On Uncertainty Quantification for Convolutional Neural Network LiDAR Localization

Mathieu Joerger, *Senior Member, IEEE*, Julian Wang, and Ali Hassani

Abstract— **In this paper, we develop and evaluate a Convolutional Neural Network (CNN)-based Light Detection and Ranging (LiDAR) localization algorithm that includes uncertainty quantification for ground vehicle navigation. This paper builds upon prior research where we used a CNN to estimate a rover's position and orientation (pose) using LiDAR point clouds (PCs). This paper presents a simplification of the LiDAR PC processing and describes a new approach for outputting a covariance matrix in addition to the rover pose estimates. Performance assessment is carried out in a structured, static lab environment using a LiDAR-equipped rover moving along a fixed, repeated trajectory.**

I. Introduction

This paper describes the design and evaluation of a new method to estimate a rover's position and orientation (pose) from a Light Detection and Ranging (LiDAR) three-dimensional (3D) point cloud (PC) using a Convolutional Neural Network (CNN). We first develop a CNN-based LiDAR localization algorithm to simultaneously determine a vehicle's pose and quantify the pose estimation uncertainty. We then implement this CNN-based covariance estimation method in a structured, known environment.

This research is intended for future autonomous navigation of vehicles, such as buses, delivery drones, farming and mining surveillance platforms, that repeatedly follow a predefined itinerary in an unstructured but known environment. LiDAR localization aims at determining a LiDAR's "pose", i.e., its position and orientation in a navigation frame, or East-North-Up (ENU) frame, given a discretized representation of the environment in sensor frame as perceived by the LiDAR.

Conventional model-based LiDAR PC navigation methods include matching techniques and landmark-based localization (LBL). On the one-hand, it is typical for scan-matching, PC-matching, and grid-based approaches to be heuristically implemented, which complicates uncertainty prediction [1-6]. On the other hand, localization error covariance matrices are readily provided in LBL when using an extended Kalman filter (EKF), but additional steps of feature extraction (FE) and data association (DA) are needed [7-9]. FE is the identification of reliable, viewpoint-invariant landmarks in the environment surrounding the LiDAR. DA is the ordering of these extracted features to match the ordering of measurements in the EKF innovation vector. We will use a LiDAR LBL method as reference algorithm when evaluating LiDAR CNNs [10-11].

Data-driven algorithms have the potential to learn how to find a LiDAR's navigation-frame pose given a sensor-frame PC, without explicitly performing the error-prone FE and DA. A major issue with neural networks, when used for localization, is their limited ability to quantify pose prediction uncertainty. Unlike model-based approaches that readily output estimation error covariance matrices, neural networks learn heuristically. We can evaluate their actual estimation errors offline by post-processing large amounts of pose error data [12]. But it remains unclear under what conditions a neural network can learn online to self-assess its LiDAR pose prediction uncertainty.

In related work, other LiDAR-based CNNs have been implemented in realistic scenarios, but composite error statistics combining varying rover-to-obstacle geometries could only be evaluated offline [13, 14]. Other CNNs have been specifically developed to quantify uncertainty, including using raw error data or using a model-based estimator's covariance output as training data [15, 16]. In [16], a matrix decomposition is implemented to ensure that the neural-network-estimated covariance matrix is positive definite. In [17], sources of uncertainty are classified and accounted for in CNN-based estimators. But, [15-17] are not implemented in LiDAR navigation applications. In [18], two-dimensional indoor LiDAR data is processed using deep neural networks, including uncertainty quantification showing limitations of the Monte-Carlo dropout technique [19]; this effort is focused on mapping performance. Reference [20] shows promising localization performance using a LiDAR CNN in GNSS-denied urban areas, but uncertainty localization performance prediction is still an issue. Missing from the literature is a detailed analysis of LiDAR CNN-estimated pose error distributions versus actual error distributions.

In response, in this paper, we design a CNN LiDAR localization process that streamlines 3D PC processing, and we develop a new training method to output elements of a pose estimation error covariance matrix. We evaluate the LiDAR CNN in comparison with a model-based approach.

The second section of this paper outlines our CNN-based LiDAR navigation architecture. It describes a new approach for parametric localization uncertainty quantification using a Cholesky factorization-based method to enforce the symmetry and positive definiteness of the pose estimation error covariance matrix. The third section is a preliminary

M. Joerger is with the Kevin T. Crofton Department of Aerospace and Ocean Engineering at the Virginia Polytechnic Institute and State University (Virginia Tech), Blacksburg, VA, USA 24061 USA (phone: 540-231-6707; fax: 540-231-9632; e-mail: joerger@vt.edu).

