Deep LiDAR localization using optical flow sensor-map correspondences

Anders Sunegård
Volvo Autonomous Solutions
Gothenburg, Sweden
anders.karlsson@volvo.com

Lennart Svensson
Chalmers University of Technology
Gothenburg, Sweden
lennart.svensson@chalmers.se

Torsten Sattler
Czech Technical
University in Prague
torsat@chalmers.se

Abstract

In this paper we propose a method for accurate localization of a multi-layer LiDAR sensor in a pre-recorded map, given a coarse initialization pose. The foundation of the algorithm is the usage of neural network optical flow predictions. We train a network to encode representations of the sensor measurement and the map, and then regress flow vectors at each spatial position in the sensor feature map. The flow regression network is straight-forward to train, and the resulting flow field can be used with standard techniques for computing sensor pose from sensor-to-map correspondences. Additionally, the network can regress flow at different spatial scales, which means that it is able to handle both position recovery and high accuracy localization. We demonstrate average localization accuracy of <0.04m position and <0.1° heading angle for a vehicle driving application with simulated LiDAR measurements, which is similar to point-to-point iterative closest point (ICP). The algorithm typically manages to recover position with prior error of more than 20m and is significantly more robust to scenes with non-salient or repetitive structure than the baselines used for comparison.

1. Introduction

With the introduction of high performance driver assistance systems and automated driving functionality, the requirement on exact position knowledge has increased to the point where satellite-based localization must be coupled with costly correction services to provide adequate accuracy. Even then, such solutions are subject to severe availability and reliability issues, due to various problems including multi-path signals and signal outages in challenging conditions.

A common alternative for high precision localization is to map areas of interest, and then localize relative the pre-recorded map. The LiDAR sensor is robust to illumination and texture variability, which is an important advantage compared to camera sensors for the localization task. Furthermore, the LiDAR sensor is useful in other tasks, such as object detection and tracking, which makes it a practical choice for autonomous vehicles.

Most localization methods divide the problem into a position retrieval stage, commonly referred to as global localization, and a local refinement stage. The coarser global localization is often left to external sensors such as satellite-based positioning, but there are LiDAR-based position retrieval methods, such as PoseMap [9] which encodes a whole scene into a descriptor and retrieve position by finding closest matches in descriptor space. Our method mostly qualifies as a local refinement algorithm, but with extended capabilities to compute position also when prior pose information is inaccurate.

Early LiDAR localization methods [15, 16] use template matching to find the rigid transform that maximizes correlation between the sensor data and the map. To achieve this, both sensor and map points are projected into 2D images from a top view perspective, and templates resulting from all transforms in a discrete search space are correlated with the map. Localization accuracy is generally sub-decimeter, but the search space must be constrained for computational complexity reasons, meaning that an accurate sensor pose prior is required.

Another option for LiDAR localization is to apply a point cloud registration method. A large body of registration methods finds a set of correspondences, i.e. pairs of matching features in sensor data and map, and computes the rigid body transformation that best aligns sensor data with the map [2,10,29,31,32]. Iterative closest point (ICP) methods [4, 23] do repeated closest distance searches to determine correspondences, and step-by-step approach an alignment. ICP and related methods suffer from a tendency to converge to a local minimum when initialization is inaccurate, and from their repeated, computationally costly correspondence searches. Fast global registration (FGR) [33] addresses such shortcomings by computing correspondences once, using local feature descriptors, and directly solves for the pose by minimizing a global objective. FGR is fast and less affected by the problem with local minimums, but it is sen-

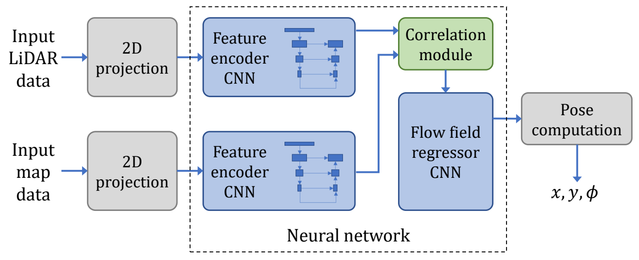

Figure 1: Schematic view of the localization pipeline for one resolution level. The 2D projection is normally sparse, especially for the LiDAR measurement, and the U-Net CNN that up-samples pooled features results in a dense feature representation. The correlation module is non-trainable and provides similarity measures for the neighborhood around each input grid position, which are used to regress the flow field that allows pose to be computed.

sitive to incorrect or ambiguous correspondence matches. Recent registration literature has applied deep learning to encode better performing descriptors [29, 31, 32] and to detect key points whose descriptors are likely to form accurate matches [29, 31]. While this has led to significant improvements in descriptor performance for registration, encoding of point descriptors that capture both the large structure shape required for global matching and the fine detail for precision localization remains unsolved to the best of our knowledge.

In this paper we combine ideas from learning based correspondence registration with efficient 2-dimensional data representations and optical flow methods. The proposed method yields robust localization performance, even with initial errors as high as 20m, combined with accuracy comparable to the best local registration methods, and high computation speed. An overview image of the method architecture is shown in Figure 1.

