SNAKE: Shape-aware Neural 3D Keypoint Field

Chengliang Zhong[1,2,3] **Peixing You**[3] **Xiaoxue Chen**[3] **Hao Zhao**[4,5] **Fuchun Sun**[2]*
Guyue Zhou[3] **Xiaodong Mu**[1] **Chuang Gan**[6] **Wenbing Huang**[7,8]
[1]Xi'an Research Institute of High-Tech [2]THUAI, Tsinghua University
[3]AIR, Tsinghua University [4] Peking University [5]Intel Labs [6]MIT
[7]Gaoling School of Artificial Intelligence, Renmin University of China
[8]Beijing Key Laboratory of Big Data Management and Analysis Methods
zhongcl19@mails.tsinghua.edu.cn; hao.zhao@intel.com;
fcsun@tsinghua.edu.cn

Abstract

Detecting 3D keypoints from point clouds is important for shape reconstruction, while this work investigates the dual question: can shape reconstruction benefit 3D keypoint detection? Existing methods either seek salient features according to statistics of different orders or learn to predict keypoints that are invariant to transformation. Nevertheless, the idea of incorporating shape reconstruction into 3D keypoint detection is under-explored. We argue that this is restricted by former problem formulations. To this end, a novel unsupervised paradigm named SNAKE is proposed, which is short for **s**hape-aware **n**eur**a**l 3D **ke**ypoint field. Similar to recent coordinate-based radiance or distance field, our network takes 3D coordinates as inputs and predicts implicit shape indicators and keypoint saliency simultaneously, thus naturally entangling 3D keypoint detection and shape reconstruction. We achieve superior performance on various public benchmarks, including standalone object datasets ModelNet40, KeypointNet, SMPL meshes and scene-level datasets 3DMatch and Redwood. Intrinsic shape awareness brings several advantages as follows. (1) SNAKE generates 3D keypoints consistent with human semantic annotation, even without such supervision. (2) SNAKE outperforms counterparts in terms of repeatability, especially when the input point clouds are down-sampled. (3) the generated keypoints allow accurate geometric registration, notably in a zero-shot setting. Codes are available at https://github.com/zhongcl-thu/SNAKE.

1 Introduction

2D sparse keypoints play a vital role in reconstruction [34], recognition [23] and pose estimation [45], with scale invariant feature transform (SIFT) [20] being arguably the most important pre-Deep Learning (DL) computer vision algorithm. Altough dense alignment using photometric or featuremetric losses is also successful in various domains [2, 38, 8], sparse keypoints are usually preferred due to compactness in storage/computation and robustness to illumination/rotation. Just like their 2D counterparts, 3D keypoints have also drawn a lot of attention from the community in both pre-DL [13, 37] and DL [16, 1, 40] literature, with various applications in reconstruction [47, 43] and recognition[28, 36].

However, detecting 3D keypoints from raw point cloud data is very challenging due to sampling sparsity. No matter how we obtain raw point clouds (e.g., through RGB-D cameras [42], stereo [4], or LIDAR [10]), they are only a discrete representation of the underlying 3D shape. This fact drives us to explore the question of *whether jointly reconstructing underlying 3D shapes helps 3D*

*Corresponding author: Fuchun Sun.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

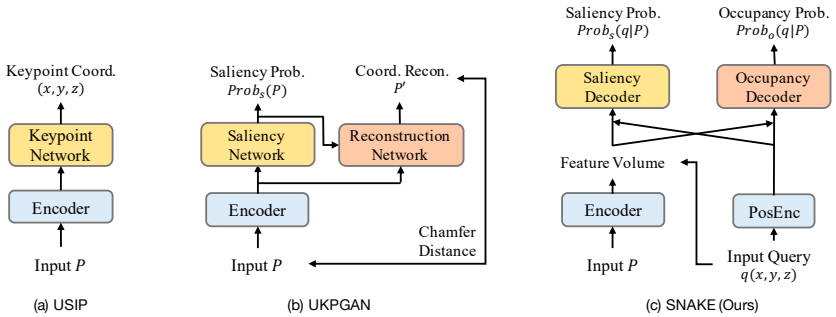

Figure 1: A comparison between existing 3D keypoint detection formulations and our newly proposed one. (a) USIP-like methods directly predict keypoint coordinates from input point clouds P. (b) UKPGAN-like methods predict saliency scores for P. Using chamfer distance, it reconstructs P coordinates based on the saliency scores and latent features. (c) Our SNAKE formulation predicts saliency probabilities and shape indicators simultaneously for each *continuous* query point q instead of *discrete* point clouds P. Sub-networks used for keypoint detection and reconstruction are shown in yellow and red, although they have different formulations. Here, the occupied points are those on the input surface.

keypoint detection. To our knowledge, former methods have seldom visited this idea. Traditional 3D keypoint detection methods are built upon some forms of first-order (e.g., density in intrinsic shape signature [46]) or second-order (e.g., curvature in mesh saliency [15]) statistics, including sophisticated reformulation like heat diffusion [35]. Modern learning-based methods rely upon the idea of consistency under geometric transformations, which can be imposed on either coordinate like USIP [16] or saliency value like D3Feat [1]. The most related method that studies joint reconstruction and 3D keypoint detection is a recent one named UKPGAN [40], yet it reconstructs input point cloud coordinates using an auxiliary decoder instead of the underlying shape manifold.

Why is this promising idea under-explored in the literature? We argue the reason is that former problem formulations are not naturally applicable for reconstructing the underlying shape surface. Existing paradigms are conceptually illustrated in Fig. 1. USIP-like methods directly output keypoint coordinates while UKPGAN-like methods generate saliency values for input point clouds. In both cases, the representations are based upon *discrete* point clouds. By contrast, we reformulate the problem using coordinate-based networks, as inspired by the recent success of neural radiance fields [22, 18, 31] and neural distance fields [24, 33]. As shown in Fig. 1-c, our model predicts a keypoint saliency value for each *continuous* input query point coordinate $q(x, y, z)$.

A direct advantage of this new paradigm is the possibility of tightly entangling shape reconstruction and 3D keypoint detection. As shown in Fig. 1-c, besides the keypoint saliency decoder, we attach a parallel shape indicator decoder that predicts whether the query point q is occupied. The input to decoders is feature embedding generated by trilinearly sampling representations conditioned on input point clouds P. Imagine a feature embedding at the wing tip of an airplane, if it can be used to reconstruct the sharp curvature of the wing tip, it can be naturally detected as a keypoint with high repeatability. As such, our method is named as **s**hape-aware **n**eur**a**l 3D **ke**ypoint field, or SNAKE.

Shape awareness, as the core feature of our new formulation, brings several advantages. (1) High repeatability. Repeatability is the most important metric for keypoint detection, *i.e.*, an algorithm should detect the same keypoint locations in two-view point clouds. If the feature embedding can successfully reconstruct the same chair junction from two-view point clouds, they are expected to generate similar saliency scores. (2) Robustness to down-sampling. When input point clouds are sparse, UKPGAN-like frameworks can only achieve reconstruction up to the density of inputs. In contrast, our SNAKE formulation can naturally reconstruct the underlying surface up to any resolution because it exploits coordinate-based networks. (3) Semantic consistency. SNAKE reconstructs the shape across instances of the same category, thus naturally encouraging semantic consistency although no semantic annotation is used. For example, intermediate representations need to be similar for successfully reconstructing different human bodies because human shapes are intrinsically similar.

To summarize, this study has the following two contributions:

- We propose a new network for joint surface reconstruction and 3D keypoint detection based upon implicit neural representations. During training, we develop several self-supervised losses that exploit the mutual relationship between two decoders. During testing, we design a gradient-based optimization strategy for maximizing the saliency of keypoints.
- Via extensive quantitative and qualitative evaluations on standalone object datasets ModelNet40, KeypointNet, SMPL meshes, and scene-level datasets 3DMatch and Redwood, we demonstrate that our shape-aware formulation achieves state-of-the-art performance under three settings: (1) semantic consistency; (2) repeatability; (3) geometric registration.

2 Related Work

3D Keypoint Detector As discussed in the introduction, 3D keypoint detection methods can be mainly categorized into hand-crafted and learning-based. Popular hand-crafted approaches [46, 32, 30] employ local geometric statistics to generate keypoints. These methods usually fail to detect consistent keypoints due to the lack of global context, especially under real-world disturbances, such as density variations and noise. USIP [16] is a pioneering learning-based 3D keypoint detector that outperforms traditional methods by a large margin. However, the detected keypoints are not semantically salient, and the number of keypoints is fixed. Fernandez et al. [9] exploit the symmetry prior to generate semantically consistent keypoints. But this method is category-specific, limiting the generalization to unseen categories and scenes. Recently, UKPGAN [40] makes use of reconstruction to find semantics-aware 3D keypoints. Yet, it recovers explicit coordinates instead of implicit shape indicators. As shown in Fig. 1, different from these explicit keypoint detection methods, we propose a new detection framework using implicit neural fields, which naturally incorporates shape reconstruction.

Implicit Neural Representation Our method exploits implicit neural representations to parameterize a continuous 3D keypoint field, which is inspired by recent studies of neural radiance fields [18, 22, 31] and neural distance fields [24, 33, 17, 44]. Unlike explicit 3D representations such as point clouds, voxels, or meshes, implicit neural functions can decode shapes continuously and learn complex shape topologies. To obtain fine geometry, ConvONet [26] proposes to use volumetric embeddings to get local instead of global features [21] of the input. Recently, similar local geometry preserving networks show a great success for the grasp pose generation [12] and articulated model estimation [11]. They utilize the synergies between their main tasks and 3D reconstruction using shared local representations and implicit functions. Unlike [11, 12] that learn geometry as an auxiliary task, our novel losses tightly couple surface occupancy and keypoint saliency estimates.

3 Method

This section presents SNAKE, a shape-aware implicit network for 3D keypoint detection. SNAKE conditions two implicit decoders (for shape and keypoint saliency) on shared volumetric feature embeddings, which is shown in Fig. 2-framework. To encourage repeatable, uniformly scattered, and sparse keypoints, we employ several self-supervised loss functions which entangle the predicted surface occupancy and keypoint saliency, as depicted in the middle panel of Fig. 2. During inference, query points with high saliency are further refined by gradient-based optimization since the implicit keypoint field is continuous and differentiable, which is displayed in Fig. 2-inference.

3.1 Network Architecture

Point Cloud Encoder As fine geometry is essential to local keypoint detection, we adopt the ConvONets [26], which can obtain local details and scale to large scenes, as the point cloud encoder denoted $f_{\theta_{en}}$ for SNAKE. Given an input point cloud $P \in \mathbb{R}^{N\times 3}$, our encoder firstly processes it with the PointNet++ [27] or alternatives like [49]) to get a feature embedding $Z \in \mathbb{R}^{N\times C_1}$, where N and C_1 are respectively the number of points and the dimension of the features. Then, these features are projected and aggregated into structured volume $Z' \in \mathbb{R}^{C_1\times H\times W\times D}$, where H, W and D are the number of voxels in three orthogonal axes. The volumetric embeddings serve as input to the 3D UNet [6] to further integrate local and global information, resulting in the output $G \in \mathbb{R}^{C_2\times H\times W\times D}$, where C_2 is the output feature dimension. More details can be found in the Appendix A.

