adam Herout

***Abstract*— We introduce a novel method for odometry estimation using convolutional neural networks from 3D LiDAR scans. The original sparse data are encoded into 2D matrices for the training of proposed networks and for the prediction. Our networks show significantly better precision in the estimation of translational motion parameters comparing with state of the art method LOAM, while achieving real-time performance. Together with IMU support, high quality odometry estimation and LiDAR data registration is realized. Moreover, we propose alternative CNNs trained for the prediction of rotational motion parameters while achieving results also comparable with state of the art. The proposed method can replace wheel encoders in odometry estimation or supplement missing GPS data, when the GNSS signal absents (e.g. during the indoor mapping). Our solution brings real-time performance and precision which are useful to provide online preview of the mapping results and verification of the map completeness in real time.**

I. Introduction

Recently, many solutions for indoor and outdoor *3D mapping* using LiDAR sensors have been introduced, proving that the problem of *odometry estimation* and *point cloud registration* is relevant and solutions are demanded. The Leica[1] company introduced Pegasus backpack equipped with multiple Velodyne LiDARs, RGB cameras, including IMU and GNSS sensors supporting the point cloud alignment. Geoslam[2] uses simple rangefinder accompanied with IMU unit in their hand-helded mapping products ZEB1 and ZEB-REVO. Companies like LiDARUSA and RIEGL[3] build their LiDAR systems primarily targeting outdoor ground and aerial mapping. Such systems require readings from IMU and GNSS sensors in order to align captured LiDAR point clouds. These requirements restrict the systems to be used for mapping the areas where GNSS sensors are available.

Another common property of these systems is offline processing of the recorded data for building the accurate 3D maps. The operator is unable to verify whether the whole environment (building, park, forest, . . .) is correctly captured and whether there are no parts missing. This is a significant disadvantage, since the repetition of the measurement process can be expensive and time demanding. Although the orientation can be estimated online and quite robustly by the IMU unit, *precise position information* requires reliable GPS signal readings including the online corrections (differential GPS, RTK, ...). Since these requirements are not met in many scenarios (indoor scenes, forests, tunnels, mining sites, etc.), the less accurate methods, like odometry estimation from wheel platform encoders, are commonly used.

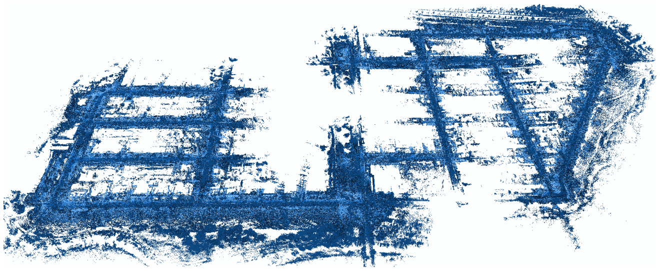

Fig. 1: Example of LiDAR point clouds registered by CNN estimated odometry. Sequence 08 of KITTI dataset [3] is presented with rotations provided by IMU.

We propose an alternative solution – a frame to frame *odometry estimation* using *convolutional neural networks* from LiDAR point clouds. Similar deployments of CNNs has already proved to be successful in ground segmentation [1] and also in vehicle detection [2] in sparse LiDAR data.

The main contribution of our work is fast, real-time and precise estimation of positional motion parameters (translation) outperforming the state-of-the-art results. We also propose alternative networks for full 6DoF visual odometry estimation (including rotation) with results comparable to the state of the art. Our deployment of convolutional neural networks for odometry estimation, together with existing methods for object detection [2] or segmentation [1] also illustrates general usability of CNNs for this type of *sparse LiDAR data*.

II. Related Work

The published methods for visual odometry estimation can be divided into two groups. The first one consists of direct methods computing the motion parameters in a single step (from image, depth or 3D data). Comparing with the second group of iterative methods, direct methods have a potential of better time performance. Unfortunately, to our best knowledge, no direct method for odometry estimation from LiDAR data have been introduced so far.

Since the introduction of notoriously known Iterative Closest Point (ICP) algorithm [4,5], many modifications of this approach were developed. In all derivatives, two basic steps are iteratively repeated until the termination conditions are met: matching the elements between 2 point clouds (originally the points were used directly) and the estimation of

All authors are with with the Faculty of Information Technology, Brno University of Technology, Czech Republic {ivelas|spanel|ihradis|herout} at fit.vutbr.cz

*This work has been supported by the Artemis JU grant agreement ALMARVI (no. 621439), the TACR Competence Centres project V3C (no. TE01020415), and the IT4IXS IT4Innovations Excellence project (LQ1602).

[1] http://leica-geosystems.com

[2] https://geoslam.com

[3] https://www.lidarusa.com, http://www.riegl.com

target frame transformation, minimizing the error represented by the distance of matching elements. This approach assumes that there actually exist matching elements in the target cloud for a significant amount of basic elements in the source point cloud. However, such assumption does not often hold for sparse LiDAR data and causes significant inaccuracies.

