An Algorithm for the SE(3)-Transformation on Neural Implicit Maps for Remapping Functions

Yijun Yuan, *Graduate Student Member, IEEE*, and Andreas Nüchter, *Member, IEEE*

***Abstract*—Implicit representations are widely used for object reconstruction due to their efficiency and flexibility. In 2021, a novel structure named neural implicit map has been invented for incremental reconstruction. A neural implicit map alleviates the problem of inefficient memory cost of previous online 3D dense reconstruction while producing better quality. However, the neural implicit map suffers the limitation that it does not support remapping as the frames of scans are encoded into a deep prior after generating the neural implicit map. This means, that neither this generation process is invertible, nor a deep prior is transformable. The non-remappable property makes it not possible to apply loop-closure techniques. We present a neural implicit map based transformation algorithm to fill this gap. As our neural implicit map is transformable, our model supports remapping for this special map of latent features. Experiments show that our remapping module is capable to well-transform neural implicit maps to new poses. Embedded into a SLAM framework, our mapping model is able to tackle the remapping of loop closures and demonstrates high-quality surface reconstruction. Our implementation is available at github for the research community.**

***Index Terms*—Mapping, SLAM.**

I. Introduction

THE reconstruction of 3D models has been explored for decades. Its developments followed the trend of low-cost, high-quality sensors, and[1] efficient computation hardware and were boosted in recent years with the progress in Deep Learning.

In the field of reconstruction, most attention has been drawn to global optimizations with bundle adjustment and loop closure [1]–[3]. Reconstructions using the Signed Distance Function (SDF) as a representation [4] have been widely accepted as a fundamental basis since Kinect Fusion [5] and VoxelHashing [6]. Recently, this basis is being challenged by the new trend of Deep Learning, as those conventional approaches have issues with the memory requirements and quality of uncomplete scans.

Relying on the high modeling ability of deep learning models, DeepSDF [7] and Occupancy Networks [8] propose implicit geometric representations that represent the shape in continuous space and thus are able to extract maps at an arbitrary resolution. Similar to deep local descriptor [9] that uses a parametric function to encode the geometry, the deep implicit model further supports prediction of the fields. These deep implicit representations have been widely sought after for their flexible use in shape reconstruction [7], shape generation [10] and more general tasks. By operating on the voxel-level, [11], [12] even provide semantics-agnostic high-quality reconstructions.

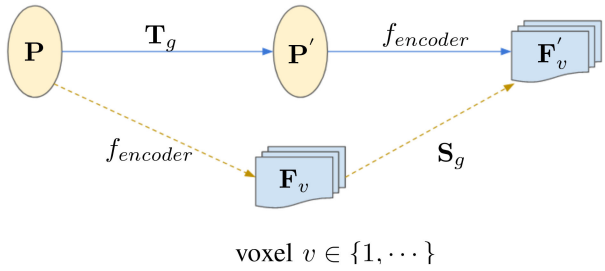


Fig. 1. Two flow paths to SE(3)-transform and deep encode the point cloud. The solid line indicates the transform-encoding path to generate implicit map of $\mathbf{T}_g$-transformed point cloud $\mathbf{P}$. The dash line shows the encoding-transform path, transforming the map of features with transformation $\mathbf{S}_g$, that is introduced in this paper.

Relying on the success of deep implicit representation, in 2021, DI-Fusion [13] and NeuralBlox [14] firstly proposed incremental neural implicit maps for 3D reconstructions. DI-Fusion especially has introduced a novel reconstruction pipeline that effectively combined the advantage of the efficiency of neural implicit representation and the robustness of field base registration. Different from the DI-Fusion that uses still the conventional SLAM pipeline, [15], [16] have proposed live-optimization with implicit representation for reconstruction which is also able to complete unseen surfaces.

However, there is still a severe limitation for an implicit representation compared with the common (point cloud, TSDF) methods: implicit representations do not support transformation. And it is this limitation that makes it hard to implement relocalization and remapping for neural implicit map reconstructions. Yuan *et al.* proposes indirect registration to evade the transformation of field during registration [17]. However, the rotation and translation is inevitable for remapping function.

In this paper, we propose a transformation algorithm for neural implicit maps to fill in this gap. As shown in Fig. 1, encoding the transformed point cloud is equivalent to first encoding the points and transforming then on the neural implicit. $\mathbf{S}_g$ is the transformation on neural implicit corresponding to the Euclidean space transformation $\mathbf{T}_g$.

Manuscript received 23 February 2022; accepted 15 June 2022. Date of publication 23 June 2022; date of current version 1 July 2022. This letter was recommended for publication by Associate Editor L. Paull and Editor J. Cuvera upon evaluation of the reviewers' comments. *(Corresponding author: Yijun Yuan.)*

The authors are with the Informatics VII: Robotics and Telematics, University of Würzburg, 97070 Wuerzburg, Germany (e-mail: yijun.yuan@uni-wuerzburg.de; andreas@nuechti.de).

Digital Object Identifier 10.1109/LRA.2022.3185383

[1]https://github.com/Jarrome/IMT_Mapping

The main challenge is the feature space transform with the corresponding alignment of the original point set. Thus, in this paper, we exploit the topic of equivariant representation [18]–[20], to implement the implicit map transformation. As the focus of SE(3)-equivariant research is not the transformation, it is actually not adequate to solve full transformation in feature space. In addition, the recent approaches work only on small examples with simple structures. Thus, our proposed model works on a map of neural implicits, i.e., a set of neural implicit functions, instead of one holistic implicit function, to evade both limitations.

The transformed implicit map is able to produce a very close result to the implicit map of the transformed point cloud. To demonstrate the advantage of our mapping model, we also embed it into a loop-closure equipped SLAM-algorithm [21].

The contributions of this paper are as follows:

- We propose a transformation algorithm for neural implicit maps.
- We implement a 3D reconstruction with this remapping model.

In the following, we first describe briefly the related work for implicit function and equivariant features. Then, we introduce our transformation algorithm for neural implicit maps. After that, experiments demonstrate the performance and we conclude this work.