Specifically, we process 2D projections of sensor and map point clouds in trained U-Net [22] CNNs, providing dense feature maps with large receptive field. Correspondences are then regressed as a flow field between the images, using a neural network structure and correlation modules similar to those used for optical flow computations in FlowNet [12] and PWC-Net [26]. We thus avoid computationally costly global searches for correspondences in descriptor space and shift the complexity to an optionally GPU-accelerated regression network. To increase accuracy and robustness to outliers, we regress uncertainty measures for each correspondence vector and use them as importance weights when computing the final pose. Global localization capabilities are achieved by a variant of the pyramid structure from PWC-Net [26], which serves to find a coarse pose estimate from an uncertain prior and gradually refines the pose.

To summarize, our contribution is three-fold: 1) We describe an optical flow formulation of the LiDAR localization problem, which leverages on learning-based optical flow and the U-Net architecture, making the method computationally efficient. 2) We apply a coarse-to-fine model structure for correspondence localization, that achieves high accuracy while being relatively insensitive to prior knowledge about the sensor position. 3) We demonstrate the use of regressed uncertainty measures for each correspondence vector to weigh their influence in the final pose computation, as such contributing to outlier insensitivity and improved localization accuracy.

2. Related work

Most LiDAR localization methods, including ours, use an off-the-shelf solution, such as a GPS receiver, to provide a initial position estimate that is subsequently refined by the method in question. We compare our method against local refinement methods which include purpose-made LiDAR localization methods and the larger body of generic point cloud registration methods.

2.1. Lidar Localization

A family of LiDAR localization methods is based on the 2D projection of LiDAR sensor data and map, where localizing is done by finding the alignment transform that maximizes correlation between the two images. Correlation is computed at each pose in a discrete search space to find the maximum correlation pose. In practise due to computational cost, this means it is necessary to make a compromise between small search radius and accuracy. The method by Levinson et al. [15, 16] uses laser reflection intensity in the map and discards the vertical coordinate. The same concept is expanded with deep learning methods in [3] where map and sensor data are encoded with a trained network. Our method makes use of 2D projections and correlation computations, but relies on shape rather than intensity to encode feature maps and uses a coarse to fine approach to avoid issues with constrained search spaces.

A 3D version [20] of normal distribution transform (NDT) [5] finds the pose by representing the point cloud map as a piece-wise continuous function in a voxel grid, to which the sensor points are aligned by minimizing a global objective. To achieve reasonable computation times, the voxel grid is typically made coarse which may limit accuracy, but the convergence valley extends beyond the 1-2m search radius of typical local refinement algorithms.

L3-Net [19] uses 3D structure tensors to detect salient features in the online LiDAR measurement and matches only the most distinct points to the map by patch correlation. Different pose hypotheses are evaluated locally for all the extracted key points and the results are aggregated in a matching volume where the most likely pose can be inferred. High localization accuracy is obtained, but like [3, 15, 16] the method is limited to local refinement.

SegMap [8] is a correspondence-based LiDAR localization method which clusters point clouds into segments that are encoded, classified, and used for correspondence matching if they fulfil certain requirements. The method achieves high matching performance when distinct segments are available. Likely due to some variability in the segment clustering, its localization accuracy does not reach sub-decimeter levels, but on the other hand the method is capable of global localization.

Visual odometry methods based on LiDAR [17, 27, 28] typically register scans using standard point cloud registration methods, further described in section 2.2. While such methods bear resemblance to localization, the scope is typically limited to consecutive scan matching, and sometimes restricted to LiDAR inherent spherical space [27,28], which is unsuitable for localization.

2.2. Point cloud registration

For a point cloud registration method to be suitable for precision localization, it must handle the difference in point density between sensor and map point clouds, and it must also handle that the sensor data only overlaps with a part of the map. These requirements lead to correspondence matching methods, where correspondence candidates from non-overlapping parts of the data can be detected and disregarded.

The class of iterative closest point (ICP) registration methods [4, 23] establishes correspondences based on closest distance measures and computes a pose in an iterative fashion. As such, ICP-methods are prone to converging to a local solution, and can be computationally expensive due to the need to re-compute correspondences iteratively. Global search extensions can be used to perform registration under larger position uncertainty, for example the branch-and-bound algorithm of [30], but such methods incur increased computation time.

An important branch of point cloud registration algorithms uses feature descriptors to form correspondence matches in descriptor space. Capable descriptor encoders allow finding correct correspondences without iterating through intermediate pose transforms, which can have great impact computation time, as demonstrated in [33]. Another consequence is that it is possible to make algorithms less sensitive to initialization and incorrect local solutions.

Learning 3D feature descriptors. Encoding of 3D structure descriptors in a way that results in distinct correspondence matching is not trivial. Recent registration methods [2, 10, 14, 18, 29, 32] train neural networks to compute descriptors such that corresponding features closely match in descriptor space. The commonly used [1, 14, 18, 21, 31] PointNet descriptor encoder directly consumes a set of points, often partitioned into smaller local neighborhoods to form descriptors for different points. Unlike a traditional CNN, whose receptive field can quickly grow through the convolution chain, the receptive field of a local PointNet descriptor is bound by its input points. Unless such descriptors are further processed, the limited receptive field may limit the global localization capabilities.

Another option for encoding descriptors, without limiting the receptive field are different flavors of continuous convolution operators, as used in [2, 29]. Continuous convolutions do not require a voxelization of the point cloud but are harder to optimize for speed on a GPU. The feature encoder of [6] discretizes the point clouds into voxel grids with simple hand crafted features in each populated voxel, which allows using 3D convolution operators, modified for sparse input, to compute a descriptor at each populated voxel. The feature encoder of our method can be viewed as a 2D adaptation of [6], with lowered computation times as a result of the decrease in dimension.

