IEEE Transactions on Robotics (T-RO) paper presented at
2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

Uni-Fusion: Universal Continuous Mapping

Yijun Yuan and Andreas Nüchter

***Abstract*—We present Uni-Fusion, a universal continuous mapping framework for surfaces, surface properties (color, infrared, etc.) and more (latent features in CLIP embedding space, etc.). We propose the first universal implicit encoding model that supports encoding of both geometry and different types of properties (RGB, infrared, features, etc.) without requiring any training. Based on this, our framework divides the point cloud into regular grid voxels and generates a latent feature in each voxel to form a Latent Implicit Map (LIM) for geometries and arbitrary properties. Then, by fusing a local LIM frame-wisely into a global LIM, an incremental reconstruction is achieved. Encoded with corresponding types of data, our Latent Implicit Map is capable of generating continuous surfaces, surface property fields, surface feature fields, and all other possible options. To demonstrate the capabilities of our model, we implement three applications: (1) incremental reconstruction for surfaces and color (2) 2D-to-3D transfer of fabricated properties (3) open-vocabulary scene understanding by creating a text CLIP feature field on surfaces. We evaluate Uni-Fusion by comparing it in corresponding applications, from which Uni-Fusion shows high-flexibility in various applications while performing best or being competitive. The project page of Uni-Fusion is available at** `https://jarrome.github.io/Uni-Fusion/`.

***Index Terms*—Mapping, RGB-D perception, Semantic scene understanding, Universal mapping**

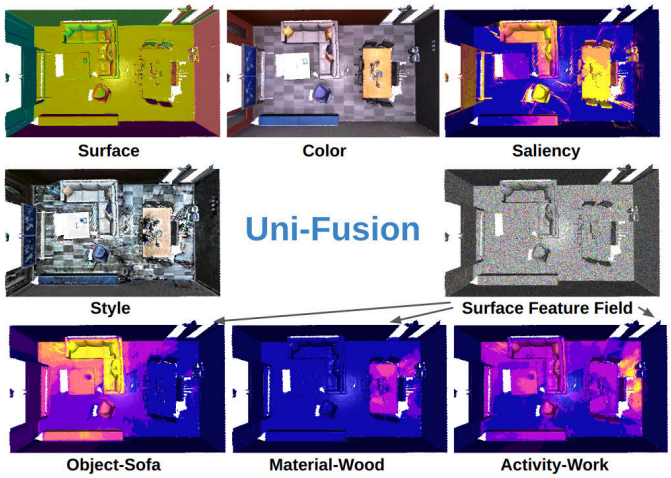


Fig. 1. One Universal Continuous Mapping for all reconstructions. Such as surface, properties including RGB, saliency, style and ..., even high dimensional features (CLIP embeddings, and etc). A rendered video is available on youtube[2].

I. Introduction

In robotics, 3D perception plays an important role in enabling robots to interact with their surroundings. To achieve this, robots must use different types of sensors and employ techniques such as reconstruction and scene understanding. These tasks require the processing of various types of information, including geometry and surface properties. However, algorithms must be designed specifically to handle these different types of data.

Therefore, at the outset, we pose the **following question**: *Is it feasible to handle all these information with a single, universal mapping model?*

Reconstruction, as one of the most prominent topics in the field, has been developing for decades. During this period, numerous works have pushed the limits [1]–[15]. Reconstruction models aim to extract the zero-level surface given a set of points. These approaches are typically based on occupancy grids and signed distance functions (SDFs). Occupancy grids are primarily used in 2D and object-level shape reconstruction, where Gaussian Process Occupancy Maps (GPOM), Gaussian Process Implicit Surface (GPIS), and Hilbert Maps have been developed to generate continuous probabilistic occupancy maps. For scene-level reconstruction, most recent works rely on SDFs. TSDF-Fusion [9], as the most widely used reconstruction model, has facilitated real-time 3D reconstruction. With the rapid development of RGB-D sensors, such as the Kinect and RealSense series, standard models have been invented [10], [11]. These models utilize discretized SDFs and the Marching Cubes algorithm [16] to generate surface meshes.

To address the high memory consumption associated with such a representation, recent techniques rely on deep neural networks to encode the geometry [12], [13], [15], [17]. By using sparse voxels in the whole scene, high-dimensional vectors are extracted for each local geometry. With this representation, researchers have proposed to fuse the map of latent vectors instead of the explicit field. The explicit field is then extracted with arbitrary resolution. Thus, such a form of representation produces a continuous mapping. These methods are called Neural Implicit Maps (NIMs).

Previous studies have demonstrated it with encoding functions pre-trained on object datasets [12], [13], [17]. Pre-training on color or other point properties becomes impractical due to the much higher complexity of the context pattern, compared to shape. Currently, the only recent solutions for continuous color reconstruction are based on back-propagation to update local latent features [15] or Neural Radiance Fields (NeRF) [18]. However, these methods require a significant number of training epochs and are not suitable for real-time applications. Therefore, this paper aims to fill the gap by

The authors are with Informatics XVII – Robotics at Julius-Maximilians-University of Würzburg, Germany. {yijun.yuan|andreas.nuechter}@uni-wuerzburg.de

This work was in parts supported by the Federal Ministry for Economic Affairs and Climate Action (BMWK) on the basis of a decision by the German Bundestag und the grant number KK5150104GM1. We also acknowledge the support by the Elite Network Bavaria (ENB) through the "Satellite Technology" academic program.

[2]https://www.youtube.com/watch?v=4Z-u7yU2ARU

introducing a universal model that directly encodes arbitrary properties without the need for time-consuming learning or training.

Our model uses Gaussian Process Regression (GPR) as its basis. Firstly, we propose to decouple the GPR by utilizing the approximation of the kernel function. Which supports the encoding of the point cluster into a single feature vector. Then, by leveraging the sparse voxel structure, we construct a local feature vector within each voxel to form a map of latents. Since our encoding-decoding function uses GPR, the entire model does not pre-touch any format of data properties. Therefore, our Uni-Fusion is applicable to arbitrary reconstruction applications.

We believe that there dose not exist such a model that can handle every aspect of robot perception. Therefore, based on this encoder, we introduce Uni-Fusion, a universal model for all types of data that generates continuous maps. A selected set of examples of "what Uni-Fusion can do" is presented in Fig. 1. With various input data, our Uni-Fusion model efficiently encodes and generates continuous maps for surfaces, colors, styles, and more. To further explore the potential of this model, we even construct a field for high-dimensional CLIP embedding [19].

The contributions of this work are as follows:

- we propose a universal encoding-decoding model for local information that does not require any training,
- we present the first universal continuous mapping model, capable of constructing continuous surfaces and various surface property fields, including high-dimensional features such as CLIP embeddings.
- we implement applications to demonstrate the capabilities of our proposed model, including: (1) incremental surface & color reconstruction, (2) 2D-to-3D transfer on a 3D canvas, (3) open-vocabulary scene understanding.

Content Overview: In Section II, we provide an overview of related works. Then, in Section III and Section IV, we present the Universal Encoder and the Uni-Fusion model, a universal continuous mapping framework built upon that encoder. Afterwards, in Section V, we showcase the broad applicability of Uni-Fusion by presenting a number of applications in different scenarios. The capabilities of Uni-Fusion are evaluated in Section VI. Finally, we outline the future directions and conclude this paper.

II. Related Works

We first discuss the development of continuous mapping from 2D scenes to 3D scenes. Then we explore the importance of recent state-of-the-art (SOTA) reconstructions utilizing neural implicit models. Next, we examine the development of kernel methods that are closely related to our approach.

A. Continuous Mapping

The practice of continuous mapping originated in 2D scenarios. Gaussian Process Occupancy Maps (GPOM) [1] use Gaussian Process Regression (GPR) to predict a continuous representation, allowing the construction of maps at arbitrary resolutions. To improve the scalability, Kim et al. [2] employ a divide and conquer strategy where GP is applied in each cluster. Then, the incremental approach to GPOM is employed by leveraging the Bayesian Committee Machine (BCM) technique [3], [4].

Meanwhile, the Hilbert Maps approach [20], [21] has been introduced. This approach is a mapping technique that does not require an explicit formulation. Hilbert Maps achieve continuous mapping by continuously optimizing parameters.

Due to the growing popularity of 3D sensors, there has been a shift in focus towards 3D scenarios. Gaussian Process Implicit Surface (GPIS), has emerged in the field [5]–[8]. These methods, given zero-level surface points, either sample points along the normal direction and assign distance values, or directly use derivative models with normals as labels (we will discuss this in more detail in our mapping Section IV-B). However, these methods focus primarily on shapes rather than entire scenes. This is due to the fact that GPR-based methods naturally incur high computational costs when dealing with large amounts of data. This is especially true when moving from 2D to 3D testing scenarios. Therefore, we leverage the concept of Neural Implicit Maps (Section II-B) to address this challenge and focus on scene reconstruction.

B. Neural implicit based Reconstruction

Neural implicit reconstruction was originally introduced for SDF and occupancy-based reconstruction. The seminal work by Park et al. [22], known as DeepSDF, employs a deep model to encode geometry prior using multi-layer perceptrons (MLPs) and extracts discretized SDF through decoding queries. It then uses an algorithm similar to Marching Cubes [16] for mesh extraction. For the latter, Occupancy Networks [23] learn to estimate the occupancy probability at positions using an implicit function. The Multiresolution IsoSurface Extraction (MISE) technique is used to generate meshes.

To enhance the efficiency of the reconstruction, DeepLS [24] uses multiple local deep priors and reconstructs based on a set of local SDFs. Jiang et al. [25] additionally proposes the use of a local implicit grid to further simplify the model complexity of the encoding. Conversely, Convolutional Occupancy Networks [26] explore the local latent by replacing

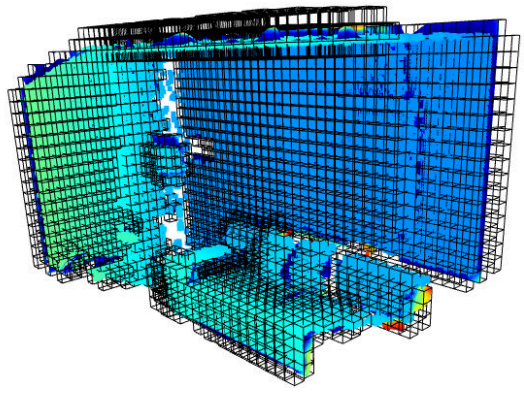


Fig. 2. Uniform, sparse voxels. Each voxel is encoded into one feature vector $\mathbf{F}_m$ [12].

the encoder-decoder with optimization on a grid of local features, thereby alleviating the burden of MLPs.

More recently, Neural Implicit Maps (NIM) have achieved real-time 3D reconstruction of real scenes. Huang et al. propose DI-Fusion [12], a neural implicit mapping method that performs incremental scene reconstruction. DI-Fusion achieves high memory efficiency and produces improved reconstructions, by fusing on maps of latent features. Similarly, NeuralBlox [27] fuses a grid of latent features with known external pose estimation. BNV-Fusion [17] further improves feature quality by incorporating post-fusion optimization. To address large scene reconstructions that may involve loops, NIM-REM [13] introduces an SE(3)-transformation algorithm for NIM and develops a mapping&remapping model that works in conjunction with bundle adjustment and loop closing. Thinking outside the box, Sucar et al. [14] propose iMAP, a novel SLAM pipeline that incorporates neural implicit-based incremental reconstruction. Using a differentiable rendering model, iMAP performs online optimization of the reconstruction by minimizing the image distances. Zhu et al. present NICE-SLAM [15], which replaces MLPs with a convolutional occupancy grid to further improve the efficiency and quality of the reconstruction.

In this work, our model also uses regularly spaced voxels, as shown in Fig. 2, for local encoding-decoding. Since we propose to use a single model for all properties, pre-training such a model is not feasible. Therefore, we employ a kernel method to achieve this objective.

C. Kernel Function Approximation

Kernel methods suffer from a scalability issue due to the $\mathcal{O}(n^3)$ time complexity required during regression. One possible solution is kernel matrix approximation, but this is beyond the scope of this paper. Another solution, kernel function approximation, aims to enhance the scalability of kernel methods by employing explicit vector maps. For example, kernel function $k(\mathbf{x}_1, \mathbf{x}_2) \approx v(\mathbf{x}_1)^T v(\mathbf{x}_2)$ with mapping function $v : \mathbb{X} \to \mathbb{R}^l$ [28]. Two approaches are commonly used for approximation: Random Fourier Features (RFFs) [29]–[33] and Nyström methods [28], [33], [34].

RFFs explicitly handle the shift-invariant kernels by mapping the data using the Fourier transform technique [33]. However, RFFs are primarily employed for shift-invariant kernels and require a large l for their operation. Furthermore, since RFFs are data-independent, they exhibit significantly worse generalization performance compared to Nyström methods [35]. In contrast, Nyström methods can approximate any positive definite kernel [28]. It relies on finding the eigenfunctions to form the approximation:

$$k(\mathbf{x}_1, \mathbf{x}_2) = \sum_{i \geq 1} \mu_i \varphi_i(\mathbf{x}_1) \varphi_i(\mathbf{x}_2),$$

where φ_i and $\mu_i \geq 0$ are eigenfunctions and eigenvalues of kernel function k with respect to the probability measure q. With the top-l eigenpairs, the Nyström method approximates the kernel function with $k(\mathbf{x}_1, \mathbf{x}_2) = \sum_{i \geq 1}^{l} \mu_i \varphi_i(\mathbf{x}_1) \varphi_i(\mathbf{x}_2)$.

However, Nyström methods are computationally expensive for medium-sized training data and require evaluating each sample N times, which is inefficient for direct regression of a continuous map.

Instead, we encode the local geometry and properties, eliminating the need for a full-space approximation. By restricting the approximation to the limited space $\mathbf{x} \in [-0.5, 0.5]^3$, and applying the function in each local region, we reduce the computational burden.

III. Decoupled Regression as an Universal Encoding

In this section, we propose a universal encoding model for point clouds. Building on this encoder, in Section IV, we introduce Universal Continuous Mapping.

A. Decoupled Gaussian Process Regression as Encoder-Decoder

Inspired by DI-Fusion [12], which introduces latent feature maps for continuous surface prediction, our goal is to generate a latent vector for each local patch. First, a universal encoder design comes from the Gaussian Process Regression (GPR), which is a widely used technique for low-dimensional regression. It has been used in various mapping models [4], [5]. Given a set of N observation points $\{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^{N}$ where point positions are $\mathbf{x}_n \in \mathbb{R}^d$ ($d = 3$ in this paper) and point properties are $\mathbf{y}_n \in \mathbb{R}^c$, GPR is used to regress a function f that best explains the data. The N points are aggregated in the $N \times d$ matrix $\mathbf{X}$, and the targets are collected in the $N \times c$ matrix $\mathbf{Y}$.