Grant [6] used planes detected in Velodyne LiDAR data as the basic elements. The planes are identified by analysis of depth gradients within readings from a single laser beam and then by accumulating in a modified Hough space. The detected planes are matched and the optimal transformation is found using previously published method [7]. Their evaluation shows the significant error ($\approx$ 1m after 15m run) when mapping indoor office environment. Douillard et al. [8] used the ground removal and clustering remaining points into the segments. The transformation estimated from matching the segments is only approximate and it is probably compromised by using quite coarse (20cm) voxel grid.

Generalized ICP (GICP) [9] replaces the standard point-to-point matching by the plane-to-plane strategy. Small local surfaces are estimated and their covariance matrices are used for their matching. When using Velodyne LiDAR data, the authors achieved $\pm 20\,$cm accuracy in the registration of pairwise scans. In our evaluation [10] using KITTI dataset [3], the method yields average error 11.5cm in the frame-to-frame registration task. The robustness of GICP drops in case of large distance between the scans ($>$ 6m). This was improved by employing visual SIFT features extracted from omnidirectional Ladybug camera [11] and the codebook quantization of extracted features for building sparse histogram and maximization of mutual information [12].

Bose and Zlot [13] are able to build consistent 3D maps of various environments, including challenging natural scenes, deploying visual loop closure over the odometry provided by inaccurate wheel encoders and the orientation by IMU. Their robust place recognition is based on Gestalt keypoint detection and description [14]. Deployment of our CNN in such system would overcome the requirement of the wheel platform and the same approach would be useful for human-carried sensory gears (Pegasus, ZEB, etc.) as mentioned in the introduction.

In our previous work [10], we proposed sampling the Velodyne LiDAR point clouds by *Collar Line Segments (CLS)* in order to overcome data sparsity. First, the original Velodyne point cloud is split into polar bins. The line segments are randomly generated within each bin, matched by nearest neighbor search and the resulting transformation fits the matched lines into the common planes. The CLS approach was also evaluated using the KITTI dataset and achieves 7cm error of the pairwise scan registration. Splitting into polar bins is also used in this work during for encoding the 3D data to 2D representation (see Sec. III-A).

The top ranks in KITTI Visual odometry benchmark [3] are for last years occupied by LiDAR Odometry and Mapping (LOAM) [15] and Visual LOAM (V-LOAM) [16] methods. Planar and edge points are detected and used to estimate the optimal transformation in two stages: fast scan-to-scan and precise scan-to-map. The map consists of keypoints found in previous LiDAR point clouds. Scan-to-scan registration enables real-time performance and only each n-th frame is actually registered within the map.

The implementation was publicly released under BSD license but withdrawn after being commercialized. The original code is accessible through the documentation[4] and we used it for evaluation and comparison with our proposed solution. In our experiments, we were able to achieve superior accuracy in the estimation of the translation parameters and comparable results in the estimation of full 6DoF (degrees of freedom) motion parameters including rotation. In V-LOAM [16], the original method was improved by fusion with RGB data from omnidirectional camera and authors also prepared method which fuses LiDAR and RGB-D data [17].

The encoding of 3D LiDAR data into the 2D representation, which can be processed by convolutional neural network (CNN), were previously proposed and used in the ground segmentation [1] and the vehicle detection [2]. We use a similar CNN approach for quite different task of visual odometry estimation. Besides the precision and the real-time performance, our method also contributes as the illustration of general usability of CNNs for sparse LiDAR data. The key difference is the amount and the ordering of input data processed by neural network (described in next chapter and Fig. 3). While the previous methods [1,2] process only a single frame, in order to estimate the transformation parameters precisely we process multiple frames simultaneously.

III. Method

Our *goal* is the estimation of transformation $\boldsymbol{T_n} = [t_n^x, t_n^y, t_n^z, r_n^x, r_n^y, r_n^z]$ representing the 6DoF motion of a platform carrying LiDAR sensor, given the current LiDAR frame $\boldsymbol{P_n}$ and N previous frames $\boldsymbol{P_{n-1}}, \boldsymbol{P_{n-2}}, \ldots, \boldsymbol{P_{n-N}}$ in form of point clouds. This can be written as a mapping Θ from the point cloud domain $\mathbb{P}$ to the domain of motion parameters (1) and (2). Each element of the point cloud $\boldsymbol{p} \in \boldsymbol{P}$ is the vector $\boldsymbol{p} = [p^x, p^y, p^z, p^r, p^i]$, where $[p^x, p^y, p^z]$ are its coordinates in the 3D space (right, down, front) originating at the sensor position. p^r is the index of the laser beam that captured this point, which is commonly referred as the "ring" index since the Velodyne data resembles the rings of points shown in Fig. 2 (top, left). The measured intensity by laser beam is denoted as p^i.

$$\boldsymbol{T_n} = \Theta(\boldsymbol{P_n}, \boldsymbol{P_{n-1}}, \boldsymbol{P_{n-2}}, \ldots, \boldsymbol{P_{n-N}}) \tag{1}$$

$$\Theta : \mathbb{P}^{N+1} \rightarrow \mathbb{R}^6 \tag{2}$$

A. Data encoding

We represent the mapping Θ by convolutional neural network. Since we use sparse 3D point clouds and convolutional neural networks are commonly designed for dense 1D and 2D data, we adopt previously proposed [1,2] *encoding* $\mathcal{E}$ (3) of 3D LiDAR data to dense matrix $\boldsymbol{M} \in \mathbb{M}$. These