Correspondence matching. Correspondences between point sets are in general not exact one-to-one mappings, i.e. the same point is not available in both sets, which arises due to the variation in sampling of points, partially overlapping sets, or different sampling densities (as is common in LiDAR localization). Therefore, pair-wise closest distance correspondences may lead to inaccurate or completely failed registration. As a counter-measure, some descriptor correspondence methods [32] find soft matches such that each source point is matched with several destination points, along with matching scores to weigh the correspondences when computing alignment transforms. Such extensions solve major issues with descriptor space matching, but the results are still limited by inaccurate point-to-point matches.

An alternative to descriptor space matching is to compute correspondence vectors, where only one or neither of the matched positions are actual points in the input point clouds. A simple example are ICP-extensions [23] that match points from the source input to planes in the destination. DeepVCP [18] computes tensors of descriptor distances between each source point and a set of different destination position hypotheses. The tensors are processed in a CNN that regresses correspondence vectors. Our method implicitly uses descriptor correlation as distance measure, and the 2D formulation allows correlation tensors to be computed in a discrete search space that is limited to the horizontal plane. Furthermore, our coarse to fine approach results in a significantly larger search space for correspondences, yet with a bounded increase in computation cost.

When parts of the point clouds contain little or no salient structure, feature descriptors become indistinct which results in matching uncertainty. A common solution [2, 18, 29, 31] is to use attention mechanisms to learn to detect points that are good candidates for correspondence matching and then give reliable correspondences higher importance in transform computation. Our method weighs flow field components by regressing uncertainty distribution parameters for each flow vector, using a log-likelihood loss as described in [11].

3. Method

Our method solves the localization problem by first estimating a flow field between the sensor and map coordinate frames, and then using the flow field to compute the sensor location, i.e., to estimate how far off our prior location is in terms of translation and angle. Specifically, we discretize the sensor and map cloud data into 2D grids from a top-view perspective with handcrafted features for each grid cell. We then use a neural network to regress a flow field, i.e., a set of 2D vectors that estimate the translation of the center of each grid cell in the sensor image into the map coordinate frame.

We assume that the map is available as a point cloud, and that the vertical direction of the sensor is known, so its data can be transformed into a coordinate system whose vertical axis is aligned with the gravitational axis. Furthermore, we require a prior on the sensor pose that is accurate within approximately 20m, and 20° heading angle. Normally, such a prior is available from satellite-based localization, or from inertial odometry based on a previous localization. The prior position defines the center point of the area of the map to extract for feature image construction.

Figure 1 summarizes our method, including the key components from the input point clouds. Using top-view projections of sensor data and map crop, feature maps are computed for both inputs. Since the 2D projection of a point cloud is in general sparse, we apply a U-Net [22] with skip connections to encode dense feature images with large receptive fields, capable of capturing large structure features. The subsequent flow field regression follows the outline of the FlowNetC model introduced in [12], where the key component is a correlation volume computation. A CNN is then used to regress flow vectors and associated covariance matrix parameters from the correlation volume output. From the regressed probabilistic flow field, the rigid transform is computed as the maximum likelihood-estimate of the pose.

The following sections describe further details of the method.

3.1. Data pre-processing

To cast the problem into the optical flow formulation, we need to transform the input points from sensor and map to appropriate coordinate systems. Using the prior information of the sensor's vertical axis and its heading we define transform operator $\mathbf{T}_{ES}$ that is used to rotate points expressed in the sensor coordinate frame S to the error frame E, which is aligned with the map axes, but with a remaining error in heading due to the prior error. The map points are extracted from a square area in the map centered in the prior position, and then translated from the map coordinate frame M to the crop frame C with origin in the prior position by constructing $\mathbf{T}_{MC}$ and applying its inverse on the extracted points. The sought sensor pose transform $\mathbf{T}_{MS}$, relative the map coordinate frame M, can be computed as

$$\mathbf{T}_{MS} = \mathbf{T}_{MC}\mathbf{T}_{CE}\mathbf{T}_{ES} \tag{1}$$

where $\mathbf{T}_{CE}$ is the to-be-computed pose correction transform that aligns the rotated sensor points with the translated map crop points.

The transformed point sets are partitioned into 2-dimensional grids in the horizontal plane, where the sensor grid contains $W_s \times H_s$ cells. The map grid is a larger $W_m \times H_m$ size to support flow vector end points outside the sensor grid borders. For each grid cell, we compute feature vectors $\mathbf{x} = [n, \tilde{z}, \sigma]^T$ of the number of contained points n, mean $\tilde{z}$ and standard deviation σ of the points' vertical coordinates, resulting in sensor and map input tensors ${}_E\mathbf{X}_s$ and ${}_C\mathbf{X}_m$.