J. Wang, is with the Kevin T. Crofton Department of Aerospace and Ocean Engineering at Virginia Tech, Blacksburg, VA, USA 24061 USA. (e-mail: julianw@vt.edu).

A. Hassani is with the Kevin T. Crofton Department of Aerospace and Ocean Engineering at Virginia Tech, Blacksburg, VA, USA 24061 USA. (e-mail: ahassani@vt.edu).

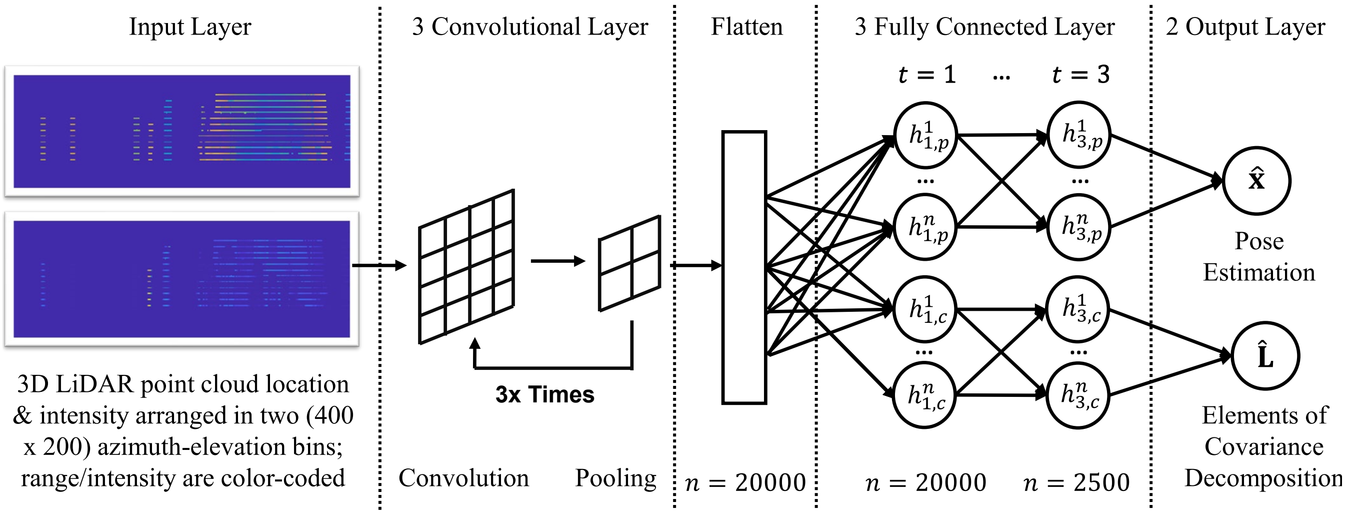


Figure 1. LiDAR CNN architecture where 3D point-clouds in the LiDAR's field of view are captured using two two-dimensional arrays of azimuth-elevation bins with color-coded ranging and intensity measurements, respectively. Outputs include pose estimates and elements of the decomposed pose estimate covariance matrix.

performance evaluation using data collected in a known laboratory environment. Because this paper's emphasis is on error distribution analysis, datasets are limited to lab experiments where true vehicle pose is continuously and accurately known. Lab testing has limitations, but it reveals variations in error distributions that would be cost-prohibitive to observe in field testing. A comparison is performed with a more conventional model-based localization technique that uses static, recognizable features. Concluding statements are given in the fourth section.

II. CNN Architecture for 3D LiDAR Point Cloud Localization with Uncertainty Quantification

A. *CNN Architecture for LiDAR PC-Based Localization*

A LiDAR PC is made of thousands of 3D point coordinates, each of which comes with a light-return-intensity measurement. Both the point coordinates and intensity values can be used for localization [11].

The 3D LiDAR PC must first be pre-processed to fit conventional image-based CNN functions that assume two-dimensional arrays as inputs. Consistent with the LiDAR's spherical scanning of the environment, we rearrange the 3D PC point locations into azimuth-elevation bins with a resolution of approximately 1-degree-by-1-degree over the LiDAR field of view. This is an improvement over the previous triple projection in [12]. This implementation also includes LiDAR light-return intensity data arranged in azimuth-elevation bins. These two arrays serve as input layer to the CNN.

Then, we process the input data with the CNN designed using PyTorch [21]. The CNN must find an estimate $\hat{\mathbf{x}}$ of the rover's actual pose $\mathbf{x}$ in a local navigation frame. It must simultaneously output elements $\hat{\mathbf{L}}$ derived from $\hat{\mathbf{P}}$, which is an estimate of the actual pose estimate error covariance matrix $\mathbf{P}$.