Shape Implicit Decoder As shown in the top panel of Fig. 2, each point $q \in \mathbb{R}^3$ from a query set Q is encoded into a C_e-dimensional vector q_e via a multi-layer perceptron that is denoted the positional

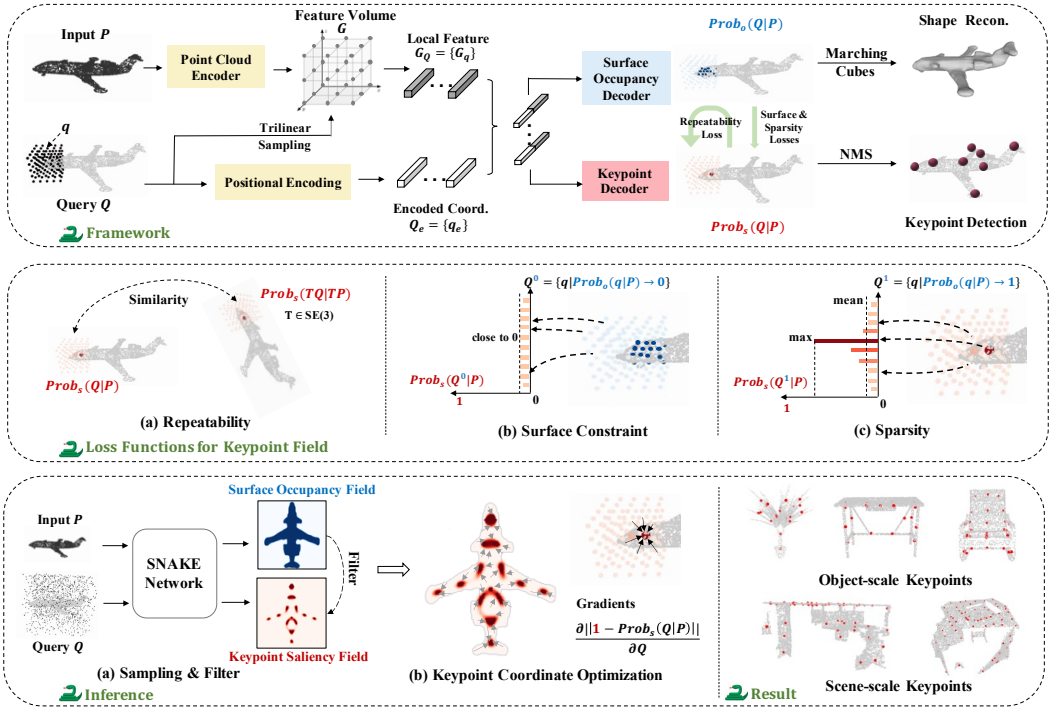

Figure 2: **Framework:** We use an implicit network to decode the surface occupancy and keypoint saliency probability simultaneously. Green arrows indicate the mutual relationships between the geometry and saliency field. Through marching cubes and non-maximum suppression (NMS), it could respectively recover the shape and detect keypoints from the input. **Loss functions for keypoint filed:** Three loss functions try to make the generated keypoint repeatable, located on the underlying surface, and sparse. **Inference:** We design a gradient-based optimization method to extract keypoints from the saliency field. **Result:** The object-scale and scene-scale keypoints after inference are displayed.

encoder $f_{\theta_{pos}}$, *i.e.* $q_e = f_{\theta_{pos}}(q)$. Then, the local feature G_q is retrieved from the feature volume G according to the coordinate of q via trilinear interpolation. The generated q_e and G_q are concatenated and mapped to the surface occupancy probability $Prob_o(q|P) \in [0, 1]$ by the occupancy decoder f_{θ_o}, as given in Eq. (1). If q is on the input surface, the $Prob_o(q|P)$ would be 1, otherwise be 0. In our formulation, the points inside the surface are also considered unoccupied.

$$f_{\theta_o}(q_e, G_q) \to Prob_o(q|P) \tag{1}$$

Keypoint Implicit Decoder Most of the process here is the same as in shape implicit decoder, except for the last mapping function. The goal of keypoint implicit decoder f_{θ_s} is to estimate the saliency of the query point q conditioned on input points P, which is denoted as $Prob_s(q|P) \in [0, 1]$ and formulated by:

$$f_{\theta_s}(q_e, G_q) \to Prob_s(q|P). \tag{2}$$

Here, saliency of the query point q is the likelihood that it is a keypoint.

3.2 Implicit Field Training

The implicit field is jointly optimized for surface occupancy and saliency estimation by several self-supervised losses. In contrast to former arts [12, 11] with a similar architecture that learn multiple tasks separately, we leverage the geometry knowledge from shape field to enhance the performance of keypoint field, as shown in the green arrows of Fig. 2. Specifically, the total loss is given by:

$$\mathcal{L} = \mathcal{L}_o + \mathcal{L}_r + \mathcal{L}_m + \mathcal{L}_s, \tag{3}$$

where $\mathcal{L}_o$ encourages the model to learn the shape from the sparse input, $\mathcal{L}_r$, $\mathcal{L}_m$ and $\mathcal{L}_s$ respectively help the predicted keypoint to be repeatable, located on the underlying surface and sparse.

Surface Occupancy Loss The binary cross-entropy loss l_{BCE} between the predicted surface occupancy $Prob_o(q|P)$ and the ground-truth label $Prob_o^{gt}$ is used for shape recovery. The queries Q are randomly sampled from the whole volume size $H \times W \times D$. The average over all queries is as follows:

$$\mathcal{L}_o = \frac{1}{|Q|} \sum_{q \in Q} l_{\mathrm{BCE}}\big(Prob_o(q|P), Prob_o^{gt}(q|P)\big), \tag{4}$$

where $|Q|$ is the number of queries Q.

Repeatability Loss Detecting keypoints with high repeatability is essential for downstream tasks like registration between two-view point clouds. That indicates the positions of keypoint are covariant to the rigid transformation of the input. To achieve a similar goal, 2D keypoint detection methods [29, 7, 45] enforce the similarity of corresponding local salient patches from multiple views. Inspired by them, we enforce the similarity of local overlapped saliency fields from two-view point clouds. Since the implicit field is continuous, we uniformly sample some values from a local field to represent the local saliency distribution. Specifically, as shown in the top and the middle part of Fig. 2, we build several local 3D Cartesian grids $\{Q_i\}_{i=1}^n$ with resolution of $H_l \times W_l \times D_l$ and size of $1/U$. We empirically set the resolution of Q_i to be almost the same as the feature volume G. As non-occupied regions are uninformative, the center of Q_i is randomly sampled from the input. Then, we perform random rigid transformation T on the P and Q_i to generate TP and TQ_i. Similar to [29], the cosine similarity, denoted as cosim, is exploited for the corresponding saliency grids of Q_i and TQ_i:

$$\mathcal{L}_r = 1 - \frac{1}{n} \sum_{i \in n} \mathrm{cosim}\big(Prob_s(Q_i|P), Prob_s(TQ_i|TP)\big). \tag{5}$$

Surface Constraint Loss As discussed in [16], 3D keypoints are encouraged to close to the input. They propose a loss to constrain the distance between the keypoint and its nearest neighbor from the input. Yet, the generated keypoints are inconsistent when given the same input but with a different density. Thanks to the shape decoder, SNAKE can reconstruct the underlying surface of the input, which is robust to the resolution change. Hence, we use the surface occupancy probability to represent the inverse distance between the query and the input. As can be seen in Fig. 2-(surface constraint), we enforce the saliency of the query that is far from input P close to 0, which is defined as

$$\mathcal{L}_m = \frac{1}{|Q|} \sum_{q \in Q} \big(1 - Prob_o(q|P)\big) \cdot Prob_s(q|P). \tag{6}$$

Sparsity Loss Similar to 2D keypoint detection methods [29], we design a sparsity loss to avoid the trivial solution ($Prob_s(Q|P)$=0) in Eq.(5)(6). As can be seen in Fig. 2, the goal is to maximize the local peakiness of the local saliency grids. As the sailency values of non-occupied points are enforced to 0 by $\mathcal{L}_m$, we only impose the sparsity loss on the points with high surface occupancy probability. Hence, we derive the sparsity loss with the help of decoded geometry by

$$\mathcal{L}_s = 1 - \frac{1}{n} \sum_{i \in n} \big(\max Prob_s(Q_i^1|P) - \mathrm{mean}\, Prob_s(Q_i^1|P)\big), \tag{7}$$

where $Q_i^1 = \{q | q \in Q_i, Prob_o(q|P) > 1 - thr_o\}$, $thr_o \in (0, 0.5]$ is a constant, and n is the number of grids. It is noted that the spatial frequency of local peakiness is dependent on the grid size $1/U$, see section 4.4. Since the network is not only required to find sparse keypoints, but also expected to recover the object shape, it would generate high saliency at the critical parts of the input, like joint points of a desk and corners of a house, as shown in the Fig. 2-result.

3.3 Explicit Keypoint Extraction

The query point q whose saliency is above a predefined threshold $thr_s \in (0, 1)$ would be selected as a keypoint at the inference stage. Although SNAKE can obtain the saliency of any query point, a higher resolution query set results in a high computational cost. Hence, as shown in Fig. 2-inference, we build a relatively low-resolution query sets Q_{infer} which are evenly distributed in the input space and further refine the coordinates of Q_{infer} by gradient-based optimization on this energy function:

$$E(Q_{\mathrm{infer}}, P) = \frac{1}{|Q_{\mathrm{infer}}|} \sum_{q \in Q_{\mathrm{infer}}} 1 - Prob_s(q|P). \tag{8}$$

Specifically, details of the explicit keypoint extraction algorithm are summarized in Alg. 1.

Algorithm 1 Optimization for Explicit Keypoint Extraction

Require: $P, Q_{\text{infer}}, f_{\theta_{en}}, f_{\theta_{pos}}, f_{\theta_o}, f_{\theta_s}$. Hyper-parameters: λ, J, thr_o, thr_s.
  Get initial $Prob_o(Q_{\text{infer}}|P)$ according to Eq.(1).
  Filter to get new query set $Q_{\text{infer}'} = \{q|q \in Q_{\text{infer}}, Prob_o(q|P) > 1 - thr_o\}$.
  for 1 to J **do**
    Evaluate energy function $E(Q_{\text{infer}'}, P)$.
    Update coordinates with gradient descent: $Q_{\text{infer}'} = Q_{\text{infer}'} - \lambda \nabla_{Q_{\text{infer}'}} E(Q_{\text{infer}'}, P)$.
  end for
  Sample final keypoints $Q_k = \{q|q \in Q_{\text{infer}'}, Prob_s(q|P) > thr_s\}$.

4 Experiment

In this section, we evaluate SNAKE under three settings. First, we compare keypoint semantic consistency across **different instances** of the same category, using both rigid and deformable objects. Next, keypoint repeatability of the **same instance** under disturbances such as SE(3) transformation, noise and downsample is evaluated. Finally, we inspect the point cloud registration task on the 3DMatch benchmark, notably in a zero-shot generalization setting. Besides, an ablation study is done to verify the effect of each design choice in SNAKE. The implementation details and hyper-parameters for SNAKE in three settings can be found in the Appendix B.

4.1 Semantic Consistency

Datasets The KeypointNet [41] dataset and meshes generated with the SMPL model [19] are utilized. KeypointNet has numerous human-annotated 3D keypoints for 16 object categories from ShapeNet [3]. The training set covers all categories that contain 5500 instances. Following [40], we evaluate 630 unseen instances from airplanes, chairs, and tables. SMPL is a skinned vertex-based deformable model that accurately captures body shape variations in natural human poses. We use the same strategy in [40] to generate both training and testing data.

Metric Mean Intersection over Union (mIoU) is adopted to show whether the keypoints across intra-class instances have the same semantics or not. For KeypointNet, a predicted keypoint is considered the same as a human-annotated semantic point if the geodesic distance between them is under some threshold. Due to the lack of human-labeled keypoints on SMPL, we compare the keypoint consistency in a pair of human models. A keypoint in the first model is regarded semantically consistent if the distance between its corresponding point and the nearest keypoint in the second model is below some threshold.