The Gaussian process assumes that the data is sampled from a multivariate Gaussian distribution, i.e.,

$$\begin{bmatrix} \mathbf{Y} \\ \mathbf{Y}_* \end{bmatrix} \sim \mathcal{N}(\mathbf{0}, \begin{bmatrix} k(\mathbf{X}, \mathbf{X}) & k(\mathbf{X}, \mathbf{X}_*) \\ k(\mathbf{X}_*, \mathbf{X}) & k(\mathbf{X}_*, \mathbf{X}_*) \end{bmatrix}). \tag{1}$$

where $(\mathbf{X}, \mathbf{Y})$ and $(\mathbf{X}_*, \mathbf{Y}_*)$ represent the observation and inference pairs. For simplicity, we denote the matrix $\mathbf{K} = k(\mathbf{X}, \mathbf{X})$, $\mathbf{K}_* = k(\mathbf{X}, \mathbf{X}_*)$, $\mathbf{K}_{**} = k(\mathbf{X}_*, \mathbf{X}_*)$.

With the derivative in [36], we obtain

$$\mathbf{Y}_* | \mathbf{X}, \mathbf{Y}, \mathbf{X}_* \sim \mathcal{N}(\mathbf{K}_*^T \mathbf{K}^{-1} \mathbf{Y}, \mathbf{K}_{**} - \mathbf{K}_*^T \mathbf{K}^{-1} \mathbf{K}_*). \tag{2}$$

When an additional Gaussian error is introduced as $\mathbf{y} = f(\mathbf{x}) + \epsilon$, the covariance of $\mathbf{X}$ is rewritten as $\mathbf{K} + \delta_n^2 \mathbf{I}$. The regression result is typically considered to be the mean

$$\mathbf{Y}_* = \mathbf{K}_*^T (\mathbf{K} + \delta_n^2 \mathbf{I})^{-1} \mathbf{Y}. \tag{3}$$

This well illustrates the challenge of using Gaussian process regression directly in large-scale reconstructions due to its $\mathcal{O}(n^3)$ time complexity, which is not feasible. Furthermore, the formula (Eq. (3)) is impractical since it requires the large input point cloud data to be maintained for the $\mathbf{K}_*$ computation.

To address the issue of high complexity and avoid the need to maintain the entire point cloud, we propose to decouple the GPR to obtain the low-dimensional latent vectors for local regions. The decoupling process involves approximating the kernel function as

$$k(\mathbf{X}, \mathbf{X}_*) \approx f_{posi}(\mathbf{X}) f_{\text{posi}}(\mathbf{X}_*)^T \tag{4}$$

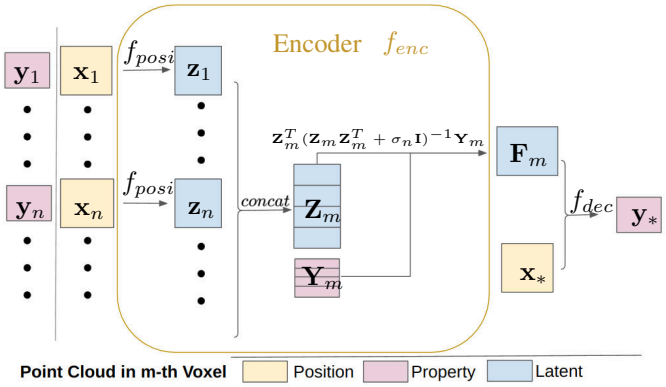


Fig. 3. Interpreting formula with a graph that is coherent to the Encoder-decoder structure in Neural Implicit Maps [12].

where $f_{\text{posi}} : \mathbb{R}^3 \to \mathbb{R}^l$. We refer to the function f_{posi} as the position encoding function, consistent with the Neural Implicit Maps model, Di-Fusion [12]. Thus, Eq. (3) is rewritten as

$$\mathbf{y}_* = f_{\text{posi}}(\mathbf{X}_*) f_{\text{enc}}(\mathbf{X}, \mathbf{Y}) \tag{5}$$

where the content encoder function is

$$f_{\text{enc}}(\mathbf{X}, \mathbf{Y}) = f_{\text{posi}}(\mathbf{X})^T (f_{\text{posi}}(\mathbf{X}) f_{\text{posi}}(\mathbf{X})^T + \delta_n^2 \mathbf{I})^{-1} \mathbf{Y} \in \mathbb{R}^{l \times c}. \tag{6}$$

The encoded feature is denoted by $\mathbf{F}_{(\mathbf{X},\mathbf{Y})} = f_{enc}(\mathbf{X}, \mathbf{Y}) \in \mathbb{R}^{l \times c}$. which serves as the basis for the construction of latent maps in the subsequent universal continuous mapping model (Section IV). Specifically, for geometry encoding we set $l = 20$ and $c = 1$, resulting in a 20-dimensional vector feature. Similarly, for color encoding, we have $l = 20$ and $c = 3$.

The decoding value for the inferred point $\mathbf{x}_*$ is expressed as

$$f_{dec}(\mathbf{x}_*, \mathbf{F}_{(\mathbf{X},\mathbf{Y})}) = f_{posi}(\mathbf{x}_*) \mathbf{F}_{(\mathbf{X},\mathbf{Y})}. \tag{7}$$

Thus, a signed distance field or surface property field is approached.

In the following we derive the approximation function.

B. Position Encoding with Approximated Kernel Function

Considering that our mapping needs to encode the local geometry & property, the encoding function only requires to touch points in a limited region ($[-.5, .5]^3$ in our case).

As we discussed in related work, Nyström methods offer greater accuracy than RFFs, because they depend on the given points. This property is well-suited to our application.

The Nyström method for kernel approximation begins with the use of eigenfunctions according to Mercer's theorem:

$$k(\mathbf{x}_1, \mathbf{x}_2) = \sum_{i \geq 1} \mu_i \psi_i(\mathbf{x}_1)^T \psi_i(\mathbf{x}_2) \tag{8}$$

where ψ_i and $\mu_i \geq 0$ are eigenfunctions and eigenvalues of kernel function k with respect to the probability measure q.

Given a set of anchor samples $\hat{\mathbf{X}} = \{\hat{\mathbf{x}}_1, \cdots, \hat{\mathbf{x}}_N\}$, we perform eigen-decomposition on the matrix $k(\hat{\mathbf{X}}, \hat{\mathbf{X}})$ to obtain its eigenpairs $\{(\lambda_i, \mathbf{u}_i)\}_{i \in \{1,\cdots,l\}}$ with rank l.

Subsequently, Nyström method produces

$$\psi_i(\mathbf{x}) = \sum_n k(\mathbf{x}, \hat{\mathbf{x}}_n) \mathbf{u}_{i,n}, i = 1, \cdots, l. \tag{9}$$

To simplify, we express the eigenfunction as

$$\psi_i(\mathbf{x}) = k(\mathbf{x}, \hat{\mathbf{X}}) \mathbf{u}_i. \tag{10}$$

Similarly, the eigenvalue is written as $\mu_i = \frac{1}{\lambda_i}$.

For clarity, we introduce the notation $\boldsymbol{\mu} = [\mu_1, \cdots, \mu_l]$ and $\Psi = [\psi_1, \cdots, \psi_l]^T$ based on Eq. (8) where $\Psi : \mathbb{R}^3 \to \mathbb{R}^l$.

To maintain consistency with Eq. (4), we set

$$f_{posi}(\mathbf{x}) = diag(\sqrt{\boldsymbol{\mu}}) \Psi(\mathbf{x}) \tag{11}$$

where $diag$ produces diagonal matrix. f_{posi} above refers to the position encoder in Eq. (6).

In this paper, we employ the Matérn kernel function [37][3]:

$$k(\mathbf{x}_1, \mathbf{x}_2) = \sigma^2 \frac{2^{1-\nu}}{\Gamma(\nu)} (\sqrt{2\nu} \frac{dist(\mathbf{x}_1, \mathbf{x}_2)}{\rho}) K_\nu (\sqrt{2\nu} \frac{dist(\mathbf{x}_1, \mathbf{x}_2)}{\rho}) \tag{12}$$

where Γ presents the gamma function, K_ν is the modified Bessel function of the second kind, $dist$ denotes the Euclidean distance, and σ and ρ are hyperparameters of the kernel function. We utilize the half integer $\nu = 3 + \frac{1}{2}$, which results in the specific function:

$$k(d) = \sigma^2 (1 + \frac{\sqrt{7}d}{\rho} + \frac{2}{5} (\frac{\sqrt{7}d}{\rho})^2 + \frac{1}{15} (\frac{\sqrt{7}d}{\rho})^3) exp(-\frac{\sqrt{7}d}{\rho}) \tag{13}$$

where $d = dist(\mathbf{x}_1, \mathbf{x}_2)$ for short.

To approximate the above kernel function, anchor points are required. We sample $N_a = 256$ points uniformly from $[-.5, .5]^3$ cube (as $\hat{\mathbf{X}}$) to compute the kernel matrix $\mathbf{K}_a = k(\hat{\mathbf{X}}, \hat{\mathbf{X}})$. Subsequently, we perform an eigendecomposition on this kernel matrix, resulting in $\mathbf{K}_a = \mathbf{U} \Lambda \mathbf{U}^T$. Lastly, $\mathbf{U}$ and Λ are utilized in Eq. (11) and Eq. (4) to form the kernel function approximation.

It is important to note that the dimension of the encoded feature, l, used in Section III-A, depends on $\mathbf{U}$. It further determines the size of the map in the next section (Section IV).

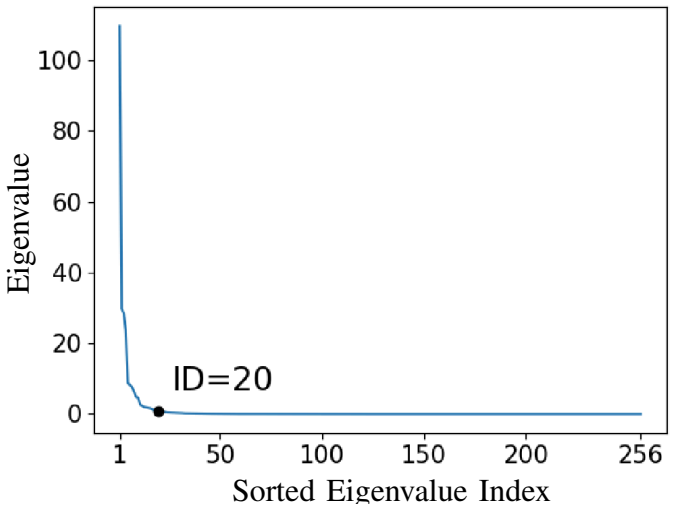


Fig. 4. Sorted eigenvalues for $\mathbf{K}_a$'s eigendecomposition.

The eigenvalues of Λ are plotted in Fig. 4, revealing that the matrix is primarily influenced by a small number of pairs

[3]https://en.wikipedia.org/wiki/Mat\'ern_covariance_function

with significant eigenvalues. Most of the eigenvalues are less than 1. Therefore, we choose $l = 20$ which is about 0.8 in this plot to approximate the kernel while maintaining a compact feature dimension.

Note that we perform a single sampling and decomposition step. Subsequently, the encoding-decoding process in Fig. 3 only requires loading and reusing the parameters for f_{posi}, f_{enc} and f_{dec}. In contrast, related works often involve pre-training on large datasets of objects [12], [17] or indoor scenes [15]. For our model, however, **no training is needed**.

IV. Universal Continuous Mapping

The previous Section III suggests a universal encoding model for different types of data. Based on this function, in this section, our Universal Continuous Mapping produces a map of latents to implicitly represent the scene. We refer to this scene representation as **Latent Implicit Maps (LIM)**, which supports surfaces, surface properties, and high-dimensional surface features.

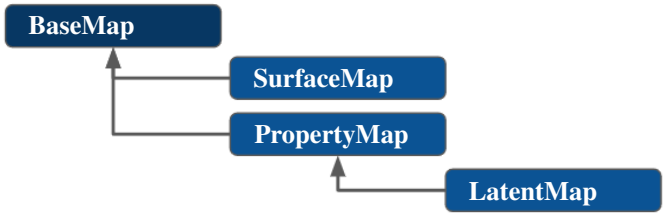


Fig. 5. Inheritance graph for the class of Latent Implicit Maps (LIM).

The inheritance graph is shown in Fig. 5. We introduce a BaseMap to handle the voxel structure (Section IV-A), dynamically allocate space and fuse maps (Section IV-E). The SurfaceMap (Section IV-B) and the PropertyMap (Section IV-C) are derived from the BaseMap and are designed to handle specific data and applications. The LatentMap (Section IV-D) is derived from PropertyMap. The main difference is that the PropertyMap works primarily with low-dimensional properties, such as color ($c = 3$), infrared ($c = 1$) data, and so on. On the other hand, the LatentMap handles much higher dimensional features, such as CLIP embeddings [19] ($c = 768$), depending on the specific application requirements.

A. Map Representation

We follow Neural Implicit Maps ([12], [13], [17]) to use uniformly spaced voxels to sparsely represent the scene. The scene is denoted as $\mathbf{V} = \{\mathbf{v}_m = (\mathbf{c}_m, \mathbf{F}_m, w_m)\}$, where m is the voxel index. Each voxel $\mathbf{v}_m$ comprises the voxel center $\mathbf{c}_m \in \mathbb{R}^3$, voxel latent feature $\mathbf{F}_m \in \mathbb{R}^{l\times c}$, and the count of observed points $w_m \in \mathbb{N}$.

Given a sequence of incremental frames as input, our model constructs local LIMs (Section IV-B, Section IV-C, Section IV-D) and fuses (Section IV-E) them into a global LIM. Then, we derive the explicit map from the global LIM.

B. Surface Mapping

Because the input point cloud $\mathbf{X}$ is located on zero-level surface, it is not adequate to recover a 3D field of scene, $f_{SDF} : \mathbb{R}^3 \to \mathbb{R}$. Therefore, we use the concept of Gaussian Process Implicit Surfaces (GPIS) [5]–[8] to incorporate derivatives into kernel or to sample additional non-zero level points. Both derivative-based and sample-based GPISs approaches utilize normal information. Hence, we first preprocess $\mathbf{X}$ to obtain normals $\mathbf{S}$.