A diagram of the LiDAR CNN architecture is displayed in Figure 1. It was designed iteratively with the intention of being as simple as possible to facilitate our understanding of the algorithm's properties and of the most sensitive parameters. Our preliminary design criterion is that the CNN should achieve an estimation performance similar to the example model-based approach in [10, 11].

In [11], static, recognizable landmarks with viewpoint-invariant features are extracted from the LiDAR PC and associated over time as the LiDAR moves in the environment. Point-features are then processed using an EKF. The algorithm in [11] serves as guidance to tune the CNN's number of layers and numbers of neurons per layer by matching output performance. Further CNN improvements using more sophisticated architectures will be investigated in future iterations of this work.

The CNN architecture includes three convolutional layers, each of which has a kernel size of 2×2 and a maximum pooling of 2×2 [12]. Their output is flattened into a 20,000×1 vector. During our iterative CNN design, we verified that the same flattened layer could be used for pose and covariance estimation without significant performance impact. The flattened layer is therefore the input to two separate threads for pose and covariance determination. Both threads comprise three fully connected layers.

Each fully connected layers'neuron has a linear mapping function, and a ReLU activation function. A neuron-dropout rate of 5% prevents overfitting the training dataset and provides robustness to changes during validation.

In each thread, the last of the fully-connected layer is fully connected to an output layer through a linear mapping function. The first thread's output is the 3×1 LiDAR pose estimate vector $\hat{\mathbf{x}}$, i.e., estimates of the LiDAR's two-dimensional horizontal position coordinates and of its orientation (or azimuth) in a local navigation frame. The outputs of the second thread are estimates of the six lower-triangular non-zero elements of the $\hat{\mathbf{L}}$ matrix defined by the Cholesky decomposition of the 3×3 a-posteriori pose covariance matrix $\hat{\mathbf{P}} = \hat{\mathbf{L}}\hat{\mathbf{L}}^T$.

B. *CNN Training for LiDAR PC-Based Uncertainty Quantification*

Training of the LiDAR CNN for the determination of the 3×1 vector $\hat{\mathbf{x}}$, which is an estimate of the actual pose vector $\mathbf{x}$ (given by a reference truth source), aims at minimizing the following loss function:

$$J_{pose} \equiv (\hat{\mathbf{x}}-\mathbf{x})^T \bar{\mathbf{P}}^{-1}(\hat{\mathbf{x}}-\mathbf{x})$$

where $\bar{\mathbf{P}}$ is the a-priori (predicted) covariance matrix of the state estimation error.

Training of the LiDAR CNN for the determination of the non-zero elements of matrix $\hat{\mathbf{L}}$ aims at minimizing the following loss function derived from the Frobenius norm of matrix $[\mathbf{I}_3-\mathbf{L}^{-1}(\hat{\mathbf{L}}\hat{\mathbf{L}}^T)\mathbf{L}^{-T}]$:

$$J_{\text{cov}} \equiv \sqrt{\text{trace}\{[\mathbf{I}_3-\mathbf{L}^{-1}(\hat{\mathbf{L}}\hat{\mathbf{L}}^T)\mathbf{L}^{-T}][\mathbf{I}_3-\mathbf{L}^{-1}(\hat{\mathbf{L}}\hat{\mathbf{L}}^T)\mathbf{L}^{-T}]^T\}}$$

where:

trace{}	: is the trace operator, i.e., sum of the diagonal elements of the matrix in argument
$\mathbf{L}$	: is a 3×3 upper-triangular matrix derived from the Cholesky factorization of the true covariance $\mathbf{P}=\mathbf{L}\mathbf{L}^T$
$\mathbf{I}_3$	: is a 3×3 identity matrix

The cost functions capture differences that we want to minimize in a two-step training process (a) between a known LiDAR pose $\mathbf{x}$ and its CNN estimate $\hat{\mathbf{x}}$, and (b) between a known LiDAR pose estimate error covariance, derived using CNN pose estimate error samples ($\mathbf{x}-\hat{\mathbf{x}}$) for a same location of the LiDAR, and the CNN's output pose estimate error covariance $\hat{\mathbf{P}}$.

In the expression of J_{cov}, the Cholesky factorization ensures symmetry and positive definiteness of $\hat{\mathbf{P}}=\hat{\mathbf{L}}\hat{\mathbf{L}}^T$. Inverses capture a normalization operation by elements of $\mathbf{L}$ required to combine position and orientation quantities that have non-identical, and non-independent distributions. The Frobenius norm gives a scalar measure of the 3×3 matrix. Other cost functions, e.g., used in references [15,16], did not provide adequate performance results in this specific LiDAR navigation implementation.