3.2. Flow field regression

For a given resolution level l, the corresponding flow field regressor function $(\mathbf{F}_{CE}, \boldsymbol{\theta}_\Sigma) = f^{(l)}({}_E\mathbf{X}_s, {}_C\mathbf{X}_m)$ is a neural network that outputs $2 \times W^{(l)} \times H^{(l)}$ flow field tensor $\mathbf{F}_{CE}$ and $3 \times W^{(l)} \times H^{(l)}$ flow covariance parameters tensor $\boldsymbol{\theta}_\Sigma$. We enumerate each spatial grid cell in the output tensors with the index $i \in [1, N^{(l)}]$ where $N^{(l)} = W^{(l)}H^{(l)}$. We use the index to denote $\mathbf{f}_i$, the flow vectors from each grid cell of $\mathbf{F}_{CE}$, $\boldsymbol{\theta}_i$, the parameters of the covariance matrix for each flow vector, and $\mathbf{p}_i$, the grid cell center point. The neural network is trained with ground truth flow fields $\mathbf{F}_{gt}$ using a log-likelihood loss

$$\mathcal{L}(\boldsymbol{\theta}_\Sigma, \mathbf{F}_{CE}, \mathbf{F}_{gt}) = \sum_{i=1}^{N} \log(\det \boldsymbol{\Sigma}(\boldsymbol{\theta}_i)) + (\mathbf{f}_i - \mathbf{f}_{i,gt})^T \boldsymbol{\Sigma}(\boldsymbol{\theta}_i)^{-1}(\mathbf{f}_i - \mathbf{f}_{i,gt}) \tag{2}$$

where parameters $\boldsymbol{\theta}_i$ of covariance matrices $\boldsymbol{\Sigma}(\boldsymbol{\theta}_i)$ at each grid cell are regression variables.

The neural network that defines regressor function $f^{(l)}({}_E\mathbf{X}_s, {}_C\mathbf{X}_m)$ is structured according to Figure 1, with a feature encoder for each input, followed by a feature correlation module, and the probabilistic flow regression module.

The encoders use a U-Net [22] structure with skip connections to encode the sparse inputs into feature maps with large receptive fields. The network has one down-sampling chain that applies 3x3 2D convolutions in 6 groups of 3 convolutional layers each. Each group halves the spatial dimensions of the tensor. The chain is followed by an up-sampling

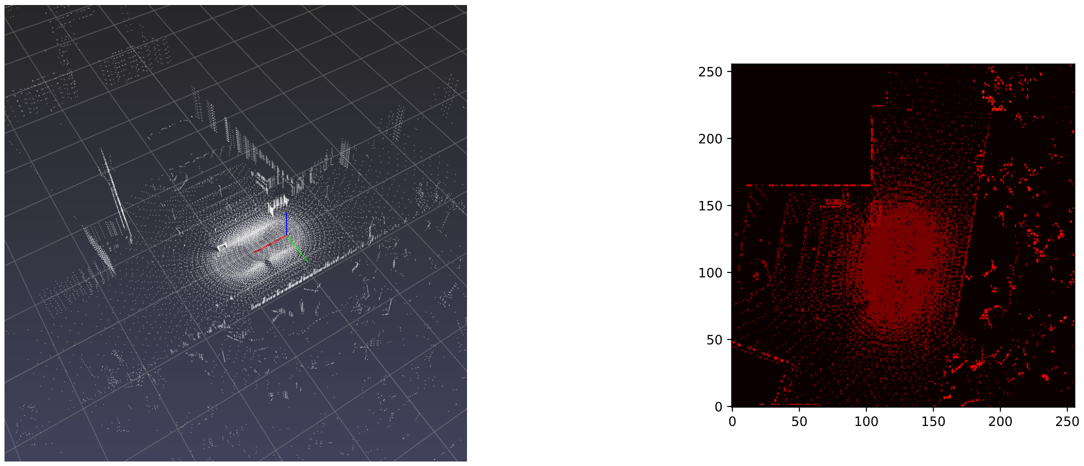

Figure 2: A sensor measurement point cloud (left) and the mean(z) channel of its corresponding top-view input feature image.

chain with the same structure as the down-sampling chain, but each group has a skip connection input from the down-sampling chain. The up-sampling chain contains groups up to the spatial dimension determined by the multi-level localization procedure.

The correlation module computes the scalar products of the feature vector at each location in the encoded sensor feature tensor and feature vectors from a set of neighboring locations around the same position in the map image. To accommodate neighbor locations outside the sensor image borders, the map data image is extracted from a larger area such that fits all neighbors. The operation results in a vector of scalar products per location in the sensor feature image, where each component is associated by a neighborhood location.

The correlation volume is input into the flow field regressor network, which has a base of 5 3x3 2D convolution layers. The base is followed by one branch of 3 convolution layers for regressing flow field $\mathbf{F}_{CE}$, and another branch of 4 layers for regressing covariance parameters tensor $\boldsymbol{\theta}_\Sigma$.

3.3. Pose computation

To compute the sensor pose from the regressed flow field, we compute the maximum likelihood estimate of pose correction transform $\mathbf{T}_{CE}$. To express the likelihood, we model the flow field vectors in terms of $\mathbf{T}_{CE}$. Given the transform $\mathbf{T}_{CE}$ and a start point $\mathbf{p}_i$ of a flow vector, we can express the true flow vector as

$$\mathbf{h}_i(\mathbf{T}_{CE}) = \mathbf{T}_{CE}\mathbf{p}_i - \mathbf{p}_i. \tag{3}$$

which we use to model the regressed flow field vector as

$$\mathbf{f}_i = \mathbf{h}_i(\mathbf{T}_{CE}) + \mathbf{e}_i \tag{4}$$

where $\mathbf{e}_i \sim \mathcal{N}(0, \boldsymbol{\Sigma}_{\theta,i})$ is the flow vector error, modelled with the regressed covariance matrices. Expressed as a probability density, we have

$$p(\mathbf{f}_i|\mathbf{T}_{CE}) = \mathcal{N}(\mathbf{f}_i; \mathbf{h}_i(\mathbf{T}_{CE}), \boldsymbol{\Sigma}_{\theta,i}). \tag{5}$$

Under the assumption that flow vectors $\mathbf{f}_i$ are conditionally independent, we can describe the distribution of the whole flow field as

$$p(\mathbf{f}_1, \mathbf{f}_2, ..., \mathbf{f}_N|\mathbf{T}_{CE}) = \prod_{i=1}^{N} p(\mathbf{f}_i|\mathbf{T}_{CE}). \tag{6}$$

While this assumption is not accurate in the general case, it is functional for our purpose of weighting flow vectors for pose computation.