1) Using Derivatives based GPIS: From [5], the derivatives of a GP are also Gaussian. Therefore, the covariance between data and derivatives is computed by differentiating the covariance function [38]. Specifically:

$$cov(\frac{\partial f_{SDF}(\mathbf{x})}{\partial x_i}, f_{SDF}(\mathbf{x}^{'})) = \frac{\partial k(\mathbf{x}, \mathbf{x}^{'})}{\partial x_i} = \frac{\partial}{\partial x_i}[f_{posi}(\mathbf{x})] f_{posi}(\mathbf{x}^{'})^T. \tag{14}$$

Additionally,

$$cov(\frac{\partial f_{SDF}(\mathbf{x})}{\partial x_i}, \frac{\partial f_{SDF}(\mathbf{x}^{'})}{\partial x_j}) = \frac{\partial^2 k(\mathbf{x}, \mathbf{x}^{'})}{\partial x_i \partial x_j^{'}} = \frac{\partial}{\partial x_i}[f_{posi}(\mathbf{x})] \frac{\partial}{\partial x_j^{'}}[f_{posi}(\mathbf{x}^{'})]^T. \tag{15}$$

Given points $\{\mathbf{x}_n\}_{n=1}^N$ with normals $\{\mathbf{s}_n\}_{n=1}^N$ and field values $\{\mathbf{y}_n = 0\}_{n=1}^N$, the position encoding function for derivatives is $f_{posi,deri}(\mathbf{x}, i) = \frac{\partial}{\partial x_i}[f_{posi}(\mathbf{x})]$. Its corresponding field value is the normal value s_i on the axis i. Therefore, we define

$$f_{posi,gpis}(\mathbf{X}) = [f_{posi}(\mathbf{X}), f_{posi,deri}(\mathbf{X}, 1), f_{posi,deri}(\mathbf{X}, 2), f_{posi,deri}(\mathbf{X}, 3)] \tag{16}$$

with regression values $\mathbf{Y}_{gpis} = [\mathbf{0}, \mathbf{s}_{\cdot,1}, \mathbf{s}_{\cdot,2}, \mathbf{s}_{\cdot,3}]^T$, where $\mathbf{0} = zeros(1, N)$, $\mathbf{s}_{\cdot,i} = [s_{1,i}, \cdots, s_{N,i}]$.

Then the local geometric encoding function is defined as

$$f_{enc,gpis}(\mathbf{X}, \mathbf{Y}, \mathbf{S}) = f_{posi,gpis}(\mathbf{X})^T (f_{posi,gpis}(\mathbf{X}) f_{posi,gpis}(\mathbf{X})^T + \sigma_n^2 \mathbf{I})^{-1} \mathbf{Y}_{gpis}. \tag{17}$$

By introducing derivatives into the kernels, the matrix size is increased by a factor of 15, while the encoded feature dimension remains low at l: $\mathbf{F}_{(\mathbf{X},\mathbf{Y},\mathbf{S})} = f_{enc,gpis}(\mathbf{X}, \mathbf{Y}, \mathbf{S}) \in \mathbb{R}^{l\times 1}$.

For inference with the points $\mathbf{x}_*$, the predictions are consistent with Eq. (7), $\mathbf{y}_* = f_{posi}(\mathbf{x}_*)\mathbf{F}_{(\mathbf{X},\mathbf{Y},\mathbf{S})}$.

2) Using Sample based GPIS: Sample-based GPIS is commonly used in GPIS research. This method avoids the computation of Jacobians and allows for smaller kernel sizes, resulting in significant reductions in computational costs in terms of time and memory.

In sample-based GPIS, given points $\{\mathbf{x}_n\}_{n=1}^N$ with normals $\{\mathbf{s}_n\}_{n=1}^N$ and field value $\{\mathbf{y}_n = 0\}_{n=1}^N$, the dataset is extended by sampling points along the normal direction. The corresponding field values are the signed distances as the sampled points move along the normal. Then, Eq. (6) is applied to the

extended points and distances ($\mathbf{X}_{ext}$, $\mathbf{Y}_{ext}$). The inference process of this model is the same as derivative-based GPIS in Section IV-B1.

For each frame, points are assigned to their corresponding voxel. Then, we encode the local geometry within each voxel using methods described in Section IV-B, either derivative-based GPIS or sample-based GPIS to obtain the voxel representation $\mathbf{v} = (\mathbf{c}, \mathbf{F}, w)$ where $\mathbf{F} \in \mathbb{R}^{l\times 1}$ represents the geometric latent vector. Subsequently, the local LIMs are fused into a global LIM according to the fusion procedure outlined in Section IV-E.

To visualize the surface result, we construct the signed distance field from the global LIM by performing inference on sample points. The sample points are generated on a grid within each voxel, with a certain resolution. By applying the Marching Cubes algorithm to the SDF, we obtain a surface mesh that represents the reconstructed surface.

C. Surface Property Fields

The previous surface mapping approach discussed in Section IV-B can be viewed as a special case of surface property mapping. However, it is important to note that these two mappings operate in different spaces. Specifically, in our implementation, we do not derive the SurfaceMap class from PropertyMap. Instead, we introduce a BaseMap that performs common operations and allows them to be specific to local map construction and visualization, e.g., meshing and coloring.

We introduce the more general mapping of surface properties. Since all points lie on the zero level of the signed distance fields, the PropertyMap naturally operates within a subspace of $\mathbb{R}^3$, the surface $\mathcal{S}$. A surface property in this context refers to any property associated with each point, such as color, infrared values, and so on. These properties are represented by the $\mathbf{y}$ value in the encoder diagram shown in Fig. 3, with a dimensionality of c.

As an example, we take the most commonly used surface property, color. Given an observed colored point cloud $\{(\mathbf{x}_n, \mathbf{q}_n)\}_{n=1}^N$ as input, where $\mathbf{q}_n$ denote the RGB color values. The corresponding surface property values are $\{\mathbf{y}_n = \mathbf{q}_n\}_{n=1}^N$. We aggregate these values into two $N \times 3$ matrices $\mathbf{X}$ and $\mathbf{Q}$. Therefore, the encoded feature for this point cloud is obtained using the following equation:

$$\mathbf{F}_{color} = f_{posi}(\mathbf{X})^T (f_{posi}(\mathbf{X}) f_{posi}(\mathbf{X})^T + \delta_n^2 \mathbf{I})^{-1} \mathbf{Q}. \quad (18)$$

Here, $\mathbf{F}_{color} \in \mathbb{R}^{l\times 3}$ represents the color feature.

Since we use $l = 20$ for the approximation function in our experiments, the color map only needs to store 20×3 float values in each voxel to represent a continuous color field. It is important to note that our model requires no training and can be applied directly to different types of data. During inference, since the field is in the surface space $\mathcal{S}$, we sample points $\mathbf{x}_*$ at arbitrary resolutions, either from a known mesh or a surface constructed from previous surface mapping (Section IV-B). Following Eq. (7), the inference point $\mathbf{x}_*$ is position encoded and multiplied by the color feature $\mathbf{F}_{color,m}$ in the corresponding voxel m to obtain its value $\mathbf{q}_* = f_{dec}(\mathbf{x}_*, \mathbf{F}_{color,m})$.

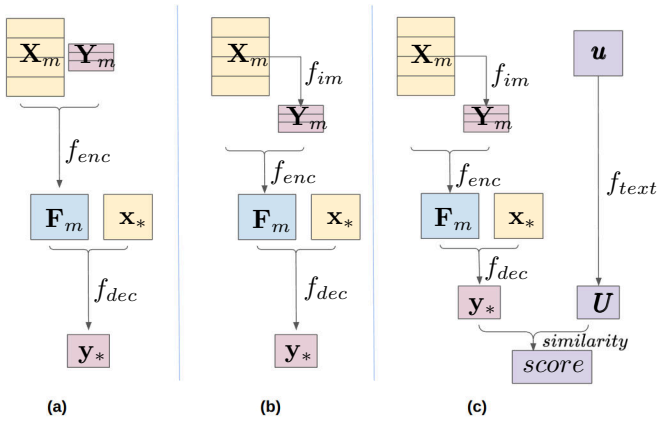


Fig. 6. Encoding-decoding diagram in various applications. (a) fold applications obtain point properties ($\mathbf{Y}_\mathrm{m}$) directly from the sensor. (b) fold applications derive point properties using a function f_{im} that captures style, saliency and etc. (c) fold applications utilize feature as $\mathbf{Y}_\mathrm{m}$ to construct a LIM for a (CLIP) feature field. Then, a text command is used to extract the semantic information.

D. Surface Feature Fields

Surface feature fields represent an extension of the previous surface property fields discussed in Section IV-C. In this extension, we broaden the scope of surface properties to include features. This demonstrates the versatility of our mapping model, as it is applied directly without the need of any training.

We start by considering an embedding function $f_{im} : \mathbb{R}^{N\times 3} \rightarrow \mathbb{R}^{N\times c}$ that processes the input data $\mathbf{X}$. We treat the feature of each point as a surface property, where $\{\mathbf{y}_n = f_{im}(\mathbf{X})_{\mathbf{x}_n} \in \mathbb{R}^{l\times c}\}_{n=1}^N$.

Following the encoding and fusion steps described in Section IV-C and Section IV-E, we construct latent implicit maps for the surface feature fields. As a result, we extract maps of features at arbitrary resolutions using the function $f_{dec}(\cdot, \mathbf{F}_\cdot)$.

We illustrate an application in Section V-C, specifically in the context of open-vocabulary scene understanding. In this application, our model constructs a CLIP space feature field on the surface, enabling it to respond to textual input. The key difference compared to surface property fields is demonstrated in Fig. 6. In Fig. 6(c), a CLIP text encoder f_{text} is added to encode the text command $\boldsymbol{u}$ into the CLIP feature $\boldsymbol{U}$. By leveraging the surface field for CLIP embeddings generated by the left branch, our model identifies the desired region by computing the similarity between features.

E. Map Fusion

We adopt the voxel-to-voxel fusion approach from Neural Implicit Maps [12] to update the LIM. The fusion operation is performed as follows:

$$\mathbf{F}_m \leftarrow \frac{\mathbf{F}_m w_m + \mathbf{F}_m^{'} w_m^{'}}{w_m + w_m^{'}}, w_m \leftarrow w_m + w_m^{'}, \quad (19)$$

where $\mathbf{v}_m = (\mathbf{c}_m, \mathbf{F}_m, w_m)$ represents the voxel m from the global LIM, and $\mathbf{v}_m^{'} = (\mathbf{c}_m^{'}, \mathbf{F}_m^{'}, w_m^{'})$ represents the voxel m from the local LIM.

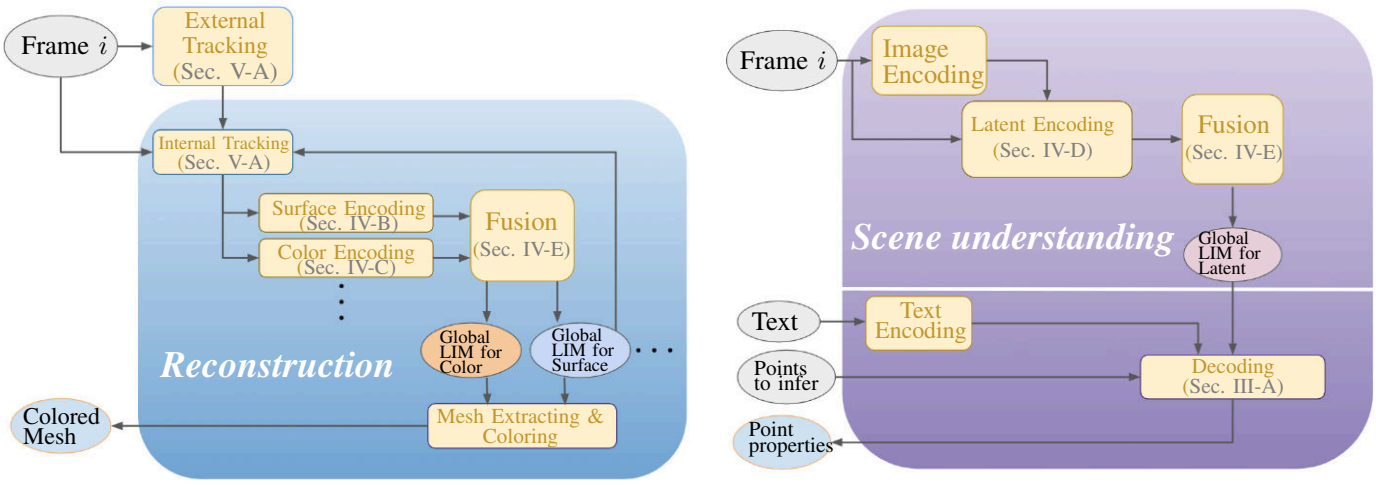


Fig. 7. Reconstruction and scene understanding applications' pipeline. On the left incremental reconstruction application, external tracking runs in parallel to reconstruction to provide coarse poses. While doing reconstruction, internal tracking refines the pose estimation fur a better surface fit. "$\cdots$" means that we can add more other properties from Section V-A2 and Section V-B into this pipeline. On the right scene understanding application, we assume that the pose of the frame is pre-known. The upper part of the white line is the fusion of LIM for the feature field. The lower part infers specific semantic information along with the text command.

V. Applications

To demonstrate the wide range of applications of our model, we have implemented the following series of applications:

1) Incremental surface & color reconstruction
2) 3D saliency detection
3) Open-vocabulary scene understanding
4) Surface infrared field
5) 3D style transfer

Starting from our motivation in inspection and service robotics, we implement 1) incremental surface & color reconstruction to visualize the robot environment. For robot exploration, we implement 2) 3D saliency detection to indicate the salient regions in maps. To recover object-level semantic information in the environment, we implement 3) open-vocabulary scene understanding to yield the regions containing the objects. Furthermore, to demonstrate the flexibility, we implement 4) surface infrared fields and 5) 3D style transfer for artistic purposes.

In Fig. 6, we classify these applications into 3 categories: (a) obtains properties directly from sensor observations, such as applications 1) and 4). (b) processes sensor data and predicts properties, such as applications 2), 5). (c) extends (b) by operating on high dimensional features, specifically application 3).

Applications 1) and 4) belong to the first category. We primarily describe 1) incremental surface & color reconstruction (Section V-A), while for 4) we can easily replace color with infrared. For the second with 2) and 5) in Section V-B, we mainly discuss the usage of fabricated properties and do not provide detailed explanations of the mapping part, as it is already covered in the previous category. Finally, we delve into the third category with application 3), which involves mapping a LIM for high-dimensional latent fields. We demonstrate the flexibility and capability of this application in Section V-C.

A. Application: Incremental Reconstruction

In this section, we present an application of incremental 3D reconstruction using RGB-D sequences. Since RGB-D sequences provide both point positions and color values, it allows us to construct two types of LIMs: one for surface (Section IV-B), and one for color (Section IV-C).

The pipeline is illustrated in Fig. 7. When an RGB-D frame i is fed into the framework, it is firstly converted into a colored point cloud ($\mathbf{X} \in \mathbb{R}^{N\times 3}, \mathbf{Q} \in \mathbb{R}^{N\times 3}$). The tracking module takes $(\mathbf{X}, \mathbf{Q})$ to estimate the current pose $\mathbf{T}$. Next, the transformed point cloud $\mathbf{X}\mathbf{T}^T$ is used as input to the surface mapping (Section IV-B), while the colored point cloud $(\mathbf{X}\mathbf{T}^T, \mathbf{Q})$ is used as input to the surface color mapping (Section IV-C), resulting in the generation of local LIMs. Using the fusion operation in Eq. (19), the local LIM is fused into the global LIM on a voxel by voxel basis.

For visualization purposes, we first sample a grid within each voxel and infer using the global surface LIM to obtain the SDF. The Marching Cube algorithm is then applied to extract the mesh.

Once the surface is reconstructed, we can sample points from it at arbitrary resolution and perform inference using the global color LIM to reconstruct the surface color.

