DeepMapping: Unsupervised Map Estimation From Multiple Point Clouds

Li Ding*† l.ding@rochester.edu

Chen Feng*‡§ cfeng@nyu.edu

†University of Rochester ‡NYU Tandon School of Engineering §Mitsubishi Electric Research Laboratories (MERL)

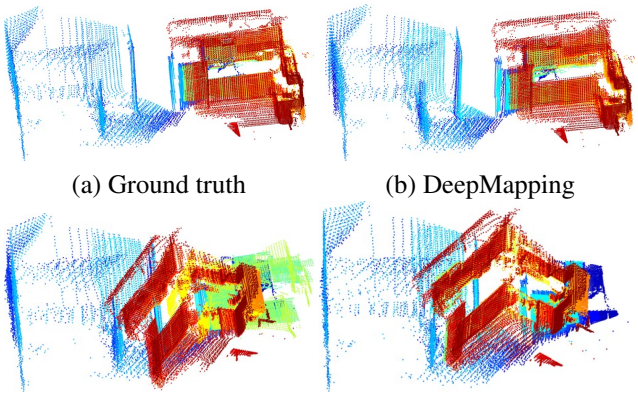
Abstract

We propose DeepMapping, a novel registration framework using deep neural networks (DNNs) as auxiliary functions to align multiple point clouds from scratch to a globally consistent frame. We use DNNs to model the highly non-convex mapping process that traditionally involves hand-crafted data association, sensor pose initialization, and global refinement. Our key novelty is that properly defining unsupervised losses to "train" these DNNs through back-propagation is equivalent to solving the underlying registration problem, yet enables fewer dependencies on good initialization as required by ICP. Our framework contains two DNNs: a localization network that estimates the poses for input point clouds, and a map network that models the scene structure by estimating the occupancy status of global coordinates. This allows us to convert the registration problem to a binary occupancy classification, which can be solved efficiently using gradient-based optimization. We further show that DeepMapping can be readily extended to address the problem of Lidar SLAM by imposing geometric constraints between consecutive point clouds. Experiments are conducted on both simulated and real datasets. Qualitative and quantitative comparisons demonstrate that DeepMapping often enables more robust and accurate global registration of multiple point clouds than existing techniques. Our code is available at http://ai4ce.github.io/DeepMapping/.

1. Introduction

Advances in deep learning have led to many state-of-the-art methods for semantic computer vision tasks. Despite those successes, their compelling improvements on the geometric aspects of computer vision are yet to be fully demonstrated (especially for registration and mapping). This is perhaps because, although being powerful, semantic representations have limitations in accurately estimating and modeling of geometric attributes of the environment. This includes, but is not limited to, estimating camera motions from a sequence of images, or registering multiple point clouds into a complete model. As opposed to semantics attributes of objects/scenes that are often categorical and thus easily described by human language, those geometric attributes are more often continuous and better described numerically, such as poses and shapes. These spatial properties and relations between objects play as vital roles as semantic ones for robotics, augmented reality, medical, and other engineering applications. Several works attempt to integrate deep learning methods into geometric vision problems [46, 53, 50, 22, 20, 31, 19]. Methods in [28, 9, 22] try to regress camera poses by training a DNN, inside which map of the environment is implicitly represented. Methods in [46, 53] propose unsupervised approaches that exploit inherent relationships between depth and motion. Despite different tasks, most approaches follow the same train-and-test pipeline that neural networks are first learned from a set of training data (either supervised or unsupervised), and then evaluated on a testing set, expecting those DNNs to be able to generalize as much as possible to untrained situations.

The essence of our discussion is an open question: Will DNNs generalize well for geometric problems especially for registration and mapping? Semantic tasks can benefit from DNNs because those related problems are defined empirically, and thus modeled and solved statistically. How-



(a) Ground truth (b) DeepMapping (c) Direct optimization (d) Multiway registration [11]

Figure 1. Global registration results from the AVD [4]. The baseline method in (c) is explained in Section 4.1 and 4.2. Each color represents one point cloud. Best viewed in color.

*This work was partially done while the authors were with MERL. And Chen Feng is the corresponding author.

ever many geometric problems are defined theoretically, and thus experiential solutions may not be adequate in terms of accuracy. Think of a simple scenario: given two images with overlapping field-of-views (FOV), without careful calculation, how accurate would a normal person be able to tell the Euclidean distance between the two camera centers? One may argue that with adequate feedback/training this can be done fairly accurately. But if this means it requires a large amount of data collection and training at each new location, the efficiency of this solution seems to be debatable.

Here we investigate another possibility of adopting powerful DNNs for the mapping/registration task. What we commonly agree from abundant empirical experiments and some theorems is that DNNs can model many arbitrarily complex mappings, and can be efficiently optimized through gradient-based methods, at least for categorical classification problems. This leads to our key idea in this paper: can we convert the complex and conventionally hand-engineered mapping/registration processes into DNNs, and solve them as if we are "training" them, although we do not necessarily expect the trained DNNs to generalize to other situations? To make this meaningful, unlike the supervised training in [28, 9, 22], we need to properly define unsupervised loss functions. Our exploration towards this line of thought shows promising results in our experiments. We summarize our contributions as follows:

- We propose DeepMapping to solve the point cloud mapping/registration problem as unsupervised end-to-end "training" of two DNNs, which is easier for parallel implementation compared to conventional methods requiring hand-crafted features and data associations.
- We convert this continuous regression problem to binary classification without sacrificing registration accuracy, using the DNNs and unsupervised losses.
- We demonstrate that DeepMapping has less dependency on good pose initialization compared with conventional baselines, through a large number of simulated and real-world experiments.

In the following, we first describe the existing registration/mapping method in Section 2, then provide the overview and details of our method in Section 3. Experiments and conclusions are in Section 4 and 5, respectively.

2. Related Work

Pairwise local registration: the methods for pairwise point cloud registration can be generally categorized into two groups: local vs. global methods. The local methods assume that a coarse initial alignment between two point clouds and iteratively update the transformation to refine the registration. The typical methods that fall into this category are the Iterative Closest Point (ICP) algorithms [6, 10, 35], probabilistic-based approaches [25, 33, 13] that model the point clouds as a probability distribution. The local methods are well-known for requiring a "warm start", or a good initialization, due to limited convergence range.

Pairwise global registration: the global methods [47, 3, 32, 51, 30, 15] do not rely on the "warm start" and can be performed on point clouds with arbitrary initial poses. Most global methods extract feature descriptors from two point clouds. These descriptor are used to establish 3D-to-3D correspondences for relative pose estimation. Robust estimations, e.g., RANSAC [18], are typically applied to handle the mismatches. The feature descriptors are either hand-crafted such as FPFH [36], SHOT [44], 3D-SIFT [39], NARF [41], PFH [37], spin images [27], or learning-based such as 3DMatch [49], PPFNet [14], and 3DFeatNet [26].

Multiple registration: in addition to pairwise registration, several methods have been proposed for multiple point clouds registration [43, 17, 24, 45, 11]. One approach is to incrementally add new a point cloud to the model registered from all previous ones. The drawback of the incremental registration is the accumulated registration error. This drift can be mitigated by minimizing a global cost function over a graph of all sensor poses [11, 43].

Deep learning approaches: recent works explore the idea of integrating learning-based approaches into mapping and localization problems. Methods in [46, 53] propose unsupervised approaches that exploit inherent relationships between depth and motion. This idea is further explored in [7, 50, 48, 31] using deep learning for visual odometry and SLAM problems. For instance, CodeSLAM [7] represents the dense geometry using a variational auto-encoder (VAE) for depth that is conditioned on the corresponding intensity image, which is later optimized during bundle adjustment. Differently, DeepMapping does not require any pre-training. [19] introduces a generative temporal model with memory system that allows the agent to memorize the scene representation and to predict its pose in a partially observed environment. Although this method and DeepMapping are both able to determine the sensor pose from the observed data, [19] requires a supervised training stage for "loading" its memory, while DeepMapping is fully unsupervised and does not follow the aforementioned train-and-test pipeline. MapNet [22] use a recurrent neural network to model the environment through a sequence of RGB-D images also in a supervised setting. The localization of camera sensor is performed using deep template matching on the discretized spatial domain that has a relatively small resolution. Unlike MapNet, besides no supervision, the proposed DeepMapping does not require any partition of the space.