The error correction transform $\mathbf{T}_{CE}$ is parameterized by translation $[x, y]^T$ and heading angle φ. The log likelihood can then be written as

$$\log L(\mathbf{T}_{CE}|\mathbf{F}_{CE}) \propto - \sum_{i=1}^{N} \left(\begin{bmatrix} x \\ y \end{bmatrix} - \boldsymbol{\mu}_{i,j} \right)^T \boldsymbol{\Sigma}_{\theta,i}{}^{-1} \left(\begin{bmatrix} x \\ y \end{bmatrix} - \boldsymbol{\mu}_{i,j} \right) \tag{7}$$

where $\boldsymbol{\mu}_i = \boldsymbol{\mu}_i(\varphi, \mathbf{p}_i, \mathbf{f}_i)$ can be evaluated for any given φ. We sample a set of M heading angle hypotheses $\varphi_j, j \in [1..M]$ from a suitable search range, and compute all $\boldsymbol{\mu}_{i,j}, i \in [1..N], j \in [1..M]$. Then $\hat{x_j}, \hat{y_j}$ that maximize Eq. 7 for each heading hypothesis φ_j are computed analytically as

$$\begin{pmatrix} \hat{x}_j \\ \hat{y}_j \end{pmatrix} = \sum_{i=1}^{N} \boldsymbol{\Sigma}\boldsymbol{\Sigma}_{\boldsymbol{\theta},i}{}^{-1} \boldsymbol{\mu}_{i,j} \tag{8}$$

where

$$\boldsymbol{\Sigma} = \left(\sum_{i=1}^{N} \boldsymbol{\Sigma}_{\boldsymbol{\theta},i}{}^{-1} \right)^{-1}. \tag{9}$$

The maximum likelihood estimate $\hat{x}, \hat{y}, \hat{\varphi}$ is found by identifying the heading hypothesis φ_j and corresponding $\hat{x_j}, \hat{y_j}$ that evaluates to the highest likelihood of all j. Finally, we construct $\hat{\mathbf{T}}_{CE}$ from the estimated parameters and compute the pose transform $\mathbf{T}_{MS}$ using (1).

3.4. Multi-scale localization

To overcome issues with the limited search space connected to the use of a correlation volume, with only a smaller impact on computational performance, we build on ideas from the optical flow method PWC-Net [26]. PWC-Net uses a coarse-to-fine approach to successively resolve flow in a pyramidal process. In our case the flow field is expected to follow a rigid transform, so the pose is estimated in each iteration, and the next iteration resolution is one step finer than the current. When the prior pose is precise, only the finest resolution flow need to be computed, but for occasional re-locating the coarser localization levels can be applied initially to bootstrap the error. In practise this means that we train multiple versions of the localization pipeline in figure 1.

4. Experimental results

4.1. Dataset extracted from CARLA

In this paper we rely entirely on synthetic data extracted from the CARLA simulator [7] for training and verification. The used version 9.8 of the CARLA simulation software includes 7 different worlds, covering urban, rural and highway scenarios. The simulation allows constructing a point cloud map that is unaffected by the quality of reference localization, and provides large quantities of annotated measurements. We use the built-in LiDAR simulation module, configured to capture 56000 points per second distributed over 32 layers covering pitch angles between -30° and 10° relative the horizontal plane. The sensor range is set to 100 meters and rotation speed to 20Hz. For each world, a point cloud map is aggregated by traversing the simulated LiDAR sensor along all road segments, in 1 meter increments, positioned 2.4m above ground level. At each position increment, a full revolution scan is collected and added to the map point cloud. The simulated LiDAR returns an instantaneous snapshot image that is unaffected by the sensor's travelling speed, so no rectification is needed. In the same way, simulated online measurement data is collected, and each training example is aggregated from 10 consecutive LiDAR scans, equalling half a second of measurement data. All in all we get 42870 unique sensor measurements and map crop samples from the 7 worlds, of which all 5772 from world 4 are used for validation and all 2013 from world 2 are used in experimental comparisons.

The dataset includes natural occlusions, such that the proximal map contains data that is not seen in sensor measurements, due to objects blocking the line of sight. Thus, our algorithm is implicitly trained to manage such occlusions, and the following evaluations test the algorithms' performance in partially occluded scenes. The opposite scenario, where the measurement scans contain data from objects that are not in the map, is not included in the dataset.

4.2. Network Training details

The training data was infinitely augmented by rotation, such that both the map image and sensor points of each sample were rotated randomly in the horizontal plane. This was found to be necessary to avoid overfitting, since the included CARLA worlds have a strong emphasis on straight roads/features/buildings in north-south or east-west directions. For training optimization we used ADAM [13] with its standard parameters. The step size was set to a fixed 0.0003. At cold start, we found it necessary to use a regular L1 loss function to find a starting point of non-trivial features.