[4] http://docs.ros.org/indigo/api/loam_velodyne/html/files.html

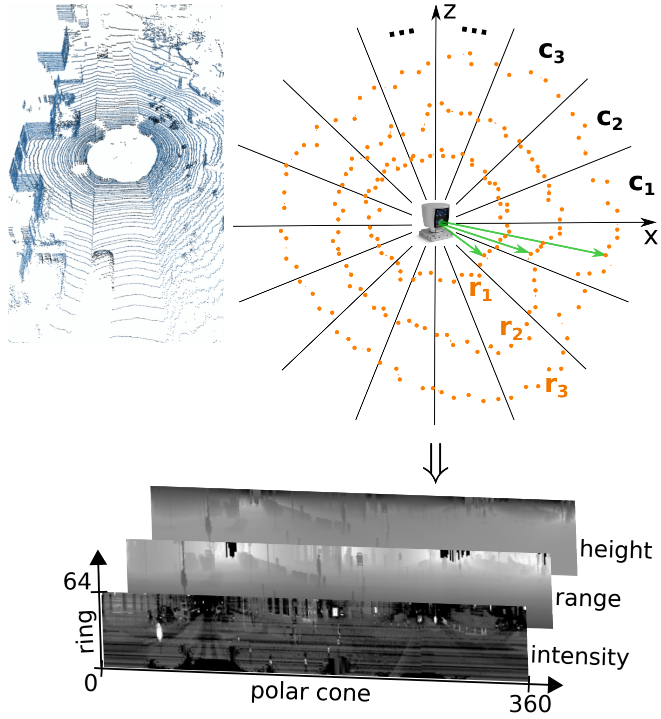

Fig. 2: Transformation of the sparse Velodyne point cloud into the multi-channel dense matrix. Each row represents measurements of a single laser beam (single ring $r_1, r_2, r_3, \ldots$) done during one rotation of the sensor. Each column contains measurements of all 64 laser beams captured within the specific rotational angle interval (polar cone $c_1, c_2, c_3, \ldots$).

encoded data are used for actual training the neural network implementing the mapping $\tilde{\Theta}$ (4, 5).

$$\boldsymbol{M} = \mathcal{E}(\boldsymbol{P}); \quad \mathcal{E} : \mathbb{P} \rightarrow \mathbb{M} \tag{3}$$

$$\boldsymbol{T_n} = \tilde{\Theta}(\mathcal{E}(\boldsymbol{P_n}), \mathcal{E}(\boldsymbol{P_{n-1}}), \ldots, \mathcal{E}(\boldsymbol{P_{n-N}})) \tag{4}$$

$$\tilde{\Theta} : \mathbb{M}^{N+1} \rightarrow \mathbb{R}^6 \tag{5}$$

Each element $\boldsymbol{m_{r,c}}$ of the matrix $\boldsymbol{M}$ encodes points of *the polar bin* $\boldsymbol{b_{r,c}} \subset \boldsymbol{P}$ (6) as a vector of 3 values: depth and vertical height relative to the sensor position, and the intensity of laser return (7). Since the multiple points fall into the same bin, the representative values are computed by averaging. On the other hand, if a polar bin is empty, the missing element of the resulting matrix is interpolated from its neighbourhood using linear interpolation.

$$\boldsymbol{m_{r,c}} = \varepsilon(\boldsymbol{b_{r,c}}); \quad \varepsilon : \mathbb{P} \rightarrow \mathbb{R}^3 \tag{6}$$

$$\varepsilon(\boldsymbol{b_{r,c}}) = \frac{\sum\limits_{\boldsymbol{p} \in \boldsymbol{b_{r,c}}} \left[p^y, \left\|p^x, p^z\right\|_2, p^i\right]}{|\boldsymbol{b_{r,c}}|} \tag{7}$$

The indexes r, c denote both the row (r) and the column (c) of the encoded matrix and the polar cone (c) and the ring index (r) in the original point cloud (see Fig. 2). Dividing the point cloud into the polar bins follows same strategy as described in our previous work [10]. Each polar bin is identified by the polar cone $\varphi(.)$ and the ring index p^r.

$$\boldsymbol{b_{r,c}} = \{\boldsymbol{p} \in \boldsymbol{P} \,|\, p^r = r \wedge \varphi(\boldsymbol{p}) = c\} \tag{8}$$

$$\varphi(\boldsymbol{p}) = \left\lfloor \frac{\text{atan}\left(\frac{p^z}{p^x}\right) + 180°}{\frac{360°}{R}} \right\rfloor \tag{9}$$

where R is horizontal angular resolution of the polar cones. In our experiments we used the resolution $R = 1°$ (and $0.2°$ in the classification formulation described below).