Other related methods such as [28, 9] solve camera localization by training DNNs to regress camera poses and test the performance in the same environment as the training images. Related to that is DSAC [8] as a differentiable alternative to traditional RANSAC for its use in pose esti-

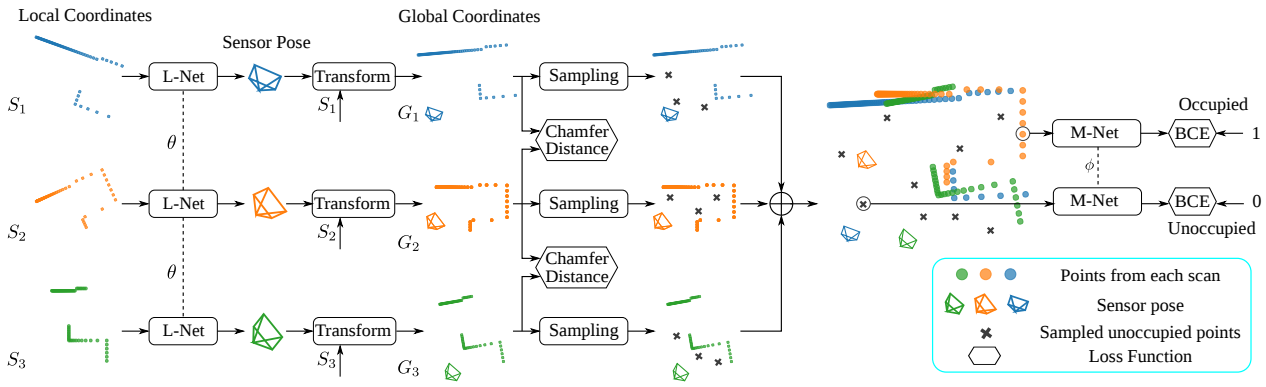


Figure 2. **DeepMapping pipeline**. Point clouds appear in different colors. Each input point cloud is fed into the shared L-Net to compute transformation parameters that map the input to the global coordinates where both occupied (colored solid circles) and unoccupied (gray cross marks) locations are sampled. The M-Net predicts the occupancy probabilities of the sampled locations. The global occupancy loss is the binary cross entropy (BCE) averaged over all sampled locations. DeepMapping is able to handle temporal information, if available, by integrating Chamfer distance loss between consecutive scan. Best viewed in color.

mation DNNs. For place recognition, semantic scene completion is used in [38] as an auxiliary task for training an image VAE for long-term robustness. The method in [20] proposes an unsupervised approach with variational Bayesian convolutional auto-encoder to model structures from point clouds. In DeepMapping, we adopt this idea to model the scene structure but use DNN rather than Bayesian inference. Other prior works include: in [16] the generative query network (GQN) shows the ability to represent the scene from a given viewpoint and rendering it from an unobserved viewpoint in the simple synthetic environments. A neuroscience study [5] uses recurrent neural networks to predict mammalian spatial behavior.

As noted, most approaches follow a train-and-test pipeline. Our approach adopts DNNs but differs from the existing methods in the way that the process of "training" in DeepMapping is equivalent to solving the point clouds registration and that once trained, we do not expect the DNNs to generalize to other situations.

3. Method

3.1. Overview

In this section, we describe the proposed DeepMapping that uses DNNs for registering multiple point clouds. Let $\mathbf{S} = \{S_i\}_{i=1}^K$ be the set of K input point clouds in D-dimensional space that are captured by 3D scanners, and the i^{th} point cloud S_i contains N_i points in sensor local frame. Given K point clouds, the goal is to register all point clouds in a common coordinate frame by estimating the sensor poses $\mathbf{T} = \{T_i\}_{i=1}^K$ for each point cloud S_i.

Conventional methods [51, 17] formulate this as an optimization problem that directly seeks the optimal sensor poses $\mathbf{T}$ to minimize the loss function

$$\mathbf{T}^\star(\mathbf{S}) = \arg\min_{\mathbf{T}} \mathcal{L}(\mathbf{T}, \mathbf{S}), \tag{1}$$

where $\mathcal{L}(\mathbf{T}, \mathbf{S})$ is the objective that scores the registration quality. As explained in Section 1, instead of directly optimizing $\mathbf{T}$, we propose to use a neural network, modeled as an auxiliary function $f_\theta(\mathbf{S})$, to estimate $\mathbf{T}$ for the input point clouds $\mathbf{S}$ where θ are the auxiliary variables to be optimized. The pose is then converted to a transformation matrix that maps S_i into its global version G_i.

Formally, we re-formulate this registration problem as finding the optimal network parameters that minimize a new objective function

$$(\theta^\star, \phi^\star) = \arg\min_{\theta,\phi} \mathcal{L}_\phi(f_\theta(\mathbf{S}), \mathbf{S}), \tag{2}$$

where $\mathcal{L}_\phi$ is a learnable objective function which will be explained in Section 3.2 and 3.4.

Figure 2 illustrates the pipeline for DeepMapping. At the heart of DeepMapping are two networks, a localization network (L-Net) and an occupancy map network (M-Net), that estimates T_i and measures the registration quality, respectively. The L-Net is the function $f_\theta : S_i \to T_i$ appeared in Equation 2 that estimates the sensor pose of a corresponding point cloud, where the parameters θ in the L-Net are shared among all point clouds. The point cloud G_i in global coordinates is obtained using the estimated sensor pose. From the transformed point clouds, we first sample both occupied and unoccupied locations. Then the locations of these samples are fed into the M-Net, to evaluate the registration performance of the L-Net. The M-Net is a binary classification network that predicts probabilities of input locations being occupied. We denote M-Net as a function m_ϕ with learnable parameters ϕ. The occupancy probabilities are used for computing the objective function $\mathcal{L}$ in Equation 2 that measures the global occupancy consistency of the transformed point clouds and thus reflects the registration quality.

One may find that moving from Equation 1 to 2 may increase the dimensionality/complexity of the problem. In

fact, we provide a simple 1D version of this problem conversion and show that using DNNs as auxiliary functions and optimizing them in higher dimensional space using gradient-based methods could enable faster and better convergence than directly optimizing the original problem. Details are included in the supplementary material.

Next, we describe the global consistency measure, network architecture for L-Net and M-Net, and the modification that allows DeepMapping to handle Lidar SLAM.

3.2. Global Consistency Measure

We propose to measure the registration quality as the global consistency of a set of point clouds using occupancy deviations among all points in the global frame. For illustration purpose only, we show this measure using a 1D occupancy grid map in Figure 3 that partitions an area into regular grid cells, but note that in fact we do not perform any global space discretization, and the global consistency is measured on a continuous 3D space using floating number coordinates from each transformed point cloud directly. In Figure 3, $\mathbf{x}_1$, $\mathbf{x}_2$, and $\mathbf{x}_3$ are three global point locations, based on which the registration quality between two point clouds G_1 and G_2 are measured. We start by feeding $\mathbf{x}_1$, $\mathbf{x}_2$, and $\mathbf{x}_3$ into the M-Net to estimate the corresponding occupancy probabilities p_1, p_2, and p_3, respectively. Each point cloud assigns a binary occupancy value to each cell indicating whether the cell is occupied by that point cloud. The occupancy status of a cell is marked as 1 if at least one point exists within the cell and 0 otherwise. The binary cross entropy B between the predicted occupancy p_i and its ground truth label y_i from each point cloud is:

$$B\left[p_i, y_i\right] = -y_i \log\left(p_i\right) - \left(1 - y_i\right) \log\left(1 - p_i\right). \quad (3)$$

Thus, the occupancy deviation is modeled as an average of binary cross entropy loss over all locations in all point clouds. Intuitively, the occupancy deviations can easily achieve small values if the accurate registration is achieved, as shown in Figure 3 (a). Conversely, if the occupancy values for a cell vary among point clouds, the corresponding average binary cross entropy will be relatively large due to difficulties of the M-Net to model "random" occupancy pattern, usually indicating misalignment among point clouds. For example, in Figure 3 (b), G_1 marks the first cell as occupied whereas G_2 assigned an unoccupied label to the same cell. This results in a large average error computed from binary cross entropy.

Since we focus on point cloud mapping, instead of using the conventional occupancy grid that discretizes the global space, we directly use the floating number coordinates of points to measure the global consistency, leading to a continuous occupancy map function $m_\phi : R^D \rightarrow [0, 1]$, $D = 2$ or 3 for 2D or 3D mapping. This modification allows computing occupancy deviations at an arbitrary scale

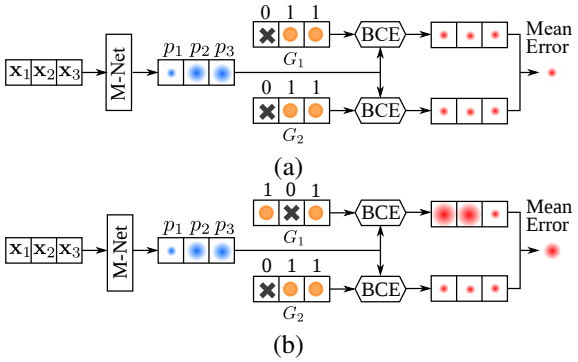


Figure 3. Global consistency measurement using occupancy deviations. (a) shows a co-aligned pair of point clouds and (b) shows a misaligned pair. Blue and red circles represent the occupancy probability and BCE error, respectively. The radius of these two circles is used to indicate the corresponding data value. Best viewed in color.