Evaluation and Results We compare SNAKE with random detection, hand-crafted detectors: ISS [46], Harris-3D [32] and SIFT-3D [30], and DL-based unsupervised detectors: USIP [16] and UKPGAN [40]. As USIP has not performed semantic consistency evaluations, we train the model with the code they provided. We follow the same protocols in [40] to filter the keypoints via NMS with a Euclidean radius of 0.1. Quantitative results are provided in Fig. 5-(a,e). SNAKE obtains higher mIoU than other methods under most thresholds on KeypointNet and SMPL. Qualitative results in Fig. 3 show our keypoints make good alignment with human annotations. Fig. 4 provides qualitative comparisons of semantically consistent keypoints on rigid and deformable objects. Owing to entangling shape reconstruction and keypoint detection, SNAKE can extract aligned representation for intra-class instances. Thus, our keypoints better outline the object shapes and are more semantically consistent under large shape variations. As shown in the saliency field projected slices, we can get symmetrical keypoints, although without any explicit constraint like the one used in [40]. Here, a projected slice is obtained by taking the maximum value of a given field along the projection direction.

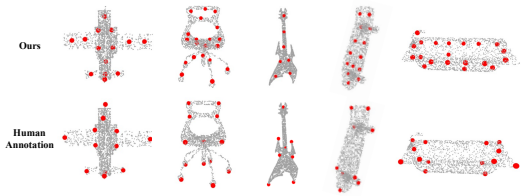

Figure 3: Comparison with human annotations on KeypointNet [41] dataset.

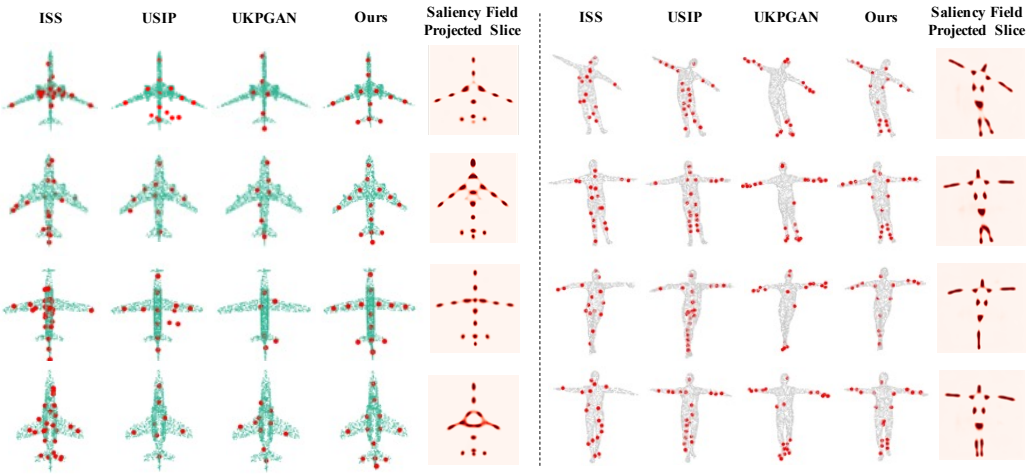

Figure 4: Semantic consistency of keypoints on rigid and deformable objects. Our keypoints are more evenly scattered on the underlying surface of objects, more symmetrical, and more semantically consistent under significant shape variations when compared to other methods. The saliency field projected slice shows that SNAKE decodes well-aligned saliency values for keypoints in different instances but with similar semantics, such as the wingtip of the airplane and the leg of the human. Here, small saliency is shown in bright red and gets darker with a larger value.

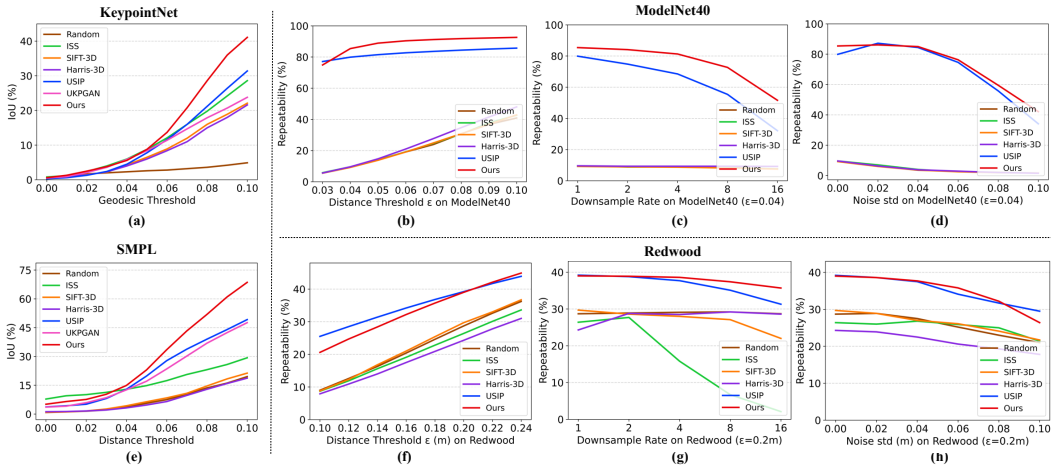

Figure 5: Quantitative results on four datasets. Keypoint semantic consistency (a)(e) on KeypointNet and SMPL. Relative repeatability for two-view point clouds with different distance threshold (b), downsample rate (c), Gaussian noise $\mathcal{N}(0, \sigma_{noise})$ (d) on ModelNet40. The results of (f)(g)(h) are tested on Redwood with the same settings in (b)(c)(d). The specific numerical results can be found in the Appendix C.2.

4.2 Repeatability

Datasets ModelNet40 [39] is a synthetic object-level dataset that contains 12,311 pre-aligned shapes from 40 categories, such as plane, guitar, and table. We adopt the official dataset split strategy. 3DMatch [43] and Redwood [5] are RGB-D reconstruction datasets for indoor scenes. Following [16], we train the model on 3DMatch and test it on Redwood to show the generalization performance. The training set contains around 19k samples and the test set consists of 207 point clouds.

Metric We adopt the relative repeatability proposed in USIP [16] as the evaluation metric. Given two point clouds captured from different viewpoints, a keypoint in the first point cloud is repeatable if its

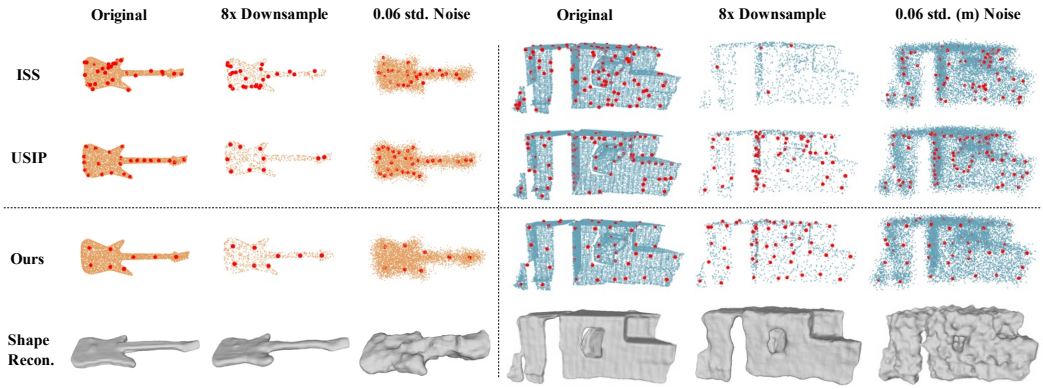

Figure 6: Visualization of keypoints under some disturbances on object-level [39] and scene-level [5] datasets compared to hand-crafted [46] and explicit representation based [16] methods. Downsample rate is 8x and the Gaussian noise scale (σ) is 0.06. The shape reconstruction via marching cubes for our occupancy field is also given. Visualization of repeatability can be found in the Appendix C.3.

distance to the nearest keypoint in the other point cloud is below a threshold ϵ. *Relative* repeatability means the number of repeatable points divided by the total number of detected keypoints.

Evaluation and Results Random detection, traditional methods and USIP are chosen as our baselines. Since UKPGAN does not provide pre-trained models on these two datasets, we do not report its results in Fig. 5 but make an additional comparison on KeypointNet, which is illustrated in the next paragraph. We use NMS to select the local peaky keypoints with a small radius (0.01 normalized distance on ModelNet40 and 0.04 meters on Redwood) for ours and baselines. We generate 64 keypoints in each sample and show the performance under different distance thresholds ϵ, downsample rates, and Gaussian noise scales. We set a fixed ϵ of 0.04 normalized distance and 0.2 meters on the ModelNet40 and Redwood dataset when testing under the last two cases. As shown in Fig. 5-(b,f), SNAKE outperforms state-of-the-art at most distance thresholds. We do not surpass USIP on Redwood in the lower thresholds. Note that it is challenging to get higher repeatability on Redwood because the paired inputs have very small overlapping regions. Fig. 5-(c,d,g,h) show the repeatability robustness to different downsample rates (d.r.) and Gaussian noise $N(0, \sigma)$ levels. SNAKE gets the highest repeatability in most cases because the shape-aware strategy helps the model reason about the underlying shapes of the objects/scenes, which makes keypoints robust to the input variations. Fig. 6 provides visualization of object-level and scene-level keypoints of the original and disturbed inputs. SNAKE can generate more consistent keypoints than other methods under drastic input changes.

We have tried to train UKPGAN (official implementation) on ModelNet40 and 3DMatch datasets from scratch but observed divergence under default hyper-parameters. As such, we provide a new experiment to compare their repeatability on the KeypointNet dataset, on which UKPGAN provided a pre-trained model. We randomly perform SE(3) transformation on the test point clouds to generate the second view point clouds. Then, we select top-32 salient keypoints with NMS (radius=0.03) in each sample and report the keypoint repeatability under different distance thresholds ϵ, downsample rates, and Gaussian noise scales. The results are summarized in Table 1, 2, which show that SNAKE achieves significant gains over UKPGAN in most cases. More discussions can be found in the Appendix C.1.

Table 1: Relative repeatability (%) with different distance thresholds ϵ on the KeypointNet dataset.

	0.03	0.05	0.07	0.09	0.10
UKPGAN	0.199	0.454	0.661	0.81	0.864
Ours	**0.643**	**0.806**	**0.892**	**0.936**	**0.948**

Table 2: Relative repeatability (%) when input point clouds are disturbed (ϵ=0.03). Here, ori. means the original input.

	ori.	d.r.=4	d.r.=8	σ=0.02	σ=0.03
UKPGAN	0.199	0.570	0.427	0.608	**0.558**
Ours	**0.643**	**0.594**	**0.525**	**0.626**	0.536

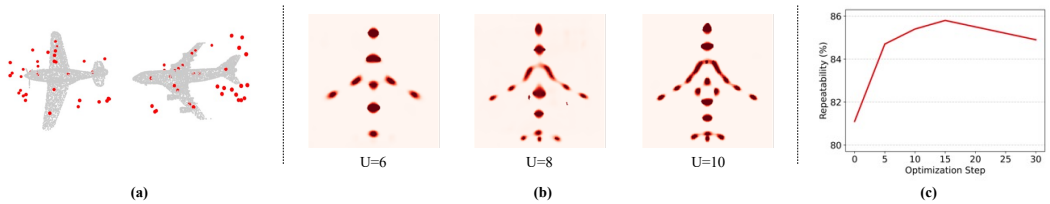

Figure 7: (a) SNAKE fails to predict semantically consistent keypoints without the occupancy decoder. (b) Saliency field slice with a different grid size of $(1/U)^3$. (c) The impact of the optimization step.

4.3 Zero-shot Point Cloud Registration

Datasets We follow the same protocols in [40] to train the model on KeypointNet and then directly test it on 3DMatch [43] dataset, evaluating how well two-view point clouds can be registered. The test set consists of 8 scenes which include some partially overlapped point cloud fragments and the ground truth SE(3) transformation matrices.