B. From regression to classification

In our preliminary experiments, we trained the network $\tilde{\Theta}$ estimating full 6DoF motion parameters. Unfortunately, such networks provided very inaccurate results. The output parameters consist of two different motion modalities – rotation and translation $\boldsymbol{T_n} = [\boldsymbol{R_n}|\boldsymbol{t_n}]$ – and it is difficult to determine (or weight) the importance of angular and positional differences in backward computation. So we decided to split the mapping into the estimation of rotation parameters $\tilde{\Theta}_{\boldsymbol{R}}$ (10) and translation $\tilde{\Theta}_{\boldsymbol{t}}$ (11).

$$\boldsymbol{R_n} = \tilde{\Theta}_{\boldsymbol{R}}(\boldsymbol{M_n}, \boldsymbol{M_{n-1}}, \ldots, \boldsymbol{M_{n-N}}) \tag{10}$$

$$\boldsymbol{t_n} = \tilde{\Theta}_{\boldsymbol{t}}(\boldsymbol{M_n}, \boldsymbol{M_{n-1}}, \ldots, \boldsymbol{M_{n-N}}) \tag{11}$$

$$\tilde{\Theta}_{\boldsymbol{R}} : \mathbb{M}^{N+1} \rightarrow \mathbb{R}^3; \quad \tilde{\Theta}_{\boldsymbol{t}} : \mathbb{M}^{N+1} \rightarrow \mathbb{R}^3 \tag{12}$$

The implementation of $\tilde{\Theta}_{\boldsymbol{R}}$ and $\tilde{\Theta}_{\boldsymbol{t}}$ by convolutional neural network is shown in Fig. 3. We use *multiple input frames* in order to improve stability and robustness of the method. Such multi-frame approach was also successfully used in our previous work [10] and comes from assumption, that motion parameters are similar within small time window ($0.1 - 0.7$s in our experiments below).

The idea behind proposed topology is the expectation that shared CNN components for pairwise frame processing will estimate the motion map across the input frame space (analogous to the optical flow in image processing). The final estimation of rotation or translation parameters is performed in the fully connected layer joining the outputs of pure CNN components.

Splitting the task of odometry estimation into two separated networks, sharing the same topology and input data, significantly improved the results – especially the precision of translation parameters. However, precision of the predicted rotation was still insufficient. The original formulations of our goal (1) can be considered as solving the *regression task*. However, the space of possible rotations between consequent frames is quite small for reasonable data (distribution of rotations for KITTI dataset can be found in Fig. 5). Such small space can be densely sampled and we can reformulate this problem to the *classification task* (13, 14).

$$R = \underset{i \in \{0, \ldots, K-1\}}{\arg\max} \; \Gamma(R_i(\boldsymbol{M_n}), \boldsymbol{M_{n-1}}) \tag{13}$$

$$\Gamma : \mathbb{M}^2 \rightarrow \mathbb{R} \tag{14}$$

where $R_i(\boldsymbol{M_n})$ represents rotation R_i of the current LiDAR frame $\boldsymbol{M_n}$ and $\Gamma(.)$ estimates the probability of R_i to be the correct rotation between the frames $\boldsymbol{M_n}$ and $\boldsymbol{M_{n-1}}$.

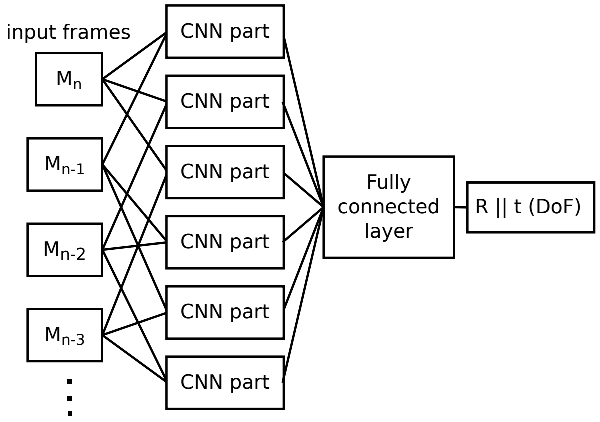

Fig. 3: Topology of the network implementing $\tilde{\Theta}_{\boldsymbol{R}}$ and $\tilde{\Theta}_{\boldsymbol{t}}$. All combinations of current $\boldsymbol{M_n}$ and previous $\boldsymbol{M_{n-1}}, \boldsymbol{M_{n-2}}, \ldots$ frames (3 previous frames in this example) are pairwise processed by the same CNN part (see structure in Fig. 4) with shared weights. The final estimation of rotation or translation parameters is done in fully connected layer joining the outputs of CNN parts.

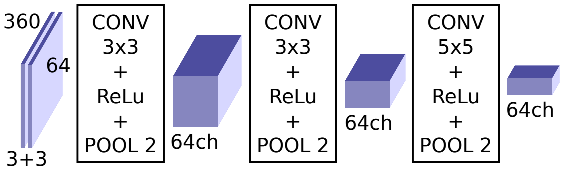

Fig. 4: Topology of shared CNN component (denoted as *"CNN part"* in Fig. 3) for processing the pairs of encoded LiDAR frames. The topology is quite shallow with small convolutional kernels, ReLu nonlinearities and max polling after each convolutional layer. The output blob size is $45 \times 8 \times 64$ $(W \times H \times Ch)$.