4.3. Baseline algorithms

We compare our localization results with the iterative closest point (ICP) [4], the more recent Fast Global Registration (FGR) [33] and normal distribution transform (NDT) [20] methods. We choose point-to-point ICP since it is a standard example of a Euclidean distance registration method, that performs well for for fine precision localization with our data sets. FGR, on the other hand, is a minimum descriptor distance registration method, that achieves both global and fine precision local registration. NDT is commonly used in robotics, providing good accuracy also with less precise initialization. We use implementations of ICP and FGR from the Intel Open3D library [34]. With FGR we use the standard fast point feature histogram (FPFH) descriptor [24] that provides strong local 3D features. For ICP we use a limit of 1000 iterations, and a stopping condition for converged solutions that limits the number of iterations to between 30 and 300 for most localizations. For NDT we use the point cloud library implementation [25].

4.4. Evaluation metrics

We divide our results into translation errors and heading errors. The translation error is computed as the distance between ground truth position and estimated position in the horizontal plane. Vertical errors returned from baselines that perform registration in 3 dimensions are discarded. The heading error is extracted from the alignment transform. Any remaining pitch or roll angle components are disregarded for fair comparisons.

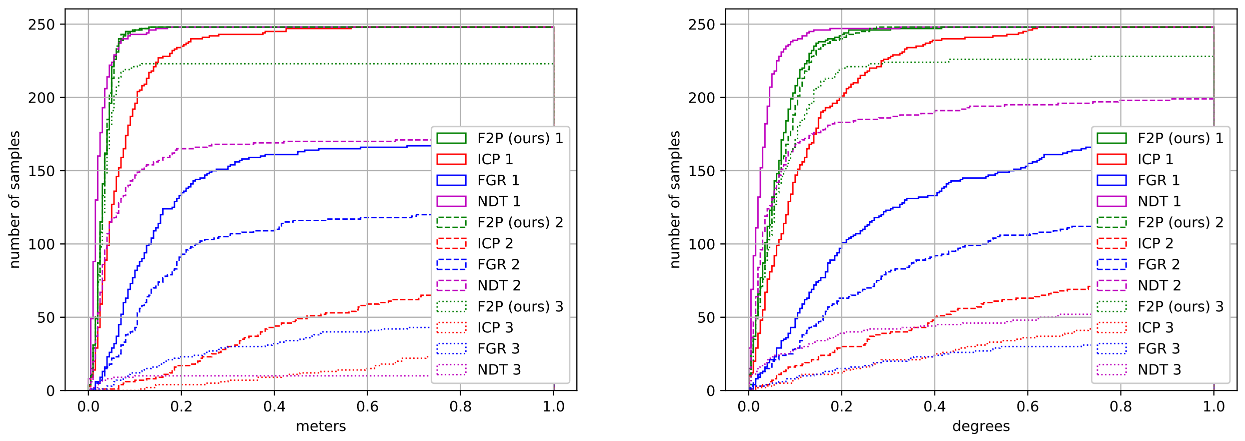

Figure 3: Cumulative histograms of translation error (left) and heading error (right) for our method (green), point-to-point ICP (red), fast global registration (blue) and NDT (magenta). Results are displayed for different initialization translation and heading angle errors: Prior 1 (2.0m, 3.5°), prior 2 (8.0m, 10°) in dashed, and prior 3 (20.0m, 20°) in dotted line.

4.5. Localization

Localization performance is evaluated using 3 different initialization errors, $2m/3.5°$, $8m/10°$ and $20m/20°$ translation/heading angle errors, denoted prior 1, 2 and 3 respectively. The translation vector has a $45°$ direction to make sure that no possible bias to translation along the map coordinate system axes can be exploited. The results are compiled in the cumulative histograms in Figure 3 that visualize translation and heading localization performance for our algorithm, referred to as flow-to-pose (F2P), in relation to each of the baselines. We also list results in Table 1, including numbers for the percentage of samples localized within a 0.1, 0.3 and 1 meters threshold.

From the histograms it is apparent that the ICP baseline performs well when initialized with a small 2m translation error, such that no alternative solutions are near. Our F2P still performs better than ICP in this setting, with 98.8% of the tested samples localized within 10cm. For larger initialization errors, ICP needs more iterations to converge, and often fails by converging to a local minimum, which is an expected result. NDT shows strong performance with fine precision localization and mostly performs better than ICP, but also struggles with inaccurate initialization. Our algorithm on the other hand is unaffected by the larger initialization error of prior 2, and has a relatively high 89.9% success rate on the most challenging prior 3. An apparent effect of the coarse-to-fine approach of our method is that when localization is successful, the accuracy is nearly the same for any of the used initialization errors. We note that the grid resolution of 20cm limits the fine accuracy compared to NDT, especially in rotation accuracy, but it remains competitive. The main advantage of FGR over ICP is that it has a larger percentage of successful localizations when initialization is less precise at 8 and 20m translation errors. However, FGR is hardly suited for localization with the used validation data, since it does not localize reliably in any setting. Likely, the repetitive structure in many scenes causes ambiguity in feature matching with the handcrafted descriptors of FGR. This demonstrates the difficulty in creating descriptors with good multi-scale matching performance.