In training, we employ a stochastic gradient descent algorithm to update the CNN's internal weights and biases to minimize the loss functions. We first train pose estimation, and then pose estimate error covariance determination. The CNN is trained using a third of the data collected using our experimental testbed. We use all samples of the LiDAR revisiting a same location to determine truth pose estimation error covariance $\mathbf{P}$ at that location. CNN performance is then validated using the other two thirds of the data.

III. Preliminary Performance Evaluation

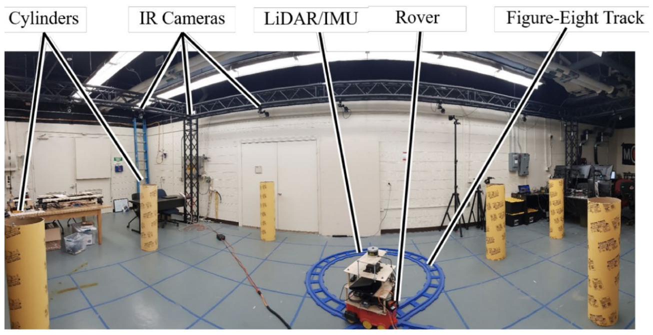
A. *Experimental Setup*

We built an experimental data collection testbed that provides a sufficient number of pose estimate samples to empirically evaluate the true pose error distribution. It comprises a rover repeatedly moving on a figure-eight track and carrying a suite of sensors. The sensor suite includes a Velodyne Puck lidar sensor VLP-16 providing a 360-degree-azimuth, 35-degree-elevation 3D scan of the environment at a 10 Hz sampling rate. A VICON infrared camera-based motion capture system tracks the LiDAR's true pose.

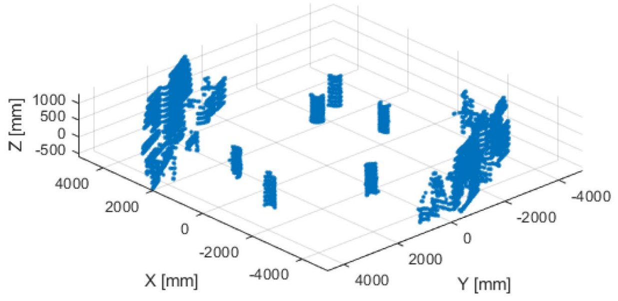
As described in Section II, for guidance in the CNN design, we use the landmark-based localization algorithm in [11]. To facilitate landmark identification in this reference algorithm, we place six cardboard cylinders around the figure-eight track. It is worth noticing that the CNN does not require PC segmentation or landmark extraction, and uses the entire PC, not only a subset of data points corresponding to landmarks. Data collection was performed over 3.5 hours, during which the rover travelled repeatedly on the figure-eight track. It completed 700 laps, the first 100 of which we use in this paper.

B. *LiDAR CNN Positioning Performance*

Out of the 100 figure-eight laps of LiDAR PC and truth data that was collected, 30 laps are used for training, and 70 laps are used for validation. As compared to [12] where we performed localization only, the CNN must learn elements of a covariance matrix in addition to the pose estimate. The model was trained on a Nvidia RTX 3090 graphics card.



(a) Test Bed Setup



(b) LiDAR Point Cloud of Test Bed

Figure 2. Testbed setup and example LiDAR point cloud collected during the experiment.

The estimated position from the LiDAR CNN over 100 laps are plotted against the true trajectory in Figure 3. The red dot-markers represent the LiDAR CNN estimate, the black line represents the true trajectory. Neither the red markers nor the black line are visible because they are underneath the error ellipses.

The gray dots represent the LiDAR PCs at one instant; it was rotated and translated from the original LiDAR-frame for representation in the navigation frame. LiDAR PCs come as inputs to the CNN in sensor frame, and the CNN must learn to perform an equivalent operation to achieve localization in the navigation frame. The six black dots around the figure-eight are landmark locations used in the reference landmark-based algorithm [11].

The zoomed-in window in Figure 3 shows the "1-sigma" CNN-estimated horizontal position estimate covariance ellipses in blue, and the true, post-processed, sample covariance ellipses in orange. Covariance ellipses represent the spread of the estimation error. In the zoomed-in window, both the true and estimated covariance matrices semi-major axes seem reasonably well aligned, with consistently larger error spread along the in-track direction as compared to the off-track direction. Demonstrating consistency between CNN-estimated and true error distributions is a major objective of this paper.