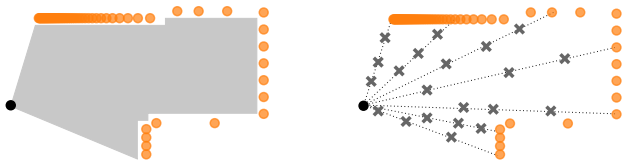


Figure 4. Sampling method. (a) Orange circles are the point cloud captured by the sensor in black. The gray region indicates the free space between the sensor and the observed point cloud. (b) Gray marks represent the sampled unoccupied points that are taken from the line segments between the sensor and the observed points.

and resolution. In this scenario, the locations of all observed/scanned points have the associated label 1 indicating occupied, whereas unoccupied points with labels 0 can be sampled from the line-of-sight. Figure 4 depicts the sampling mechanism for a Lidar scanner. In Figure 4 (a), the gray region indicates the free space where the laser beams traverse and orange points are the captured point cloud where the laser beams hit the object. We denote $s\left(G_i\right)$ as a set of sampled points from G_i that lie on the laser beam emitted from the scanner, illustrated as gray cross marks in Figure 4 (b). These sampled points are associated with label 0 indicating unoccupied locations.

By combining the occupancy deviations and the sampling function, the loss used in Equation 2 is defined as

$$\mathcal{L}_{cls} = \frac{1}{K} \sum_{i=1}^{K} B\left[m_\phi\left(G_i\right), 1\right] + B\left[m_\phi\left(s\left(G_i\right)\right), 0\right], \quad (4)$$

where G_i is a function of L-Net parameters θ, and with an slight abuse of notation $B\left[m_\phi\left(G_i\right), 1\right]$ here means the average BCE error for all points in point cloud G_i, and $B\left[m_\phi\left(s\left(G_i\right)\right), 0\right]$ means the average BCE error for correspondingly sampled unoccupied locations.

For minimizing Equation 4, we adopt the gradient-based optimization because the loss function $\mathcal{L}_{cls}$ is differentiable

for both θ and ϕ. In each forward pass, the input point clouds $\mathbf{S}$ are processed with the L-Net and transformed into the global space. Both occupied and unoccupied locations are randomly sampled and the error is computed accordingly. During the backward pass, we back-propagate the error of occupancy deviations and update the network parameters θ and ϕ.

3.3. Network Architecture

L-Net: the goal of the localization network, L-Net, is to estimate the sensor pose T_i in a global frame. The L-Net consists of a latent feature extraction module followed by multi-layer perceptron (MLP) that outputs sensor pose T_i. This module depends on the format of the input point cloud S_i. If S_i is an organized point cloud that is generated from range images such as disparity maps or depth images, then S_i is organized as an array of points in which the spatial relationship between adjacent points is preserved. Therefore, we apply a convolutional neural network (CNN) to extract the feature vector of point cloud followed by a global pooling layer to aggregate local features to a global one. In the case where the inputs are a set of unorganized point clouds without any spatial order, we adopt PointNet [34] architecture for extracting features from point cloud. We remove the input and feature transformations appeared in [34] and use a shared MLP that maps each D-dimensional point coordinates to a high dimensional feature space. A global pooling layer is applied across all points for feature aggregation.

The extracted latent feature vector is processed with a 3-layer MLP with the output channels corresponding to the degree of freedom (DOF) of sensor movement. The convolution and MLP layers include ReLU activation function except for the last layer in the feature extraction module and the final output layer.

M-Net: the occupancy map network, M-Net, is a binary classification network that takes as input the location coordinates in the global space and predicts the corresponding occupancy probability for each input location. The M-Net is a shared MLP with input channel D and output channel 1. Each MLP layer is followed by a ReLU activation function except for the output layer that includes a sigmoid function to output occupancy probability.

3.4. Extension to Lidar SLAM

Equation 4 treats the input point clouds $\mathbf{S}$ as an unordered set of scans instead of a temporally ordered sequence. In some applications, the temporal information may be available. For example, Lidar SLAM uses laser scanners to explore the unknown environment and captures an ordered sequence of point clouds at different time t.

Now we extend DeepMapping to exploit such a temporal relationship. The underlying assumption is that the consecutive scans of point clouds are expected to have a reasonable overlapping with each other, which normally holds in the SLAM settings [42, 40]. We utilize the geometric constraints among point clouds that are temporally close to each other. Specifically, we adopt the Chamfer distance as the metric that measures the distance between two point clouds G_i and G_j in the global coordinates

$$\begin{aligned} d\left(G_i, G_j\right) = & \frac{1}{N_i} \sum_{\mathbf{x}_i \in G_i} \min_{\mathbf{x}_j \in G_j} \|\mathbf{x}_j - \mathbf{x}_i\|_2 \\ & + \frac{1}{N_j} \sum_{\mathbf{x}_j \in G_j} \min_{\mathbf{x}_i \in G_i} \|\mathbf{x}_i - \mathbf{x}_j\|_2. \end{aligned} \tag{5}$$

The Chamfer distance measures the average distance between each point in one point cloud to its nearest point in the other. The minimization of the Chamfer distance $d\left(G_i, G_j\right)$ results in a pairwise alignment between point clouds G_i and G_j. To integrate the Chamfer distance into DeepMapping, we modify the objective function in 2 as

$$(\theta^\star, \phi^\star) = \arg\min_{\theta,\phi} \mathcal{L}_{cls} + \lambda \mathcal{L}_{ch}, \tag{6}$$

where λ is a free parameter to control the relation of two loss and $\mathcal{L}_{ch}$ is defined as the average Chamfer distance between each point cloud G_i and its temporal neighbors G_j

$$\mathcal{L}_{ch} = \sum_{i=1}^{K} \sum_{j \in \mathcal{N}(i)} d\left(G_i, G_j\right). \tag{7}$$

4. Experiments

We verify the utility of DeepMapping on two datasets: a simulated 2D Lidar point cloud dataset and a real 3D dataset called Active Vision Dataset (AVD) [4]. We implemented DeepMapping with PyTorch [1]. The network parameters are optimized using Adam optimizer [29] with a learning rate of 0.001 on a NVidia K80.

For quantitative comparison, we use two metrics: the absolute trajectory error (ATE) and the point distance between the ground truth point cloud and the registered one. The ATE assesses the global consistency of the estimated trajectory, and the point distance measures the geometric distance between the corresponding points from the ground truth and the registered point cloud. Since the estimated position can be defined in an arbitrary coordinates, a rigid transformation is determined to map the points in estimated coordinates to the ground truth coordinates using the closed-form solution proposed in [23]. Once aligned in the same coordinates,the ATE and the point distance are then defined as the root-mean-square error in sensor positions and in point locations, respectively.

4.1. Experiments on 2D Simulated Point Cloud

Dataset: to simulate the 2D Lidar point clouds captured by a virtual Lidar scanner, we create a large environment represented by a 1024×1024 binary image, as shown in Figure 5. The white pixels indicate the free space whereas the

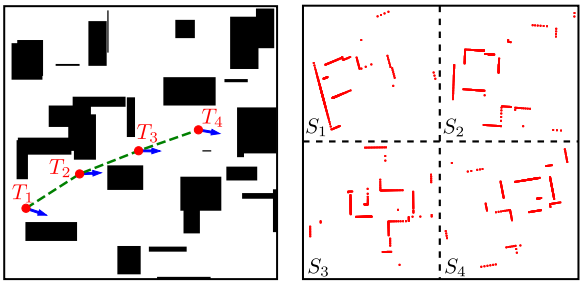


Figure 5. Illustration of 4 scans captured at time $20, 70, 120, 170$. Left: the binary image shows the simulated environment with size of 1024×1024. The sensor poses are shown in red circles and blue arrows. The green dash line indicates the trajectory of the sensor. Right: observed point clouds captured at corresponding poses.

black ones correspond to obstacles. We assume a moving robot equipped with a horizontal Lidar scanner to explore this environment. We first sample the trajectory of robot movement that consists of a sequence of poses. At each pose in the trajectory, the Lidar scanner spreads laser beam over the FOV and the laser beam is reflected back whenever it hit the obstacles within its path. Each observed point is computed as the intersection point between the laser ray and boundary of obstacles. The scanning procedure yields a set of 2D points in the robot's local coordinates.

In this experiment, we set the FOV of the Lidar scanner to $360°$ and assume an ideal scanner with an unlimited sensing range. If no object exists along the laser ray, we return the point where the ray hits the image boundary. We create 3 binary maps of virtual environment and sample in total 75 trajectories. 50 trajectories have 128 poses and the remaining 25 trajectories contain 256 poses where point clouds are scanned. Each scan contains 256 points. The transformation T_i is parameterized by a 2D translation vector (t_x, t_y) and a rotation angle α.