1) Tracking: According to Zhu et al. [15], the current implicit scene representation-based tracking models, such as iMAP, DI-Fusion and NICE-SLAM, still have a performance gap compared to state-of-the-art tracking approaches such as BAD-SLAM and ORB-SLAM2. Therefore, instead of following the neural implicit models to track with frame-to-model or ray-tracing based optimization, we incorporate ORB-SLAM2 in a separate thread to provide a pose prior. Hence, we call this external tracking.

Note that the primary focus of ORB-SLAM2 is on localization not scene reconstruction. This means that direct use of ORB-SLAM2 provides coarse surface reconstruction. We

further use colored point cloud registration (CPCR) [39] as a tracking refiner.

In our implementation, ORB-SLAM2 runs independently over all frames. Every few frames, CPCR tracks an initially posed colored point cloud to compute the odometry within a local window. Mapping is then done in the same thread. Therefore, the latter is called internal tracking.

2) Other types of datas: Application 4) uses a point cloud with infrared information, which is a straightforward modification of the color-based approach. LIM feature dimension is also correspondingly reduced. This flexibility allows different types of point cloud properties to be integrated into the continuous mapping pipeline.

B. Application: 2D-to-3D Transfer

Applications such as 2) and 4) can be easily integrated with application 1) incremental reconstruction (Section V-A) by incorporating the fabricated result together with the point cloud. For instance, given RGB-D frames, we detect saliency or transfer image styles to generate a fabricated X image. Here, X represents saliency, style, or other properties. By combining X with depth information through unprojection, we assign the fabricated values to corresponding points, resulting in point pairs ($\mathbf{X}$, $\mathbf{Q}_X$).

Similar to the reconstruction pipeline in Fig. 7, we employ encoding (Section III-A) and fusion (Eq. (19)) to construct a global LIM for the fabricated properties X. This global LIM represents a surface X fields that is utilized for subsequent inference.

While it is possible to similarly transfer a 2D semantic image to 3D, it may not be feasible in practice due to the need for multiple passes of different categories of semantic information on the same dataset (such as object, usability, etc.). Therefore, in the following section, we demonstrate the construction of a surface feature field for scene understanding application that satisfies various requirements through a single mapping pass.

C. Application: Open-vocabulary Scene Understanding

This application follows OpenScene and CLIP-Field [40], [41], which learn to predict dense features for 3D scene points, where the features are co-embedded with text and image pixels in CLIP feature space. Inspired by this, we design the mapping for surface feature fields (Section IV-D). The distinction between surface property fields is illustrated in Fig. 6. We use the pre-trained OpenSeg model [19] to obtain the function f_{im} that produce CLIP feature for each image point. Then we encodes the voxel latent $\mathbf{F}_m$ from the CLIP features $\mathbf{Q}_m$ and their corresponding positions $\mathbf{X}_m$. During inference, given an open-vocabulary text input $\boldsymbol{u}$ and a position input $\mathbf{x}_*$, we obtain the CLIP space features and determine the semantic property of the point based on similarity computation. The pipeline of this application is illustrated in Fig. 7.

In this application, the image encoding function f_{im} and the text encoding function f_{text} are obtained from pre-trained model, while f_{enc} and f_{dec} in our Uni-Fusion framework are deterministic functions. With these functions, our model constructs a continuous field for the CLIP feature on surface.

A very interesting and relevant work for Uni-Fusion's scene understanding application is VLMaps [42]. While VLMaps produces a 2D map, our model produces a surface CLIP feature field in 3D space.

VI. Experiments

In this section, we demonstrate the wide range of applications and the high capabilities of Uni-Fusion. First, we evaluate Uni-Fusion in application 1) Incremental surface and color reconstruction, comparing its performance with SOTAs. For applications 2) and 5), which are new topics, no specific benchmarks are available. Therefore, we showcase the performance on existing results. Next, we implement application 3) and compare it with SOTA zero-shot semantic segmentation models. Finally, for application 4), since infrared data is not commonly used, we collect our own dataset containing infrared values and show all applications on this data.

A. Implementation Details

In the experiments, we use our sample-based GPIS for local geometry encoding. For each point, two additional points are sampled along normal direction, one positive and one negative, with distance $d_s = 0.1$ in the local voxel's normalized space. Compared to derivative-based GPIS, our sample-based GPIS is more efficient in both space and time. For the encoder, we randomly sample 256 anchor points from the range $[-0.5, 0.5]^3$. We utilize the first 20 eigenpairs, resulting in a feature dimension of 20. The model selection process is discussed in the ablation study.

Different latent maps use different granularities. For the surface LIM, we use a voxel size of $5\mathrm{cm}$. For color which requires later comparison to NeRF, we use a voxel size of $2\mathrm{cm}$. For other property LIM and feature LIM, we use a voxel size of $10\mathrm{cm}$.

For smooth reconstruction, the encoded voxel is designed overlapped following [12]. The encoded voxel uses twice the voxel size, resulting in a half-space overlap with each neighboring voxel. During meshing, SDFs are retrieved and interpolated from the overlapped voxels [12]. While for the remaining properties, we sample only from its own voxel part.

The implementation runs on a PC with AMD Ryzen 9 5950X 16-core CPU and an Nvidia Geforce RTX 3090 GPU (24 GB). Online collaboration with OpenSeg (takes 15 GB GPU memory) utilizes one other 3090 GPU solely for OpenSeg to avoid out of memory.

B. Datasets

We evaluate incremental reconstruction on the ScanNet dataset [43], TUM RGB-D dataset [44], and Replica dataset [14]. Using MSG-Net [45]'s material set, we transfer styles to the 3D canvas. For open-vocabulary scene understanding, we evaluate on ScanNet segmentation data [46] and S3DIS dataset [47].

1) ScanNet [43]: ScanNet is a densely annotated RGB-D video dataset. It is captured with the structure sensor [48] and contains 1513 scenes for training and validation. For each scene, both images and a 3D mesh is provided, along with their 2D and 3D semantic annotations.

ScanNet provides 312 scenes for validation, which contains a wide range of different room structures. It has now been widely used in the thorough evaluation of the performance of reconstruction and semantic segmentation.

2) TUM RGB-D [44]: TUM RGB-D is a benchmark to mainly evaluate the tracking performance. It is captured with Microsoft Kinect sensor together with ground-truth trajectory from the sensor.

3) Replica [14]: The Replica dataset refers to iMAP's pre-processed dataset [14]. It is a synthetic rendered RGB-D dataset from given 3D models. The advantage of including this dataset is that Replica does not have motion blur. This is better to evaluate the capability of the algorithms on reconstructing surface color.

4) MSG-Net Style [45]: MSG-Net provides material images for transfering the styles. We select 21style fold for demonstration. These images are given in Fig. 14 together with our result.

5) ScanNet Point Cloud Segmentation Data [46]: For point cloud semantic segmentation benchmarking, PointNet++ [46] preprocesses the original ScanNet [43] and generates sub-sampled point clouds and corresponding annotations for each scene.

6) S3DIS [47] and 2D-3D-S [49]: S3DIS is a semantic segmentation dataset for 3D point clouds. Which is also a subset of the 2D-3D-S dataset. The 2D-3D-S dataset is a multi-modality dataset containing 2D, 2.5D and 3D domains. This dataset is densely annotated with semantic labels.

Note that 2D-3D-S's 2D captures is not a RGB-D video as ScanNet. 2D-3D-S's images only have small overlap. Therefore, it is only suitable for semantic segmentation and not for incremental reconstruction.

C. Baselines

For online surface mapping evaluation, we select TSDF-Fusion [9], iMAP [14], SOTA DI-Fusion [12] and BNV-Fusion [17] as four baseline methods.

For the color field, we choose TSDF-Fusion [9], σ-Fusion [50], iMAP [14], NICE-SLAM [15] and even the recent hot Neural Radiance Fields model NeRF-SLAM [18] as five baselines. While including NeRF in the comparison may not be entirely fair, we want to show how Uni-Fusion narrows the performance gap.

For the scene understanding application, we evaluate generalized zero-shot point cloud semantic segmentation with ZSLPC [51], DeViSe [52] and SOTA 3DGenZ [53] for comparison.

D. Metrics

For incremental reconstruction, we evaluate the geometric reconstruction using **Accuracy**, **Completeness**, and **F1 score** according to SOTA BNV-Fusion. It firstly uniformly samples $100,000$ points from the reconstruction and ground truth meshes respectively. Then **Accuracy** (**Completeness**) measures the percentage of reconstruction-to-groundtruth (groundtruth-to-reconstruction) distances that are lower than 2.5cm threshold. **F1 score** is the harmonic mean of accuracy and completeness. For tracking performance, we use **ATE RMSE**.

To evaluate color reconstruction, we follow SOTA on this topic, NeRF to render both depth and RGB images to compare the image level **Depth L1** and **RGB PSNR**.

To compare scene understanding, we follow zero-shot point cloud semantic segmentation SOTA 3DGenZ to evaluate the **Intersection-of-Union (IoU)** and **Accuracy**.

E. Reconstruction Results

In our evaluation, we first use the ScanNet validation set with 312 sequences to thoroughly test the geometric reconstruction on a wide variety of scenes. Then, we use TUM RGB-D to compare our modified tracking model with related works. Since the tracking part is not the contribution of this paper, we give a brief overview of the tracking results. To fairly evaluate the color reconstruction, we compare with related works, including NeRF, on the high-quality rendered Replica dataset.

1) Evaluation on ScanNet Dataset [43]: We use the 312 different scenes from the ScanNet validation set to evaluate the performance of surface reconstruction. We follow the pure mapping SOTA BNV-Fusion to take every 10th posed frame as input. Without using any learning (as iMAP, DI-Fusion, and BNV-Fusion do) or any post-optimization (as BNV-Fusion does), our Uni-Fusion is capable to achieve precise continuous mapping performance.

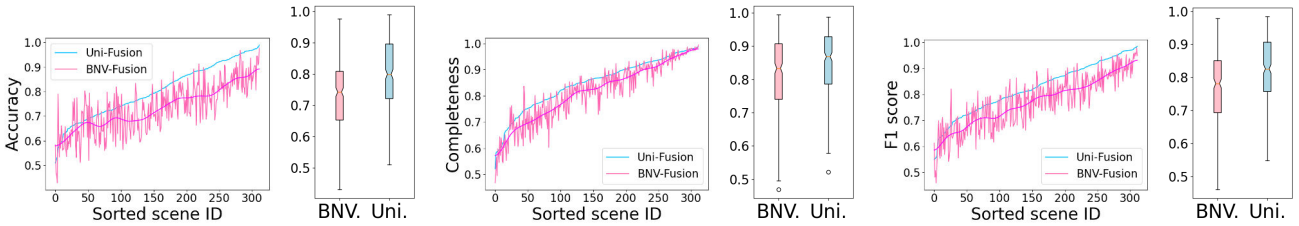
The results presented in Table I demonstrate that our Uni-Fusion outperforms the SOTA method BNV-Fusion in terms of $+6$ **higher accuracy**. However, our model does not outperform BNV-Fusion in terms of completeness, since BNV-Fusion incorporates completion in post-optimization. Nevertheless, Uni-Fusion's completion is still much higher than one other optimization based model iMAP. Overall, our Uni-Fusion model achieves higher F1-scores, which quantifies the overall quality of reconstruction.

It is important to note that the SOTA BNV-Fusion is not capable of real-time performance as it requires post-optimization of all fed frames. On the other hand, the real-time model Di-Fusion exhibits much worse results without using post-optimization. In contrast, our **real-time** model, **Uni-Fusion** achieves **much better** reconstruction quality than these approaches even without post-optimization.

We additionally run BNV-Fusion's official implementation (emphasized with *) on the 312 videos from ScanNet and conduct a scene-wise comparison in Section VI-E. Our result is shown with the light blue curve, BNV-Fusion's result is shown with pink. The scene index is sorted based on the score value of Uni-Fusion. To enhance visual clarity, we apply a smoothing technique to BNV-Fusion's curve and presented it with dark pink color. The comparison clearly shows that Uni-Fusion is overall performing better than BNV-Fusion. In addition, we use box-plots to analyze the statistics alongside the

TABLE I
SURFACE COMPARISON ON SCANNET [43]. SCORES ARE FETCHED FROM [17]. * INDICATES THE RESULT FROM OUR RUNS OF THE OFFICIAL BNV-FUSION CODE.

Method	Pre-Train with extra dataset	Train with sequences	Real-time	Accuracy (%) ↑	Completeness (%) ↑	F1 score ↑
TSDF Fusion [54]	None	None	✓	73.83	85.85	78.84
iMAP [14]	None	Online train		68.96	82.12	74.96
DI-Fusion [12]	Object Pretrain	None	✓	66.34	79.65	72.97
BNV-Fusion [17]	Object Pretrain	Post Optimization		74.90	**88.12**	80.56
BNV-Fusion* [17]	Object Pretrain	Post Optimization		73.42	81.75	77.18
Uni-Fusion (Ours)	None	None	✓	**80.43**	84.91	**82.44**



(a) Accuracy (b) Completeness (c) F1 score

Fig. 8. Quantitative comparison on 312 scenes of the ScanNet validation set. We demonstrate the performance of SOTA BNV-Fusion and our Uni-Fusion. We sort Uni-Fusion's evaluation value and reordered all of the scores. The zigzag pink is the BNV-Fusion result; we also plot a deep-pink smoothed curve for better visualization.

curve plot. Uni-Fusion's scores show a higher concentration on the plots. While the difference in completeness may be less pronounced, Uni-Fusion's box plot is smaller and positioned relatively higher. This indicates that Uni-Fusion achieves a more stable completeness result, while BNV-Fusion is more likely to achieve low completeness in some cases.

In summary, our Uni-Fusion model achieves superior results across almost all 312 scenes in terms of accuracy, completeness and F1-score. This finding is consistent with the observation presented in Table I with BNV-Fusion*, that Uni-Fusion outperforms the official implementation of BNV-Fusion in all metrics.

We show reconstruction on selected scenes from ScanNet in Fig. 9. Both BNV-Fusion and our Uni-Fusion are able to produce high quality reconstructions. However, we observe that BNV-Fusion generates numerous small meshes on walls, resulting in the appearance of small particles in the reconstruction. We attribute this behavior to BNV-Fusion's use of very small voxel size ($0.02\mathrm{m}$) to achieve a high score. This is further supported by the fact that their mesh averages $247\,\mathrm{MB}$, while ours averages only $54\,\mathrm{Mb}$. Furthermore, our Uni-Fusion's mesh is smoother and also provides highly-accurate color to the mesh which is not available for this surface SOTA.

Besides, we evaluate the storage cost of the latent representation. BNV-Fusion's latents require an average storage of $228\,\mathrm{MB}$ across the 312 scenes of ScanNet, while Uni-Fusion achieves significantly lower storage requirements with an average of only $9\,\mathrm{MB}$.

2) Tracking Evaluation on TUM RGB-D Dataset [44]: In the above test, we evaluate the performance of pure mapping. Although tracking is not the contribution focus in our paper, it is still part of the incremental reconstruction model. We follow the novel incremental reconstruction model NICE-SLAM [15] to evaluate the camera tracking on the small scale TUM RGB-D dataset. Our Uni-Fusion uses a coarse-to-fine strategy for 3D reconstruction tracking. From Table II, it demonstrates overall better ATE RMSE than other implicit representation models.