II. Related Work

A. Deep Implicit Representations

Algorithms for Implicit Representation can be divided into two categories: First, the most widely used branch is the DeepSDF [7]. For SDF, the geometry prior is encoded with MLP and then fed into another model together with query points to extract the signed distance values with a discretized distance field. A mesh is then extracted using the Marching Cubes algorithm [22]. For non-closed shapes which is more general for point cloud data, an unsigned distance field neural model is introduced without indicating inside-outside [23]. The second category is Occupancy Networks [8]. For Occupancy Networks, different from the distances in the SDF model, the probability of occupancy at a certain position is estimated from the implicit function. Then a Multiresolution IsoSurface Extraction (MISE) is implemented to obtain meshes.

To efficiently reconstruct intricate surfaces, DeepLS [11] introduces a local deep geometry prior and performs the reconstruction with a set of local learned continuous SDFs. Similarly, [12] proposes a local implicit grid for reconstruction. Note that, one advantage of the local implicit model is that it relieves the pressure on the encoding model. As a local surface is much more simple compared to a whole complicated scene, such a local strategy is trained on a simple synthetic object dataset and generalized then to the real complex scene.

In 2021, DI-Fusion [13] moves one step further and leads the research to a real reconstruction of a scene. It is the first implicit function research that realizes the incremental reconstruction of a scene. More importantly, they alleviate the memory inefficiency of SDF representation by updating a map of latent features while extracting distance values for registration and visualization, yielding a new direction of 3D reconstruction with Deep Learning. Similarly, NeuralBlox [14] also proposes to fuse the grid of latent features. Given an external state estimation with noise, its latent code fusion still shows a robust performance. iMAP [15] proposes a non-conventional SLAM pipeline with implicit neural representation for incremental reconstruction. The main component of it is a differentiable rendering model. With an online optimization, the reconstruction is optimized by repeatedly minimizing the rendering distance to observed images.

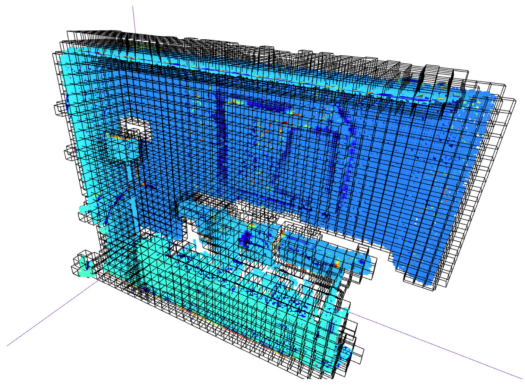


Fig. 2. PLIVox representation from DI-Fusion [13].

In this work, we build on top of DI-Fusion by using their PLIVox representation as in Fig. 2.

B. Equivariant Feature

Equivariance is a novel concept for 3D point clouds. The target is a universal representation of objects with different poses to avoid exhaustive data augmentation.

We follow the definition in SE(3)-Transformers [19]: Given a set of transformations $\mathbf{T}_g : \mathcal{V} \to \mathcal{V}$ for $g \in G$, where G is an abstract group, a function $\phi : \mathcal{V} \to \mathcal{Y}$ is equivariant if for g, there exists a transformation $\mathbf{S}_g : \mathcal{Y} \to \mathcal{Y}$ such that

$$\mathbf{S}_g[\phi(v)] = \phi(\mathbf{T}_g[v]) \text{ for all } g \in G, v \in \mathcal{V}. \tag{1}$$

From their definition, an SO(3)-equivariant function ϕ follows $\mathbf{S_R}\phi(v) = \phi(\mathbf{R}[v])$ with function $\mathbf{S}$ an operation to produce same result as aligning the point cloud. While for the translation-equivariant, for convenience, the $\mathbf{S}$ operation is usually defined as identity mapping.

As the translation is reduced with the relative position, most of works mainly focus on SO(3)-equivariant representations [18], [24]–[26] with steerable kernel bases. SE(3)-Transformers [19] leverages the advance of self-attention on large point sets and graphs with various point numbers.

Realizing equivariance by learning, those works are restricted to using convolutions and relative positions of neighboring points. Vector Neurons (VNN) firstly introduce a whole group of network layers that produce SO(3)-equivariant features [20]. It is flexible to reconstruct PointNet [27] or DGCNN [28] with VNN layers. This provides us a good basis as it is able to

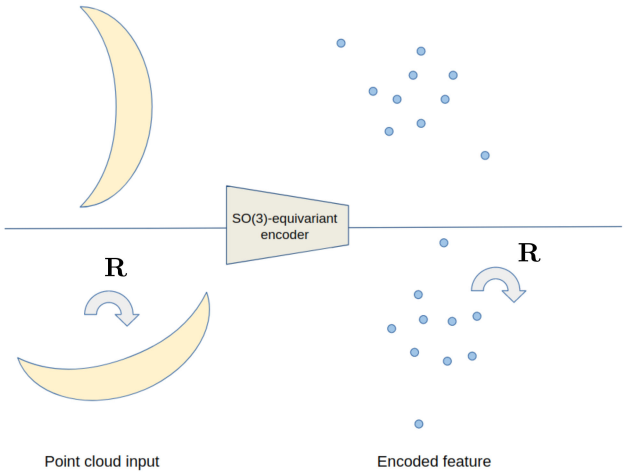


Fig. 3. SO(3)-equivariant representation for point cloud.

function as point-encoder. In this work, we mainly use three rotation-equivariant operations: VNLinear, VNLeaKyReLU, and mean-pooling. For example, with the input $\mathbf{V} \in \mathbb{R}^{C\times 3}$, VNLinear parameter $\mathbf{W}_l \in \mathbb{R}^{C'\times C}$, VNLinear produces the output $\mathbf{W}_l\mathbf{V}$ where $(\mathbf{W}_l\mathbf{V})\mathbf{R} = \mathbf{W}_l(\mathbf{V}\mathbf{R})$ is rotation-equivariant. LeaKyReLU produces each output vector-neuron $\mathbf{v}'_c \in \mathbf{V}' = f_{\text{LeaKyReLU}}(\mathbf{V})$ with separate parameters $\mathbf{W}_c \in \mathbb{R}^{1\times C}$ and $\mathbf{U}_c \in \mathbb{R}^{1\times C}$ where $c \in \{1,\dots,C\}$. For each vector-neuron $\mathbf{v}'_c \in \mathbb{R}^{1\times 3}$, it maps the input feature $\mathbf{V}$ to the feature $\mathbf{q}_c = \mathbf{W}_c\mathbf{V} \in \mathbb{R}^{1\times 3}$ and direction $\mathbf{k}_c = \mathbf{U}_c\mathbf{V} \in \mathbb{R}^{1\times 3}$. Thus it produces

$$\mathbf{v}' = \begin{cases} \mathbf{q}_c & \text{if } \langle \mathbf{q}_c, \mathbf{k}_c\rangle \geqslant 0 \\ \mathbf{q}_c - \left\langle \mathbf{q}_c, \frac{\mathbf{k}_c}{\|\mathbf{k}_c\|} \right\rangle \frac{\mathbf{k}_c}{\|\mathbf{k}_c\|} & \text{otherwise,} \end{cases} \tag{2}$$

with the output $\mathbf{V}' = [\mathbf{v}'_c]_{c=1}^{C}$ [20]. Mean-pooling is an average on the same dimension of all points, therefore it is naturally rotation-equivariant.