Metric To evaluate geometric registration, we need both keypoint detectors and descriptors. Thus, we combine an off-the-shelf and state-of-the-art descriptor D3Feat [1] with our and other keypoint detectors. Following [40], we compute three metrics: Feature Matching Recall, Inlier Ratio, and Registration Recall for a pair of point clouds.

Evaluation and Results As baselines, we choose random detection, ISS, SIFT-3D, UKPGAN, and D3Feat. Note that D3Feat is a task-specific learning-based detector trained on the 3DMatch dataset, thus not included in this zero-shot comparison. Ours and UKPGAN are trained on the synthetic object dataset KeypointNet only. The results are reported under different numbers of keypoints (*i.e.*, 2500, 1000, 500, 250, 100). The NMS with a radius of 0.05m is used for D3Feat, UKPGAN, and ours. As shown in Table 3, SNAKE outperforms other methods consistently under three metrics. For registration recall and inlier ratio, we achieve significant gains over UKPGAN and other traditional keypoint methods. Notably, when the keypoints are high in numbers, SNAKE even outperforms D3Feat which has seen the target domain. Local shape primitives like planes, corners, or curves may be shared between objects and scenes, so our shape-aware formulation allows a superior generalization from objects to scenes.

Table 3: Registration result on 3DMatch. We combine the off-the-shelf descriptor D3Feat [1] and different keypoint detectors to perform two-view point cloud registration.

		Feature Matching Recall (%)					Registration Recall (%)					Inlier Ratio (%)				
Detector	Descriptor	2500	1000	500	250	100	2500	1000	500	250	100	2500	1000	500	250	100
D3Feat	D3Feat	95.6	94.5	94.3	93.3	90.6	84.4	84.9	82.5	79.3	67.2	40.6	42.7	44.1	45.0	45.6
Random	D3Feat	95.1	94.5	92.8	90.0	81.2	83.0	80.0	77.0	65.5	38.8	38.6	33.6	28.9	23.6	17.3
ISS	D3Feat	95.2	94.4	93.4	90.1	81.0	83.5	79.2	76.0	64.3	37.2	38.2	33.5	28.8	23.9	17.4
SIFT	D3Feat	94.9	94.0	93.0	91.2	81.3	84.0	79.9	76.1	60.9	38.6	38.4	33.6	28.8	23.3	17.4
UKPGAN	D3Feat	94.7	94.2	93.5	92.6	85.9	82.8	81.4	77.1	69.7	47.4	38.8	35.5	34.0	33.1	27.7
Ours	D3Feat	**95.5**	**95.0**	**94.7**	**92.9**	**89.5**	**85.1**	**83.7**	**81.2**	**74.6**	**50.9**	**41.3**	**39.0**	**37.0**	**33.5**	**30.0**

4.4 Ablation Study

Loss Function Table 4 reports the performance w.r.t. designs of loss functions. (Row 1) If the surface occupancy decoder is removed, the surface constraint cannot be performed according to Eq.(6), so they are removed simultaneously. Although the model could detect significantly repeatable keypoints on ModelNet40 [39], it fails to give semantically consistent keypoints on KeypointNet [41]. Fig. 7-a shows that SNAKE is unable to output symmetric and meaningful keypoints without the shape-aware technique. That indicates the repeatability could not be the only criterion for keypoint detection if an implicit formulation is adopted. (Row 2-4) Each loss function for training keypoint field is vital for keypoint detection. Note that the model gives a trivial solution (0) for the saliency field and cannot extract distinctive points when removing the sparsity loss.

Grid Size and Volumetric Resolution The grid size $1/U$ controls the number of keypoints because $\mathcal{L}_s$ enforces the model to predict a single local maxima per grid of size $(1/U)^3$. Fig. 7-b shows

Table 4: Ablations for the designs of loss function. occ. = occupancy, sur. = surface, rep. = repeatability, spa. = sparsity and rr. = relative repeatability.

	rr. (%) on [39]			mIoU (%) on [41]		
Threshold ϵ	0.04	0.05	0.06	0.08	0.09	0.1
w/o occ. & sur.	**0.92**	**0.94**	**0.95**	0.22	0.25	0.28
w/o sur.	0.28	0.36	0.42	**0.31**	0.35	0.39
w/o rep.	0.22	0.28	0.34	0.30	0.35	0.39
w/o spa.	0	0	0	0	0	0
w/ all	0.85	0.89	0.90	0.30	**0.37**	**0.42**

Table 5: Impact of different local grid size used in the $\mathcal{L}_o$ and $\mathcal{L}_s$ on ModelNet40.

U	4	6	8	10
rr. (%) (ϵ=0.04)	0.79	**0.85**	0.79	0.77

Table 6: Impact of different global volumetric resolution on ModelNet40.

$H(=W=D)$	32	48	64	80
rr. (%) (ϵ=0.04)	0.62	0.79	**0.85**	0.78

different saliency field slices obtained from the same input with various $1/U$. When U is small, SNAKE outputs fewer salient responses, and more for larger values of U. We also give the relative repeatability results on ModelNet40 under distance threshold $\epsilon = 0.04$ in Table 5, indicating that $U = 6$ gives the best results. From Table 6, we can see that higher resolution improves performance. However, the performance drops when it reaches the resolution of 80. The potential reason is as such: the number of queries in a single grid increases when the resolution becomes higher, as mentioned in 3.2. In this case, finer details make the input to cosine similarity too long and contain spurious values.

Optimization Step and Learning Rate Fig. 7-c shows the importance of optimization (see Alg. 1) for refining keypoint coordinates on the ModelNet40 dataset. It is noted that too many optimization steps will not bring more gains but increase the computational overhead. In this paper, we set the number of update steps to 10. The learning rate for optimization is also key to the final result. When the learning rate is set to 0.1, 0.01, 0.001 and 0.0001, the relative repeatability (%) on ModelNet40 dataset with the same experimental settings as Table 6 are 0.002, 0.622, 0.854 and 0.826, respectively. In addition, the comparison of computation cost of baselines and ours can be found in the Appendix D.

5 Conclusion and Discussion

We propose SNAKE, a method for 3D keypoint detection based on implicit neural representations. Extensive evaluations show our keypoints are semantically consistent, repeatable, robust to downsample, and generalizable to unseen scenarios. **Limitations.** The optimization for keypoint extraction during inference requires considerable computational cost and time, which may not be applicable for use in scenarios that require real-time keypoint detection. **Negative Social Impact.** The industry may use the method for pose estimation in autonomous robots. Since our method is not perfect, it may lead to wrong decision making and potential human injury.

6 Acknowledgments

This research is jointly supported by following projects: the National Science and Technology Major Project of the Ministry of Science and Technology of China (No.2018AAA0102900); the Key Field R&D Program of Guangdong Province (No.2021B0101410002); Sino-German Collaborative Research Project Crossmodal Learning (NSFC 61621136008/DFG SFB/TRR169); the National Natural Science Foundation of China (No.62006137); Beijing Outstanding Young Scientist Program (No.BJJWZYJH012019100020098). We would like to thank Pengfei Li for discussions about implicit field learning. We would also like to thank the anonymous reviewers for their insightful comments.

References

[1] Xuyang Bai, Zixin Luo, Lei Zhou, Hongbo Fu, Long Quan, and Chiew-Lan Tai. D3feat: Joint learning of dense detection and description of 3d local features. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6359–6367, 2020.

[2] Simon Baker and Iain Matthews. Lucas-kanade 20 years on: A unifying framework. *International journal of computer vision*, 56(3):221–255, 2004.

[3] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

[4] Xuelian Cheng, Yiran Zhong, Mehrtash Harandi, Yuchao Dai, Xiaojun Chang, Hongdong Li, Tom Drummond, and Zongyuan Ge. Hierarchical neural architecture search for deep stereo matching. *Advances in Neural Information Processing Systems*, 33:22158–22169, 2020.

[5] Sungjoon Choi, Qian-Yi Zhou, and Vladlen Koltun. Robust reconstruction of indoor scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5556–5565, 2015.

[6] Özgün Çiçek, Ahmed Abdulkadir, Soeren S Lienkamp, Thomas Brox, and Olaf Ronneberger. 3d u-net: learning dense volumetric segmentation from sparse annotation. In *International conference on medical image computing and computer-assisted intervention*, pages 424–432. Springer, 2016.

[7] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 224–236, 2018.

[8] Jakob Engel, Jurgen Sturm, and Daniel Cremers. Semi-dense visual odometry for a monocular camera. In *Proceedings of the IEEE international conference on computer vision*, pages 1449–1456, 2013.

[9] Clara Fernandez-Labrador, Ajad Chhatkuli, Danda Pani Paudel, Jose J Guerrero, Cédric Demonceaux, and Luc Van Gool. Unsupervised learning of category-specific symmetric 3d keypoints from point sets. In *European Conference on Computer Vision*, pages 546–563. Springer, 2020.

[10] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[11] Zhenyu Jiang, Cheng-Chun Hsu, and Yuke Zhu. Ditto: Building digital twins of articulated objects from interaction. *arXiv preprint arXiv:2202.08227*, 2022.

[12] Zhenyu Jiang, Yifeng Zhu, Maxwell Svetlik, Kuan Fang, and Yuke Zhu. Synergies between affordance and geometry: 6-dof grasp detection via implicit representations. *Robotics: science and systems*, 2021.

[13] Andrew E Johnson and Martial Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE Transactions on pattern analysis and machine intelligence*, 21(5):433–449, 1999.

[14] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[15] Chang Ha Lee, Amitabh Varshney, and David W Jacobs. Mesh saliency. In *ACM SIGGRAPH 2005 Papers*, pages 659–666. 2005.

[16] Jiaxin Li and Gim Hee Lee. Usip: Unsupervised stable interest point detection from 3d point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 361–370, 2019.

[17] Pengfei Li, Yongliang Shi, Tianyu Liu, Hao Zhao, Guyue Zhou, and Ya-Qin Zhang. Semi-supervised implicit scene completion from sparse lidar. *arXiv preprint arXiv:2111.14798*, 2021.

[18] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. *Advances in Neural Information Processing Systems*, 33:15651–15663, 2020.

[19] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. Smpl: A skinned multi-person linear model. *ACM Trans. Graph.*, 34(6), oct 2015.

[20] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[21] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4460–4470, 2019.

[22] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European conference on computer vision*, pages 405–421. Springer, 2020.

[23] David Nister and Henrik Stewenius. Scalable recognition with a vocabulary tree. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 2161–2168. Ieee, 2006.

[24] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 165–174, 2019.

[25] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[26] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. In *European Conference on Computer Vision*, pages 523–540. Springer, 2020.

[27] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017.

[28] Hossein Rahmani, Arif Mahmood, Q Du Huynh, and Ajmal Mian. Hopc: Histogram of oriented principal components of 3d pointclouds for action recognition. In *European conference on computer vision*, pages 742–757. Springer, 2014.

[29] Jerome Revaud, Cesar De Souza, Martin Humenberger, and Philippe Weinzaepfel. R2d2: Reliable and repeatable detector and descriptor. *Advances in Neural Information Processing Systems*, 32, 2019.

[30] Blaine Rister, Mark A Horowitz, and Daniel L Rubin. Volumetric image registration from invariant keypoints. *IEEE Transactions on Image Processing*, 26(10):4900–4910, 2017.

[31] Katja Schwarz, Yiyi Liao, Michael Niemeyer, and Andreas Geiger. Graf: Generative radiance fields for 3d-aware image synthesis. *Advances in Neural Information Processing Systems*, 33:20154–20166, 2020.

[32] Ivan Sipiran and Benjamin Bustos. Harris 3d: a robust extension of the harris operator for interest point detection on 3d meshes. *The Visual Computer*, 27(11):963–976, 2011.

[33] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in Neural Information Processing Systems*, 33:7462–7473, 2020.