TABLE II
TRACKING ON TUM RGB-D [44]. ATE RMSE [cm] (↓) IS USED AS THE EVALUATION METRIC.

	`fr1/desk`	`fr2/xyz`	`fr3/office`
iMAP [14]	4.9	2.0	5.8
DI-Fusion [12]	4.4	2.3	15.6
NICE-SLAM [15]	2.7	1.8	3.0
Ours	1.8	0.5	2.1
BAD-SLAM [55]	1.7	1.1	1.7
Kintinuous [56]	3.7	2.9	3.0
ORB-SLAM2 [57]	**1.6**	**0.4**	**1.0**

On the other hand, there also exist high accuracy algorithms from SLAM. By additionally using bundle adjustment and loop-closing techniques, their tracking quality is much better than all of the implicit based models. Our coarse-to-fine strategy obtains a good score because, first it ensures it does not easly lose track. Second, it is more suitable for surface fitting.

This further supports our test on the Replica dataset.

3) Evaluation on Replica RGB-D Dataset [14]: In this evaluation, we compare with implicit reconstruction (TSDF-Fusion, σ-Fusion) and latent implicit reconstruction models (iMAP, NICE-SLAM) that support color. Additionally, we include a large-scale NeRF model, NeRF-SLAM, in the comparison. It is important to note that NeRF is SOTA in the view-synthesis task, which gives it an unfair advantage over other models because it learns light directions and does not really model a surface. However, we include NeRF in this evaluation to demonstrate that Uni-Fusion significantly reduces the gap.

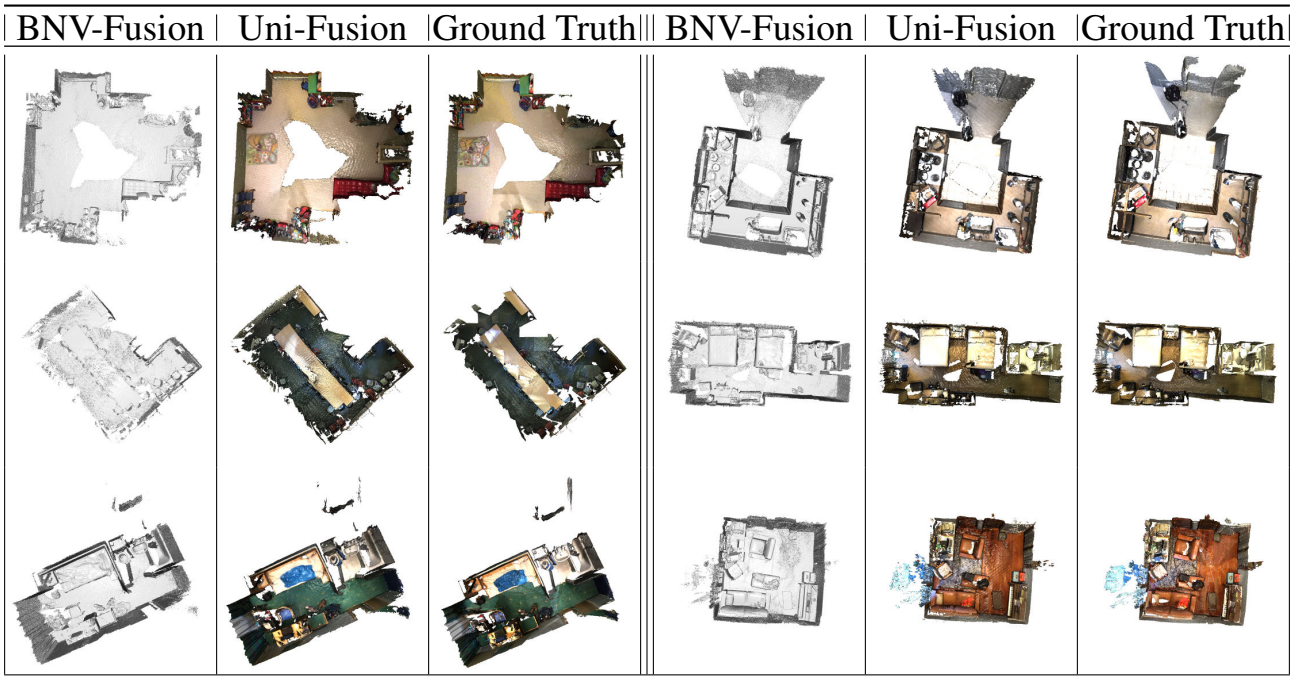


Fig. 9. Surface reconstruction on ScanNet dataset.

TABLE III
GEOMETRIC (L1) AND PHOTOMETRIC (PSNR) EVALUATION ON THE REPLICA DATASET [14].

		office-0	office-1	office-2	office-3	office-4	room-0	room-1	room-2	Avg.
Non-continuous mapping method										
TSDF-Fusion [9]	**Depth L1** [cm] ↓	14.11	10.50	30.89	28.92	22.83	23.51	20.94	23.34	21.88
	PSNR [dB] ↑	11.16	15.92	4.86	5.68	5.46	3.43	4.51	5.57	7.07
σ-Fusion [50]	**Depth L1** [cm] ↓	13.80	10.21	22.27	28.70	22.21	21.92	19.28	22.40	20.10
	PSNR [dB] ↑	11.16	15.92	4.86	5.69	5.46	3.45	4.51	5.57	7.08
Continuous mapping method										
iMAP* [14]	**Depth L1** [cm] ↓	6.43	7.41	14.23	8.68	6.80	5.70	4.93	6.94	7.64
	PSNR [dB] ↑	7.39	11.89	8.12	5.62	5.98	5.66	5.31	5.64	6.95
Nice-SLAM [15]	**Depth L1** [cm] ↓	1.51	0.93	8.41	10.48	2.43	2.53	3.45	2.93	4.08
	PSNR [dB] ↑	22.44	25.22	22.79	22.94	24.72	**29.90**	**29.12**	19.80	24.61
Uni-Fusion (Ours)	**Depth L1** [cm] ↓	**0.79**	**0.56**	**1.59**	**2.71**	**1.66**	**1.94**	**0.69**	**1.80**	**1.47**
	PSNR [dB] ↑	**33.88**	**33.31**	**25.84**	**26.01**	**28.14**	24.02	26.20	**27.17**	**28.07**
Neural radiance field method										
NeRF-SLAM [18]	**Depth L1** [cm] ↓	2.49	1.98	9.13	10.58	3.59	2.97	2.63	2.58	4.49
	PSNR [dB] ↑	**48.07**	**53.44**	**39.30**	**38.63**	**39.21**	**34.90**	**36.95**	**40.75**	**41.40**

TABLE IV
DIFFERENCES AMONG DIFFERENT SURFACE & COLOR RECONSTRUCTION MODELS.

Method	Pre-Train with extra dataset	Train with sequences	Real-time	Direct Output	Light direction	Render
TSDF-Fusion	None	None	✓	Discrete TSDF		Ray Rasterization
σ-Fusion	None	None	✓	Discrete TSDF		Ray Rasterization
iMAP	None	Online Train		MLPs		Volumetric Rendering
NICE-SLAM	Pretrain with indoor dataset	Online Train		Neural Implicit Grid		Volumetric Rendering
NeRF-SLAM	None	Train hundred epochs	-	NeRF	✓	Volumetric Rendering
Uni-Fusion	None	None	✓	Latent Implicit Map		Ray Rasterization

Notably, NeRF-SLAM embeds external tracking model [50], [58] to provide poses while using SOTA NeRF implementation Instant-ngp [59] for NeRF construction.

Uni-Fusion tracks and follows the same setting as in ScanNet test to take every 10 frames for mapping. NICE-SLAM and NeRF-SLAM create depth and color using volumetric rendering. In Uni-Fusion, we cast rays from the virtual camera onto our result surface mesh for the depth image. The cast points are then inferred using Uni-Fusion's color LIM to obtain color results.

According to Table III, Uni-Fusion demonstrate the best Depth L1 on all scenes with an average of **1.47**cm **depth L1**. This is a **177% boost** compared to the second best model.

Moreover, excluding NeRF, our Uni-Fusion also shows the best performance in modeling the colors, achieving an average PSNR of 28.07dB.

However, it is strange that NICE-SLAM loses details while in two cases, it shows better PSNR than Uni-Fusion. To highlight the true result, we provide rendered images in Fig. 10. It is evident that our Uni-Fusion accurately models the details of painting, carpet and quilt, while NICE-SLAM only roughly models the average color.

In addition, from Fig. 10, Uni-Fusion's rendering quality is as precise as NeRF. The painting, carpet and quilt in Uni-Fusion's results are very similiar to the original appearance. The green window highlights the regions of interest. Uni-Fusion reproduces the high-quality appearances that are very close to NeRF in terms of qualitative evaluation. However, Uni-Fusion still has a quantitative score gap to NeRF's color rendering (41.4dB), despite the highly comparable qualitative results to NeRF and ground truth. We attribute this difference to three main factors: **1.** Uni-Fusion does not model the light directions to points, which is essential to NeRF. **2.** NeRF optimizes image quality by focusing primarily on color rather than depth, which is evident from its higher color rendering score but much worse depth rendering compared to Uni-Fusion. **3.** Uni-Fusion does not support hole filling, which results in black holes in the rendered images.

We summarize the differences between Uni-Fusion and other models in Table IV. Similar to TSDF-Fusion and σ-Fusion, Uni-Fusion is a forward method that does not require any training of map representation, i.e., pre- or on-line training. It shares similarities with NICE-SLAM and NeRF-SLAM in producing an implicit map with a set of latents, that outputs results at arbitrary resolution. However, Uni-Fusion differs in the extraction of the signed distance field, as each query value is directly inferred using the corresponding ruling latent, while NICE-SLAM and NeRF-SLAM use a much denser grid of features for interpolation during volumetric rendering based inference.

Like TSDF-Fusion and σ-Fusion, our Uni-Fusion is a real-time algorithm, whereas iMAP, NICE-SLAM and NeRF-SLAM are not capable of running in real time. NeRF-SLAM claims to be real-time, which is questionable as it still needs hundreds of epochs training after feeding the data.

However, optimization with backpropagation allows for pixel-to-pixel learning, which is theoretically superior to the regression and fusion strategy. Although Uni-Fusion demonstrates its high ability to model colors, exploring NeRF-like post-optimization would be a promising direction for further improvements.

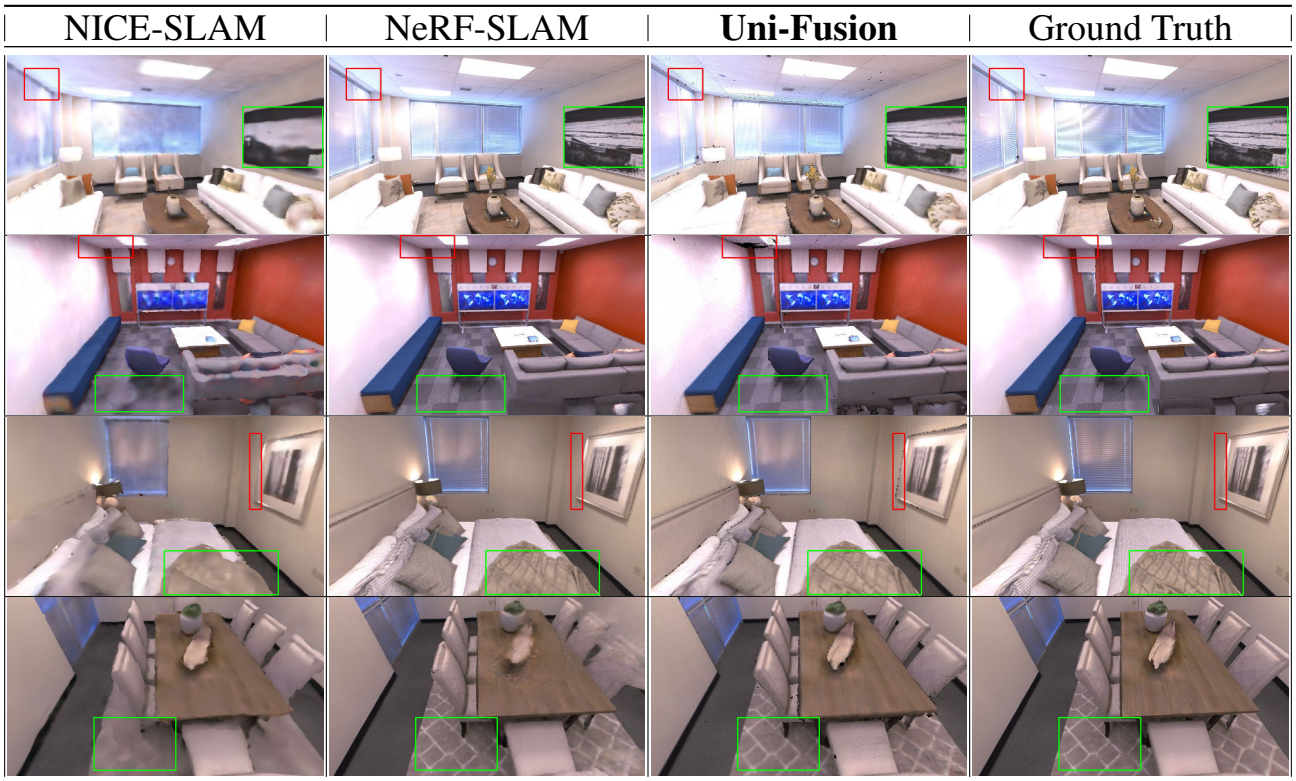


Fig. 10. Demonstration of color rendering on the Replica dataset. Fine appearances are highlighted in green window. Small defects are in a red box.

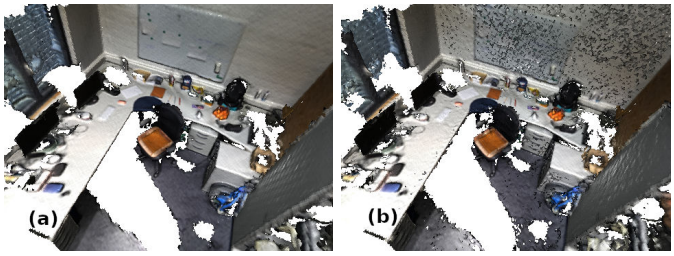


Fig. 11. Ablation study on surface construction basis. (a) Sample based. (b) Derivative based.