As the original goal of the equivariance concept is to provide universal features, i.e., $\mathbf{S} = \mathbf{I}$ is adequate for translation. However, with a different focus of transforming the feature space, this setting is not applicable. In this paper, only the feature rotation resorts to the SO(3)-equivariant architecture VNN [20] and functions as Fig. 3. The translation is solved using other techniques.

III. Methodology

We follow DI-Fusion [13] to use the evenly-spaced voxels (PLIVoxs) to represent the map. $\mathbf{V} = \{\mathbf{v}_m = (\mathbf{c}_m, \mathbf{F}_m, w_m)\}$ with $\mathbf{c}_m \in \mathbb{R}^3$, $\mathbf{F}_m \in \mathbb{R}^L$, $w \in \mathbb{N}$ the voxel centriod, latent representation of observed geometry and observation count respectively.

A. SO(3)-Equivariant Features

Our encoder-decoder neural network Φ follows the design of encoder-decoder in DI-Fusion [13]. ϕ_e encodes points in a PLIVox, and ϕ_d predicts distance mean and standard deviation for query points. But different from DI-Fusion that is using a simple PointNet structure, to realize the transformation on the neural implicit map, we propose to use VNN layers [20] to build an SO(3)-equivariant encoder.

For each local voxel, it uses the points $\mathbf{P}_m$ and the norm $\mathbf{S}_m$ as an input. Two branches of VNN-MLPs are respectively applied on $\mathbf{P}_m$ and $\mathbf{S}_m$ and produce features $\mathbf{F}_{\mathbf{P}_m} \in \mathbb{R}^{l\times 3}$ and $\mathbf{F}_{\mathbf{S}_m} \in \mathbb{R}^{l\times 3}$. Then by concatenating $\mathbf{F}_{\mathbf{P}_m}$ and $\mathbf{F}_{\mathbf{S}_m}$ along the l axis, it achieves $\mathbf{F}_m \in \mathbb{R}^{2l\times 3}$. A point encoder ϕ_p is given in Fig. 5. The local point set encoder ϕ_e produces the mean-pooling of the ϕ_p output in $\mathbf{P}_m$.

In this encoder-decoder we changed the encoder with VNN to realize the SO(3)-equivariant functionality. For more details about the decoder network and Conditional Neural Processes-style training, please refer to DI-Fusion [13].

B. Neural Implicit Mapping Module

Neural Implicit Mapping Module consists of Encoding (Sec. III-A), Fusion (Sec. III-B3), Removal (Sec. III-B3), and Transforming (Sec. III-B1) functions.

The input frame to this module is firstly encoded into a local neural implicit map and fused to the global map. When a loop is detected, remapping a certain frame requires removing, transforming, and fusing the corresponding local neural implicit map.

A diagram of our mapping module is given in Fig. 4. The neural implicit mapping module serves as a mapping module for SLAM.

1) Transformation to Global Coordinate: Our transformation algorithm of Neural Implicit Map consists of two steps: the grid transformation and the feature rotation.

As demonstrated in Fig. 6, given the transformation $\mathbf{T}$ of the local map $\mathbf{V}_l$ to global coordinates, the update is actually on the center $\mathbf{c}$ and its corresponding implicit feature $\mathbf{F}$ for each PLIVox. For PLIVox voxel $\mathbf{v}_m \in \mathbf{V}_l$, center (grid coordinate) $\mathbf{c}_m$ directly transforms

$$\mathbf{c}_m \leftarrow \mathbf{T} \cdot \mathbf{c}_m. \tag{3}$$

Afterwards, for the feature $\mathbf{F}_m \in \mathbb{R}^{2l\times 3}$, as the feature is always positioned at the voxel center, the rotation is left to solve. Thus

$$\mathbf{F}_m \leftarrow (\mathbf{R} \cdot \mathbf{F}_m^T)^T. \tag{4}$$

However, only transforming the local neural implicit frame is not sufficient to update the global map. Because the transformed voxel grid for the local frame may not be consistent with the grid of the global map.

2) Interpolation to Global Grid: As shown in Fig. 7, there is a small gap between global and local grid coordinates. Therefore, we need to additionally interpolate the local features on the global grid.

Since the distance between the local and the target voxel is small, and points involved for encoding are actually in a 2-times voxel length region around its voxel in implementation, we propose to align voxels by linearizing the function $\phi_e(\mathbf{P} + \mathbf{t})$. Each $\mathbf{v}_m$ in the local grid will contribute to the close neighbor $\mathbf{v}_n$ in the global grid:

$$\mathbf{F}_{\mathbf{v}_n} = \phi_e(\mathbf{P}_m + \mathbf{t}_{m,n}), \quad \mathbf{t}_{m,n} = \mathbf{c}_n - \mathbf{c}_m \tag{5}$$

