
Extending DeepSDF for automatic 3D shape retrieval and similarity transform estimation.

Oladapo A. Afolabi, Allen Y. Yang, S. Shankar Sastry

Department of Electrical Engineering and Computer Sciences
University of California, Berkeley
`{oafolabi,yang,sastry}@eecs.berkeley.edu`

Abstract

Recent advances in computer graphics and computer vision have found successful application of deep neural network models for 3D shapes based on signed distance functions (SDFs) that are useful for shape representation, retrieval, and completion. However, this approach has been limited by the need to have query shapes in the same canonical scale and pose as those observed during training, restricting its effectiveness on real world scenes. We present a formulation to overcome this issue by jointly estimating shape and similarity transform parameters. We conduct experiments to demonstrate the effectiveness of this formulation on synthetic and real datasets and report favorable comparisons to the state of the art. Finally, we also emphasize the viability of this approach as a form of 3D model compression.

1 Introduction

The rapid growth of low-cost LIDAR or RGB-D sensors (e.g., Intel RealSense and Apple iPad Pro LIDAR) has led to the growth of applications that require building 3D models of real-world scenes. Such models are useful for virtual immersion and automatic content creation in Augmented and Virtual Reality (AR/VR) where the user experience can be customized to individuals' own spaces. A typical pipeline of the task currently would include Simultaneous Localization and Mapping to generate an SDF representation of the space. Subsequently, mesh or pointcloud models of the scene may be extracted from the SDFs using variants of the Marching Cubes algorithm[25].

While very accurate models can be produced by this pipeline, it is also well known that the models may suffer from missing parts and holes due to occlusion and insufficient scanning coverage of the scene. Furthermore, pointcloud or mesh representations of the scene may not be organized in a way that is intuitive to edit (i.e., they are unordered), and may take up large amounts of memory to store. These characteristics present a significant bottleneck for AR/VR applications to transmit customized user models in real time and to virtually interact with the models.

To alleviate this problem, multiple lines of work such as [46, 20, 24, 4] have suggested that we decompose the scene into objects and their poses and scales. For example, one popular approach is to replace object pointcloud with a suitable Computer Aided Design (CAD) model from a database. The benefit of this is that the model is more likely to be complete and can be stored using its index in the database rather than a full mesh model. However, since models in CAD databases are most likely stored in a canonical pose and scale [8], this approach is only useful when one can estimate the appropriate scale and pose together with recognizing the model category in the database. This leads to a *joint shape retrieval and transform estimation* (JSRTE) problem, which is the focus of this paper.

Another drawback of the above approach is that given a scene scan, there is no guarantee of finding similar models in the database. Thus we may lose a great deal of representation power if we only rely on referencing the database indexes. Nevertheless, it has been proposed that deep neural network (DNN) based 3D generative models such as [1, 29, 15] allow one to produce a larger set of shapes than are available during training (or in the database) by learning a latent space that allows for

interpolation and generalization of shapes. What this implies is that we may now be able to obtain the benefit of using a CAD model representation without sacrificing too much representation power.

In this work we utilize an SDF-based 3D DNN model to formulate the JSRTE problem constrained on similarity transform (rigid body transform and scale) as an optimization problem. We present a gradient-descent based solution and compare our results to the state of the art for shape retrieval and/or transform estimation problems. We show this approach produces excellent results on synthetic and real world data. Our contributions are as follows:

1. Formulate JSRTE problem as a novel joint optimization problem.
2. Demonstrate a parameterization of the problem allowing for easy incorporation in popular DNN frameworks.
3. Conduct experiments on synthetic and real datasets showing the effectiveness of our approach on JSRTE problem and as a form of 3D data compression.

2 Related Work

2.1 3D Shape Alignment using Signed Distance Functions

Aligning 3D shapes is a well studied problem with varying solutions depending on the representations of the 3D shape. For example, when the geometry of a 3D object is represented using pointcloud, solutions for estimating rigid-body and orientation-preserving similarity transformations have been proposed [17, 18, 3, 49, 47].