Similar approach was previously used in the task of human age estimation [18]. Instead of training the CNN to estimate the age directly, the image of person is classified to be $0, 1, \ldots, 100$ years old.

The implementation of Γ comparator by a convolutional network can be found in Fig. 6. In next sections, this network will be referred as *classification CNN* while the original one will be referred as *regression CNN*. We have also experimented with the classification-like formulation of the problem using the original CNN topology (Fig. 3) without sampling and applying the rotations, but this did not bring any significant improvement.

For the classification network we have experienced better results when wider input (horizontal resolution $R = 0.2°$) is provided to the network. This affected also properties of the convolutional component used (the CNN part), where wider convolution kernels are used with horizontal stride (see Fig. 7) in order to reduce the amount of data processed by the fully connected layer.

Although the space of observed rotations is quite small (approximately $\pm 1°$ around x and z axis, and $\pm 4°$ for y axis, see Fig. 5), sampling densely (by fraction of degree) this subspace of 3D rotations would result in thousands of possible rotations. Because such amount of rotations would be infeasible to process, we decided to estimate the rotation around each axis separately, so we trained 3 CNNs implementing (13) for rotations around x, y and z axis separately. These networks share the same topology (Fig. 6).

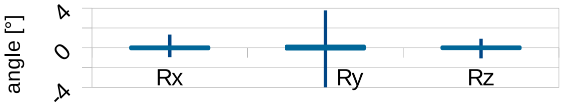

Fig. 5: Rotations (min-max) around x, y, z axis in training data sequences of KITTI dataset.

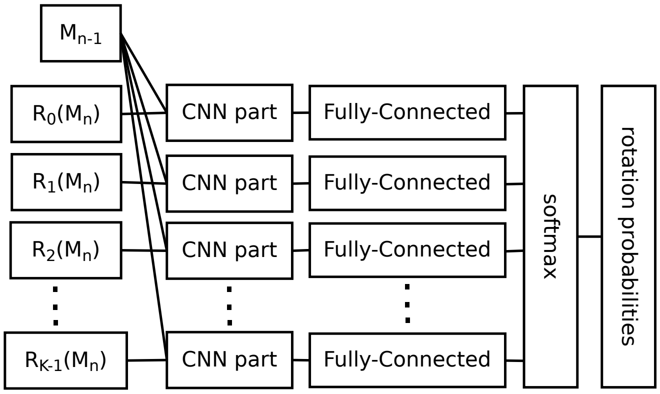

Fig. 6: Modification of original topology (Fig. 3) for precise estimation of rotation parameters. Rotation parameter space (each axis separately) is densely sampled into K rotations $R_0, R_1, \ldots, R_{K-1}$ and applied to current frame $\boldsymbol{M_n}$. CNN component and fully connected layer are trained as comparators Γ with previous frame $\boldsymbol{M_{n-1}}$ estimating probability of given rotation. All CNN parts (structure in Fig. 7) and fully connected layers share the weights of the activations.

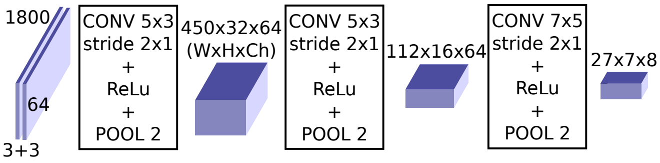
1800
64
3+3
CONV 5x3 stride 2x1 + ReLu + POOL 2
450x32x64 (WxHxCh)
CONV 5x3 stride 2x1 + ReLu + POOL 2
112x16x64
CONV 7x5 stride 2x1 + ReLu + POOL 2
27x7x8

Fig. 7: Modification of convolutional component for classification network. Wider input (angular resolution $R = 0.2°$) and wider convolution kernels with horizontal stride are used.

In the formulation of our classification problem (13), the final decision of the best rotation R^* is done by max polling. Since Γ estimates the probability of rotation angle $p(R_i)$ (15), assuming the normal distribution we can compute also maximum likelihood solution by weighted average (16).

$$p(R_i) = \Gamma(R_i(\boldsymbol{M_n}), \boldsymbol{M_{n-1}}) \tag{15}$$

$$R^* = \frac{\sum\limits_{i \in \boldsymbol{S_W}} p(R_i).R_i}{\sum\limits_{i \in \boldsymbol{S_W}} p(R_i)} \tag{16}$$

$$\boldsymbol{S_W} = \underset{\boldsymbol{S}=\{i_0,\ldots,i_0+W\}}{\arg\max} \sum_{i \in \boldsymbol{S}} p(R_i) \tag{17}$$

Moreover, this estimation can done for a window of fixed size W which is limited only for the highest rotation probabilities (17). Window of size 1 results in max polling.

C. Data processing

For training and testing the proposed networks, we used encoded data from Velodyne LiDAR sensor. As we mentioned before, the original raw point clouds consist of x, y and z coordinates, identification of laser beam which captured given point and the value of laser intensity reading. The encoding into 2D representation transforms x and z coordinates (horizontal plane) into the depth information and horizontal angle represented by range channel and the column index respectively in the encoded matrix. The intensity readings and y coordinates are directly mapped into the matrix channels and laser beam index is represented by encoded matrix row index. This means that our encoding (besides the aggregating multiple points into the same polar bin) did not cause any information loss.