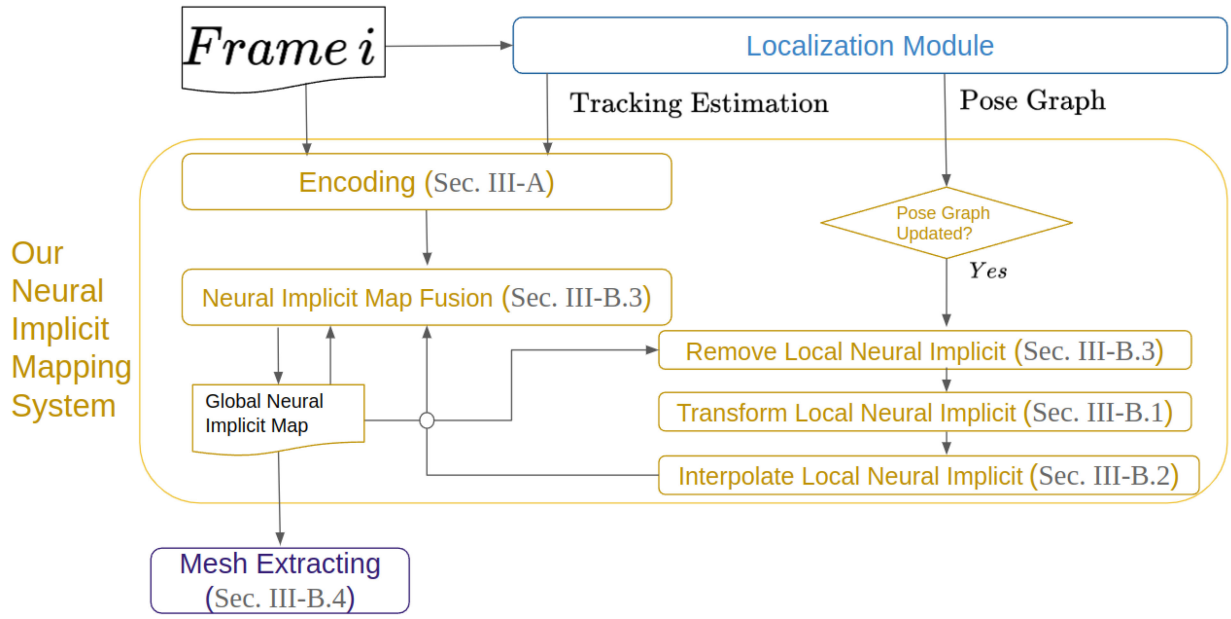


Fig. 4. Pipeline for SLAM embedding our mapping module. SLAM module provides the point cloud P_i' and the pose table $\{T_1, \ldots, T_i\}$ for the mapping module with keyframe i.

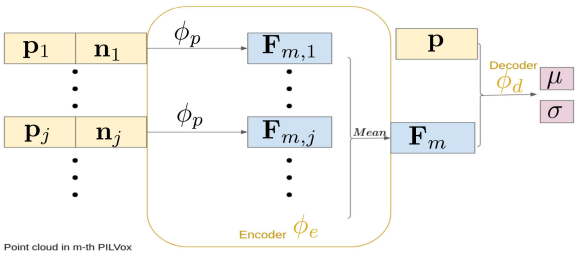


Fig. 5. The structure of encoder-decoder neural network. $\mathbf{p_j}$, $\mathbf{n_j}$ are point xyz and norm for certain point $\mathbf{p}_j$ in one m-th PLIVox. $\mathbf{p}$ are point for inference and μ, σ are estimated distance value and its standard derivation.

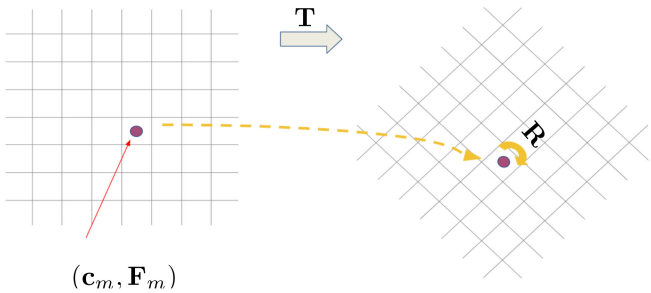


Fig. 6. Demonstration of the transformation on the neural implicit map. The voxel grid is transformed to a new position. $\mathbf{F}_m$ rotates since $\mathbf{F}_m$ is still positioned at the center of a voxel in the grid after the transformation (center is transformed).

Then by linearizing the right side of (5), we yield

$$\mathbf{F}_{\mathbf{v}_n} = \mathbf{F}_{\mathbf{v}_m} + \frac{\partial}{\partial \mathbf{t}}[\phi_e(\mathbf{P}_m)]\mathbf{t}_{m,n}. \tag{6}$$

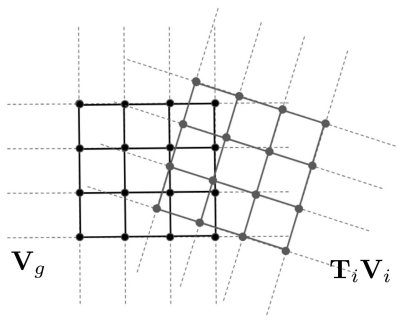


Fig. 7. The transformed local grid is not well-fitted with the global grid. Here we show the center of the grid for better demonstration.

Here we denote the Jacobian as $\mathbf{J}_m = \frac{\partial}{\partial \mathbf{t}}[\phi_e(\mathbf{P}_m)]$, for which $\mathbf{J}_m \in \mathbb{R}^{2l\times 3\times 3}$.

Following the feature metric of PointNetLK [29], we approximate each column of the Jacobian using

$$\mathbf{J}_{m,p} \approx \frac{\phi_e(\mathbf{P}_m + \mathbf{t}_p) - \phi_e(\mathbf{P}_m)}{\Delta t} \in \mathbb{R}^{2l\times 3} \tag{7}$$

where $\mathbf{t}_p \in \{[\Delta t, 0, 0], [0, \Delta t, 0], [0, 0, \Delta t]\}$.

Then the Jacobian of the feature over the translation is

$$\mathbf{J}_m = [\mathbf{J}_{m,1}\ \mathbf{J}_{m,2}\ \mathbf{J}_{m,3}]. \tag{8}$$

To note that the Jacobian is computed together with the implicit feature which is *before* the transformation (with $\mathbf{T}$). Thus each column of Jacobian need a pre-transformation $\mathbf{J}_{m,p} \leftarrow \mathbf{J}_{m,p}\mathbf{R}^T$ together with its feature transformation in (4). In addition, the translation bias $\mathbf{t}_{m,n}$ in (6) cannot be directly multiplied with $\mathbf{J}_m$. An inverse rotation is required, that is $\mathbf{J}_m\mathbf{R}^T\mathbf{t}_{m,n}$. We then

keep the formulation (6) still valid by rewriting the Jacobian as $\mathbf{J}_m \leftarrow \mathbf{J}_m\mathbf{R}^T$.