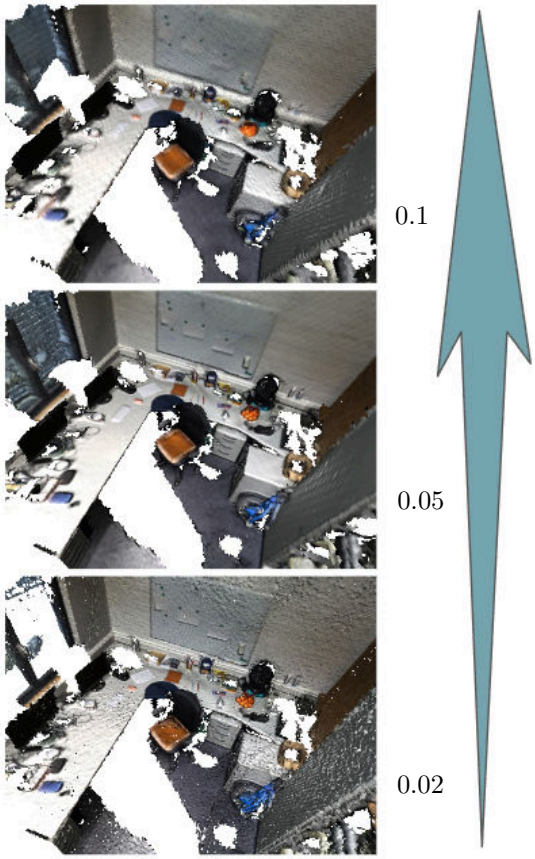


Fig. 12. Ablation study on voxel size.

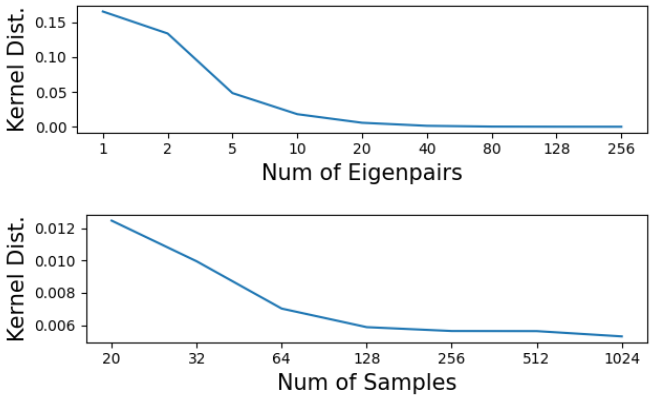


Fig. 13. Ablation study on number of eigenpairs and number of anchor samples in kernel approximation.

F. Ablation study

1) Sample-based or Derivative-based: We select the surface model with our own recorded sequences. All settings are detailed in Section VI-A. As shown in Fig. 11, reconstructions of Yijun's office are demonstrated. While both models are capable of construction, the derivative-based model introduces a lot of noise to the surface. This issue arises, because, for smoothness purpose, we follow Di-Fusion [12] to build voxels that overlap with their neighbors, leading to redundant voxels near the surface. For these redundant voxels, no center sample is provided and thus the derivative-based surface construction builds poor SDFs on unknown region of the voxels.

In contrast, the sample-based surface construction does not encounter this problem because it adds more points within the voxels, enabling the construction of very smooth surfaces. We observe well-constructed and accurately colored objects such as the whiteboard, the chair, the school bag and even the oranges.

2) Voxel size: While testing of the office scene, we vary the voxel size from low to high. From Fig. 12, when a low voxel size $0.02\mathrm{m}$ is used, the surface appears rough. As the voxel size increases, the smoothness improves. However, when a voxel size of $0.1\mathrm{m}$ is employed, the surface appears blurry. Considering Uni-Fusion produces a surface color field, the quality of surface directly impacts the coloring. Thus, further increasing the voxel size will result in deteriorated surface quality.

Therefore, in the above given experiments, $0.05\mathrm{m}$ voxel size is used for surface construction. Additionally, it should be clarified that each voxel used for encoding actually has a size of $0.1\mathrm{m}$ due to the employment of overlapped voxels.

3) Number of eigenpairs and anchor points: The kernel approximation is affected by the number of eigenpairs and anchor points. Uni-Fusion treats the approximation module as a cohesive entity, expecting it to behave like a real kernel. Therefore, we conduct the ablation study at the module level.

This feature dimension l corresponds to the number of eigenpairs retained during kernel approximation. We employ a uniform sampling of 256 anchor points for the approximation. To access the accuracy in recovering the original kernel, we randomly sample 2000 test samples in $[-.5, .5]^3$ and compute the matrix $\mathbf{K}$ using original Matérn Kernel. Our approximation, denoted as $\hat{\mathbf{K}}$, is then evaluated. We calculate the Mean Absolute Error (MAE) between the two kernels and observe the curve presented in Fig. 13. From the figure, we find that the error decreases fast until l reaches 20, and beyond that point, the improvement becomes marginal. Considering that the result for $l = 20$ is very close to that at $l = 40$ while requiring only half the storage space, the optimal selection for l is 20.

Similarly, we perform a module-level ablation for number of anchor points. With l fixed at 20, a minimum of 20 samples is required. Fig. 13 demonstrates that the approximation shows minimal improvement beyond 256 samples. Additionally, since this number of anchor samples only affects the computation time and not the storage space, it is advisable to select a the large value such as 256, but not the largest, to improve efficiency without sacrificing accuracy.

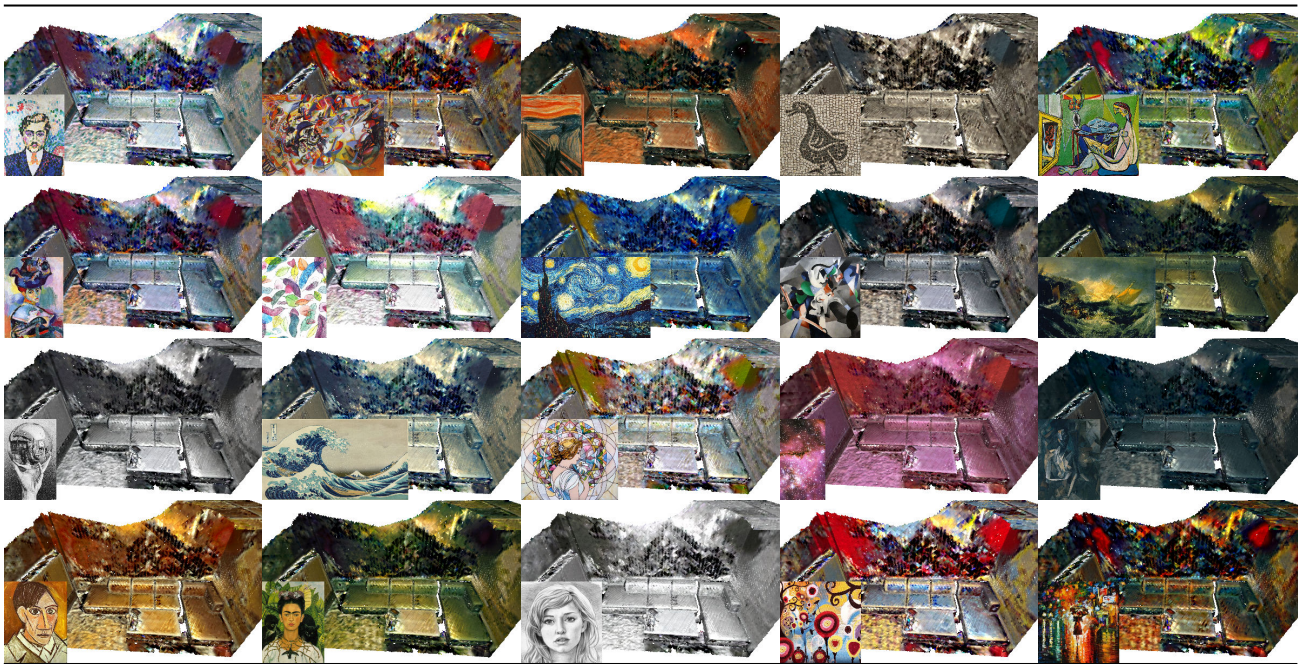


Fig. 14. Style transfer on 3D canvas.

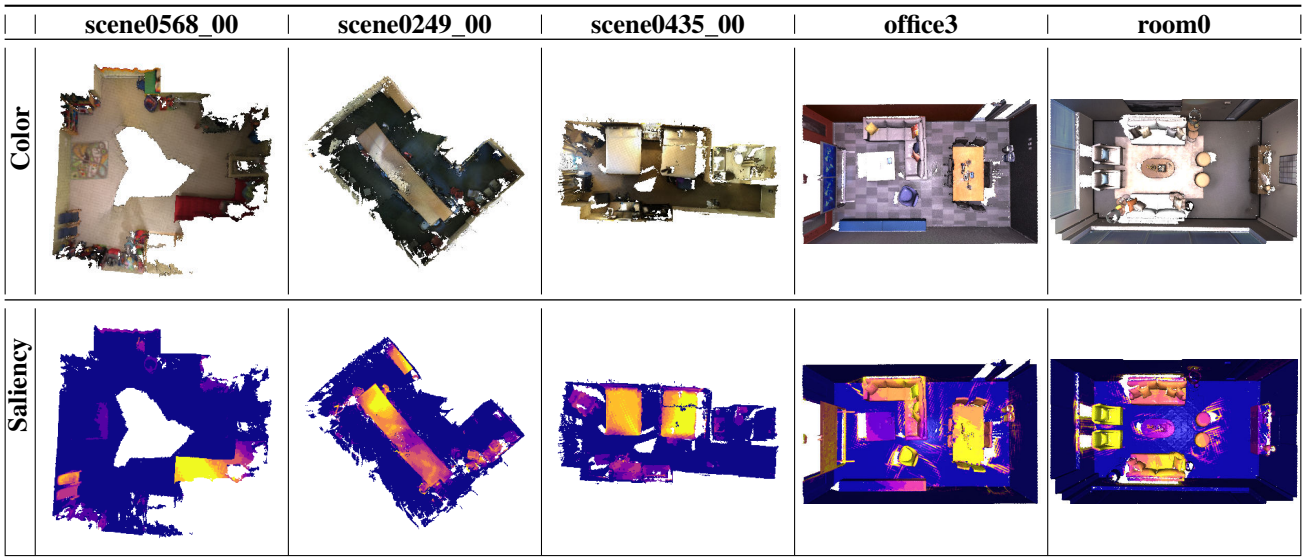


Fig. 15. Saliency transfer on the 3D canvas. The top row shows the result colored meshes. The bottom row shows the saliency meshes.

G. Results for 2D-to-3D transfer

In addition to surface and color, Uni-Fusion also allows continuous mapping of other fabricated data. Therefore, we apply Uni-Fusion to style data and saliency data in order to achieve style and saliency transfer on the 3D canvas.

In Fig. 14, we present artistic painting on the 3D canvas. Each frame undergoes style transfer using MSG-Net [45], and Uni-Fusion is used to construct the style LIM for the surface style field. The test scene is "office0" from the Replica dataset captured from a single custom view. 20 images are used to provide the style and are attached at lower left corner accordingly. Our Uni-Fusion successfully constructs style meshes that closely resemble the taste of the supplied style images. For instance, with pure style or abstract painting, the 3D "canvas" shows a very similar style. Among the style images, with our favorite style, located in the middle of the fourth row, Uni-Fusion produces a high quality 3D sketch painting.

In Fig. 15, we demonstrate saliency detection in 3D. In this figure we select three scenes from ScanNet and two scenes from Replica. Similarly, we detect saliency on each frame using InSPyReNet [60] and construct saliency LIM for surface saliency field.

In the second row, high saliency regions are colored in yellow, indicating the object of interest. This information can be used to guide a robot's navigation in 3D scene. For example, in the first column, the sofa, chair, curtain, and television in the room certainly attract more attention in daily life. In the

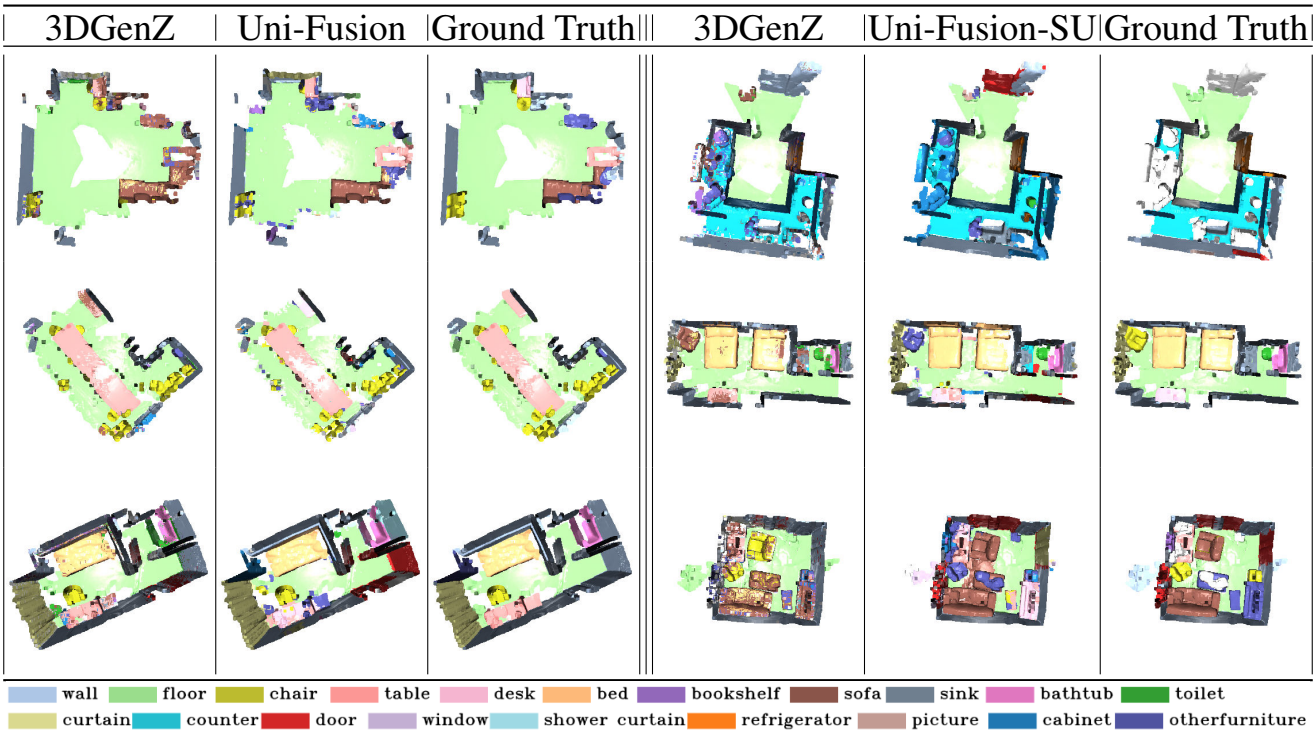


Fig. 16. Demonstration of semantic segmentation on the ScanNet dataset. Selected scenes are consistent with Fig. 9

second column, a meeting room, the long desk and chairs are obviously the main components. Similarly, in the third column showcasing a hotel room, the bed stands out, along with the sofa, desk, lamp, and TV in the fourth column, and the sofas and chairs in the last column.

H. Scene Understanding Results

Saliency detection effectively highlights the objects of interest. This is also considered part of 3D semantic understanding. However, as the semantics categories vary, fusing different categories of semantics into multiple LIMs can be inefficient. Therefore, in this section, we utilize Uni-Fusion to fuse and construct a surface field for high-dimensional CLIP embeddings. With a single LIM, we can generate different semantic results based on corresponding commands. Since now our Uni-Fusion works with OpenSeg for scene understanding purposes, we call it Uni-Fusion-SU.