Baseline: we compare DeepMapping with the baseline incremental iterative closest point (ICP) with point-to-point distance metrics [6] and point-to-plane distance metrics [10]. To justify the advantage of the neural network based optimization, we also test another baseline method, referred to as the direct optimization. Specifically, we remove the L-Net f_θ and perform the optimization of the same loss function with respect to the sensor poses $\mathbf{T}$ rather than network parameters θ.

Implementation: the detailed architecture of the L-Net consists of C(2)-C(64)-C(128)-C(1024)-M(1024)-FC(512)-FC(256)-FC(3), where C(n) denotes 1D atrous convolutions with kernel size 3 and dilation rate of 2, M(n) denotes 1D max-pooling layer over n points, FC(n) denotes fully-connected layer with n output. The M-Net can be described as FC(2)-FC(64)-FC(512)-FC(512)-FC(256)-FC(128)-FC(1). We do not optimize the network architectures. More ablation studies are included in supplementary material. To sample unoccupied points, we randomly select 19 points on each laser ray starting from the scanner. For the baselines, we run the pairwise ICP algorithm on every two consecutive point clouds. The final transformation T_i is obtained by concatenating all prior transformations. To compare DeepMapping with the direct optimization, we use the same optimizer and the learning rate. To ensure the same initialization between DeepMapping and the direct optimization, we run DeepMapping with only on forward pass (no back-propagation is performed) and save the output sensor poses. These poses are used as the starting point for the direct optimization. The free parameter λ is set to 10.

Results: Figure 6 shows the qualitative comparison of 4 trajectories simulated from the dataset. Note that while these trajectories are sampled from the same binary image (environment), the registrations are processed independently, i.e., we "train" the network for each trajectory and do expect the generalization ability if it is tested on other trajectories. As shown, the baseline method of direction optimization on sensor pose fails to find the transformations leading to globally inaccurate registration. While the incremental ICP algorithms are able to procedure a noisy registration, the error of the increment ICP algorithms accumulated over frames. As opposed to the baselines, the proposed DeepMapping accurately aligns multiple point clouds by utilizing neural networks (the L-Net) for the optimization. Once the registration is converged, we feed the all points (both occupied and unoccupied) in the global coordinates into the M-Net to estimated the occupancy probability, which is then converted to an image of occupancy map shown in the third column in Figure 6. The unexplored regions are shown in gray. The estimated occupancy map agrees with registered point clouds.

The box plots in Figure 7 show the quantitative results of the ATE and the point distance. Note that the data is plotted with the logarithmic scale for the y-axis. DeepMapping has the best performance in both metrics, achieving a median ATE of 5.6828 pixels and a median point distance of 6.5029, which significantly outperforms the baseline methods.

4.2. Experiments on the Active Vision Dataset

Dataset: DeepMapping is also tested on a real 3D dataset: Active Vision Dataset [4] that provides RGB-D images for a variety of scans of indoor environment. Unlike other RGB-D datasets [42, 40] that capture RGB-D videos around the scene, the AVD uses a robotic platform to visit a set of discrete points on a rectangular grid with a fixed width of 300mm. At each point, 12 views of images are captured by rotating the camera every $30°$. This data collection procedure leads to a low frame rate and relatively small overlapping between two images.

The ground truth camera intrinsic and extrinsic parame-

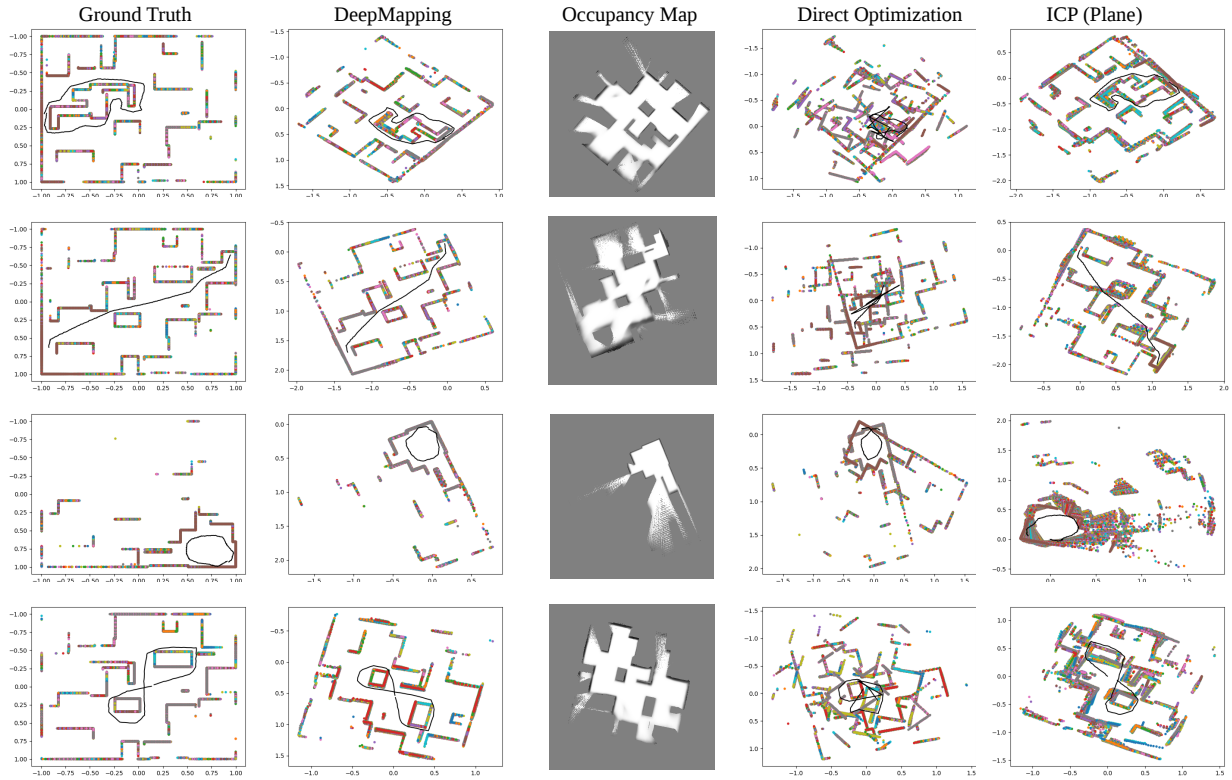
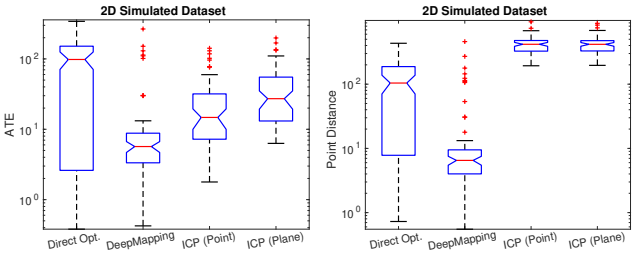


Figure 6. Qualitative results from the 2D simulated dataset. The black lines are the trajectories of sensor. The third column show the images of occupancy maps for each trajectory that are estimated by the M-Net. The black, while, and gray pixels show the occupied, unoccupied, and unexplored locations. Note that the results of each trajectory cam be defined in arbitrary coordinate systems and do not necessarily aligned with ground truth. More visualization results including failure cases are included in supplementary material. Best viewed in color.



(a) (b)

Figure 7. Box plots of the ATE and the point distance on the 2D simulated dataset. In each box, the red line indicates the median. The top and bottom blue edges of the box show the first (25th percentile) and the third (75th percentile) quartiles, respectively. Note the logarithmic scale for the y-axis.

ters are provided. We use the camera intrinsic to covert the depth map to point cloud representation and do not use any information provided by color images. We randomly move or rotate the camera sensor in the space and collect 105 trajectories from the dataset. Each trajectory contains either 8 or 16 point clouds.

Baseline: we compare DeepMapping with the baseline method of multiway registration [11] The ICP algorithms perform poorly because of the low overlapping rate between consecutive point clouds and thus are not included in this section. The multiway registration aligns multiple point clouds via pose graph optimization where each node represents an input point cloud and the graph edges connect two point clouds. We follow the same procedure in Section 4.1 to test the performance of direct optimization with respect to sensor pose on the AVD. For DeepMapping and the direction optimization, we compare two loss functions: only the BCE loss $\mathcal{L}_{bce}$ (by setting λ to 0), and the combination of the BCE loss $\mathcal{L}_{bce}$ and the Chamfer distance $\mathcal{L}_{ch}$. For the latter one, we set λ to 0.1 because of the small overlapping between two scans.

Implementation: the L-Net architecture consists of C(3)-C(64)-C(128)-C(256)-C(1024)-AM(1)-FC(512)-FC(256)-FC(3), where C(n) denotes 2D atrous convolutions with kernel size 3 and dilation rate of 2 and AM(1) denotes 2D adaptive max-pooling layer. The M-Net has the same structure as that used in Section 4.1, except for the input layer that has 3 nodes rather than 2. We sample 35 points on each laser ray. The multiway registration [11] is implemented using Open3D library [52].