In our implementation, for each target grid $\mathbf{v}_n$ in $\mathbf{V}_g$ that has neighbors with center distance $<$ voxelSize, we find its K nearest neighbors $\mathbf{v}_m$ where $m \in \{c_1, \cdots c_K\}$ in the local grid. Here c_i denotes the PLIVox index of neighbors. Then we interpolate

$$\mathbf{F}_n = \sum_{m\in\{c_1,\cdots c_K\}} s_{n,m}(\mathbf{F}_m + \mathbf{J}_m\mathbf{t}_{n,m}) \tag{9}$$

where $s_{n,m} = \frac{\exp(-||\mathbf{t}_{n,m}||^2)}{\sum\exp(-||\mathbf{t}_{n,\cdot}||^2)}$. Moreover, the voxel point number w is required for the following global neural implicit map updating (III-B3), thus should also be recorded

$$w_n = \sum_m s_{n,m}\cdot w_m. \tag{10}$$

3) Map Removal & Fusion: DI-Fusion provided an updating of the neural implicit map in a voxel-to-voxel manner. As we transform and fit the local grid to the global grid in previous Sec. III-B1 Sec. III-B2, we are now ready for the neural implicit map updating.

Since the global map has been fused previously with the local map $\mathbf{V}_k$, i.e., after a pose update, the global map $\mathbf{V}_g$ requires a local removal, and afterwards a local fusion with the updated neural implicit map. We similarly formulate the removal formula. The removal and fusion are done as following:

$$\mathbf{F}_m \leftarrow \frac{\mathbf{F}_m w_m \mp \mathbf{F}_m^k}{w_m \mp w_m^k}, \quad w_m \leftarrow w_m \mp w_m^k \tag{11}$$

4) Mesh Extracting: We follow DI-Fusion [13] to build the reconstruction model from the neural implicit map. First, signed distance fields are generated for each PLIVox by using the decoder ϕ_d from Sec. III-A. Then with the one complete signed distance field, the Marching Cube algorithm is used to extract the mesh model.

The whole pipeline is plotted in Fig. 4. When frame i is processed by the SLAM system, an external localization module (see experiment section) is required to track the camera and maintain the pose graph. In each frame, our neural implicit mapping module encodes the frame and fuses it into the global implicit representation. When there is a loop closure, it updates the sequence of poses and our mapping system checks the pose of each frame. If the pose of a certain frame is updated, the mapping module will *(1) remove the old local neural implicit map* of that frame, *(2) transform to a new pose and interpolate* to produce a new local neural implicit map from the original copy of the local map, and *(3) fuse this new local map into global*.

IV. Experiments

A. Setting

1) Datasets: Three datasets have been used in our experiments. The object dataset ShapeNet [30] is used for training purpose. The RGB-D dataset ICL-NUIM [31] and Replica [32] are utilized for quantitative evaluation.

a) ShapeNet [30]: ShapeNet is a rich-annotated large variety 3D shape dataset. We follow [13] to select 6 categories (bookshelf, display, sofa, chair, lamp, and table) and 100 samples to train the encoder and decoder model. For more details about the pre-procession of this data, please refers to [13].

b) ICL-NUIM [31]: ICL-NUIM is a widely used RGB-D dataset for SLAM and Reconstruction. It contains living rooms and office room scenes. From which the living room scene contains a ground truth surface model. So this living room scene is widely involved in research for surface comparison. We use lr-kt[0-3] with synthetic noise for standard surface comparison.

c) Replica [32]: The Replica data set is a highly photo-realistic 3D indoor scene reconstruction dataset. We use iMAP's [15] 8 sequences (5 offices and 3 apartments) from Replica. The 8 sequences contain rendered 2000 RGB-D frames each. Different from ICL-NUIM sequences that do not repeatedly record certain views, because of the live-optimization of iMAP, the replica sequences cover each direction and surface multiple times. We extensively implement our model on this dataset to demonstrate the reconstruction quality.

2) Implementation Details: All of the experiment is implemented on a NUC-computer (CPU-i7-10710 U 1.10 GHz, 32 GB memory, GTX2080Ti-12 GB). We follow the DI-Fusion to set PLIVox parameters: voxel-size $= 0.1\,m$. For mesh extraction, we also set the same σ threshold $\sigma_D = 0.06$ to fairly compare with DI-Fusion. For encoding, we set the length of a feature to $9 \times 3 \times 2$ for the VNN SO(3)-equivariant feature. Three VNN-linear operations are required on each point and normal to get the two 9×3 features for each point in PLIVox. More specifically, the point encoder sequentially VNLinear(1,32)$\rightarrow$ VNLeakyReLU$\rightarrow$VNLinear(32, 32)$\rightarrow$VNLeakyReLU $\rightarrow$ VNLinear(32,9) results in a 9×3 size feature at point and normal branch respectively. Then, by mean-pooling and concatenating, one 18×3 size point set feature is obtained for that PLIVox. For decoding and optimization loss, we follow the same as DI-Fusion to predict both mean and variance and train the whole encoder-decoder network in a similar strategy as the Conditional Neural Processes [33]. For interpolation, we set the candidate number $K = 8$. For mesh extracting, resolution$= 4$ for the space grid in each PLIVox used.

To test on the Replica dataset, we set $\sigma_D = 0.15$ and resolution$= 3$.

3) Training: Even with the different encoder structures, our model is actually trained with the same setting as DI-Fusion [13] on ShapeNet data.

4) Testing: During testing, the pre-trained encoder-decoder is transferred to the new scene of indoor scale reconstruction. We have two tests, one is about the functionality of the transformation, and the other is about the incremental reconstruction. For the first test in Sec. IV-B, we use the mapping module for a single frame.