[34] Noah Snavely, Steven M Seitz, and Richard Szeliski. Modeling the world from internet photo collections. *International journal of computer vision*, 80(2):189–210, 2008.

[35] Jian Sun, Maks Ovsjanikov, and Leonidas Guibas. A concise and provably informative multi-scale signature based on heat diffusion. In *Computer graphics forum*, volume 28, pages 1383–1392. Wiley Online Library, 2009.

[36] Supasorn Suwajanakorn, Noah Snavely, Jonathan J Tompson, and Mohammad Norouzi. Discovery of latent 3d keypoints via end-to-end geometric reasoning. *Advances in neural information processing systems*, 31, 2018.

[37] Federico Tombari, Samuele Salti, and Luigi Di Stefano. Performance evaluation of 3d keypoint detectors. *International Journal of Computer Vision*, 102(1):198–220, 2013.

[38] Philippe Weinzaepfel, Jerome Revaud, Zaid Harchaoui, and Cordelia Schmid. Deepflow: Large displacement optical flow with deep matching. In *Proceedings of the IEEE international conference on computer vision*, pages 1385–1392, 2013.

[39] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015.

[40] Yang You, Wenhai Liu, Yong-Lu Li, Weiming Wang, and Cewu Lu. Ukpgan: Unsupervised keypoint ganeration. *arXiv preprint arXiv:2011.11974*, 2020.

[41] Yang You, Yujing Lou, Chengkun Li, Zhoujun Cheng, Liangwei Li, Lizhuang Ma, Cewu Lu, and Weiming Wang. Keypointnet: A large-scale 3d keypoint dataset aggregated from numerous human annotations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13647–13656, 2020.

[42] Aviad Zabatani, Vitaly Surazhsky, Erez Sperling, Sagi Ben Moshe, Ohad Menashe, David H Silver, Zachi Karni, Alexander M Bronstein, Michael M Bronstein, and Ron Kimmel. Intel® realsense™ sr300 coded light depth camera. *IEEE transactions on pattern analysis and machine intelligence*, 42(10):2333–2345, 2019.

[43] Andy Zeng, Shuran Song, Matthias Nießner, Matthew Fisher, Jianxiong Xiao, and Thomas A. Funkhouser. 3dmatch: Learning local geometric descriptors from rgb-d reconstructions. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 199–208, 2017.

[44] Hao Zhao, Rene Ranftl, Yurong Chen, and Hongbin Zha. Transferable end-to-end room layout estimation via implicit encoding. *arXiv preprint arXiv:2112.11340*, 2021.

[45] Chengliang Zhong, Chao Yang, Fuchun Sun, Jinshan Qi, Xiaodong Mu, Huaping Liu, and Wenbing Huang. Sim2real object-centric keypoint detection and description. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 5440–5449, 2022.

[46] Yu Zhong. Intrinsic shape signatures: A shape descriptor for 3d object recognition. In *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*, pages 689–696. IEEE, 2009.

[47] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Fast global registration. In *European conference on computer vision*, pages 766–782. Springer, 2016.

[48] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3D: A modern library for 3D data processing. *arXiv:1801.09847*, 2018.

[49] Yi Zhou, Chenglei Wu, Zimo Li, Chen Cao, Yuting Ye, Jason Saragih, Hao Li, and Yaser Sheikh. Fully convolutional mesh autoencoder using efficient spatially varying kernels. *Advances in Neural Information Processing Systems*, 33:9251–9262, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 5.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Section 5.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See https://github.com/zhongcl-thu/SNAKE.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix B.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See Appendix C.2.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix D.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes] See Appendix E.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] See Appendix E.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] See Appendix E.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] See Appendix E.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

Appendix

A Network Architecture

Following [26], our implementation is a compilation of PointNet++ [27], 3D UNet [6], positional encoder and implicit surface occupancy decoder. The architecture of the implicit keypoint decoder is designed to be the same as the surface occupancy decoder. The dimensions of the feature embedding Z and Z' are both set to 32, *i.e.*, $C_1 = C_2 = 32$. And each point from a query set is also encoded into a 32-dimensional feature vector. More details can be found in the code we provide.

B Implementation Details

B.1 Training

SNAKE is implemented in PyTorch [25] using the Adam [14] optimizer with a mini-batch size of b on 4 NVIDIA A100 GPUs for el epochs. We use a learning rate of 10^{-4} for the first ef epochs, which is dropped ten times for the remainder. As discussed in Sec. 3.2 (repeatability loss), we perform random rigid transformation T on the input P to generate a second view input TP. Then, we use some data augmentation on TP to increase data diversity by downsampling with a random rate between 0 and 4, and Gaussian noise. Training hyper-parameters on four datasets are provided in Table 7.

In our formulation, occupied points are those on the input surface, and the others are considered all unoccupied, including the points inside the surface. Therefore, we can only use input point clouds to learn the surface occupancy model. Specifically, we randomly sample the positives from the input point cloud. The negatives are randomly sampled in the unit 3D space. Although some of the negatives are indeed on the surface of the object, their number is so limited compared to the whole query sets that they do not affect the training.

Table 7: **Training and testing hyper-parameters.** Sem.=Semantic consistency evaluation, Rep.=Repeatability evaluation, Reg.=Registration evaluation, KeypN.=KeypointNet [41], ModelN.=ModelNet40 [39].

Setting	Training Set	Test Set	N	$H/W/D$	$H_l/W_l/D_l$	U	n	b	ef/el	thr_o	thr_s	λ	J
Sem.	KeypN.	KeypN.	2048	64/64/64	8/8/8	8	500	16	40/60	0.5	0.7	10^{-3}	10
	SMPL [19]	SMPL	2048	64/64/64	8/8/8	8	500	16	20/30	0.5	0.7	10^{-3}	10
Rep.	ModelN.	ModelN.	5000	64/64/64	8/8/8	6	500	16	40/60	0.5	0.7	10^{-3}	10
	3DMatch [43]	Redwood [5]	10000	100/100/100	10/10/10	8	150	6	15/20	0.5	0.7	10^{-3}	10
Reg.	KeypN.	3DMatch	2048	64/64/64	6/6/6	12	500	16	40/60	0.5	0.4	10^{-3}	10

B.2 Testing

For the SMPL dataset, the correspondence between the paired point clouds can be generated by SMPL vertex index. Since the keypoint SNAKE generates may not be in the input point cloud (we enforce the keypoint scatter on the underlying surface of the input), we take the point closest to the generated keypoint in the input as the final keypoint. We use the same strategy on the 3DMatch dataset when performing geometric registration because D3feat [1] predicts descriptors for each point in the input. The testing hyper-parameters are shown in Table 7.

C Results

C.1 Additional comparison with UKPGAN on keypoint repeatability

Due to the absence of pretrained model on the ModelNet40 and 3DMatch dataset, we do not report the keypoint repeatability of UKPGAN [40] on the main paper. We have tried to train UKPGAN (official implementation) on the ModelNet40 and 3DMatch datasets from scratch but observed divergence under default hyper-parameters. The training always reports NaN losses in early epochs. This instability also implies limitations in implementing the idea of joint reconstruction and keypoint

detection with GAN-based methods. As such, we provide a new experiment to compare their repeatability on the KeypointNet dataset, on which the UKPGAN provided a pre-trained model.

Tabke 1 and Table 2 in the main paper show that SNAKE achieves significant gains over UKPGAN in most cases. Interestingly, when the inputs are disturbed, the performance of UKPGAN increases rather than decreases. Via visualizing the results in Fig. 8, we find that when the input point clouds are disturbed, the keypoints predicted by UKPGAN are clustered in a small area, which improves the repeatability of keypoints but fails to cover the input uniformly. This illustrates that the GAN-based method adopted by UKPGAN to control the keypoint sparsity is not robust to input point cloud disturbance. The keypoints of ours still remain meaningful under the drastic changes of inputs.

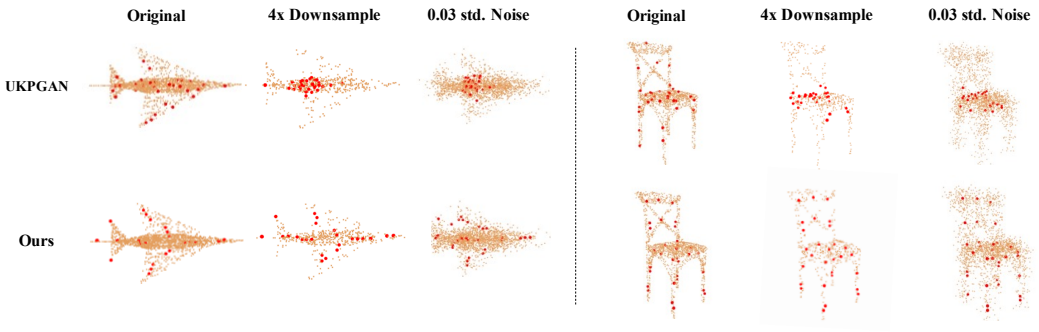

Figure 8: Keypoints of the KeypointNet data under some input disturbances.

C.2 Quantitative Results

The specific numerical results on semantic consistency and repeatability are summarized in Table 9-15, which correspond to Figure 5 in the main paper. We present the mean and standard deviation of our results over 6 models trained under different random seeds.

C.3 Qualitative Visualization of Saliency Field and Keypoints

We show more qualitative results on keypoint semantic consistency between intra-class instances from rigid objects plane, guitar, motorcycle, and deformable human shapes in Figure 10- 13. Owing to entangling shape reconstruction and keypoint detection, SNAKE can extract aligned representation for intra-class instances. As shown in Figure 14- 19, we provide more visualizations of keypoints under some disturbances on object-level (ModelNet40) and scene-level (Redwood) datasets. It can be seen that SNAKE can generate more consistent keypoints than other methods under significant variations of inputs. We also show the detected keypoints of the same object/scene from different views to demonstrate the repeatability of keypoint in Figure 20- 22.

C.4 Qualitative Visualization of Surface Occupancy Field and Shape Reconstruction

As shown in Figure 9, we show visualizations of the occupancy field and shape reconstruction on the ModelNet40 dataset. These five samples are taken from the unseen test set. As shown by the second row, only points on the input surface have a high occupancy value, and the other points (inside or outside of the surface) have a near-zero occupancy value. Under our definition, two surfaces can be obtained through the marching cube, and we only show the outer surface.

D Computation Cost

As shown in Table 8, we report the time taken to generate keypoints of hand-crafted detector ISS, deep-learning (DL) based methods USIP [16], UKPGAN [40] and ours. ISS [46] is implemented by Open3d [48] and deployed on an AMD EPYC 7742 64-Core CPU. DL-based methods are deployed on an NVIDIA GeForce RTX 3090 GPU. USIP requires the lowest computational time to generate keypoints, while UKPGAN requires the highest cost since it takes much time to compute smoothed density values. The inference time of our model is comparable to ISS when we do not refine the

keypoint by optimization (J=0), and the repeatability is still as high as around 81% when the input point number is 4096. The time increases with the increasing number of optimization iterations J. As discussed before, when J becomes larger (below 15), the performance of keypoint gets better. It suggests that there is a trade-off between keypoint performance and inference speed of our method. The GPU memory cost (MB) for USIP, UKPGAN, and SNAKE during a single batch inference is 3747, 10727, and 2785, which illustrates that SNAKE requires the lowest GPU memory cost to generate keypoints.

Table 8: Average time (s) taken to compute keypoints from input point clouds on ModelNet40 dataset. The hyper-parameters of ours can be found in the Table 7. Decimals in parentheses in italics are relative repeatability (%). Here, the experiment setting is the same as in Sec. 4.2.