The term "1-sigma" refers to the fact that if the position estimate vector was a bivariate normally distributed random vector, we would expect 40% of the sample data to be within n the "1-sigma" covariance ellipse (we would expect 68% of the data to be within +/- 1 sigma bounds for a one-dimensional random variable).

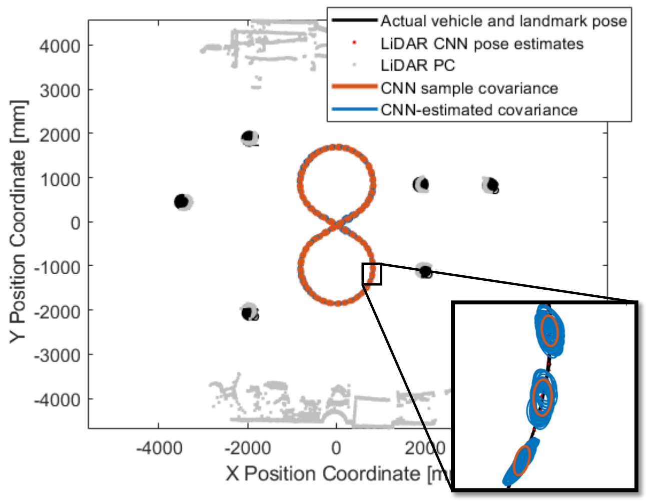


Figure 3. Rover trajectory and covariance ellipses estimated using the CNN-based LiDAR localization algorithm described in Section II (showing horizontal position coordinates only, no heading angle estimate) over 100 repeated figure-eight laps; the estimated covariance ellipses (blue) are good approximations of the true covariance ellipses (orange) .

C. LiDAR CNN Pose Performance Comparison with Model-Based Methods

Figure 4 is a comparison of CNN localization versus a more conventional LiDAR EKF that uses extracted point-features for localization. Figure 4 shows both cross-track positioning errors on the upper charts, and heading angle (or azimuth angle) estimation errors on the lower charts. It does so for the EKF and CNN-based approaches on the left-hand-side and right-hand-side charts, respectively. In all cases, the cross-track and heading angle deviations are lower than 30 mm and 2 deg, respectively. We focused on cross-track positioning deviations because this direction is of primary concern in ground-vehicle lane-centering tasks.

Figure 4(a) shows increased error and error covariance when the LiDAR crosses the tracks' intersection. This occurs in part because of higher vibrations and of landmark feature-extraction errors in the EKF-based approach. After further investigation, we determined that another, more significant source of errors comes from the LiDAR's warm-up period [22].

Figure 5 was generated similar to Figure 4, but we removed the few laps impacted by the LiDAR warm-up period, and we changed the y-axes scales. Both the EKF-based approach in Figure 5(a) and the CNN localization in Figure 5(b) show lower errors as compared to Figure 4. The reduction in error deviation is of factor three, which cannot be neglected, and would have been difficult to identify in a field test.

Position and heading angle error estimation in Figure 4(a) are driven by feature extraction errors. Feature extraction is the process of identifying recognizable landmarks in the environment, in this case, cardboard cylinders. The data is segmented to find the few datapoints originating from the landmarks, and the central axis of the cylinder is computed [11]. Few point features are extracted to minimize the risk of mistaking one landmark from another in data association, i.e., when sending the point feature measurements to the EKF. The localization process can be more robust when using the entire point cloud as in Figure 4(b), which is an advantage of the CNN over this specific EKF-based implementation. Other implementations that are not landmark-based could be considered at the cost of higher computation load (e.g., [6]).

LiDAR warm-up-induced error variations are removed in Figure 5. The true sample error covariance envelopes (black lines) capture the main error variations as the LiDAR's viewpoint changes with vehicle motion. This true covariance envelope is fairly consistent for the EKF-based and CNN-based approaches, with slight increases in heading angle estimation error deviations at times 4s-to-6s and 14s-16s when the vehicle crosses the tracks intersection.

In Figure 5(b), the red lines represent the CNN-estimated covariance envelopes, which are constituent over multiple laps, and also match the true envelope fairly well, although there is a slight time offsets between black and red lines. We further investigate this offset in the next section.