Results: The visual comparison of multiple point clouds

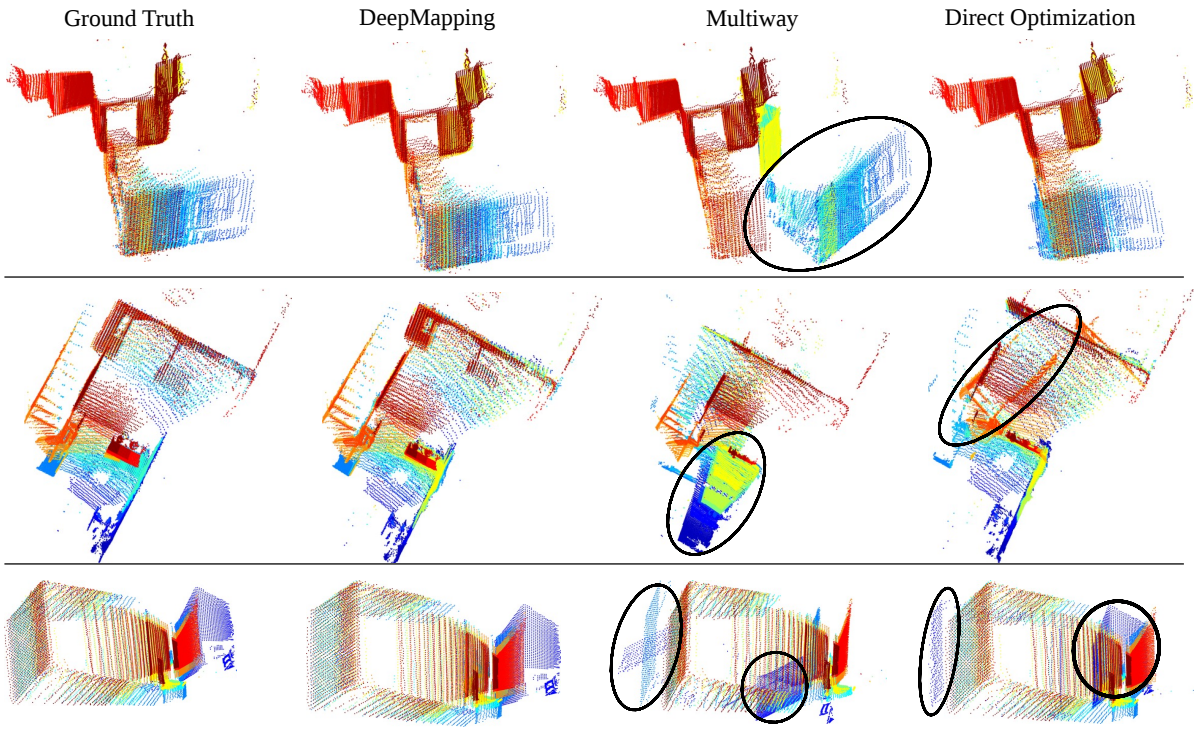
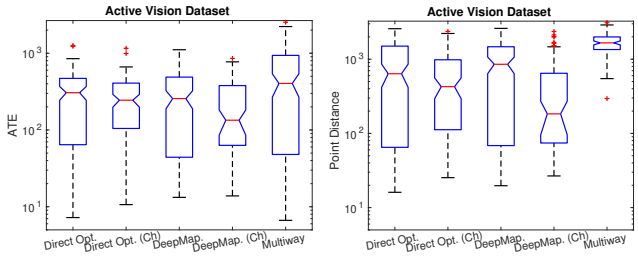


Figure 8. Qualitative results of multiple point clouds registration tested on the AVD [4] dataset. The black ellipses highlight the misaligned parts in baselines. Each color represents one point cloud. More visualizations are shown in supplementary material. Best viewed in color.

registrations for different algorithms is shown in Figure 8. The misaligned point clouds are highlighted with black ellipses. The stairs from the direction optimization, for example, are not correctly registered, as shown in the second row in Figure 8. The multiway registration [11] and the direction optimization fails to aligned several planar structures that are shown in the last row in Fig 8.

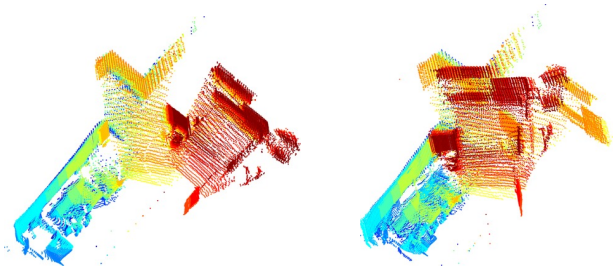
We show the box plots of the ATE and the point distance in Figure 9. The multiway registration [11] performs poorly that has the large error in both metrics. Comparing the direction optimization with DeepMapping, we show the advantage of using neural networks for optimization. In addition, integrating Chamfer distance into DeepMapping is also helpful to improve the registration accuracy.



(a) (b)

Figure 9. Box plots of the ATE and the point distance on the AVD [4]. The methods without use the BCE only and the ones with "Ch" combine the Chamfer distance. Legends are the same as those in Figure 6. Note the logarithmic scale for the y-axis.

In Figure 10, we show a sample failure case in which DeepMapping does not align all input point clouds properly. While the red frame is misaligned, the remaining point clouds are still registered. The misalignment is due to the convergence of network parameters into a local minima.



(a) Ground truth (b) DeepMapping

Figure 10. Sample failure registration from the AVD [4]. The red frame is misaligned due the convergence to a local minima.

5. Conclusion

In this paper, we propose DeepMapping to explore a possible direction for integrating deep learning methods into multiple point clouds registration that could also be informative for other related geometric vision problems. The novelty of our approach lies in the formulation that converts solving the registration problem into "training" some DNNs using properly defined unsupervised loss functions, with promising experimental performances.

Acknowledgment

The authors gratefully acknowledge the helpful comments and suggestions from Yuichi Taguchi, Dong Tian, Weiyang Liu, and Alan Sullivan.

References

[1] PyTorch. https://pytorch.org/. 5

[2] A. Agrawal, R. Verschueren, S. Diamond, and S. Boyd. A rewriting system for convex optimization problems. *J. Control and Decision*, 5(1):42–60, 2018. 11

[3] D. Aiger, N. J. Mitra, and D. Cohen-Or. 4-points congruent sets for robust pairwise surface registration. In *ACM Trans. Graphics*, volume 27, page 85, 2008. 2

[4] P. Ammirato, P. Poirson, E. Park, J. Kosecka, and A. C. Berg. A dataset for developing and benchmarking active vision. In *Proc. the IEEE Intl. Conf. on Robotics and Auto.*, 2017. 1, 5, 6, 8, 11, 12, 13, 15

[5] A. Banino, C. Barry, B. Uria, C. Blundell, T. Lillicrap, P. Mirowski, A. Pritzel, M. J. Chadwick, T. Degris, J. Modayil, et al. Vector-based navigation using grid-like representations in artificial agents. *Nature*, 557(7705):429, 2018. 3

[6] P. J. Besl and N. D. McKay. A method for registration of 3-D shapes. *IEEE Trans. Pattern Anal. Mach. Intel.*, 14(2):239–256, 1992. 2, 6

[7] M. Bloesch, J. Czarnowski, R. Clark, S. Leutenegger, and A. J. Davison. CodeSLAM - learning a compact, optimisable representation for dense visual SLAM. In *IEEE Intl. Conf. Comp. Vision, and Pattern Recog.*, June 2018. 2

[8] E. Brachmann, A. Krull, S. Nowozin, J. Shotton, F. Michel, S. Gumhold, and C. Rother. DSAC - differentiable ransac for camera localization. In *IEEE Intl. Conf. Comp. Vision, and Pattern Recog.*, July 2017. 3

[9] S. Brahmbhatt, J. Gu, K. Kim, J. Hays, and J. Kautz. Geometry-aware learning of maps for camera localization. In *IEEE Intl. Conf. Comp. Vision, and Pattern Recog.*, June 2018. 1, 2

[10] Y. Chen and G. Medioni. Object modelling by registration of multiple range images. *Image and Vision Computing*, 10(3):145–155, 1992. 2, 6

[11] S. Choi, Q. Zhou, and V. Koltun. Robust reconstruction of indoor scenes. In *IEEE Intl. Conf. Comp. Vision, and Pattern Recog.*, June 2015. 1, 2, 7, 8, 11, 13

[12] D.-A. Clevert, T. Unterthiner, and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (ELUs). In *Intl. Conf. Learning Representations*, 2015. 11

[13] M. Danelljan, G. Meneghetti, F. Shahbaz Khan, and M. Felsberg. A probabilistic framework for color-based point set registration. In *IEEE Intl. Conf. Comp. Vision, and Pattern Recog.*, pages 1818–1826, 2016. 2