Furthermore, we use the same data normalization (18) and rescaling as we used in our previous work [1].

$$\overline{h} = \frac{y^i}{H}; \qquad \overline{d} = \log(d) \tag{18}$$

This applies only to the vertical height h and depth d, since the intensity values are already normalized to interval $(0; 1)$. We set the height normalization constant to $H = 3$, since in the usual scenarios, the Velodyne (model HDL-64E) captures vertical slice approximately 3m high.

In our preliminary experiments, we trained the convolutional networks without this normalization and rescaling (18) and we also experimented with using the 3D point coordinates as the channels of CNN input matrices. All these approaches resulted only in worse odometry precision.

IV. Experiments

We implemented the proposed networks using *Caffe*[5] deep learning framework. For training and testing, data from the KITTI odometry benchmark[6] were used together with provided development kit for the evaluation and error estimation. The LiDAR data were collected by Velodyne HDL-64E sensor mounted on top of a vehicle together with IMU sensor and GPS localization unit with RTK correction signal providing precise position and orientation measurements [3]. Velodyne spins with frequency 10Hz providing 10 LiDAR scans per second. The dataset consist of 11 data sequences where ground truth is provided. We split these data to training (sequences 00-07) and testing set (sequences 08-10). The rest of the dataset (sequences 11-21) serves for benchmarking purposes only.

The error of estimated odometry is evaluated by the development kit provided with the KITTI benchmark. The data sequences are split into subsequences of $100, 200, \ldots, 800$ frames ($10, 20, \ldots, 80$ seconds duration). The error e_s of each subsequence is computed as (19).

$$e_s = \frac{\|\boldsymbol{E_s}, \boldsymbol{C_s}\|_2}{l_s} \tag{19}$$

	CNN-t	CNN-R	CNN-Rt	Forward time [s/frame]	
N	error	error	error	GPU	CPU
1	0.0184	0.3794	0.3827	0.004	0.065
2	0.0129	0.2752	0.2764	0.013	0.194
3	0.0111	0.2615	0.2617	0.026	0.393
5	**0.0103**	0.2646	0.2656	0.067	0.987
7	0.0130	0.2534	0.2546	0.125	1.873

TABLE I: Evaluation of regression networks for different size of input data – N is the number of previous frames. The convolutional networks were used to determine the translation parameters only (column CNN-t), the rotation only (CNN-R) and both the rotation and translation (CNN-Rt) parameters for KITTI sequences 00-08. Error of the estimated odometry together with the processing time of single frame (using CPU only or GPU acceleration) is presented.

Window size W	Odom. error	Window size	Odom. error
1 (max polling)	0.03573	9	0.03704
3	**0.03433**	11	0.03712
5	0.03504	13	0.03719
7	0.03629	all	0.03719

TABLE II: The impact of window size on the error of odometry, when the rotation parameters are estimated by classification strategy. Window size $W = 1$ is equivalent to the max pooling, maximal likelihood solution is found also when "*all*" probabilities are taken into the account without the window restriction.

where $\boldsymbol{E_s}$ is the expected position (from ground truth) and $\boldsymbol{C_s}$ is the estimated position of the LiDAR where the last frame of subsequence was taken with respect to the initial position (within given subsequence). The difference is divided by the length l_s of the followed trajectory. The final error value is the average of errors e_s across all the subsequences of all the lengths.

First, we trained and evaluated regression networks (topology described in Fig. 3) for direct estimation of rotation or translation parameters. The results can be found in Table I. To determine the error of the network predicting translation or rotation motion parameters, the missing rotation or translation parameters respectively were taken from the ground truth data since the evaluation requires all 6DoF parameters.

Evaluation shows that proposed CNNs predict the translation (*CNN-t* in Table I) with high precision – the best results were achieved for network taking the current and $N = 5$ previous frames as the input. The results also show, that all these networks outperform LOAM (error 0.0186, see evaluation in Table III for more details) in the estimation of translation parameters. On contrary, this method is unable to estimate rotations (*CNN-R* and *CNN-Rt*) with sufficient precision. All networks except the largest one ($N < 7$) are capable of realtime performance with GPU support (GeForce GTX 770 used) and the smallest one also without

[5] caffe.berkeleyvision.org

[6] www.cvlibs.net/datasets/kitti/eval_odometry.php

	Translation only			Rotation and translation			
Seq. #	**LOAM-full**	**LOAM-online**	**CNN-regression**	**LOAM-full**	**LOAM-online**	**CNN-regression**	**CNN-classification**
00	0.0152	0.0193	0.0084	0.0225	0.0516	0.2877	0.0302
01	0.0368	0.0255	0.0079	0.0396	0.0385	0.1492	0.0444
02	0.0383	0.0293	0.0076	0.0461	0.0550	0.2290	0.0342
03	0.0120	0.0117	0.0166	0.0191	0.0294	0.0648	0.0494
04	0.0076	0.0085	0.0089	0.0148	0.0150	0.0757	0.0177
05	0.0092	0.0096	0.0056	0.0184	0.0246	0.1357	0.0235
06	0.0088	0.0130	0.0036	0.0160	0.0335	0.0812	0.0188
07	0.0137	0.0155	0.0077	0.0192	0.0380	0.1308	0.0177
Train average	0.0214	0.0197	0.0077	0.0287	0.0433	0.1970	0.0303
08	0.0107	0.0145	0.0096	0.0239	0.0349	0.2716	0.0289
09	0.0368	0.0380	0.0098	0.0322	0.0430	0.2373	0.0494
10	0.0213	0.0196	0.0128	0.0295	0.0399	0.2823	0.0327
Test average	0.0186	0.0208	**0.0102**	**0.0268**	0.0376	0.2655	0.0343