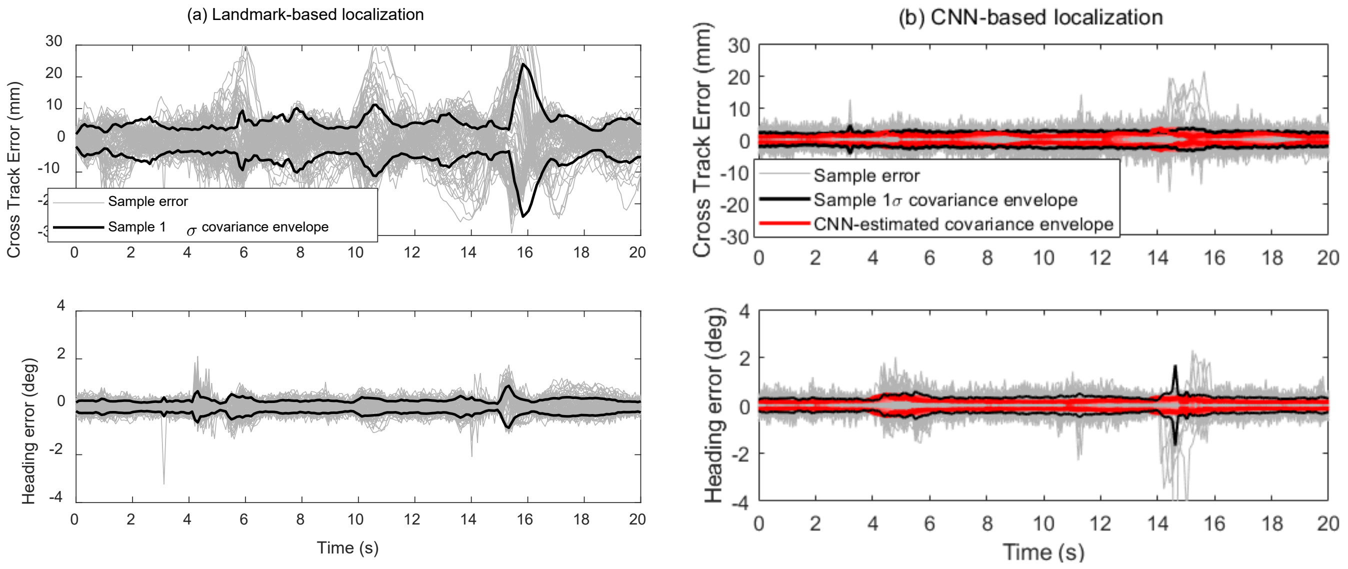


Figure 4. Cross-track and heading angle estimation errors obtained: (a) using LiDAR EKF; (b) using LiDAR CNN. Overall, the estimate errors are small, but significant feature extraction error variations impact the EKF-based approach. The LiDAR CNN method does not require explicit feature extraction. The LiDAR CNN uses the entire point-cloud, whereas the LiDAR EKF localization methods only uses datapoints corresponding to identified landmarks.