Alternatively, and more pertinent to our work, 3D shapes can be aligned using signed distance functions (SDFs). [42] provided a good overview of some of these methods. For example, [40, 7] presented techniques to estimate rigid-body transformations between two shapes represented using SDFs. [27, 10] further tackled the problem of estimating rigid-body transformation and scale using SDFs, the same type of transformation studied in this paper. Their algorithms made use of geometric moments and the Fourier transform. In comparison, we develop a gradient-descent based algorithm to find a solution to an optimization formulation. Similarly, [28, 19] also made use of gradient-descent algorithms on optimization problems to estimate similarity transformations between SDFs. Yet, none of the works above explicitly allow for solving shape retrieval and transformation estimation in one optimization problem. Rather, they only deal with transformation estimation given the shape model.

2.2 3D Features for Shape Retrieval and Alignment

There exists a body of work that explicitly deals with shape retrieval. Classical shape retrieval algorithms such as [38, 16, 44] usually follow a two-step approach: The first step involves shape recognition from a database by making use of shape descriptors such as [45, 35, 2]. The second step involves a correspondence search using 3D keypoints and descriptors followed by registration. It has been argued in [39] that dense alignment via SDFs can produce much better results since it does not make use of a correspondence search which may introduce errors. Our work favors this approach. We will compare our superior results against classical 3D feature based retrieval techniques.

More recent work has shifted from using hand-crafted descriptors to data driven ones such as [50, 12]. Alternative data driven approaches [31] make use of deep neural networks (DNNs) to replace the recognition module entirely. However, the sequential approach of recognition and then registration does not allow for corrections of errors in the registration phase caused by a mismatch during the recognition stage. To address this problem, [5] proposed an end-to-end shape retrieval and alignment module. In comparison, although our work also jointly optimizes over the space of shapes and transforms, the formulation is not a feed-forward neural network but an optimization problem that makes use of a neural network model. The main benefit of this is that the optimization process can be viewed as introducing feedback that allows iterative minimization of errors from predictions made by the network. A feed-forward module can only make a prediction once per input and has no mechanism to minimize errors after training. More importantly, the space of shapes over which we optimize allows us to retrieve objects that were not seen during training.

3 DeepSDF-based Shape Retrieval and Similarity Transform Estimation

3.1 Problem Statement

Assume that we are given a set of shapes $\mathbb{S} = \{S_1, S_2, S_3, \dots, S_n\}$ belonging to the same class, all in the canonical scale and pose (e.g., a set of chairs all normalized to lie in the unit sphere with forward

directions aligned with the $+y$ axis). Further assume the shapes are represented as meshes although our approach also works for other representations as long as we can extract a signed distance function from the representation.

Then, for a new shape S belonging to the same class but not necessarily in \mathbb{S} or in the same canonical scale and pose, we would like to firstly build a dataset \mathbb{D} of shapes using \mathbb{S} , and secondly find an item in \mathbb{D} and a corresponding similarity transformation that best approximates S .

Note that we do not enforce $\mathbb{D} = \mathbb{S}$, but only require that \mathbb{D} is constructed using only \mathbb{S} . Typical shape retrieval problems tend to set $\mathbb{D} = \mathbb{S}$. However, as we will show, this may restrict the richness of \mathbb{D} .

Our approach assumes that all given shapes have an associated SDF. This can be easily obtained, for example, from a triangle mesh representation of the shape.¹

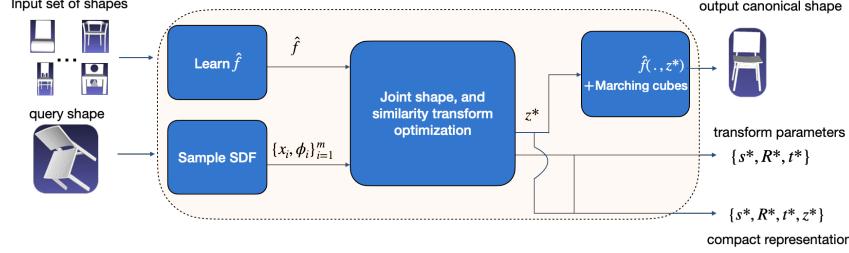


Figure 1: Overview of the proposed JSRTE algorithm.