[14] H. Deng, T. Birdal, and S. Ilic. PPFNet: Global context aware local features for robust 3D point matching. In *IEEE Intl. Conf. Comp. Vision, and Pattern Recog.*, June 2018. 2

[15] G. Elbaz, T. Avraham, and A. Fischer. 3D point cloud registration for localization using a deep neural network auto-encoder. In *IEEE Intl. Conf. Comp. Vision, and Pattern Recog.*, pages 2472–2481. IEEE, 2017. 2

[16] S. A. Eslami, D. J. Rezende, F. Besse, F. Viola, A. S. Morcos, M. Garnelo, A. Ruderman, A. A. Rusu, I. Danihelka, K. Gregor, et al. Neural scene representation and rendering. *Science*, 360(6394):1204–1210, 2018. 3

[17] G. D. Evangelidis, D. Kounades-Bastian, R. Horaud, and E. Z. Psarakis. A generative model for the joint registration of multiple point sets. In *Euro. Conf. on Comp. Vision*, pages 109–122. Springer, 2014. 2, 3

[18] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, 1981. 2

[19] M. Fraccaro, D. J. Rezende, Y. Zwols, A. Pritzel, S. Eslami, and F. Viola. Generative temporal models with spatial memory for partially observed environments. In *Intl. Conf. on Mach. Learning*, 2018. 1, 2

[20] V. Guizilini and F. Ramos. Learning to reconstruct 3D structures for occupancy mapping from depth and color information. *Intl. J. of Robotics Research*, 2018. 1, 3

[21] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE Intl. Conf. Comp. Vision, and Pattern Recog.*, June 2016. 12

[22] J. F. Henriques and A. Vedaldi. MapNet: An allocentric spatial memory for mapping environments. *IEEE Intl. Conf. Comp. Vision, and Pattern Recog.*, 2018. 1, 2

[23] B. K. Horn. Closed-form solution of absolute orientation using unit quaternions. *J. Opt. Soc. Am. A*, 4(4):629–642, 1987. 5

[24] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, et al. Kinectfusion: real-time 3D reconstruction and interaction using a moving depth camera. In *ACM Symp. User Interface Software and Technology*, pages 559–568, 2011. 2

[25] B. Jian and B. C. Vemuri. A robust algorithm for point set registration using mixture of gaussians. In *IEEE Intl. Conf. Comp. Vision*, volume 2, pages 1246–1251, 2005. 2

[26] Z. Jian Yew and G. Hee Lee. 3DFeat-Net: Weakly supervised local 3D features for point cloud registration. In *Euro. Conf. on Comp. Vision*, September 2018. 2

[27] A. E. Johnson and M. Hebert. Using spin images for efficient object recognition in cluttered 3D scenes. *IEEE Trans. Pattern Anal. Mach. Intel.*, (5):433–449, 1999. 2

[28] A. Kendall, M. Grimes, and R. Cipolla. PoseNet: A convolutional network for real-time 6-DOF camera relocalization. In *IEEE Intl. Conf. Comp. Vision*, December 2015. 1, 2

[29] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Intl. Conf. Learning Representations*, 2015. 5

[30] H. Lei, G. Jiang, and L. Quan. Fast descriptors and correspondence propagation for robust global point cloud registration. *IEEE Trans. Image Proc.*, 26(8):3614–3623, 2017. 2

[31] J. Li, H. Zhan, B. M. Chen, I. Reid, and G. H. Lee. Deep learning for 2D scan matching and loop closure. In *IEEE Intl. Conf. Intel. Robots and Sys.*, pages 763–768, Sept 2017. 1, 2

[32] N. Mellado, D. Aiger, and N. J. Mitra. Super 4PCS fast global pointcloud registration via smart indexing. In *Comp. Graphics Forum*, volume 33, pages 205–215, 2014. 2

[33] A. Myronenko and X. Song. Point set registration: Coherent point drift. *IEEE Trans. Pattern Anal. Mach. Intel.*, 32(12):2262–2275, 2010. 2

[34] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. PointNet: Deep learning on point sets for 3D classification and segmentation. In *IEEE Intl. Conf. Comp. Vision, and Pattern Recog.*, July 2017. 5, 11

[35] S. Rusinkiewicz and M. Levoy. Efficient variants of the ICP algorithm. In *3D Digital Imaging and Modeling*, pages 145–152, 2001. 2

[36] R. B. Rusu, N. Blodow, and M. Beetz. Fast point feature histograms (FPFH) for 3D registration. In *Proc. the IEEE Intl. Conf. on Robotics and Auto.*, pages 3212–3217, 2009. 2

[37] R. B. Rusu, N. Blodow, Z. C. Marton, and M. Beetz. Aligning point cloud views using persistent feature histograms. In *IEEE Intl. Conf. Intel. Robots and Sys.*, pages 3384–3391, 2008. 2

[38] J. L. Schönberger, M. Pollefeys, A. Geiger, and T. Sattler. Semantic visual localization. *ISPRS J. Photographic and Remote Sensing*, 2018. 3

[39] P. Scovanner, S. Ali, and M. Shah. A 3-dimensional SIFT descriptor and its application to action recognition. In *ACM Intl. Conf. Multimedia*, pages 357–360, 2007. 2

[40] S. Song, S. P. Lichtenberg, and J. Xiao. Sun RGB-D: A RGB-D scene understanding benchmark suite. In *IEEE Intl. Conf. Comp. Vision, and Pattern Recog.*, pages 567–576, 2015. 5, 6

[41] B. Steder, R. B. Rusu, K. Konolige, and W. Burgard. NARF: 3D range image features for object recognition. In *Wksp. on Defining and Solving Realistic Perception Problems in Personal Robotics at IEEE Intl. Conf. Intel. Robots and Sys.*, volume 44, 2010. 2

[42] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of RGB-D SLAM systems. In *IEEE Intl. Conf. Intel. Robots and Sys.*, Oct. 2012. 5, 6

[43] P. W. Theiler, J. D. Wegner, and K. Schindler. Globally consistent registration of terrestrial laser scans via graph optimization. *ISPRS J. Photographic and Remote Sensing*, 109:126–138, 2015. 2

[44] F. Tombari, S. Salti, and L. Di Stefano. Unique signatures of histograms for local surface description. In *Euro. Conf. on Comp. Vision*, pages 356–369, 2010. 2

[45] A. Torsello, E. Rodola, and A. Albarelli. Multiview registration via graph diffusion of dual quaternions. In *IEEE Intl. Conf. Comp. Vision, and Pattern Recog.*, pages 2441–2448, 2011. 2

[46] B. Ummenhofer, H. Zhou, J. Uhrig, N. Mayer, E. Ilg, A. Dosovitskiy, and T. Brox. DeMoN: Depth and motion network for learning monocular stereo. In *IEEE Intl. Conf. Comp. Vision, and Pattern Recog.*, volume 5, page 6, 2017. 1, 2

[47] J. Yang, H. Li, D. Campbell, and Y. Jia. Go-ICP: A globally optimal solution to 3D ICP point-set registration. *IEEE Trans. Pattern Anal. Mach. Intel.*, 38(11):2241–2254, Nov 2016. 2

[48] N. Yang, R. Wang, J. Stückler, and D. Cremers. Deep virtual stereo odometry: Leveraging deep depth prediction for monocular direct sparse odometry. In *Euro. Conf. on Comp. Vision*, pages 835–852, 2018. 2

[49] A. Zeng, S. Song, M. Nießner, M. Fisher, J. Xiao, and T. Funkhouser. 3Dmatch: Learning local geometric descriptors from RGB-D reconstructions. In *IEEE Intl. Conf. Comp. Vision, and Pattern Recog.*, pages 199–208, 2017. 2

[50] H. Zhou, B. Ummenhofer, and T. Brox. DeepTAM: Deep tracking and mapping. In *Euro. Conf. on Comp. Vision*, September 2018. 1, 2

[51] Q.-Y. Zhou, J. Park, and V. Koltun. Fast global registration. In *Euro. Conf. on Comp. Vision*, pages 766–782, 2016. 2, 3

[52] Q.-Y. Zhou, J. Park, and V. Koltun. Open3D: A modern library for 3D data processing. *arXiv:1801.09847*, 2018. 8

[53] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe. Unsupervised learning of depth and ego-motion from video. In *IEEE Intl. Conf. Comp. Vision, and Pattern Recog.*, volume 2, page 7, 2017. 1, 2

Supplementary

A. Interpretation of Our Method

Given the differences between the problem formulations in Equation 1 and 2 , it is natural to ask why we use the neural network f_θ to estimate the sensor pose $\mathbf{T}$ instead of directly optimization. In this section, we attempt to provide a simple potential interpretation of the benefit introduced by our formulation.