Input Point #	ISS	USIP	UKPGAN	Ours		
				J=0	J=5	J=10
2048	0.07 (*0.088*)	0.006 (*0.748*)	14.41	0.08 (*0.795*)	0.50 (*0.835*)	0.81 (*0.851*)
4096	0.11 (*0.096*)	0.007 (*0.799*)	36.80	0.09 (*0.811*)	0.50 (*0.850*)	0.83 (*0.864*)

E Illustrations on the Assets We Used and Released

The license of assets we used is as follows: (a) MIT License for KeypointNet dataset. (b) Software Copyright License for non-commercial scientific research purposes on SMPL-Model. (c) GPL-3.0 License for ModelNet40, 3DMatch, Redwood dataset, and USIP. (d) Microsoft research license for 3DMatch registration benchmark.

All existing datasets and codes we used in this paper are allowed for research and do not contain personally identifiable information or offensive content. Note that SMPL only has human shapes without the identity information of the person, such as the face or body texture. Our code is released under the MIT license.

Table 9: mIoU (%) with different geodesic distance thresholds on the KeypointNet dataset. This table corresponds to Figure 5-(a) in the main paper.

	0.00	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09	0.1
Random	0.005	0.010	0.017	0.020	0.023	0.026	0.028	0.032	0.036	0.042	0.049
ISS	**0.008**	**0.012**	0.024	**0.040**	**0.060**	0.088	0.121	0.160	0.198	0.242	0.286
SIFT3D	0.005	0.010	0.015	0.022	0.043	0.065	0.089	0.120	0.160	0.189	0.221
Harris3D	0.005	0.010	0.014	0.023	0.040	0.060	0.084	0.110	0.150	0.180	0.216
USIP	0.003	0.006	0.013	0.024	0.045	0.078	0.116	0.160	0.212	0.264	0.314
UKPGAN	0.005	0.009	0.021	0.036	0.059	0.084	0.114	0.147	0.179	0.207	0.238
Ours	0.006±0.000	**0.012±0.000**	**0.025±0.001**	0.039±0.001	0.058±0.001	**0.091±0.002**	**0.144±0.005**	**0.214±0.005**	**0.291±0.005**	**0.361±0.002**	**0.412±0.002**

Table 10: mIoU (%) with different Euclidean distance thresholds on SMPL mesh. This table corresponds to Figure 5-(e) in the main paper.

	0.00	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09	0.1
Random	0.008	0.011	0.015	0.021	0.038	0.056	0.075	0.103	0.136	0.161	0.195
ISS	**0.078**	**0.095**	**0.101**	0.113	0.129	0.148	0.174	0.206	0.231	0.258	0.293
SIFT3D	0.009	0.011	0.016	0.026	0.043	0.064	0.084	0.108	0.146	0.183	0.213
Harris3D	0.012	0.013	0.016	0.021	0.032	0.047	0.065	0.097	0.129	0.159	0.187
USIP	0.037	0.043	0.051	0.081	0.129	0.198	0.278	0.338	0.390	0.440	0.492
UKPGAN	0.036	0.041	0.059	0.085	0.126	0.171	0.235	0.302	0.369	0.424	0.476
Ours	0.063±0.018	0.079±0.019	0.094±0.023	**0.128±0.028**	**0.182±0.036**	**0.255±0.041**	**0.355±0.041**	**0.457±0.046**	**0.557±0.043**	**0.639±0.037**	**0.704±0.036**

Table 11: Relative repeatability (%) with different distance thresholds on the ModelNet40 dataset. This table corresponds to Figure 5-(b) in the main paper.

	0.03	0.04	0.05	0.06	0.07	0.08	0.09	0.1
Random	0.056	0.094	0.14	0.191	0.249	0.308	0.368	0.429
ISS	0.058	0.096	0.14	0.192	0.247	0.306	0.367	0.427
SIFT3D	0.055	0.092	0.138	0.191	0.249	0.308	0.369	0.429
Harris3D	0.056	0.096	0.147	0.21	0.277	0.347	0.415	0.48
USIP	**0.771**	0.799	0.815	0.827	0.836	0.844	0.851	0.857
Ours	0.763±0.011	**0.864±0.009**	**0.897±0.007**	**0.910±0.005**	**0.917±0.005**	**0.923±0.005**	**0.927±0.005**	**0.930±0.005**

Table 12: Relative repeatability (%) when the input is randomly downsampled by some rates on the ModelNet40 dataset. This table corresponds to Figure 5-(c) in the main paper.

	1	2	4	8	16
Random	0.094	0.093	0.093	0.091	0.092
ISS	0.096	0.088	0.088	0.083	0.076
SIFT3D	0.092	0.089	0.087	0.082	0.075
Harris3D	0.096	0.093	0.093	0.093	0.092
USIP	0.799	0.748	0.685	0.554	0.321
Ours	**0.864±0.009**	**0.851±0.009**	**0.820±0.008**	**0.730±0.009**	**0.528±0.012**

Table 13: Relative repeatability (%) when the input is disturbed by Gaussian noise $N(0, \sigma)$ on the ModelNet40 dataset. This table corresponds to Figure 5-(d) in the main paper.

	0.00	0.02	0.04	0.06	0.08	0.10	0.12
Random	0.094	0.062	0.038	0.027	0.021	0.016	0.014
ISS	0.096	0.061	0.037	0.025	0.02	0.016	0.015
SIFT3D	0.092	0.06	0.036	0.025	0.019	0.016	0.014
Harris3D	0.096	0.063	0.038	0.029	0.02	0.015	0.015
USIP	0.799	**0.872**	**0.844**	0.746	0.558	0.341	0.192
Ours	**0.864±0.009**	0.869±0.008	0.841±0.015	**0.766±0.013**	**0.619±0.041**	**0.464±0.049**	**0.354±0.045**

Table 14: Relative repeatability (%) with the different distance thresholds (m) on the Redwood dataset. This table corresponds to Figure 5-(f) in the main paper.

	0.1	0.12	0.14	0.16	0.18	0.2	0.22	0.24
Random	0.09	0.126	0.163	0.204	0.246	0.287	0.326	0.362
ISS	0.087	0.119	0.156	0.191	0.228	0.264	0.301	0.336
SIFT3D	0.088	0.123	0.168	0.21	0.254	0.297	0.33	0.367
Harris3D	0.079	0.109	0.14	0.175	0.209	0.243	0.278	0.31
USIP	**0.255**	**0.285**	**0.314**	**0.342**	**0.368**	0.392	0.417	0.439
Ours	0.205±0.005	0.246±0.007	0.286±0.008	0.323±0.008	0.359±0.009	**0.393±0.010**	**0.425±0.010**	**0.454±0.009**

Table 15: Relative repeatability (%) when the input is randomly downsampled by some rates on the Redwood dataset. This table corresponds to Figure 5-(g) in the main paper.

	1	2	4	8	16
Random	0.287	0.289	0.291	0.292	0.287
ISS	0.264	0.277	0.158	0.067	0.021
SIFT3D	0.297	0.286	0.28	0.271	0.22
Harris3D	0.243	0.288	0.285	0.292	0.286
USIP	0.392	0.388	0.377	0.351	0.313
Ours	**0.393±0.010**	**0.394±0.008**	**0.391±0.009**	**0.381±0.008**	**0.362±0.007**

Table 16: Relative repeatability (%) when the input is disturbed by Gaussian noise $N(0, \sigma)$ on the Redwood dataset. This table corresponds to Figure 5-(h) in the main paper.

	0.00	0.02	0.04	0.06	0.08	0.10
Random	0.287	0.289	0.275	0.252	0.23	0.21
ISS	0.264	0.26	0.268	0.259	0.25	0.214
SIFT3D	0.297	0.289	0.27	0.261	0.241	0.217
Harris3D	0.243	0.239	0.225	0.206	0.193	0.178
USIP	0.392	0.386	0.375	0.341	0.317	**0.295**
Ours	**0.393±0.010**	**0.392±0.008**	**0.381±0.009**	**0.359±0.009**	**0.318±0.007**	0.256±0.013

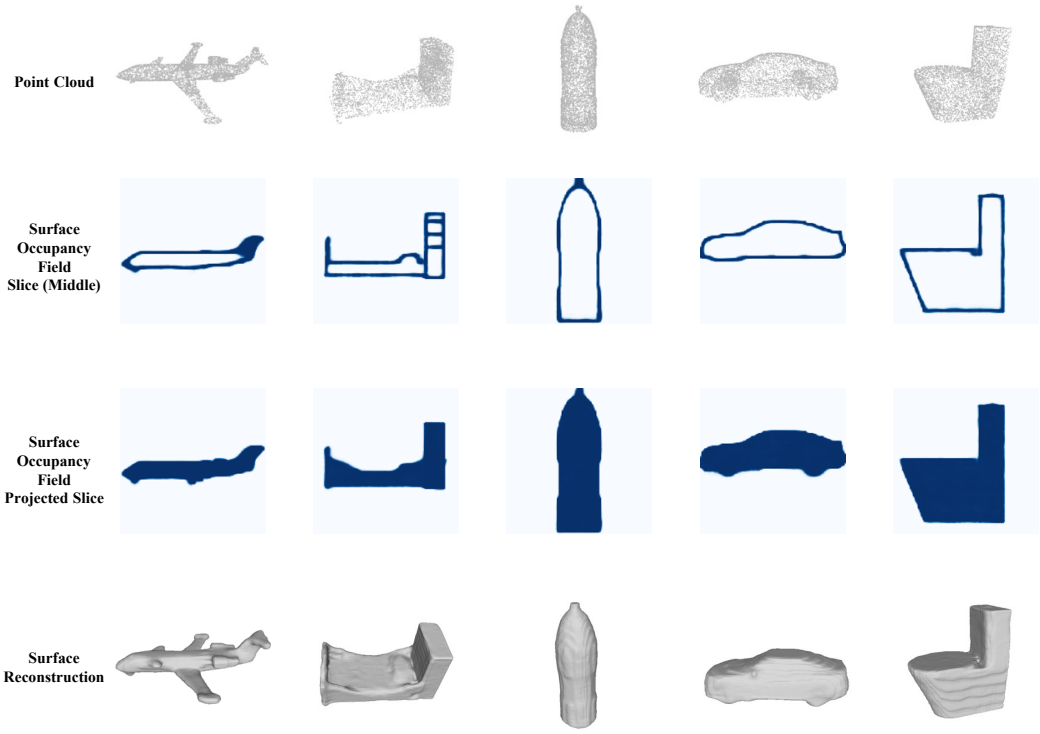

Figure 9: Visualization for surface occupancy field and surface reconstruction of test instances (unseen) from ModelNet40 dataset. The second row shows the middle slice of the surface occupancy field of these objects. The third row shows the projected surface occupancy field on the same slice by taking the maximum value. The fourth row shows the outer surface reconstructed by applying marching cubes on the surface occupancy field, using a threshold of 0.4.