As shown in Fig. 1, the input to our algorithm is a query shape S and a set of canonical shapes \mathbb{S} . Its output are the optimal shape representation in canonical pose and scale from \mathbb{D} as a mesh, the optimal similarity transformation, and a compact representation of the query shape for data compression.

3.2 Notation and Preliminaries

A 3D similarity transformation $g \in Sim(3)$ is a tuple (s, R, t) , with $s \in \mathbb{R}^+$, $R \in SO(3)$, $t \in \mathbb{R}^3$, and R can be written as a 3×3 real matrix. Then, g acting on a point $x \in \mathbb{R}^3$ is given by

$$g(x) = sRx + t. \quad (1)$$

Let S be a solid 3D shape with a closed surface such that we have a well-defined notion of inside and outside. In addition, let $\partial\Omega_S$ represent the set of points on the surface of S , Ω_S represent the set of points on the surface and interior of S , and Ω_S^c the complement of Ω_S . Then an SDF is an implicit

$$\text{representation of the shape defined as: } \Phi(x, \Omega_S) = \begin{cases} -\min_{y \in \partial\Omega_S} \|x - y\|_2, & \text{if } x \in \Omega_S; \\ \min_{y \in \partial\Omega_S} \|x - y\|_2, & \text{if } x \in \Omega_S^c. \end{cases}$$

Intuitively, for a fixed shape S , its SDF at a point x tells us the distance from x to the surface of S , with the sign determined by whether x lies in the interior of S .

Let S, S' be two shapes related by a similarity transformation g as defined in (1), i.e., $\partial\Omega_{S'} = \{g(x) | x \in \partial\Omega_S\}$. Under this condition, it is well known [28] that

$$\Phi(g(x), \Omega_{S'}) = s\Phi(x, \Omega_S). \quad (2)$$

In other words, SDFs are invariant to rotations and translations, but vary proportionally with isotropic scaling. The constant of proportionality is the scaling factor s .

3.3 Approach

Our proposed approach first learns a dataset \mathbb{D} from the input set of shapes \mathbb{S} and then formulates and solves an optimization problem to jointly estimate the transformation and shape parameters. Inspired by [29], we also learn a generative model and latent space of shapes. The benefit to this is that the latent space is more expressive than the input set of shapes \mathbb{S} . Specifically, the latent space has been shown in [29] to extrapolate the shapes in \mathbb{S} and produce new shapes of the same class but not contained in \mathbb{S} . The generative model also allows us to formulate searching over the latent space as a tractable optimization problem. In addition, since the dimension of the latent space is relatively small and a similarity transformation involves only a few parameters, we can simultaneously obtain a compact representation of the query shape.

¹Please refer to the supplementary for more details on SDF extraction and sampling.

3.3.1 Approximating SDFs with DNNs and Learning a Latent Dataset

Given an object class of shapes in a canonical pose and scale, $\mathbb{S} = \{S_1, S_2, S_3, \dots, S_n\}$, one can learn a generative model $\hat{f}(x, z) \approx \Phi(x, h(z))$ using DNNs [29]. Here $x \in \mathbb{R}^3$ is a point and $z \in \mathbb{R}^d$ is a latent vector with one-to-one correspondence with the shapes in \mathbb{S} . $h(z)$ is a mapping that assigns each latent vector in \mathbb{R}^d to a shape in \mathbb{S} , i.e., $h : \mathbb{R}^d \rightarrow \mathcal{P}(\mathbb{R}^3)$ such that $\forall S \in \mathbb{S} \exists z \in \mathbb{R}^d : h(z) = \Omega_S$. The dimension d is a design choice, and we refer to \mathbb{R}^d as the latent space. It was also shown in [29] that in learning \hat{f} , one learns a latent space that may generate new shapes not in the training dataset.

The implication of these results is that we can choose the latent space as the dataset for our algorithm. Also, given a shape S in the same canonical scale and pose and a set of m samples of its SDF $\chi_S = \{(x_i, \phi_i = \Phi(x_i, \Omega_S))\}_{i=1}^m$, S can be estimated by solving the following problem [29]:

$$\min_{z \in \mathbb{R}^d} \sum_{(x_i, \phi_i) \in \chi_S} |\hat{f}(x_i, z) - \phi_i|. \quad (3)$$

The above problem can be viewed as a special case of shape retrieval, since one can convert the latent vector back into a mesh by using \hat{f} to generate SDF samples of the shape and using [25] to convert it to a mesh. Moreover, in the event that we have incomplete knowledge of the SDF of S , this formulation gives some robustness and has been shown to be useful for shape completion as well.