The basic inspiration comes from an optimization technique known as changing variables [2] that can convert an originally non-convex optimization problem to an equivalent convex one. In their example, a geometric program can be converted to a linear program by substituting exponential functions as original variables. In our formulation, we combine this idea with neural networks by replacing the optimization variables $\mathbf{T}$ with $f_\theta(\mathbf{S})$ and transforming the objective function from Equation 1 to 2. While we do not expect that the replacement of variables $\mathbf{T}$ with neural network parameters θ yields a convex problem, we observe that this transformation is beneficial to finding optimal solutions to the original problem.

We conduct a simple 1D experiment to illustrate this observation. Consider a problem of finding the optimal value of $x \in \mathbb{R}$ that minimizes $\mathcal{L}(x)$, a non-convex objective function with multiple local minima shown as the black line in Figure 11. Specifically, the objective function is defined as

$$\mathcal{L}(x) = \frac{1}{2}x^2 + 5\sin(10x) + 20\sin(x).$$

In this experiment, we compare two optimization methods, i.e., the proposed network-based optimization and the direct optimization. For network-based optimization, we introduce a MLP, f_θ, which consists of FC(10)-FC(20)-FC(30)-FC(40)-FC(1). Each MLP layer is followed by an ELU [12] activation function except for the output layer. The MLP has one node in the input and the output layer to replace the variable x with $f_\theta(z)$, resulting in another problem with optimization variable z. To ensure the same starting point, the direct optimization is initialized with $x_0 = f_{\theta_0}(z_0)$ where θ_0 and z_0 are the initial values of network parameters θ and variable z, respectively. We use gradient descent with a learning rate of 2×10^{-4} and run 1000 iterations. For the network-based optimization, we jointly update the network parameters θ and z.

The cyan point shows the result using gradient descent optimization that is performed directly on $\mathcal{L}(x)$, which is trapped in a local minimum. The function $\mathcal{L}(f_{\theta^\star}(z))$ with the optimal $\theta^\star$ found in the network-based optimization is plotted as the blue dash line in Figure 11. We take $f_{\theta^\star}(z^\star)$ to retrieve the optimal point $x^\star$ for $\mathcal{L}(x)$. The red plus and red circle in Figure 11 correspond to $z^\star$ and $x^\star$, respectively. The green star symbols show the values of x during the 1000 gradient-descent iterations.

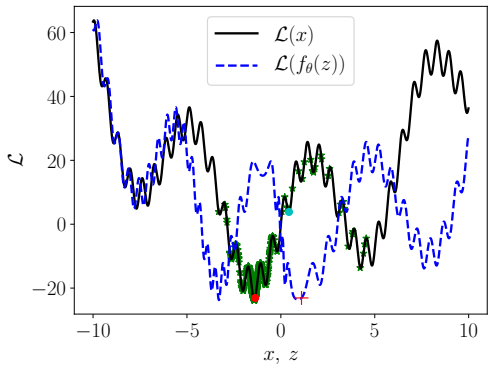


Figure 11. A 1D example to show the effectiveness of our neural-network-based conversion of a optimization problem into a higher dimension one. Red point shows the optimal solution found in the converted problem, while the cyan point shows the gradient descent optimum in the original problem. Please refer to Section A for a detailed explanation of the figure. Best viewed in color.

Notice the distribution of the green star symbols, visualizing the "sampled locations" in the domain, x, of the original problem. It is interesting to see that our conversion leads to a wider search range in the original problem domain, while keeping the same number of function evaluations of the original problem $\mathcal{L}(\cdot)$ as in direct gradient descent.

B. Ablation Studies

In this section, we conduct several ablation studies to investigate the effects of various network architectures in DeepMapping using the AVD [4]. For quantitative comparison, we choose absolute trajectory error (ATE) as metrics and include the ATE from baseline multiway registration method [11].

Feature extraction module in the L-Net: we compare the effects of feature extraction module, i.e., CNN-based architecture and PointNet-based architecture [34]. The CNN-based network consists of C(3)-C(64)-C(128)-C(256)-C(1024)-AM(1), where C(n) denotes 2D atrous convolutions with kernel size 3 and dilation rate of 2 and AM(1) denotes 2D adaptive max-pooling layer. The PointNet-based architecture is FC(3)-FC(64)-FC(128)-FC(256)-FC(1024)-AM(1) where FC(n) is fully-connected layer with n output nodes.

The box plot in Figure 12 depict the quantitative results of the ATE. As shown, CNN-based architecture achieves better performance with a median error of 134.07mm than PointNet-based architectures that has a median error of 207.84mm. This is not supervising because CNN is able to explore local structure information from neighborhood pixels while PointNet is a per-point function performing on each point independently.

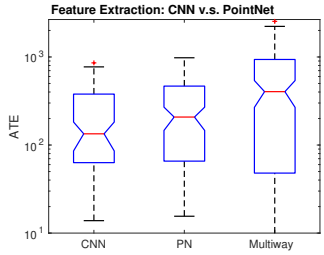


Figure 12. Quantitative comparison of the ATE between CNN-based and PointNet-based architectures in the L-Net, tested on the AVD [4].

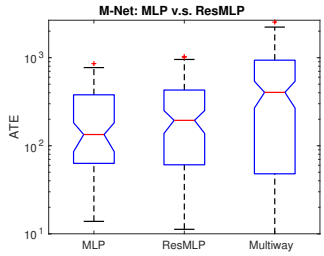


Figure 13. Quantitative comparison of the ATE between MLP and ResMLP in the M-Net, tested on the AVD [4].

Architecture of the M-Net: the proposed DeepMapping uses MLP in the M-Net to predict the occupancy status in the global coordinates. We compare this architecture with ResMLP that integrate the idea of deep residual networks [21]. ResMLP consists of a stack of basic residual blocks where each residual block, denoted as RB(n), contains two fully-connected layers with the same number of output nodes n. The detailed ResMLP architecture can be described as RB(64)-RB(64)-RB(64)-RB(128)-RB(128). As shown in Figure 13, MLP has a marginal improvement over ResMLP in terms of the ATE and therefore is adopted in the proposed DeepMapping.

Depth and width of MLP in the M-Net: the depth and width of MLP are defined as the number of layers in the MLP and the number of output nodes from each layer. To investigate the influence of layer depth, we fixed the layer width to 64 and test MLP with depths 4, 5, 6, and 7. Figure 14 show the corresponding results of the ATE. As shown, increasing MLP depth is beneficial to reducing the ATE. For example, MLP with depth 6 has a lower error than those with depth 4 and 5. However, deeper networks may deteriorate the performance and make it difficult to optimize. To compare the effect of MLP width, we fixed the depth to 4 and choose MLP with width 32, 64, 96, and 128. As shown in Figure 15. MLP with a width of 128 achieves the best performance.

C. More Results on 2D Simulated Point Cloud

Figure 16 shows additional qualitative comparisons of registration results on the 2D simulated dataset. As shown,

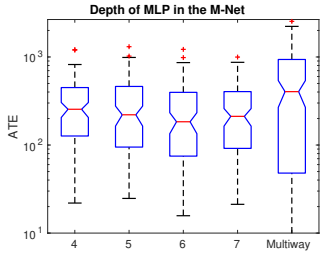


Figure 14. Quantitative comparison of different depths of MLP in the M-Net, tested on the AVD [4]. The layer width is fixed to 64.

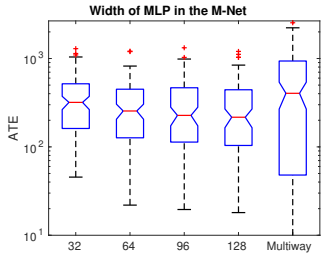


Figure 15. Quantitative comparison of different width of MLP in the M-Net, tested on the AVD [4]. All MLP have the same depth of 4 layers.

both the direct optimization and the incremental ICP with point-to-point metric fails to register all point clouds. The proposed DeepMapping, however, is more robust and accurate than baseline methods. The last two rows in Figure 16 show two cases where all methods fail to find correct registration.

Table 1 reports the average execution time and the success rate for different methods to register 128 point clouds. A registration of multiple point clouds is considered successful if the ATE is less than a threshold of 20 pixel, which is about 2% of the image size (1024×1024). The success rate is then defined as the ratio of the number of successful registration to the total number of test cases. All methods are tested on a machine with a 2.1GHz Intel® Xeon® CPU E5-2695 v4 CPU. We use a NVidia K80 for "training" DeepMapping and the direct optimization. While DeepMapping takes a long time to find optimal registration, the success rate is significantly higher than those of baselines.

	DeepMapping	Direct Opt.	ICP (Point)	ICP (Plane)
Runtime	133min	80min	6.48s	12.35s
Succ. Rate	84.2%	31.5%	36.0%	53.3%

Table 1. Average runtime and success rate for different methods tested on the 2D simulated dataset.

We also test two initialization methods, random initialization and zero initialization, for the direct optimization. Both methods have worse performances than the initialization which is the same as DeepMapping.