A side-effect of using the 2-dimensional map representation is that our method is virtually agnostic to initialization errors in the vertical direction, whereas the baselines are equally affected by errors in the z-coordinate.

4.6. Flow field uncertainty measure

The effect of flow vector covariance for computation of pose from flow field is evaluated by comparing poses computed by using regressed covariance matrices with poses from covariance matrices set to identity, putting equal weight on each flow vector. On average over the validation set, the regressed covariance contributed to an improvement of translation error from 3.5 to 3.2 cm and heading error from $0.090°$ to $0.062°$. The rather modest improvements can be explained by the flow field regressor CNN which is trained with flow vector annotations that are computed from the correct rigid transformation, leading to a network that regresses flow vectors that are mostly coherent with a rigid transformation. This means that there are rarely any wildly incorrect flow vectors, and when errors occur they affect neighboring flow vectors.

Qualitatively, the error covariances clearly correlate with the regressed flow field errors as demonstrated in heatmaps of error and covariance trace respectively in Figure 4.

Prior error trans./rot.	2m / 3.5°				8m / 10°				20m / 20°			
	F2P	ICP	FGR	NDT	F2P	ICP	FGR	NDT	F2P	ICP	FGR	NDT
Median trans. err (m)	0.032	0.053	0.174	**0.019**	**0.033**	5.071	1.088	0.067	**0.036**	16.69	12.70	17.36
Mean trans. err (m)	0.035	0.076	2.496	**0.026**	**0.035**	5.605	5.133	2.527	**2.199**	16.57	13.60	17.17
Median rot. err (deg)	0.051	0.088	0.311	**0.023**	0.056	2.930	1.106	**0.045**	**0.059**	31.49	14.71	17.08
Mean rot. err (deg)	0.062	0.124	1.912	**0.031**	**0.068**	8.666	6.162	2.106	**0.482**	26.16	19.21	16.42
% <0.1m trans. err	**98.8**	77.4	31.0	98.0	**99.2**	2.4	16.5	58.1	**89.1**	0.0	4.8	3.6
% <0.3m trans. err	**100.0**	98.0	60.9	**100.0**	**100.0**	12.1	42.3	67.7	**89.9**	2.4	11.7	4.0
% <1.0m trans. err	**100.0**	**100.0**	69.0	**100.0**	**100.0**	29.8	50.0	69.0	**89.9**	14.9	19.8	4.0
% <0.1° rot. err	81.9	57.3	17.3	**96.4**	**75.8**	6.5	11.3	66.9	**66.9**	3.2	3.2	11.3
% <0.3° rot. err	99.2	91.1	49.6	**99.6**	**100.0**	15.7	32.7	75.0	**90.3**	8.5	8.1	16.9
% <1.0° rot. err	**100.0**	**100.0**	71.0	**100.0**	**100.0**	35.5	49.6	80.2	**91.9**	19.4	15.3	22.2
Time median (s)	**0.51**	0.96	0.71	2.70	**0.63**	3.35	0.92	5.76	**0.78**	4.93	2.59	10.58
Time mean (s)	**0.52**	1.01	0.79	2.76	**0.66**	4.10	1.07	8.62	**0.78**	5.67	2.81	17.52

Table 1: Translation error, rotation error and computation time performance for our method (F2P), point-to-point ICP, fast global registration (FGR) and NDT. Results are displayed for different initialization translation and heading angle errors.

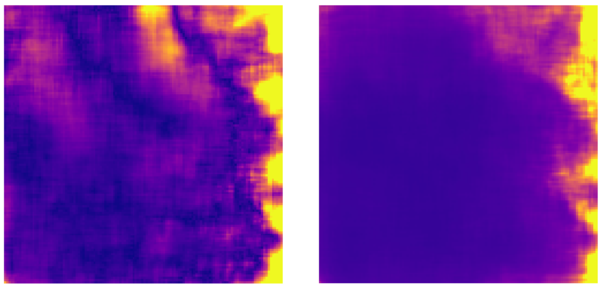
Figure 4: Left: The error of the predicted flow field. Right: The square root of the covariance trace at each flow vector.

4.7. Computational efficiency

We evaluate the validation data for baselines and our method on CPU to get comparable figures. Average computation times are listed in Table 1. Our method has a clearly bounded increase in computation time with increasing initialization errors. The increase stems from the use of coarse localization steps with more uncertain priors. Since the coarser localization networks use smaller dimension feature representations, they compute fast and add only a fraction to the total time. ICP in general needs more iterations when initialization error is large, due to the longer path to the convergence point, which results in computation time for the largest initialization error being five times the time for localizing with the finest error. For FGR the largest contribution comes from the computation of the feature descriptors, which is done only once. Therefore the computation time of FGR is mostly proportional to the point cloud size, which means that a more uncertain pose prior results in longer computation time due to the larger map crop used. Computation times of the used NDT implementation are around 5-15 times slower than our algorithm.

Overall, our method has the fastest computation times in all scenarios. The advantage increases with larger initial uncertainty. There is also a possibility to run the most demanding computations on a graphics processor which could further lower the computation times.