However, the above restriction to shapes in the canonical pose and scale limits the usefulness of this work in real world applications, since most objects are not in a canonical scale and pose. While it is possible to learn a new generative model on a larger dataset containing scaled and transformed versions of shapes in \mathbb{S} , we find that this is an inefficient approach to solving the problem. Our main contribution in this work is to present a principled way to overcome this problem without making use of any more data or computation for training.

3.3.2 Joint Optimization of Shape and Transform Parameters

Our main idea is to modify (3) to include an optimization over similarity transform parameters that is also easy to implement. Using (2) and (3), we formulate the problem as:

$$\min_{s, R, t, z} \sum_{(x_i, \phi_i) \in \chi_S} \left| s \hat{f} \left(\frac{R^{-1}(x_i - t)}{s}, z \right) - \phi_i \right| \quad (4)$$

with $s \in \mathbb{R}^+$, $R \in SO(3)$, $t \in \mathbb{R}^3$, $z \in \mathbb{R}^d$. Then one may use gradient-descent algorithms to solve the problem. However, care must be taken since R does not live in a vector space. While it is possible to properly take gradient steps in $SO(3)$ [11], we take an alternative approach to parameterize R .

Using the axis-angle representation for rotation matrices, we know that for any rotation matrix R , there exists $\omega : \|\omega\|_2 = 1$ and $\theta \in [-\pi, \pi] : R = \exp(\hat{\omega}\theta)$, where $\omega = [\omega_1, \omega_2, \omega_3]^\top$, $\hat{\omega} = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix}$. Since ω lies on the unit sphere, we may parameterize it using spherical coordinates as $\omega_1 = \sin(\psi)\cos(\rho)$, $\omega_2 = \sin(\psi)\sin(\rho)$, $\omega_3 = \cos(\psi)$, where, to avoid potential confusion due to overloading symbols, we have used ψ to represent the polar angle and ρ to represent the azimuthal angle. Consequently, we may write

$$R(\psi, \rho, \theta) = \exp \left(\begin{bmatrix} 0 & -\cos(\psi) & \sin(\psi)\sin(\rho) \\ \cos(\psi) & 0 & -\sin(\psi)\cos(\rho) \\ -\sin(\psi)\sin(\rho) & \sin(\psi)\cos(\rho) & 0 \end{bmatrix} \theta \right) \quad (5)$$

and the initial problem (4) as

$$\min_{s, \psi, \rho, \theta, t, z} \sum_{(x_i, \phi_i) \in \chi_S} \left| s \hat{f} \left(\frac{[R(\psi, \rho, \theta)]^{-1}(x_i - t)}{s}, z \right) - \phi_i \right|. \quad (6)$$

In this way, no special handling of gradients has to be taken. We find that this explicit formulation allows for easy implementation and incorporation into common deep learning frameworks with automatic differentiation such as [30]. We summarize our JSRTE algorithm in Algorithm 1.

3.4 Implementation

We have implemented JSRTE algorithm using PyTorch [30] and report some implementation details here for reproducibility. Specifically, the matrix exponential $\exp(\hat{\omega}\theta)$ is expressed by Rodrigues'

formula [34, 26]: $\exp(\hat{\omega}\theta) = I + \hat{\omega}\sin(\theta) + \hat{\omega}^2(1 - \cos(\theta))$. Adam[23] as implemented in [30] is adopted to solve the optimization problem (6). To ensure that the search is in the feasible set of the problem, $s \in [\underline{s}, \bar{s}]$ can be restricted as $\underline{s} = 0.01, \bar{s} = 10$. \hat{f} is learned using open source code provided by [29, 13], with latent vectors in \mathbb{R}^{256} . For all objects, we have trained on subsets of the corresponding Shapenet [8] class. To encourage convergence of the network, [29] has used a penalty term on the norm of z weighed by 10^{-4} . We also include this term in solving (6). We generally make use of the same SDF extraction and sampling approach used in [29] (see supplementary for detail).