For the second test in Sec. IV-C, ORB-SLAM2 RGB-D is utilized to provide the localization. Then we use two benchmarks to evaluate the performance: ICL-NUIM and Replica [32]. ICL-NUIM is the most widely used standard test which has a standard surface model and metrics tools for comparison. On the standard ICL-NUIM benchmark, we compare with DVO-SLAM [34],

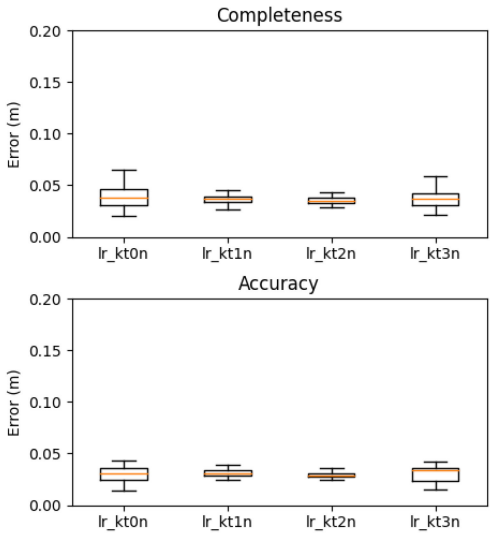


Fig. 8. Accuracy and completeness. Accuracy metrics show how close encode-transform extracted points to points in transform-encode branch are. While completeness metrics show how close transform-encode extracted points to points in encode-transform branch are.

Surfel Tracking [35], ElasticFusion [3], BundleFusion [2], and DI-Fusion [13].

On Replica dataset, we follow the paper iMap [15] to select the data and compare it with iMAP as a baseline. Values are taken from the iMAP paper as its source is not released.

B. Evaluate the Functionality of Transformation on Neural Implicit Maps

As introduced in Fig. 1, there are actually two paths to encode point clouds into the neural implicit map with transformation. To evaluate the functionality of our transformation algorithm, we generate neural implicit maps in both branches and then measure the performance with respect to accuracy and completeness between the reconstruction from the result neural implicit maps. Accuracy shows the average distance between the sampled reconstruction points in the encode-transform path and the nearest points in the transform-encode path. Completeness shows the average distance between the sampled points from transform-encode reconstruction and the nearest points in the encode-transform path. We select lr-kt[0-3] as the test sequences, and GT-trajectory to provide the transformation accordingly. After generating the neural implicit maps, the decoder is used to generate the Signed Distance Field and the Marching Cubes algorithm is used to generate the surface. Each frame is recorded separately to compute the surface error. We also draw the error for each frame, the distribution of error is shown in Fig. 8. It is clear that our method retains a similar reconstruction with the transformation on the neural implicit maps.

For the completeness especially, the very low error shows that our transformation-on-implicit well-reconstructs the surface region compared to the transformation-then-implicit build. However, there still exists an $\sim 3\,cm$ error. The success of the incremental reconstruction in the following experiment shows that this is more a small surface generation effect in the whole reconstruction.

TABLE I
COMPARISON OF SURFACE ERROR ON ICL-NUIM [31] BENCHMARK (MEASURED IN CENTIMETERS)

	lr kt0	lr kt1	lr kt2	lr kt3
DVO-SLAM [34]	3.2	6.1	11.9	5.3
RGB-D SLAM [36]	4.4	3.2	3.1	16.7
MRSMap [37]	6.1	14	9.8	24.8
Kintinuous [38]	1.1	0.8	0.9	15.0
ElasticFusion [3]	0.7	0.7	**0.8**	2.8
DI-Fusion [13]	**0.6**	1.5	1.1	4.5
Ours	1.2	1.58	1.0	**1.2**

The bold entities shows the best result of the comparison methods.

C. Evaluate the Incremental Reconstruction Performance

1) ICL-NUIM Test: In this part, we evaluate our model on the ICL-NUIM benchmark with synthetic noise added. We use a surface error to metric for the difference between reconstruction and the ground-truth model. The quantitative evaluation is demonstrated in Tab. I. We observe that the neural implicit map based algorithm achieves high accuracy compared to others. However, in lr-kt3 which contains loops, DI-Fusion does not exceed ElasticFusion. But our model gets the best score with 1.2. To note that, our model is able to detect and remap the start-end loop on lr-kt3, which is reflected on the scores, 1.2 cm, exceeding the rest. This demonstrates that our model is able to *address the problem of DI-Fusion which is not compatible to a loop closure module*. We find that our model scores similar on lr-kt1, 2 with DI-Fusion. It also approves that our VNN-encoder well-represent the feature while holding the SO(3)-equivariant functionality.

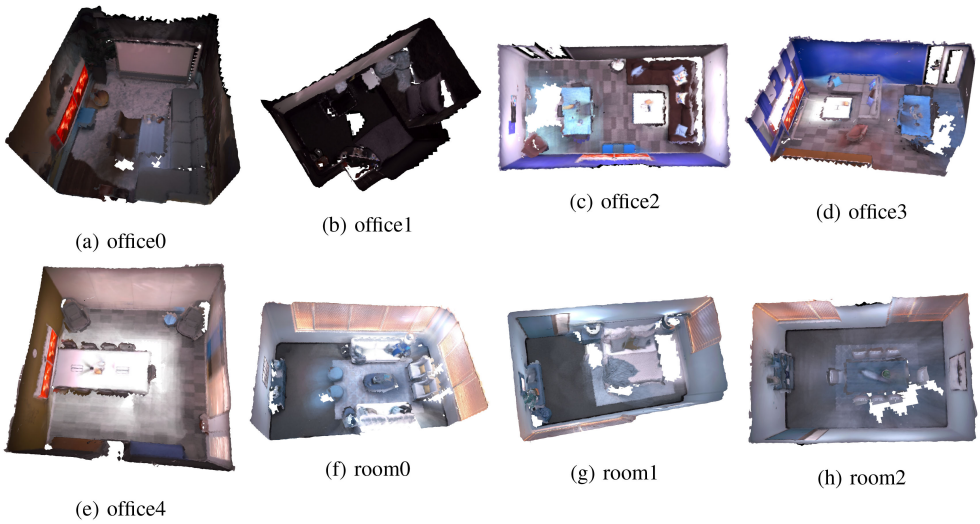
2) Replica Test: We also evaluate our model on the iMAP [15] Replica Dataset sequences. The metrics follow iMAP on accuracy, completion, and completion ratio. The completion ratio is an important metric because the ground truth model contains the ceiling which is mostly non-touched in data sequences. In Tab. II, we see that our model scores best on all accuracy tests, and best on most completion and completion ratio. On the average scores, our model achieves best on accuracy and completion ratio. Our average completion does not exceed iMAP.

Note that iMAP is a live-training model with differential rendering. It is naturally capable to complete the blocked region of point clouds. Our model does not have this point cloud completion function. This explains why iMAP exceeds ours on completion. But its higher completion but lower or similar completion ratio of iMAP means the guess of unobserved surface usually fails.