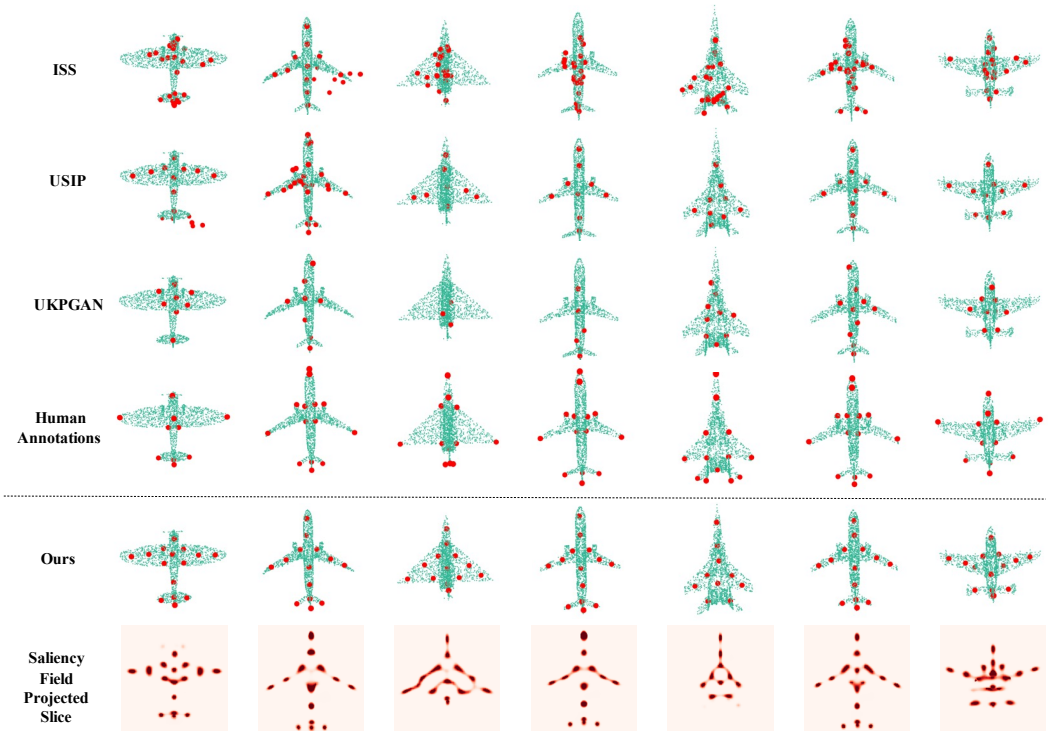

Figure 10: Keypoint semantic consistency comparison on the plane.

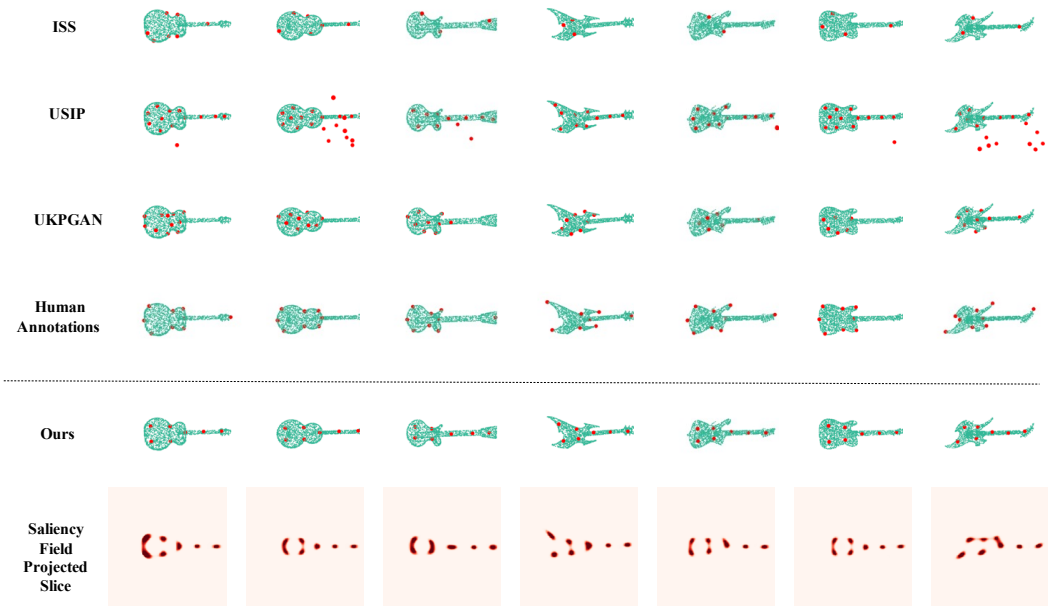

Figure 11: Keypoint semantic consistency comparison on the guitar.

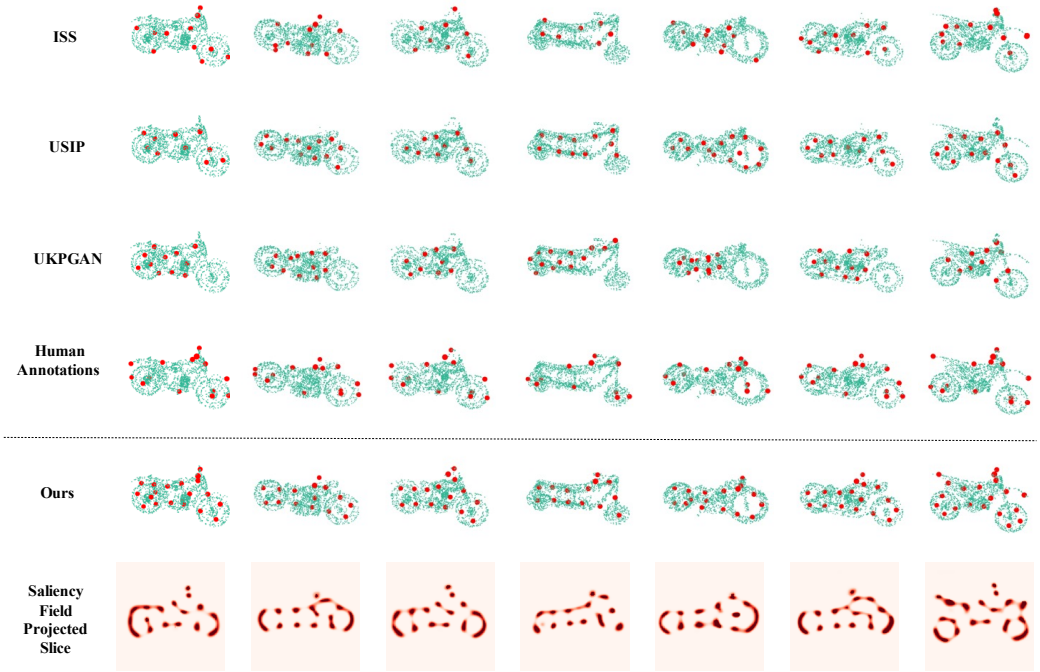

Figure 12: Keypoint semantic consistency comparison on the motorcycle.

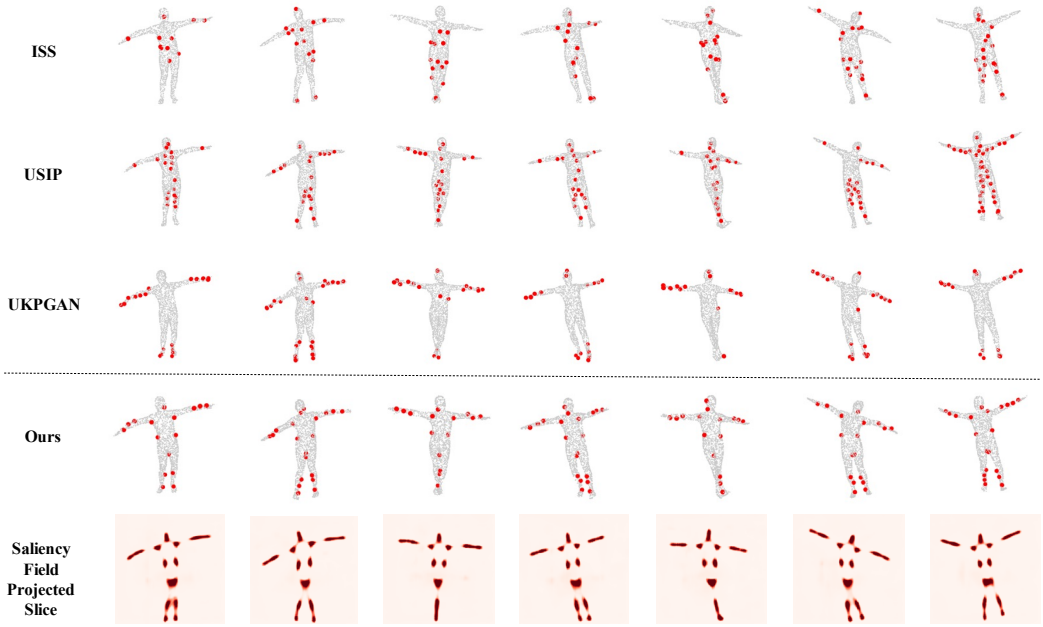

Figure 13: Keypoint semantic consistency comparison on the human shape.

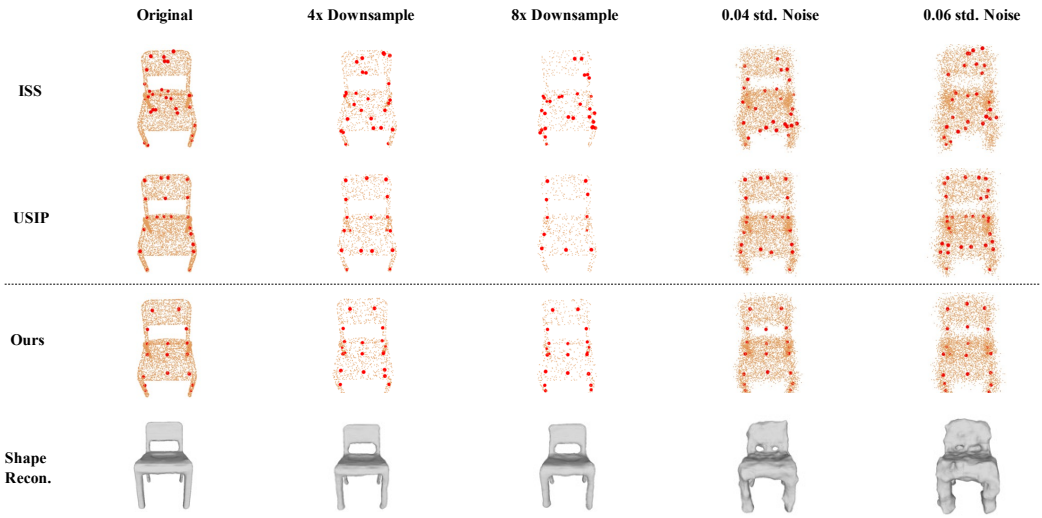

Figure 14: Keypoints of the chair under some input disturbances.

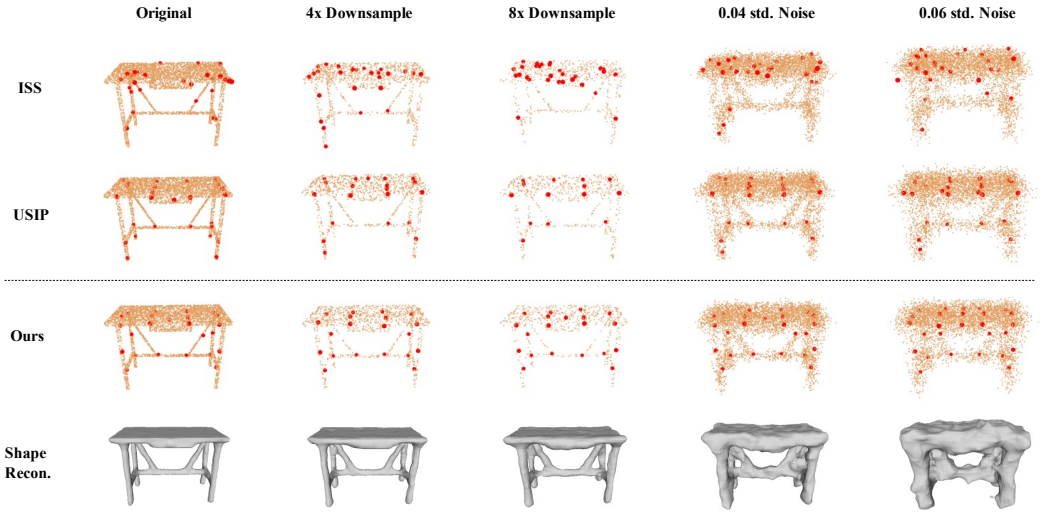

Figure 15: Keypoints of the desk under some input disturbances.