1) Semantic Segmentation: We first evaluate our model on generalized zero-shot point cloud semantic segmentation application. Generalized Zero-Shot Learning (GZSL) differs from Zero-Shot Learning (ZSL) in that ZSL only predicts classes unseen during training, while GZSL predicts both unseen and seen classes [53]. Therefore, comparing our results with GZSL SOTAs provides a better understanding of the potential of Uni-Fusion-SU, as it does not train on both seen and unseen.

This test uses ScanNet and S3DIS datasets for benchmarking. It is important to note that the **compared baselines are trained on the corresponding datasets**. Our Uni-Fusion-SU uses OpenSeg to provide the 2D image level feature

TABLE V
GZSL SEMANTIC SEGMENTATION RESULTS. SCORES ARE IN %. † INDICATE 3DGENZ'S ADAPTION OF THE METHOD. NOTE THAT, UNI-FUSION-SU DOES NOT EVEN TRAIN WITH THE SEEN CLASSES.

	Training set		Inference input	ScanNet			S3DIS		
	Backbone	Classifier		*Seen*	*Unseen*	*All*	*Seen*	*Unseen*	*All*
Supervised methods with different levels of supervision									
Full supervision	$seen \cup unseen$	$seen \cup unseen$	Point Cloud	43.3	51.9	45.1	74.0	50.0	66.6
ZSL backbone	$seen$	$seen \cup unseen$	Point Cloud	41.5	39.2	40.3	60.9	21.5	48.7
ZSL-trivial	$seen$	$seen$	Point Cloud	39.2	0.0	31.3	70.2	0.0	48.6
Generalized zero-shot-learning methods									
ZSLPC-Seg [51]†	$seen$	$unseen$	Point Cloud	28.2	0.0	22.6	65.6	0.0	45.3
DeViSe-3DSeg [52]†	$seen$	$unseen$	Point Cloud	20.0	0.0	16.0	70.2	0.0	48.6
3DGenZ [53]	$seen$	$seen \cup unseen$	Point Cloud	32.8	7.7	27.8	53.1	7.3	**39.0**
Zero-shot learning + map fusion									
Uni-Fusion-SU (Ours)	None	None	Sparse Frames	31.0	**41.9**	**32.9**	31.3	**24.0**	29.0

ebmedding. Although **Uni-Fusion-SU** is also zero-shot, **it does not touch any ScanNet or S3DIS annotations**.

We demonstrate the mIoU scores in Table V. In particular, our model achieves best results among the zero-shot learning methods on the ScanNet dataset and remains competitive with fully supervised methods.

Furthermore, we provide results specifically for the unseen classes in Table VI. Although not as good as the fully supervised approach, Uni-Fusion-SU performs much better than 3DGenZ. In addition, our Uni-Fusion-SU demonstrates high precision in classes such as sofa and Toilet, even when compared to the fully supervised model.

TABLE VI
CLASSWISE GZSL SEMANTIC SEGMENTATION PERFORMANCE (%) ON THE SCANNET UNSEEN SPLIT.

		Bookshelf	**Desk**	**Sofa**	**Toilet**	mean
FSL (Fully supervise)	IoU	56.9	30.0	57.4	63.4	51.9
3DGenZ (Zero-shot)	IoU	6.3	3.3	13.1	8.1	7.7
Uni-Fusion-SU (Ours)	IoU	38.3	16.8	51.7	60.9	41.9
3DGenZ (Zero-shot)	Acc.	13.4	5.9	49.6	26.3	23.8
Uni-Fusion-SU (Ours)	Acc.	61.9	29.6	67.4	91.6	62.6

However, in the S3DIS dataset, our model does not outperform 3DGenZ and other methods as shown in Table V.

Even in the result of unsceened data, as presented in Table VII, we observe that Uni-Fusion-SU hardly finds some classed, e.g. Beam and Column, which are not commonly annotated objects. However, for common objects like sofa and window, our model performs much better.

TABLE VII
CLASSWISE GZSL SEMANTIC SEGMENTATION PERFORMANCE (%) ON THE S3DIS UNSEEN SPLIT.

		Beam	**Column**	**Sofa**	**Window**	mean
FSL (Fully supervise)	IoU	63.1	10.2	54.1	72.4	50.0
3DGenZ (Zero-shot)	IoU	13.9	2.4	4.9	8.1	7.3
Uni-Fusion-SU (Ours)	IoU	5.5	0.02	57.4	32.9	24.0
3DGenZ (Zero-shot)	Acc.	20.0	9.1	62.4	23.7	28.8
Uni-Fusion-SU (Ours)	Acc.	41.5	0.02	78.3	42.1	40.5

We present the results of the semantic segmentation in Fig. 16. It is evident that, 3DGenZ's result contains more noise, as seen in the spotted sofa, bed and other objects. Conversely, Uni-Fusion-SU's results are generally smoother and more precise.

2) Scene Understanding with Different Properties: The main contribution of this application is that, Uni-Fusion is the first model to construct a continuous mapping of high-dimensional embeddings onto the surface without the need for any training of the map representation. In the previous experiment (Section VI-H1), we evaluate the performance of generalized zero-shot semantic segmentation. However, the potential of Uni-Fusion goes beyond semantic segmentation. By constructing a LIM, we obtain a surface CLIP feature field. This enables us to query various semantic categories such as **Object, Room Type, Material, Affordance and Activity** without requiring multiple LIMs or re-running the model.

We present the results in Fig. 17, where we query object (desk, sofa), activity (work), affordance (sittable), and material (wood). Uni-Fusion-SU accurately identifies and highlights the object and material regions. However, for less specific commands such as work or sittable, the model provides a wider range of results with less confidence (indicated by dull yellow). Nevertheless, the suggested options are also roughly correct.

I. Time

We run all of the applications in a single pass using our captured office sequences and evaluate the time cost of construction and fusion of each LIM. The average time cost across frames is shown in Table VIII.

TABLE VIII
TIME REQUIRED FOR EACH FRAME.

	Surface	Color	Infrared	Style	Saliency	Latent	Internal Track
Time (s)	0.100	0.038	0.045	0.048	0.045	0.011	0.225

Using depth and property images of size 720×1280 as input, it is evident from the table, that our model operates at a frequency of ~ 10Hz for surface (sample mode) LIM construction and integration. It alse achieves a frequency of over 20Hz for color, infrared, style, and saliency. These results demonstrate the suitability of Uni-Fusion for real-time applications.

However, our internal tracking process takes around 0.225s per frame, which is relatively slower compared to the mapping module. Nevertheless, Uni-Fusion uses external tracking to prevent tracking loss, enabling our internal tracking and mapping to operate at a lower frequency. As a result, the entire model can be effectively applied in real-time in various scenarios.

VII. EXTENSIVE EXPERIMENT ON OUR OWN DATASET

In previous experiments, we evaluate the capabilities of Uni-Fusion in different applications. To further demonstrate its effectiveness in robotic environmental understanding, we capture our own dataset to show all applications together.

We capture two scenes: The office and apartment of the first author using a Microsoft Kinect Azure. RGB-D and infrared video are captured. After calibration, RGB, depth, infrared inputs have resolution of 720×1280. Uni-Fusion tracks and reconstructs all applications in one pass. While office data has been involved in ablation study (Section VI-F), we showcase all applications using the apartment dataset, as depicted in Fig. 18.

For better visualization, the ceiling of reconstruction is removed. The top row of images presents the colored mesh with room details, the infrared mesh revealing the lighting effect, and the saliency reconstruction highlighting objects crucial for navigation. Additionally, we select the second style from Fig. 14 for style transfer to the apartment canvas. As a result, the wooden floor in the room is colored with dark green. The whole apartment is in a warm style.

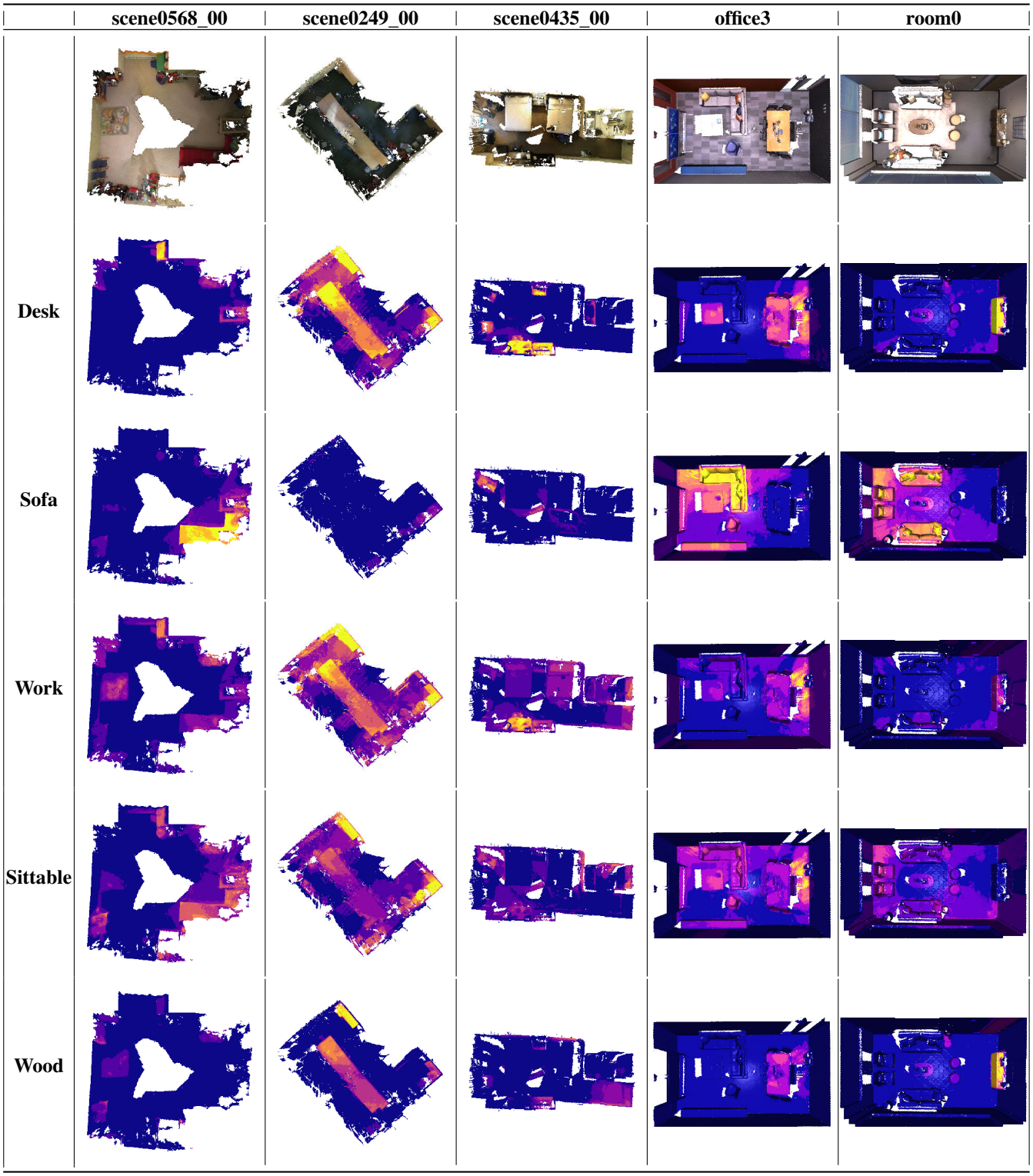


Fig. 17. Demonstration of the original mesh, highlighted semantic mesh given various queries.

The remaining results are generated from the surface field of the CLIP embeddings. We issue commands to locate objects, e.g., where is the sofa, desk and coat. In addition, it easily identifies affordances such as being sittable. For material, it successfully detects the wooden floor in each room.

VIII. Limitations and Future Work

1) Remapping: Uni-Fusion currently lacks support for deintegrating local LIM from global LIM, which is essential for incorporating bundle adjustment or loop closing techniques. In addition, the current state of Uni-Fusion does not allow the transformation of LIMs as demonstrated by NIM-REM [13]. To enhance a better quality and to facilitate large scale mapping, loop closing and bundle adjustment are future

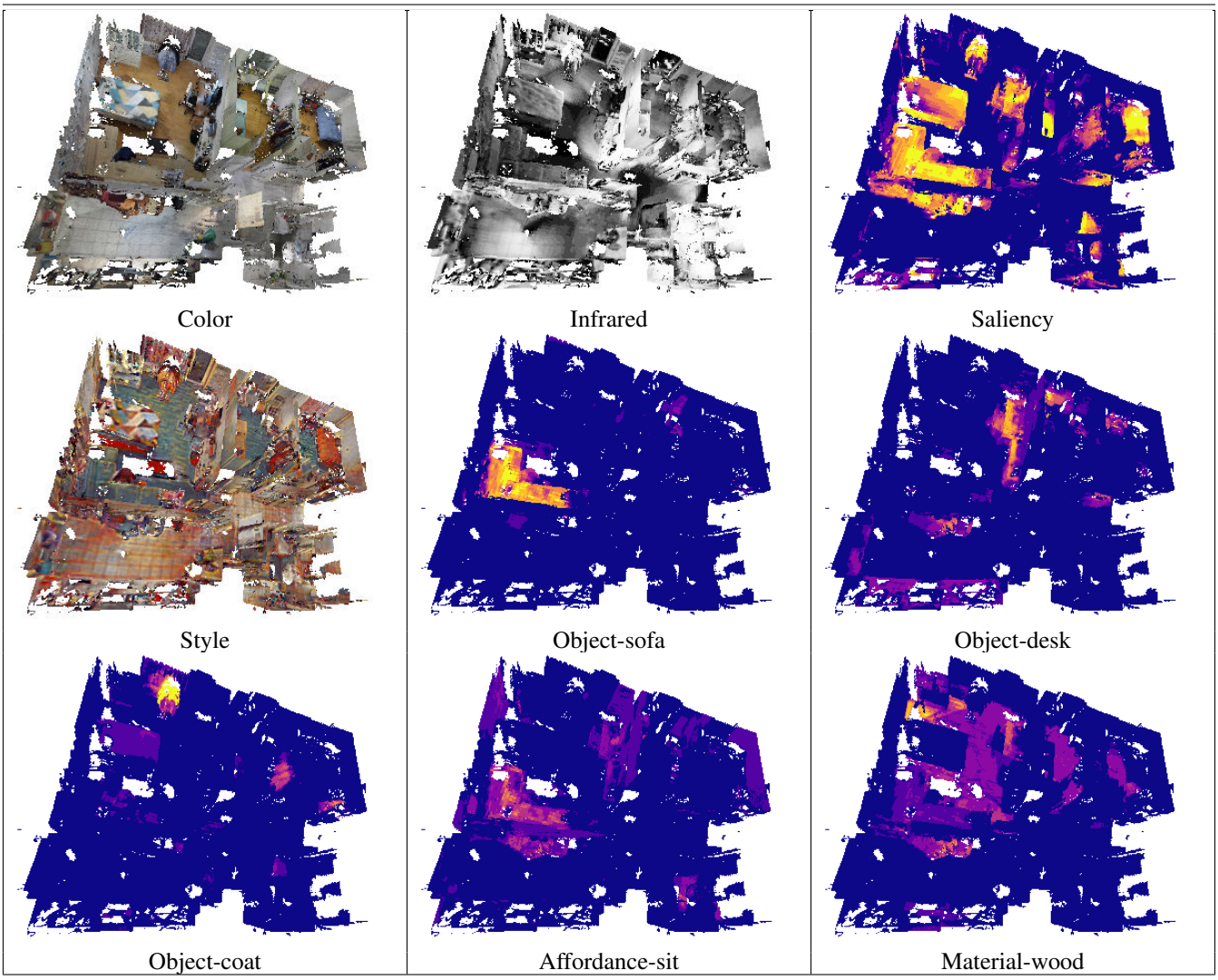


Fig. 18. Demonstration on the captured apartment data.

targets.