Some results are textured and plotted at Fig. 9.

D. Efficiency Test

1) Time Cost: Time efficiency of encoding and remapping directly influences the usage of our model in online reconstruction.



(a) office0 (b) office1 (c) office2 (d) office3

(e) office4 (f) room0 (g) room1 (h) room2

Fig. 9. Reconstruction Demonstration. Our post-processed textures are averaged from projected image colors.

TABLE II
RECONSTRUCTION TEST ON REPLICA DATASET [32]

		room-0	room-1	room-2	office-0	office-1	office-2	office-3	office-4	Avg.
iMAP* [15]	**Acc.** [cm] ↓	3.58	3.69	4.68	5.87	3.71	4.81	4.27	4.83	4.43
	Comp. [cm] ↓	5.06	4.87	5.51	**6.11**	**5.26**	**5.65**	5.45	6.59	**5.56**
	Comp. Ratio [< 5cm %] ↑	83.91	83.45	75.53	77.71	**79.64**	77.22	77.34	**77.63**	79.06
Ours	**Acc.** [cm] ↓	**2.05**	**1.74**	**1.97**	**2.03**	**1.63**	**2.10**	**2.75**	**3.07**	**2.17**
	Comp. [cm] ↓	**3.75**	**3.41**	**4.60**	9.68	8.73	**5.67**	**4.77**	**5.14**	5.72
	Comp. Ratio [< 5cm %] ↑	**86.59**	**87.60**	**83.57**	**79.28**	78.14	**77.76**	**78.19**	74.16	**80.66**

*Values taken from [15].
The bold entities shows the best result of the comparison methods.

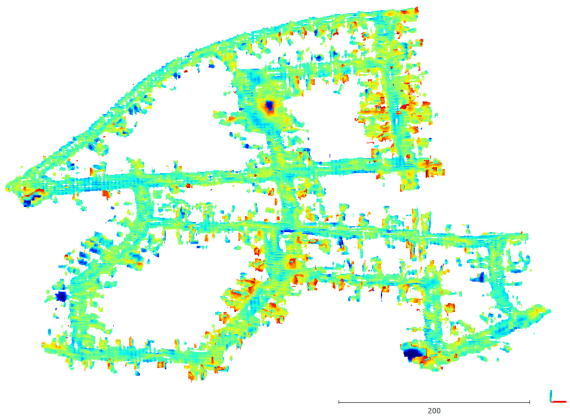


Fig. 10. Incremental reconstruction result on KITTI-odometry sequence 00.

Thus we recorded the time cost of lr-kt3 test from Sec. IV-C1 that contains a large loop.

For each frame that is fed into the mapping model, it is encoded as a local neural implicit map. The recorded encoding time is $\mathbf{0.0077s} \pm 0.0019\,\mathbf{s}$ per frame.

When a loop is detected, certain frames require being removed, transformed and fused again onto the global map. This is accomplished with neural implicit maps.

Per frame removal from global map takes $\mathbf{0.0040s} \pm 0.00038\,\mathbf{s}$. Per frame transformation and interpolation take $\mathbf{0.018s} \pm 0.0039\,\mathbf{s}$. Per frame fusion to global map takes $\mathbf{0.0032} \pm 0.00044\,\mathbf{s}$.

Thus the encoding processes around 130 **Hz**, and the remapping processes around 50 **Hz** on our NUC-computer. Therefore this remapping can be well-adequate for the online application.

2) Space Cost: The space cost mainly consists of the Network (encoder, decoder), Neural Implicit Map, and Meshing.

We evaluate this by saving network parameters and neural implicit maps into files. The parameter files are 31.5 kB for encoder and 207 kB for decoder. We save full result map from lr-kt3 test with $torch.save$, merely 29.2 MB are taken.

During the encoding, we fetched the network parameters that take 26.5 kB. Points are passed to our VNN-Encoder. As we previously provided the specific layers. The space cost is computed as $n \times \max\{1, 32, 32, 9\} \times 3 \times 2 = 192n$ $float32$ as an intermediate buffer is not reserved. Thus we count the points into the encoder to compute the encoding buffer $105MB \pm 59\,MB$.

We do not count the space cost of the mesh extracting, as in Fig. 4, it is for visualization and can be operated externally.

E. Demonstration on Campus-Scale Reconstruction

We are also interested in scenarios that other methods cannot do. We see that for indoor scenes, many sequences do not contain large loops for the front-end restructuring. But when it turns to outdoor LiDAR SLAM, such as KITTI-odometry [39], loop closure shows vital significance to remove the accumulated error

for a long trajectory. Thus, we attempt to produce a neural implicit mapping on such a scene to further reveal the capability of our algorithm. The LiDAR localization model PyICP-SLAM[2] is utilized to provide tracking estimation and the pose graph. Due to the scale difference to the indoor scene and limited available memory, we use a voxel size of $4\,m$, other hyperparameters are preserved. A reconstruction on KITTI-odometry sequence 00 is given in Fig. 10.

V. Conclusion

In this paper, we have presented a neural implicit mapping module that does support loop closing. By utilizing an SO(3)-equivariant encoder, we are able to implement SE(3)-transformations directly on the neural implicit maps. In combination with an interpolation step, our mapping module supports updating the neural implicit map when the pose of certain frame changes without touching the original 3D point cloud. In addition, we showed in our experiments, that our SO(3)-equivariant encoder takes the responsibility of generating neural implicit maps, and based on that, our transforming module functions well and provides high-quality reconstruction with and without a loop closure.

References

[1] Y.-P. Cao, L. Kobbelt, and S.-M. Hu, "Real-time high-accuracy three-dimensional reconstruction with consumer RGB-D cameras," *ACM Trans. Graph.*, vol. 37, no. 5, pp. 1–16, 2018.

[2] A. Dai, M. Nießner, M. Zollhöfer, S. Izadi, and C. Theobalt, "BundleFusion: Real-time globally consistent 3D reconstruction using on-the-fly surface reintegration," *ACM Trans. Graph.*, vol. 36, no. 4, 2017, Art. no. 76.

[3] T. Whelan, S. Leutenegger, R. Salas-Moreno, B. Glocker, and A. Davison, "ElasticFusion: Dense SLAM without a pose graph," *Robot. Sci. Syst.*, 2015.