Algorithm 1 Joint Shape Retrieval and Transform Estimation (JSRTE)

```

1: function JSRTE( $\mathbb{S}, S, s^0, \rho^0, \psi^0, \theta^0, t^0, z^0$ )
2:    $\hat{f}, \mathbb{D} \leftarrow \mathbb{S}$                                  $\triangleright$  Learn generative model and latent space from data (offline).
3:    $\{x_i, \phi_i\}_{i=1}^m \leftarrow S$                    $\triangleright$  Sample SDF from query shape
4:   Initialization                                      $\triangleright$  Initialize (6) using provided parameters
5:   Solve (6) for  $s^*, R^*, t^*, z^*$                  $\triangleright$  Use gradient descent to solve (6)
6:    $\{\bar{x}_i, \hat{f}(\bar{x}_i, z^*)\}_{i=1}^n \leftarrow \hat{f}, z^*, \bar{x}_i \in [-1, 1]^3$      $\triangleright$  Generate SDF samples at  $z^*$  using  $\hat{f}$ 
7:    $S^* \leftarrow \text{MARCHINGCUBES}(\{\bar{x}_i, \hat{f}(\bar{x}_i, z^*)\}_{i=1}^n)$        $\triangleright$  Generate mesh from SDF samples with [25]
8:   return  $s^*, R^*, t^*, z^*, S^*$ 
```

4 Experiments

We perform three experiments to validate our approach for shape and transformation estimation. The first experiment seeks to evaluate the performance of the proposed algorithm when we initialize its parameters within reasonable bounds of the groundtruth. The second experiment evaluates our algorithm on real world data and compares it to state-of-the-art benchmarks. The third experiment assesses the viability of our approach as a means for 3D data compression.

4.1 Synthetic data

Gradient-descent based methods are susceptible to converging to locally optimal solutions. So it is important to find good initialization points for the variables. For this experiment, we want to observe the behavior of our algorithm when initialized within reasonable bounds of the groundtruth.

We sample objects from the chair, table, sofa, and bed categories from ShapeNet [8]. First, 30 objects from each category are sampled. Then for each object a random scale and pose are assigned by sampling the groundtruth parameters, $s \sim \text{Uniform}([\frac{1}{2}, 2])$, $\psi, \rho, \theta \sim \text{Uniform}(-\pi, \pi)$, and $\|t\|_2 \sim \text{Uniform}(0, 4)$. The direction of t is chosen uniformly from the surface of a unit sphere. For each shape and transform pair, we solve the shape retrieval problem using our algorithm 50 times, with each trial using a random initialization. We carry out two sub-experiments to highlight scenarios where our algorithm is applicable. In the first scenario, we assume that we know the axis of rotation (and consequently ψ, ρ), but all other parameters are unknown. This is a reasonable assumption since many objects lie upright on flat surfaces such as floors or tables. One may use the normal to the surface as the axis of rotation. In the second scenario, we relax this assumption and introduce some uncertainty in the axis of rotation. The scenarios are used to randomly sample the initialization for our algorithm.

Concretely, we refer to the initialization parameters as $s^0 = s(1 + \Delta s)$, $\psi^0 = \psi + \Delta\psi$, $\rho^0 = \rho + \Delta\rho$, $\theta^0 = \theta + \Delta\theta$, $t^0 = t + \Delta t$. The Δ variables correspond to the difference between the ground-truth and initialization parameters. We then sample the Δ variables uniformly from ranges determined by each scenario. Table 1 below summarizes these ranges for the different scenarios.

Table 1: Parameter ranges for the difference between initialization points and groundtruth values.

Scenario	Δs	$\Delta\psi$	$\Delta\rho$	$\Delta\theta$	$\ \Delta t\ (m)$
Known rotation axis	[0, 0.3)	N/A	N/A	$[-\frac{\pi}{9}, \frac{\pi}{9}]$	[0, 0.15)
Unknown rotation axis	[0, 0.3)	$[-\frac{\pi}{36}, \frac{\pi}{36}]$	$[-\frac{\pi}{36}, \frac{\pi}{36}]$	$[-\frac{\pi}{9}, \frac{\pi}{9}]$	[0, 0.15)

In all scenarios, We choose the direction of Δt uniformly from the surface of a unit sphere. $z^0 \sim \text{Normal}(\mathbf{0}, \sigma\mathbf{I})$, $\sigma = 0.01$. We use the corresponding ShapeNet [8] model class as input shape set \mathbb{S} .