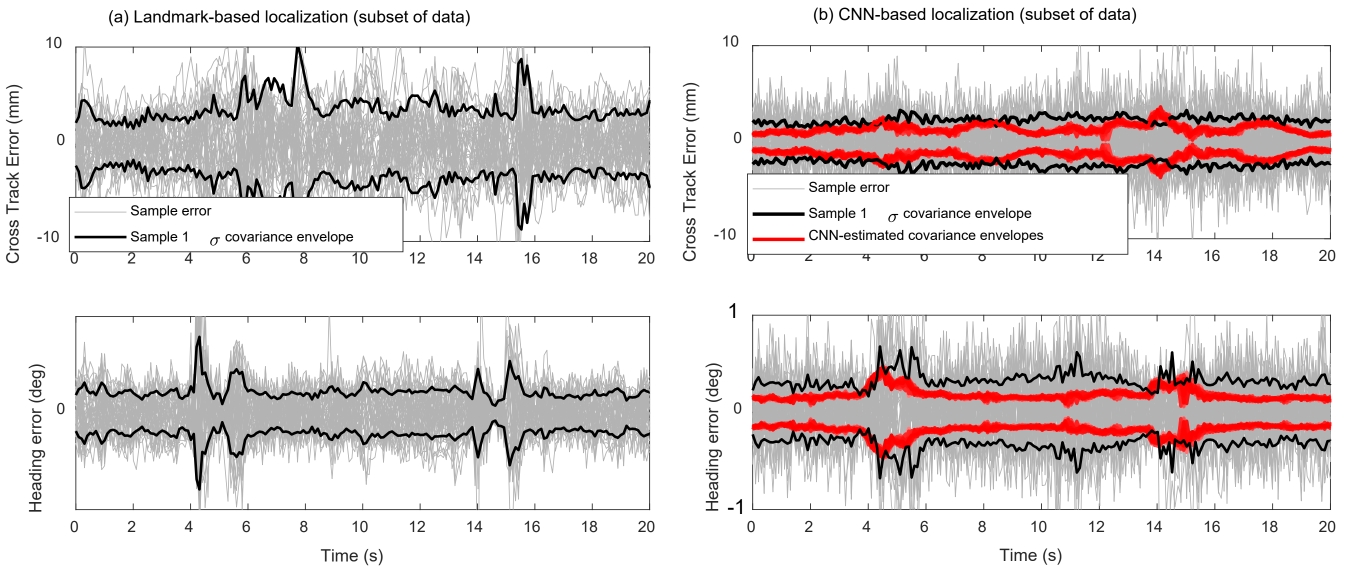


Figure 5. After removing laps impacted by the LiDAR warm-up period, this figure shows the cross-track and heading angle estimation errors obtained: (a) using LiDAR EKF; and (b) using LiDAR CNN. The estimated error covariance envelope for LiDAR CNN (red lines in (b)) is consistent over multiple laps (mostly overlapping) and is consistent with the true covariance envelope (black line). It is worth noticing that the Y-axis scales changed as compared to Figure 4.

D. LiDAR CNN Pose Performance Comparison with Model-Based Methods

Figure 6 shows the LiDAR CNN covariance envelope estimated during training (light-green line), which does not perfectly fit the true covariance (black line). This suggests that the CNN is not over-fitted during training. The red lines are the estimated covariance obtained during validation.

The figure shows 30 overlapping and indistinguishable green covariance envelopes and another more than 30 red covariance envelopes. They are both self-consistent over multiple laps. They are also consistent in magnitude with the true covariance envelope (black). The red curves are also consistent with the green envelopes derived during training, but there appears to be a slight time offset that we suspect is caused by imperfect calibration of our experiments.

Further evaluation of the experimental setup and data processing algorithm will be performed in future work to find the source of this apparent slight timing inconsistency.

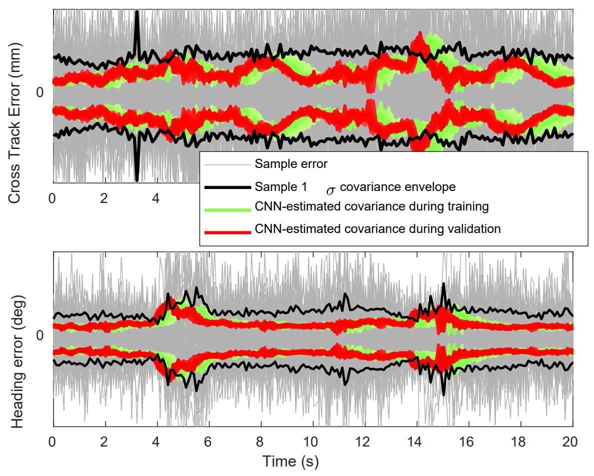


Figure 6. Cross-track and heading angle estimation errors, true error covariance, and estimated error covariance obtained using a LiDAR CNN. The estimated error covariance envelope achieved during training (green lines) and validation (red lines) are consistent with the true covariance envelope (black line) and are consistent over multiple laps. The apparent timing offset between green and red lines will be investigated in future work.

IV. Conclusion

In this paper, we designed a new Convolutional Neural Network (CNN) architecture, which we trained to directly estimate position and orientation (pose) from a LiDAR point cloud (PC) without explicit extraction of environmental features. The CNN architecture is streamlined to take as inputs two arrays corresponding to the LiDAR's distance and light-return intensity measurements over a range of azimuth and elevation angles that define the sensor's field of view. The CNN outputs include both pose estimates and pose estimation uncertainty in the form of elements of an estimation error covariance matrix.

We carried out a preliminary analysis of the LiDAR CNN's performance using data collected in a known lab environment. We evaluated the LiDAR CNN performance in comparison with a more conventional EKF-based method that extracts and associates landmark features from the LiDAR PC for localization. The lab test helped analyze the system, and illustrated the CNN's robustness to sensor errors during the LiDAR warm-up period. Future work will aim at quantifying the LiDAR CNN's ability to evaluate pose uncertainty in realistic ground vehicle environments.

References

[1] F. Wang and Z. Zhao, "A Survey of Iterative Closest Point Algorithm," *2017 Chinese Automation Congress (CAC)*, Jinan, China, 2017, pp. 4395-4399, doi: 10.1109/CAC.2017.8243553.

[2] J. Levinson and S. Thrun, "Robust vehicle localization in urban environments using probabilistic maps," *IEEE Int.Conference on Robotics and Automation (ICRA)*, 2010, pp. 4372–4378.