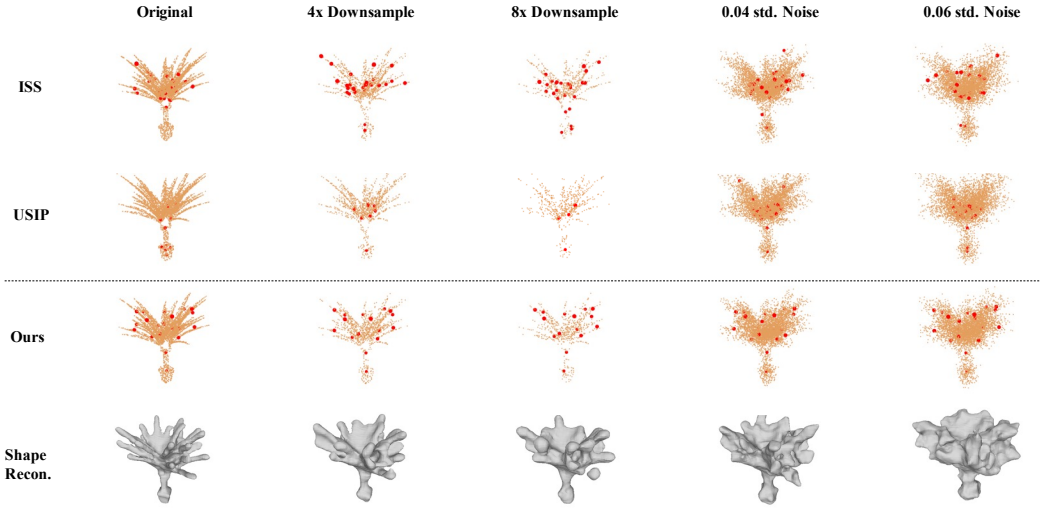

Figure 16: Keypoints of the flower under some input disturbances.

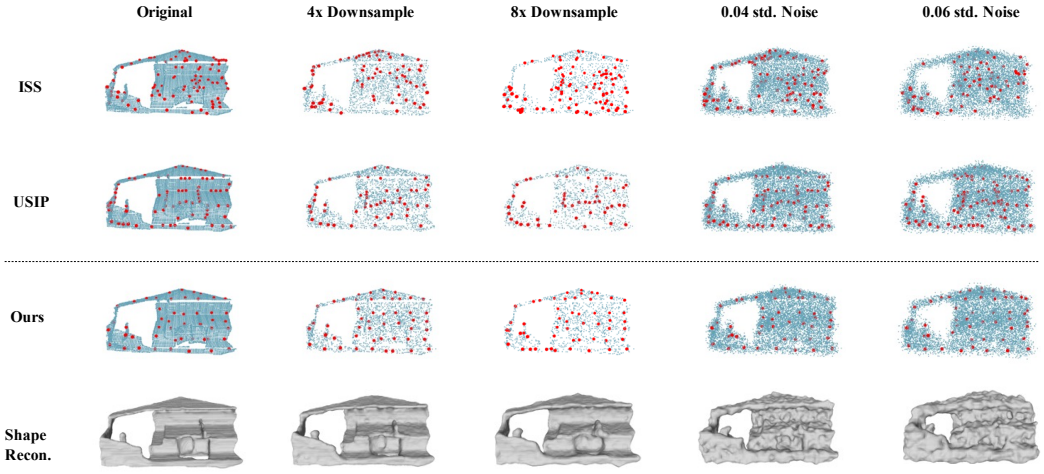

Figure 17: Keypoints of the indoor scene (1) under some input disturbances.

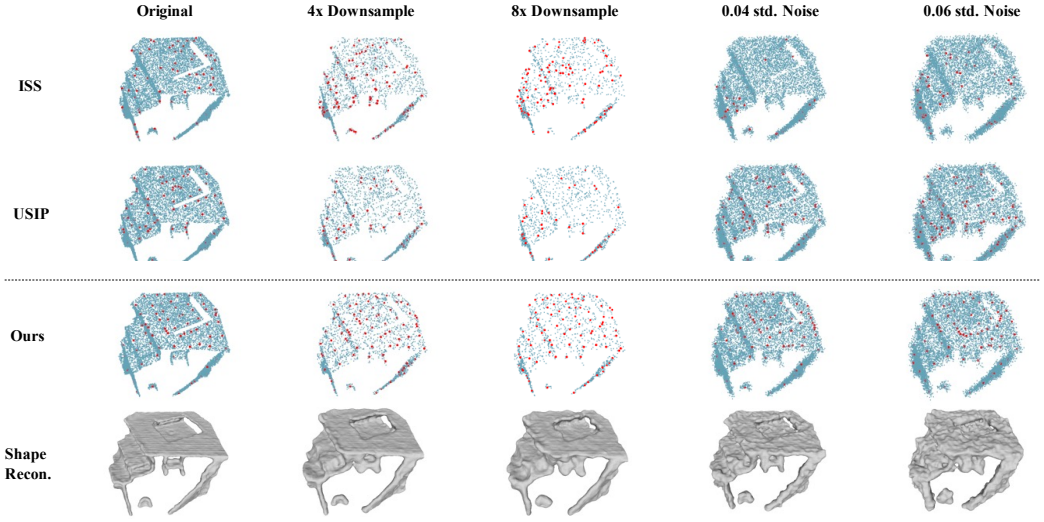

Figure 18: Keypoints of the indoor scene (2) under some input disturbances.

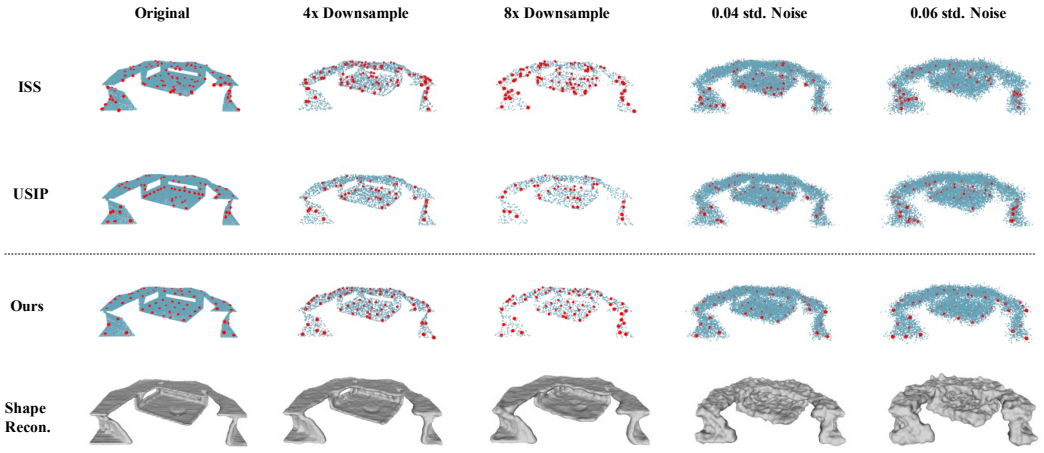

Figure 19: Keypoints of the indoor scene (3) under some input disturbances.

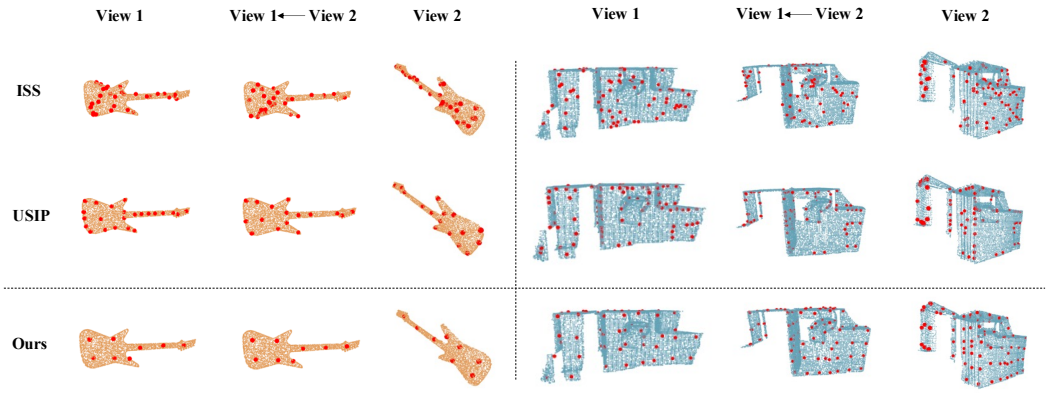

Figure 20: Keypoints repeatability comparison when the input is not corrupted. Note that in the Redwood dataset (right panel), two-view point clouds are partially overlapped.

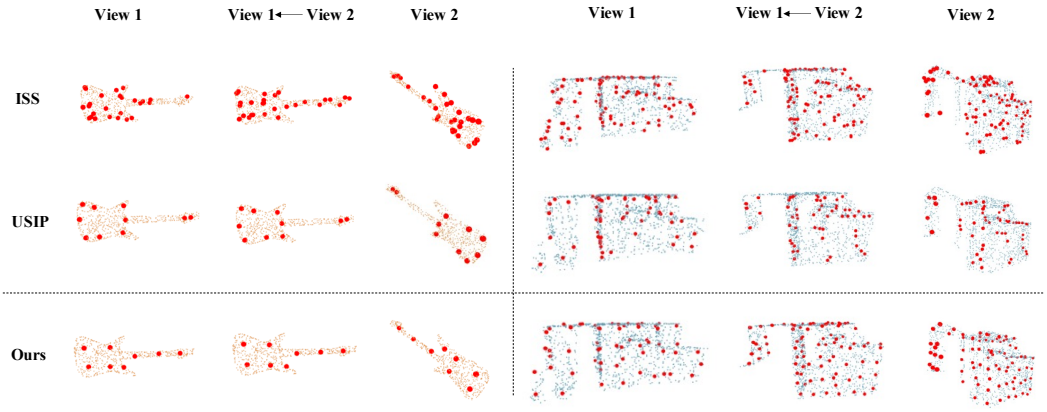

Figure 21: Keypoints repeatability comparison when the input is 8x down sampled. Note that in the Redwood dataset (right panel), two-view point clouds are partially overlapped.

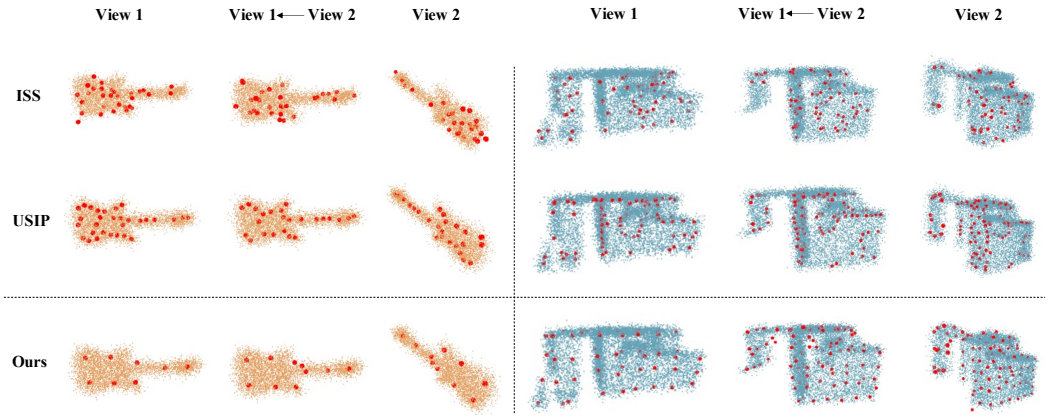

Figure 22: Keypoints repeatability comparison when the input is added Gaussion noises (std=0.06). Note that in the Redwood dataset (right panel), two-view point clouds are partially overlapped.