TABLE III: Comparison of the odometry estimatation precision by the proposed method and LOAM for sequences of the KITTI dataset [3] (sequences $00 - 07$ were used for training the CNN, $08 - 10$ for testing only). LOAM was tested in the on-line mode (LOAM-online) when the time spent for single frame processing is limited to Velodyne fps (0.1s/frame) and in the full mode (LOAM-full) where each frame is fully registered within the map. Both the regression (CNN-regression) and the classification (CNN-classification) strategies of our method are included. When only translation parameters are estimated, our method outperforms LOAM. On the contrary, LOAM outperforms our CNN odometry when full 6DoF motion parameters are estimated.

any acceleration (running on i5-6500 CPU). Note: Velodyne standard framerate is 10fps.

We also wanted to explore, whether CNNs are capable to predict full 6DoF motion parameters, including rotation angles with sufficient precision. Hence the classification network schema shown in Fig. 6 was implemented and trained also using the Caffe framework. The network predicts probabilities for densely sampled rotation angles. We used sampling resolution $0.2°$, what is equivalent to the horizontal angular resolution of Velodyne data in the KITTI dataset. Given the statistics from training data shown in Fig. 5, we sampled the interval $\pm1.3°$ of rotations around x and z axis into 13 classes, and the interval $\pm5.6°$ into 56 classes, including approximately 30% tolerance.

Since the network predicts the probabilities of given rotations, the final estimation of the rotation angle is obtained by max polling (13) or by the window approach of maximum likelihood estimation (16,17). Table II shows that optimal results are achieved when the window size $W = 3$ is used.

We compared our CNN approach for odometry estimation with the LOAM method [15]. We used the originally published ROS implementation (see link in Sec. II) with a slight modification to enable KITTI Velodyne HDL-64E data processing. In the original package, the input data format of Velodyne VLP-16 is “hardcoded”. The results of this implementation is labeled in Table III as *LOAM-online*, since the data are processed online in real time (10fps). This real-time performance is achieved by skipping the full mapping procedure (registration of the current frame against the internal map) for particular input frames.

Comparing with this original online mode of LOAM method, our CNN approach achieves better results in estimation of both translation and rotation motion parameters. However, it is important to mention, that our classification network for the orientation estimation requires 0.27s/frame when using GPU acceleration.

The portion of skipped frames in the LOAM method depends on the input frame rate, size of input data, available computational power and affects the precision of estimated odometry. In our experiments with the KITTI dataset (on the same machine as we used for CNN experiments), 31.7% of input frames is processed by the full mapping procedure.

In order to determine the full potential of the LOAM method, and for fair comparison, we made further modifications of the original implementation, so the mapping procedure runs for each input frame. Results of this method are labeled as *LOAM-full* in Table III and, in estimation of all 6DoF motion parameters, it outperforms our proposed CNNs. However, the prediction of translation parameters by our regression networks is still significantly more precise and faster. And the average processing time of a single frame by the LOAM-full method is 0.7s. The visualization of estimated transformations can be found in Fig. 8.

We have also submitted the results of our networks (i.e. the regression CNN estimating translational parameters only and the classification CNN estimating rotations) to the KITTI benchmark together with the outputs we achieved using the LOAM method in the online and the full mapping mode. The results are similar as in our experiments – best performing LOAM-full achieves 3.49% and our CNNs 4.59% error. LOAM-online performed worse than in our experiments with error 9.21%. Interestingly, the error of our refactored original implementation of LOAM is more significant than errors reported for the original submission of the LOAM