2) Visual Language Navigation: Uni-Fusion serves as a solid foundation for reconstruction and scene understanding in the context of Visual-Language Robot Navigation (VLN). While existing work produces a 2D embedding map [42], Uni-Fusion excels in constructing a 3D embedding map of the scene. As a result, Uni-Fusion empowers the robot with a deeper understanding of the scene. In our future work, we intend to explore applications such as navigation.

IX. Conclusion

In this paper, we have introduced Uni-Fusion, a novel universal model for all continuous mapping applications. Without any training, Uni-Fusion constructs Latent Implicit Maps that support geometry and arbitrary properties. Moving one step further to scene understanding, Uni-Fusion is also the first model that is capable of constructing continuous maps with high-dimensional embeddings without the training of map representation. With such a basis, we have implemented several applications, including a high-quality incremental surface and color reconstruction application, a 2D-to-3D transfer of fabricated properties, and an open-vocabulary scene understanding application.

References

[1] S. T. O'Callaghan and F. T. Ramos, "Gaussian process occupancy maps," *The Intl. Journal of Robotics Research*, vol. 31, no. 1, pp. 42–62, 2012.

[2] S. Kim and J. Kim, "Continuous occupancy maps using overlapping local gaussian processes," in *2013 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2013, pp. 4709–4714.

[3] M. Ghaffari Jadidi, J. Valls Miro, and G. Dissanayake, "Gaussian processes autonomous mapping and exploration for range-sensing mobile robots," *Autonomous Robots*, vol. 42, pp. 273–290, 2018.

[4] Y. Yuan, H. Kuang, and S. Schwertfeger, "Fast gaussian process occupancy maps," in *2018 15th Intl. Conf. on Control, Automation, Robotics and Vision (ICARCV)*. IEEE, 2018, pp. 1502–1507.

[5] W. Martens, Y. Poffet, P. R. Soria, R. Fitch, and S. Sukkarieh, "Geometric priors for gaussian process implicit surfaces," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 373–380, 2016.

[6] B. Lee, C. Zhang, Z. Huang, and D. D. Lee, "Online continuous mapping using gaussian process implicit surfaces," in *2019 Intl. Conf. on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 6884–6890.

[7] L. Wu, K. M. B. Lee, L. Liu, and T. Vidal-Calleja, "Faithful euclidean distance field from log-gaussian process implicit surfaces," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2461–2468, 2021.

[8] J.-P. A. Ivan, T. Stoyanov, and J. A. Stork, "Online distance field priors for gaussian process implicit surfaces," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 8996–9003, 2022.

[9] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," in *Proc. of the 23rd annual conf. on Computer graphics and interactive techniques*, 1996, pp. 303–312.

[10] 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 *Proc. of the 24th annual ACM symposium on User interface software and technology*, 2011, pp. 559–568.

[11] 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 Transactions on Graphics (ToG)*, vol. 36, no. 4, p. 1, 2017.

[12] J. Huang, S.-S. Huang, H. Song, and S.-M. Hu, "Di-fusion: Online implicit 3d reconstruction with deep priors," in *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, 2021, pp. 8932–8941.

[13] Y. Yuan and A. Nüchter, "An algorithm for the se (3)-transformation on neural implicit maps for remapping functions," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 7763–7770, 2022.

[14] E. Sucar, S. Liu, J. Ortiz, and A. J. Davison, "imap: Implicit mapping and positioning in real-time," in *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision*, 2021, pp. 6229–6238.

[15] Z. Zhu, S. Peng, V. Larsson, W. Xu, H. Bao, Z. Cui, M. R. Oswald, and M. Pollefeys, "Nice-slam: Neural implicit scalable encoding for slam," in *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, 2022, pp. 12 786–12 796.

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

[17] K. Li, Y. Tang, V. A. Prisacariu, and P. H. Torr, "Bnv-fusion: Dense 3d reconstruction using bi-level neural volume fusion," in *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, 2022, pp. 6166–6175.

[18] A. Rosinol, J. J. Leonard, and L. Carlone, "Nerf-slam: Real-time dense monocular slam with neural radiance fields," *arXiv preprint arXiv:2210.13641*, 2022.

[19] G. Ghiasi, X. Gu, Y. Cui, and T.-Y. Lin, "Scaling open-vocabulary image segmentation with image-level labels," in *Computer Vision–ECCV 2022: 17th European Conf., Tel Aviv, Israel, October 23–27, 2022, Proc., Part XXXVI*. Springer, 2022, pp. 540–557.

[20] R. Senanayake and F. Ramos, "Bayesian hilbert maps for dynamic continuous occupancy mapping," in *Conf. on Robot Learning*. PMLR, 2017, pp. 458–471.

[21] W. Zhi, L. Ott, R. Senanayake, and F. Ramos, "Continuous occupancy map fusion with fast bayesian hilbert maps," in *2019 Intl. Conf. on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 4111–4117.

[22] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, "Deepsdf: Learning continuous signed distance functions for shape representation," in *Proc. of the IEEE/CVF conf. on computer vision and pattern recognition*, 2019, pp. 165–174.

[23] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger, "Occupancy networks: Learning 3d reconstruction in function space," in *Proc. of the IEEE/CVF conf. on computer vision and pattern recognition*, 2019, pp. 4460–4470.

[24] R. Chabra, J. E. Lenssen, E. Ilg, T. Schmidt, J. Straub, S. Lovegrove, and R. Newcombe, "Deep local shapes: Learning local sdf priors for detailed 3d reconstruction," in *Computer Vision–ECCV 2020: 16th European Conf., Glasgow, UK, August 23–28, 2020, Proc., Part XXIX 16*. Springer, 2020, pp. 608–625.

[25] C. Jiang, A. Sud, A. Makadia, J. Huang, M. Nießner, T. Funkhouser *et al.*, "Local implicit grid representations for 3d scenes," in *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, 2020, pp. 6001–6010.

[26] S. Peng, M. Niemeyer, L. Mescheder, M. Pollefeys, and A. Geiger, "Convolutional occupancy networks," in *Computer Vision–ECCV 2020: 16th European Conf., Glasgow, UK, August 23–28, 2020, Proc., Part III 16*. Springer, 2020, pp. 523–540.

[27] S. Lionar, L. Schmid, C. Cadena, R. Siegwart, and A. Cramariuc, "Neuralblox: Real-time neural representation fusion for robust volumetric mapping," in *2021 Intl. Conf. on 3D Vision (3DV)*. IEEE, 2021, pp. 1279–1289.

[28] Z. Deng, J. Shi, and J. Zhu, "Neuralef: Deconstructing kernels by deep neural networks," in *International Conference on Machine Learning*. PMLR, 2022, pp. 4976–4992.

[29] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," *Advances in neural information processing systems*, vol. 20, 2007.

[30] ——, "Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning," *Advances in neural information processing systems*, vol. 21, 2008.

[31] F. X. X. Yu, A. T. Suresh, K. M. Choromanski, D. N. Holtmann-Rice, and S. Kumar, "Orthogonal random features," *Advances in neural information processing systems*, vol. 29, 2016.

[32] M. Munkhoeva, Y. Kapushev, E. Burnaev, and I. Oseledets, "Quadrature-based features for kernel approximation," *Advances in neural information processing systems*, vol. 31, 2018.

[33] D. P. Francis and K. Raimond, "Major advancements in kernel function approximation," *Artificial Intelligence Review*, vol. 54, no. 2, pp. 843–876, 2021.

[34] C. Williams and M. Seeger, "Using the nyström method to speed up kernel machines," *Advances in neural information processing systems*, vol. 13, 2000.

[35] T. Yang, Y.-F. Li, M. Mahdavi, R. Jin, and Z.-H. Zhou, "Nyström method vs random fourier features: A theoretical and empirical comparison," *Advances in neural information processing systems*, vol. 25, 2012.

[36] C. K. Williams and C. E. Rasmussen, *Gaussian processes for machine learning*. MIT press Cambridge, MA, 2006, vol. 2, no. 3.

[37] M. G. Genton, "Classes of kernels for machine learning: a statistics perspective," *Journal of machine learning research*, vol. 2, no. Dec, pp. 299–312, 2001.

[38] E. Solak, R. Murray-Smith, W. Leithead, D. Leith, and C. Rasmussen, "Derivative observations in gaussian process models of dynamic systems," *Advances in neural information processing systems*, vol. 15, 2002.

[39] J. Park, Q.-Y. Zhou, and V. Koltun, "Colored point cloud registration revisited," in *Proc. of the IEEE intl. conf. on computer vision*, 2017, pp. 143–152.

[40] S. Peng, K. Genova, C. Jiang, A. Tagliasacchi, M. Pollefeys, T. Funkhouser *et al.*, "Openscene: 3d scene understanding with open vocabularies," *arXiv preprint arXiv:2211.15654*, 2022.

[41] N. M. M. Shafiullah, C. Paxton, L. Pinto, S. Chintala, and A. Szlam, "Clip-fields: Weakly supervised semantic fields for robotic memory," *arXiv preprint arXiv:2210.05663*, 2022.

[42] C. Huang, O. Mees, A. Zeng, and W. Burgard, "Visual language maps for robot navigation," in *ICRA*, 2023.

[43] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, "Scannet: Richly-annotated 3d reconstructions of indoor scenes," in *Proc. of the IEEE conf. on computer vision and pattern recognition*, 2017, pp. 5828–5839.

[44] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of rgb-d slam systems," in *2012 IEEE/RSJ intl. conf. on intelligent robots and systems*. IEEE, 2012, pp. 573–580.

[45] H. Zhang and K. Dana, "Multi-style generative network for real-time transfer," in *Proc. of the European Conf. on Computer Vision (ECCV) Workshops*, 2018, pp. 0–0.

[46] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *Advances in neural information processing systems*, vol. 30, 2017.

[47] I. Armeni, O. Sener, A. R. Zamir, H. Jiang, I. Brilakis, M. Fischer, and S. Savarese, "3d semantic parsing of large-scale indoor spaces," in *Proc. of the IEEE conf. on computer vision and pattern recognition*, 2016, pp. 1534–1543.

[48] Occipital, "Occipital: The structure sensor," 2016.

[49] I. Armeni, S. Sax, A. R. Zamir, and S. Savarese, "Joint 2d-3d-semantic data for indoor scene understanding," *arXiv preprint arXiv:1702.01105*, 2017.

[50] A. Rosinol, J. J. Leonard, and L. Carlone, "Probabilistic volumetric fusion for dense monocular slam," in *Proc. of the IEEE/CVF Winter Conf. on Applications of Computer Vision*, 2023, pp. 3097–3105.

[51] A. Cheraghian, S. Rahman, and L. Petersson, "Zero-shot learning of 3d point cloud objects," in *2019 16th Intl. Conf. on Machine Vision Applications (MVA)*. IEEE, 2019, pp. 1–6.

[52] A. Frome, G. S. Corrado, J. Shlens, S. Bengio, J. Dean, M. Ranzato, and T. Mikolov, "Devise: A deep visual-semantic embedding model," *Advances in neural information processing systems*, vol. 26, 2013.

[53] B. Michele, A. Boulch, G. Puy, M. Bucher, and R. Marlet, "Generative zero-shot learning for semantic segmentation of 3d point clouds," in *2021 Intl. Conf. on 3D Vision (3DV)*. IEEE, 2021, pp. 992–1002.

[54] Q.-Y. Zhou, J. Park, and V. Koltun, "Open3d: A modern library for 3d data processing," *arXiv preprint arXiv:1801.09847*, 2018.

[55] T. Schops, T. Sattler, and M. Pollefeys, "Bad slam: Bundle adjusted direct rgb-d slam," in *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, 2019, pp. 134–144.
[56] T. Whelan, M. Kaess, M. Fallon, H. Johannsson, J. Leonard, and J. McDonald, "Kintinuous: Spatially extended kinectfusion," 2012.
[57] R. Mur-Artal and J. D. Tardós, "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras," *IEEE transactions on robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.
[58] Z. Teed and J. Deng, "Droid-slam: Deep visual slam for monocular, stereo, and rgb-d cameras," *Advances in neural information processing systems*, vol. 34, pp. 16 558–16 569, 2021.
[59] T. Müller, A. Evans, C. Schied, and A. Keller, "Instant neural graphics primitives with a multiresolution hash encoding," *ACM Transactions on Graphics (ToG)*, vol. 41, no. 4, pp. 1–15, 2022.
[60] T. Kim, K. Kim, J. Lee, D. Cha, J. Lee, and D. Kim, "Revisiting image pyramid structure for high resolution salient object detection," in *Proc. of the Asian Conf. on Computer Vision*, 2022, pp. 108–124.

Yijun Yuan is a third year PhD student in Prof. Andreas Nüchter's group at Julius Maximilian University Würzburg. He received Bachelor's and Master's Degree from ShanghaiTech university in 2018 and 2021. Yijun has experience on Rescue Robotics and Metrical&Topological Mapping and in PhD studies he focuses on dense SLAM methods using various sensors.

Andreas Nüchter is professor and chair of robotics at University of Würzburg. Before fall 2022 he was associated professor (tenured) for telematics at University of Würzburg and before summer 2013 he headed as assistant professor the Automation group at Jacobs University Bremen. Prior he was a research associate at University of Osnabrück. Further past affiliations were with the Fraunhofer Institute for Autonomous Intelligent Systems (AIS, Sankt Augustin), the University of Bonn, from which he received the diploma degree in computer science in 2002 and the Washington State University. He holds a doctorate degree (Dr. rer. nat) from University of Bonn. Andreas works on robotics and automation, cognitive systems and artificial intelligence. His main research interests include reliable robot control, 3D environment mapping, 3D vision, and laser scanning technologies for various applications, e.g. for planetary exploration, safety security and rescue robotics, or underwater inspection. Andreas developed fast 3D scan matching algorithms that enable robots to perceive and map their environment in 3D representing the pose with 6 degrees of freedom. The capabilities of these robotic SLAM approaches were demonstrated at RoboCup Rescue competitions, ELROB and several other events. He is a member of the GI and the IEEE.