[4] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," in *Proc. 23rd Annu. Conf. Comput. Graph. Interactive Techn.*, 1996, pp. 303–312.

[5] R. A. Newcombe *et al.*, "KinectFusion: Real-time dense surface mapping and tracking," in *Proc. IEEE 10th Int. Symp. Mixed Augmented Reality*, 2011, pp. 127–136.

[6] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger, "Real-time 3D reconstruction at scale using voxel hashing," *ACM Trans. Graph.*, vol. 32, no. 6, pp. 1–11, 2013.

[7] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, "DeepSDF: Learning continuous signed distance functions for shape representation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 165–174.

[8] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger, "Occupancy networks: Learning 3D reconstruction in function space," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 4460–4470.

[9] Y. Yuan, D. Borrmann, J. Hou, Y. Ma, A. Nüchter, and S. Schwertfeger, "Self-supervised point set local descriptors for point cloud registration," *Sensors*, vol. 21, no. 2, 2021, Art. no. 486.

[10] Z. Chen and H. Zhang, "Learning implicit fields for generative shape modeling," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 5939–5948.

[11] R. Chabra *et al.*, "Deep local shapes: Learning local SDF priors for detailed 3D reconstruction," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 608–625.

[12] C. Jiang, A. Sud, A. Makadia, J. Huang, M. Nießner, and T. Funkhouser, "Local implicit grid representations for 3D scenes," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 6001–6010.

[13] J. Huang, S.-S. Huang, H. Song, and S.-M. Hu, "Di-Fusion: Online implicit 3D reconstruction with deep priors," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 8932–8941.

[14] S. Lionar, L. Schmid, C. Cadena, R. Siegwart, and A. Cramariuc, "Neural-blox: Real-time neural representation fusion for robust volumetric mapping," in *Proc. Int. Conf. 3D Vis.*, 2021, pp. 1279–1289.

[15] E. Sucar, S. Liu, J. Ortiz, and A. J. Davison, "IMAP: Implicit mapping and positioning in real-time," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 6229–6238.

[16] Z. Zhu *et al.*, "Nice-SLAM: Neural implicit scalable encoding for SLAM," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 12786–12796.

[17] Y. Yuan and A. Nüchter, "Indirect point cloud registration: Aligning distance fields using a pseudo third point set," *IEEE Robot. Automat. Lett.*, vol. 7, no. 3, pp. 7075–7082, Jul. 2022.

[18] N. Thomas *et al.*, "Tensor field networks: Rotation-and translation-equivariant neural networks for 3D point clouds," 2018, *arXiv:1802.08219*.

[19] F. Fuchs, D. Worrall, V. Fischer, and M. Welling, "Se (3)-transformers: 3D roto-translation equivariant attention networks," *Adv. Neural Inf. Process. Syst.*, vol. 33, pp. 1970–1981, 2020.

[20] C. Deng, O. Litany, Y. Duan, A. Poulenard, A. Tagliasacchi, and L. J. Guibas, "Vector neurons: A general framework for so (3)-equivariant networks," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 12200–12209.

[21] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: An open-source slam system for monocular, stereo, and RGB-D cameras," *IEEE Trans. Robot.*, vol. 33, no. 5, pp. 1255–1262, Oct. 2017.

[22] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3D surface construction algorithm," *ACM siggraph Comput. Graph.*, vol. 21, no. 4, pp. 163–169, 1987.

[23] J. Chibane *et al.*, "Neural unsigned distance fields for implicit function learning," *Adv. Neural Inf. Process. Syst.*, vol. 33, pp. 21638–21652, 2020.

[24] R. Kondor, Z. Lin, and S. Trivedi, "Clebsch–gordan nets: A fully fourier space spherical convolutional neural network," *Adv. Neural Inf. Process. Syst.*, vol. 31, pp. 10117–10126, 2018.

[25] C. Esteves, C. Allen-Blanchette, A. Makadia, and K. Daniilidis, "Learning so (3) equivariant representations with spherical CNNS," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 52–68.

[26] M. Weiler, M. Geiger, M. Welling, W. Boomsma, and T. S. Cohen, "3D steerable CNNS: Learning rotationally equivariant features in volumetric data," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, 2018.

[27] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 652–660.

[28] A. V. Phan, M. Le Nguyen, Y. L. H. Nguyen, and L. T. Bui, "DGCNN: A convolutional neural network over large-scale labeled graphs," *Neural Netw.*, vol. 108, pp. 533–543, 2018.

[29] Y. Aoki, H. Goforth, R.A. Srivatsan, and S. Lucey, "PointNetLK: Robust & efficient point cloud registration using pointnet," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 7163–7172.

[30] A. X. Chang *et al.*, "Shapenet: An information-rich 3D model repository," 2015, *arXiv:1512.03012*.

[31] A. Handa, T. Whelan, J. McDonald, and A. J. Davison, "A benchmark for RGB-D visual odometry, 3D reconstruction and slam," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2014, pp. 1524–1531.

[32] J. Straub *et al.*, "The replica dataset: A digital replica of indoor spaces," 2019, *arXiv:1906.05797*.

[33] M. Garnelo *et al.*, "Conditional neural processes," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1704–1713.

[34] C. Kerl, J. Sturm, and D. Cremers, "Dense visual SLAM for RGB-D cameras," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2013, pp. 2100–2106.

[35] M. Keller, D. Lefloch, M. Lambers, S. Izadi, T. Weyrich, and A. Kolb, "Real-time 3D reconstruction in dynamic scenes using point-based fusion," in *Proc. Int. Conf. 3D Vis.-3DV*, 2013, pp. 1–8.

[36] F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, and W. Burgard, "An evaluation of the RGB-D SLAM system," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2012, pp. 1691–1696.

[37] J. Stückler and S. Behnke, "Multi-resolution surfel maps for efficient dense 3D modeling and tracking," *J. Vis. Commun. Image Representation*, vol. 25, no. 1, pp. 137–147, 2014.

[38] T. Whelan, M. Kaess, H. Johannsson, M. Fallon, J. J. Leonard, and J. McDonald, "Real-time large-scale dense RGB-D SLAM with volumetric fusion," *Int. J. Robot. Res.*, vol. 34, no. 4–5, pp. 598–626, 2015.

[39] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 3354–3361.

[2]https://github.com/gisbi-kim/PyICP-SLAM