D. More Results on the Active Vision Dataset

Figure 17 shows additional visual comparison tested on the AVD [4]. The black ellipse highlights the region corresponding to misaligned parts from baseline methods. Table 2 lists the average execution time and the success rate to register 16 point clouds from the AVD. In this experiment, a registration is considered to be successful if the ATE is less than $450mm$. The hardware configuration is identical to those in Section C. As shown, the success rate from DeepMapping is higher than the rate from multiway registration [11].

	DeepMapping	Direct Opt.	Multiway [11]
Runtime	84min	71min	42.49s
Succ. Rate	80.0%	77.1%	58.1%

Table 2. Average runtime and success rate for different methods tested on the AVD.

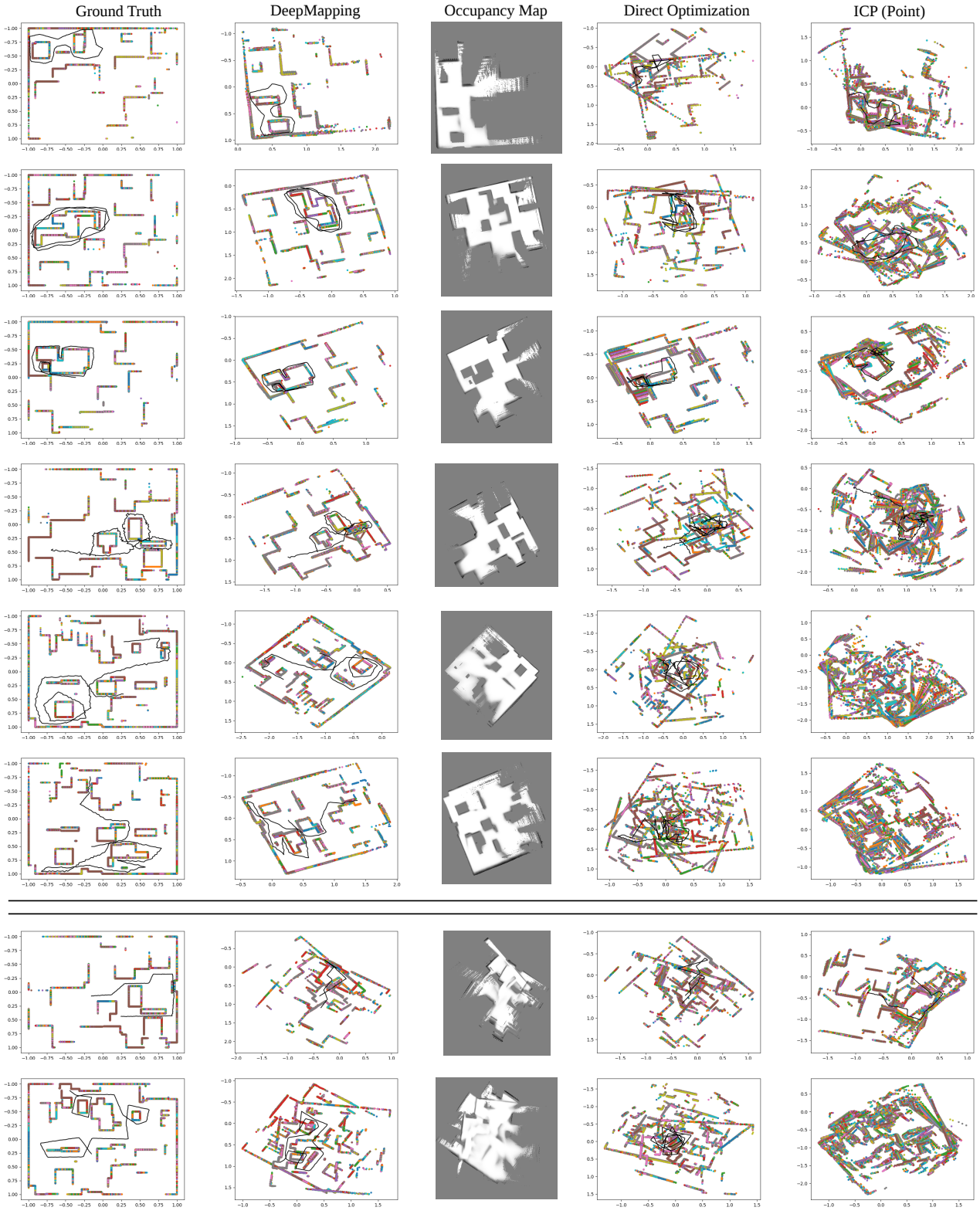


Figure 16. Additional visual comparisons of multiple point clouds registration from the 2D simulated dataset. The black lines are the trajectories of sensor. The third column show the images of occupancy maps for each trajectory that are estimated by the M-Net. The black, while, and gray pixels show the occupied, unoccupied, and unexplored locations, respectively. Note that the results of each trajectory cam be defined in arbitrary coordinate systems and do not necessarily aligned with ground truth. The last two rows show the failure cases where all three methods fail to find the optimal registrations. Best viewed in color.

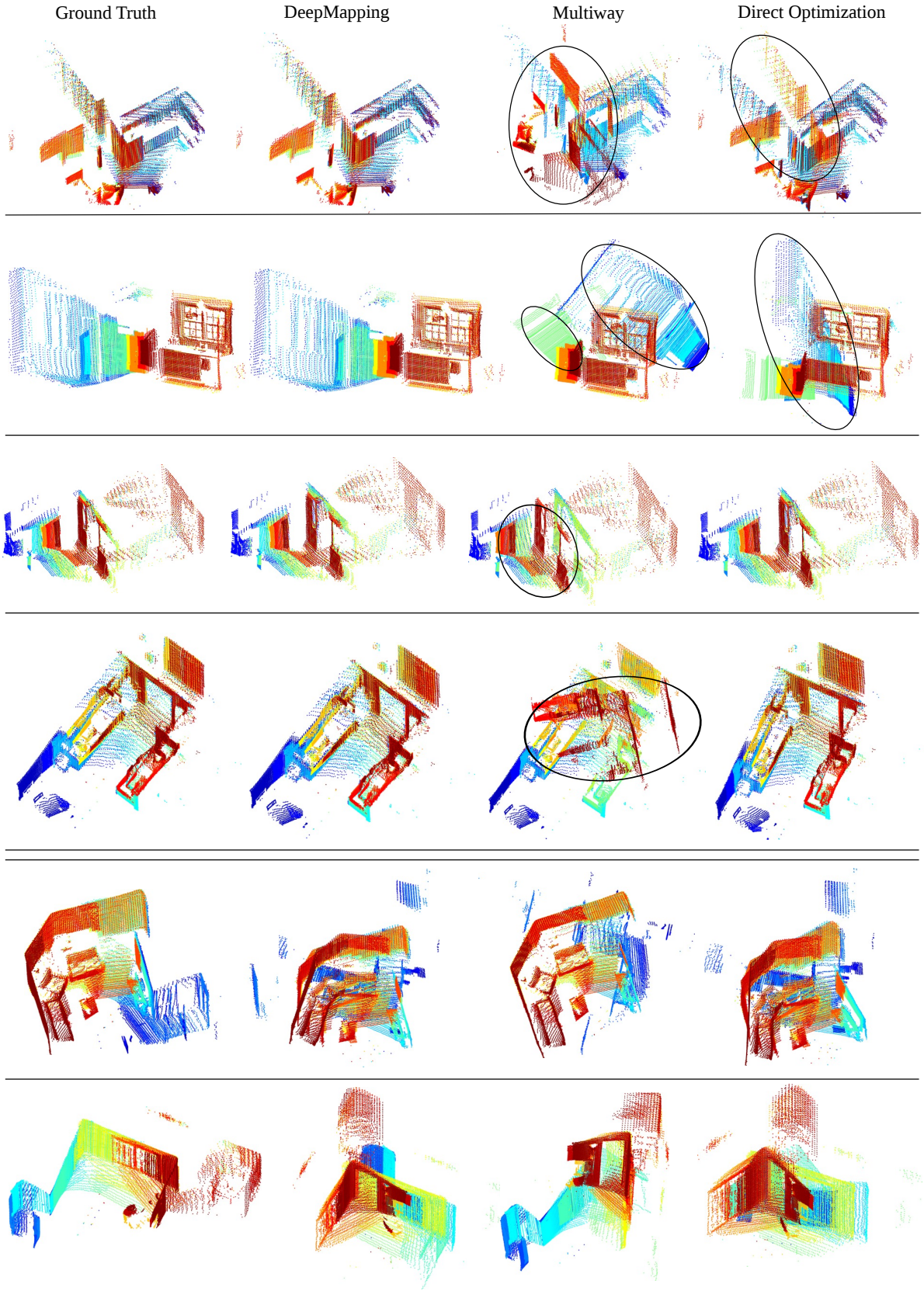


Figure 17. Additional visual comparisons of multiple point clouds registration from the AVD [4]. The black ellipses highlight the misaligned parts in baselines. Each color represents one point cloud. The last two rows show the failure cases where all three methods fail to find the optimal registrations. Best viewed in color.