[3] J. Castorena and S. Agarwal, "Ground-edge-based LIDAR localization without a reflectivity calibration for autonomous driving," *IEEE Robot. Autom. Lett.*, vol. 3, no. 1, pp. 344–351, 2018.

[4] R. W. Wolcott and R. M. Eustice, "Fast lidar localization using multiresolution gaussian mixture maps," *IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 2814–2821.

[5] Z. Luo, M. V. Mohrenschildt, S. Habibi, "A Probability Occupancy Grid Based Approach for Real-Time LiDAR Ground Segmentation," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 3, pp. 998-1010, March 2020, doi: 10.1109/TITS.2019.2900548.

[6] A. Hassani and M. Joerger, "A new point-cloud-based lidar/imu localization method with uncertainty evaluation," *Proceedings of the 34th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2021)*, 2021, pp. 636–651.

[7] A. Hata and D. Wolf, "Road marking detection using lidar reflective intensity data and its application to vehicle localization," *17th International Conference on Intelligent Transportation Systems (ITSC)*, 2014, pp. 584–589.

[8] M. Joerger, M. Jamoom, M. Spenko, B. Pervan, "Integrity of Laser-Based Feature Extraction and Data Association," *Proceedings of IEEE/ION PLANS 2016*, Savannah, GA, April 2016, pp. 557-571.

[9] M. Joerger, B. Pervan, "Continuity Risk of Feature Extraction for Laser-Based Navigation," *Proceedings of the 2017 International Technical Meeting of the Institute of Navigation*, Monterey, California, January 2017, pp. 839-855.

[10] A. Hassani, M. Joerger, G. D. Arana, and M. Spenko, "LiDAR data association risk reduction, using tight integration with INS," *Proceedings of the 31st International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS+ 2018)*, 2018, pp. 2467–2483.

[11] A. Hassani, N. Morris, M. Spenko, and M. Joerger, "Experimental integrity evaluation of tightly-integrated IMU/LiDAR including return-light intensity data," *Proceedings of the 32nd International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS+ 2019)*, Miami, FL, 2019.

[12] J. Wang, A. Hassani, M. Joerger, "Navigation Performance of a LiDAR-Based CNN in a Known Environment" *Proceedings of the International Technical Meeting of the Institute of Navigation (ION ITM 2022)*, 2022.

[13] W. Lu, Y. Zhou, G.Wan, S. Hou, and S. Song, "L3-net: Towards learning based lidar localization for autonomous driving," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 6389–6398.

[14] I. A. Barsan, S. Wang, A. Pokrovsky, and R. Urtasun, "Learning to localize using a lidar intensity map," *2nd Conference on Robot Learning (CoRL)*, 2018.

[15] R.L. Russell, C. Reale, "Multivariate Uncertainty in Deep Learning," *IEEE Transaction on Neural Networks and Learning Systems*, 2021.

[16] Liu, K., Ok, K., Vega-Brown, W., & Roy, N. (2018, May). Deep inference for covariance estimation: Learning gaussian noise models for state estimation. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1436-1443, 2018.

[17] Y. Gal and Z. Ghahramani, "Dropout As a Bayesian Approximation: Representing Model Uncertainty in Deep Learning," *Proceedings of the 33rd International Conference on International Conference on Machine Learning, ICML'16*, vol. 48, New York, NY, 2016, pp. 1050–1059.

[18] F. Verdoja, J. Lundell and V. Kyrki, "Deep Network Uncertainty Maps for Indoor Navigation," *2019 IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids)*, 2019, pp. 112-119, .

[19] A. Kendall, and Y. Gal, "What uncertainties do we need in bayesian deep learning for computer vision?" *31st Conference on Neural Information Processing Systems (NIPS 2017)*, Long Beach, CA, USA, 2017.

[20] A. Schlichting, and U. Feuerhake, "Global vehicle localization by sequence analysis using lidar features derived by an autoencoder," *2018 IEEE Intelligent Vehicles Symposium (IV2018)*, 2018, pp. 656-661.

[21] Torch Contributors, "BCELOSS", Technical function description, 2019. Available online at https://pytorch.org/docs/stable/generated/torch.nn.BCELoss.html

[22] S. Cattini, D. Cassanelli, L. D. Cecilia, L. Ferrari and L. Rovati, "A Procedure for the Characterization and Comparison of 3-D LiDAR Systems," *IEEE Transactions on Instrumentation and Measurement*, vol. 70, 2021, pp. 1-10, doi: 10.1109/TIM.2020.3043114.