5. Conclusion

We present a robust and accurate localization method, built on learning-based optical flow field regression enhanced with a multi-scale coarse-to-fine refinement process. Evaluations show that localization accuracy is similar to normal distribution transform (NDT) and iterative closest point (ICP), which are commonly used to refine the pose found by other localization methods. At the same time, we obtain global localization performance that is unmatched by fast global registration, which shows that our learnt features, used in a multi-scale process, can find correct correspondences between sensor readings and map, even when the geometric scene structure lacks salient features. Furthermore, the neural network implementation is competitive in computation time, as a result of the efficient 2-dimensional point cloud representations.

Acknowledgements: This work was supported by
1) the project *Highly Automated Freight Transports* (AutoFreight) funded by Vinnova FFI [2016-05413],
2) the EU H2020 RICAIP (No. 857306) and EU Reg. Dev. Fund IMPACT (No.CZ.02.1.01/0.0/0.0/15 003/0000468) projects,
3) the Chalmers AI Research Centre (CHAIR),
4) the Swedish Foundation for Strategic Research (Semantic Mapping and Visual Navigation for Smart Robots, and
5) the Wallenberg Al, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation).

References

[1] Y. Aoki, H. Goforth, R. A. Srivatsan, and S. Lucey. Pointnetlk: Robust & efficient point cloud registration using pointnet. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7163–7172, 2019.

[2] X. Bai, Z. Luo, L. Zhou, H. Fu, L. Quan, and C.-L. Tai. D3feat: Joint learning of dense detection and description of 3d local features. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6359–6367, 2020.

[3] I. A. Barsan, S. Wang, A. Pokrovsky, and R. Urtasun. Learning to localize using a lidar intensity map. In *CoRL*, pages 605–616, 2018.

[4] P. J. Besl and N. D. McKay. Method for registration of 3-d shapes. In *Sensor fusion IV: control paradigms and data structures*, volume 1611, pages 586–606. International Society for Optics and Photonics, 1992.

[5] P. Biber and W. Straßer. The normal distributions transform: A new approach to laser scan matching. In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453)*, volume 3, pages 2743–2748. IEEE, 2003.

[6] C. Choy, J. Park, and V. Koltun. Fully convolutional geometric features. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 8958–8966, 2019.

[7] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017.

[8] R. Dubé, A. Cramariuc, D. Dugas, H. Sommer, M. Dymczyk, J. Nieto, R. Siegwart, and C. Cadena. Segmap: Segment-based mapping and localization using data-driven descriptors. *The International Journal of Robotics Research*, 39(2-3):339–355, 2020.

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

[10] Z. Gojcic, C. Zhou, J. D. Wegner, and A. Wieser. The perfect match: 3d point cloud matching with smoothed densities. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5545–5554, 2019.

[11] E. Ilg, O. Cicek, S. Galesso, A. Klein, O. Makansi, F. Hutter, and T. Brox. Uncertainty estimates and multi-hypotheses networks for optical flow. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 652–667, 2018.

[12] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2462–2470, 2017.

[13] D. P. Kingma and J. L. Ba. Adam: Amethod for stochastic optimization.

[14] A. Kurobe, Y. Sekikawa, K. Ishikawa, and H. Saito. Corsnet: 3d point cloud registration by deep neural network. *IEEE Robotics and Automation Letters*, 5(3):3960–3966, 2020.

[15] J. Levinson, M. Montemerlo, and S. Thrun. Map-based precision vehicle localization in urban environments. In *Robotics: science and systems*, volume 4, page 1. Citeseer, 2007.

[16] J. Levinson and S. Thrun. Robust vehicle localization in urban environments using probabilistic maps. In *2010 IEEE International Conference on Robotics and Automation*, pages 4372–4378. IEEE, 2010.

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

[18] W. Lu, G. Wan, Y. Zhou, X. Fu, P. Yuan, and S. Song. Deepvcp: An end-to-end deep neural network for point cloud registration. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 12–21, 2019.

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

[20] M. Magnusson, A. Lilienthal, and T. Duckett. Scan registration for autonomous mining vehicles using 3d-ndt. *Journal of Field Robotics*, 24(10):803–827, 2007.

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

[22] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[23] S. Rusinkiewicz and M. Levoy. Efficient variants of the icp algorithm. In *Proceedings third international conference on 3-D digital imaging and modeling*, pages 145–152. IEEE, 2001.

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

[25] R. B. Rusu and S. Cousins. 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011.

[26] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8934–8943, 2018.

[27] Q. M. Thomas, O. Wasenmüller, and D. Stricker. Delio: Decoupled lidar odometry. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 1549–1556. IEEE, 2019.

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

[29] Y. Wang and J. M. Solomon. Deep closest point: Learning representations for point cloud registration. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3523–3532, 2019.

[30] J. Yang, H. Li, D. Campbell, and Y. Jia. Go-icp: A globally optimal solution to 3d icp point-set registration. *IEEE transactions on pattern analysis and machine intelligence*, 38(11):2241–2254, 2015.

[31] Z. J. Yew and G. H. Lee. 3dfeat-net: Weakly supervised local 3d features for point cloud registration. In *European Conference on Computer Vision*, pages 630–646. Springer, 2018.

[32] Z. J. Yew and G. H. Lee. Rpm-net: Robust point matching using learned features. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11824–11833, 2020.

[33] Q.-Y. Zhou, J. Park, and V. Koltun. Fast global registration. In *European Conference on Computer Vision*, pages 766–782. Springer, 2016.

[34] Q.-Y. Zhou, J. Park, and V. Koltun. Open3D: A modern library for 3D data processing. *arXiv:1801.09847*, 2018.