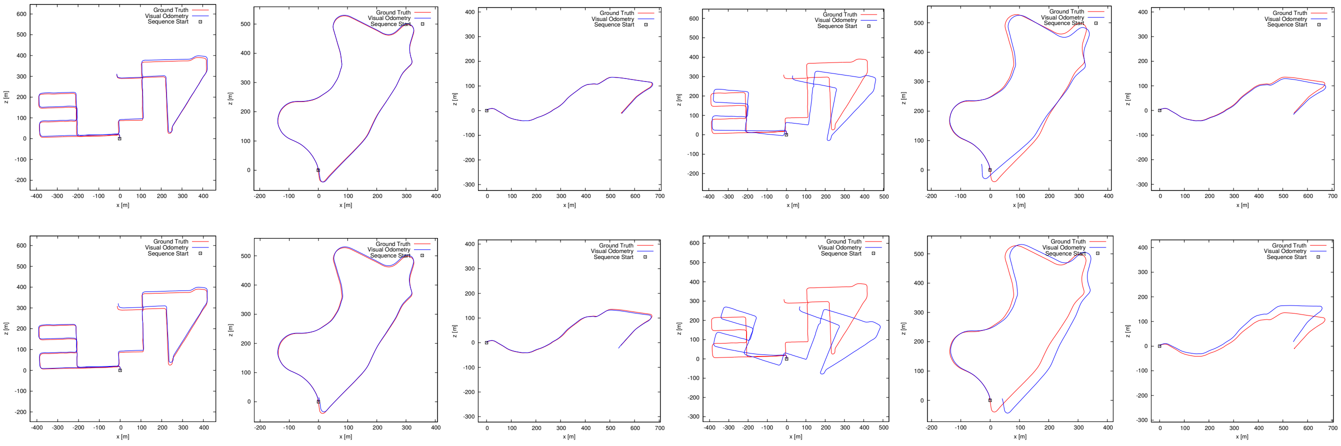

Fig. 8: The example of LOAM results (top) and our CNNs (bottom row) for KITTI sequences used for testing $(08 - 10)$. When only translation parameters are estimated (first 3 columns), both methods achieves very good precision and the differences from ground truth (red) are barely visible. When all 6DoF motion parameters are estimated (columns $4 - 6$), better performance of loam LOAM can be observed.

authors. This is probably caused by a special tuning of the method for the KITTI dataset which has been never published and authors unfortunately refused to share both the specification/implementation used and the outputs of their method with us.

V. Conclusion

This paper introduced novel method of odometry estimation using convolutional neural networks. As the most significant contribution, networks for very fast real-time and precise estimation of translation parameters, beyond the performance of other state of the art methods, were proposed. The precision of proposed CNNs was evaluated using the standard KITTI odometry dataset.

Proposed solution can replace the less accurate methods like odometry estimated from wheel platform encoders or GPS based solutions, when GNSS signal is not sufficient or corrections are missing (indoor, forests, etc.). Moreover, with the rotation parameters obtained from the IMU sensor, results of the mapping can be shown in a preview for online verification of the mapping procedure when the data are being collect.

We also introduced two alternative network topologies and training strategies for prediction of orientation angles, enabling complete visual odometry estimation using CNNs in a real time. Our method benefits from existing encoding of sparse LiDAR data for processing by CNNs [1,2] and contributes as a proof of general usability of such a framework.

In the future work, we are going to deploy our odometry estimation approaches in real-word online 3D LiDAR mapping solutions for both indoor and outdoor environments.

References

[1] M. Velas, M. Spanel, M. Hradis, and A. Herout, "CNN for very fast ground segmentation in Velodyne lidar data," 2017. [Online]. Available: http://arxiv.org/abs/1709.02128

[2] B. Li, T. Zhang, and T. Xia, "Vehicle detection from 3D lidar using fully convolutional network," *CoRR*, vol. abs/1608.07916, 2016. [Online]. Available: http://arxiv.org/abs/1608.07916

[3] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *Int. Journal of Robotics Research (IJRR)*, 2013.

[4] Y. Chen and G. Medioni, "Object modelling by registration of multiple range images," *Image Vision Comput.*, vol. 10, pp. 145–155, 1992.

[5] P. J. Besl and N. D. McKay, "A method for registration of 3-D shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, Feb 1992.

[6] 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.

[7] 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.

[8] 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.

[9] A. Segal, D. Haehnel, and S. Thrun, "Generalized-icp," in *Proceedings of Robotics: Science and Systems*, Seattle, USA, June 2009.

[10] M. Velas, M. Spanel, and A. Herout, "Collar line segments for fast odometry estimation from velodyne point clouds," in *IEEE Int. Conference on Robotics and Automation*, May 2016, pp. 4486–4495.

[11] 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.

[12] G. Pandey, J. R. McBride, S. Savarese, and R. M. Eustice, "Toward mutual information based automatic registration of 3D point clouds," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2012, pp. 2698–2704.

[13] M. Bosse and R. Zlot, "Place recognition using keypoint voting in large 3D lidar datasets," in *2013 IEEE International Conference on Robotics and Automation*, May 2013, pp. 2677–2684.

[14] M. Bosse and R. Zlot, "Keypoint design and evaluation for place recognition in 2D lidar maps," *Robotics and Autonomous Systems*, vol. 57, no. 12, pp. 1211 – 1224, 2009, inside Data Association.

[15] J. Zhang and S. Singh, "Loam: Lidar odometry and mapping in real-time," in *Robotics: Science and Systems Conference (RSS 2014)*, 2014.

[16] J. Zhang and S. Singh, "Visual-lidar odometry and mapping: Low-rift, robust, and fast," in *IEEE ICRA*, Seattle, WA, 2015.

[17] J. Zhang, M. Kaess, and S. Singh, "Real-time depth enhanced monocular odometry," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sept 2014, pp. 4973–4980.

[18] R. Rothe, R. Timofte, and L. V. Gool, "Deep expectation of real and apparent age from a single image without facial landmarks," *International Journal of Computer Vision (IJCV)*, July 2016.