To quantitatively measure the performance of our algorithm, we use the F-score between the output shape (transformed by our predicted transform parameters) and the test shape as recommended in

[43]. The F-score measures the similarity between 3D surfaces as the harmonic mean between precision and recall. A point on the predicted surface is considered a true positive if it is within a threshold ϵ of the groundtruth surface. In this work, we have set this threshold as 5% of the scale of the groundtruth object. We refer to this score as F@5%. The F-score ranges from 0 to 1, with higher numbers indicating more closely matched shapes.

For each shape and transform pair, we categorize the F-score into bins and count the number of initializations out of 50 that fall into each bin. Then for each bin, we report aggregate statistics using box plots in Fig. 2. The position of each box-plot on the x-axis corresponds to the rightmost edge of its bin. The leftmost edge of its bin is the previous point on the x-axis.

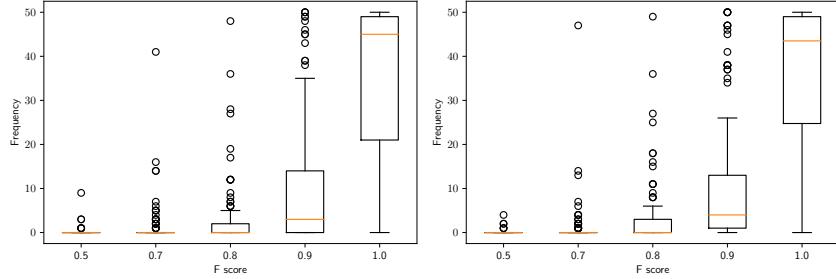


Figure 2: Box plot of F@5% from experiments described in Table 1. **Left:** Scenario with a known axis of rotation; **Right:** Scenario with an unknown axis of rotation. The median for each bin is shown as an orange line in the box, with outliers shown as circles. Each box is placed on the right edge of its corresponding bin, the left edge of each bin is the right edge of the preceding bin.

Fig. 2 corresponds to scenarios with a known and unknown axis of rotation respectively. In both scenarios, we observe that our algorithm performs well with most F@5% scores greater than 0.8. We also observe a slight degradation in performance in the scenario with an unknown axis of rotation, as is to be expected due to more uncertainty. Intuitively, the results imply that with a known axis of rotation, if we know the amount of rotation up to $\pm 20^\circ$, the translation up to ± 15 cm and the scale up to 30% of the true scale, we can expect our algorithm to perform very well. We also perform more experiments to test our algorithm in other parameter ranges and report those as well as qualitative results in the supplementary.

4.2 Real Data

In the next set of experiments, we evaluate our algorithm on real data. In this paper, we demonstrate the results on the chair subset of the Redwood dataset [9]. The Redwood dataset contains scans (represented as triangle meshes) of real objects in varying scales and poses. We have manually segmented the dataset so that we remove other objects in the background, leaving behind only the floor and the main object in each scene. We assume that all objects in the scene lie on the floor and are upright. This allows us to obtain an estimate for ω as the normal to the floor plane. In this way, we collect 89 shapes and samples of their SDFs. However, we use 31 randomly selected shapes from these 89 for validating the baselines described below, leaving only 58 for testing. We discarded all other scenes that could not be segmented properly. For each shape, we initialize s using the radius of its bounding sphere and t using the center of the bounding sphere. We estimate ω using the normal vector to the floor plane and z is initialized by drawing from $\text{Normal}(\mathbf{0}, \sigma\mathbf{I})$, $\sigma = 0.01$. The only parameter left is θ . For θ we grid up the space in increments of 30° and run our algorithm for each value of $\theta^0 \in [0, \frac{\pi}{6}, \frac{\pi}{12}, \dots, 2\pi]$. We then select the best result from each of these initialization of θ using the F@5% score.

We compare our work to three different approaches for shape retrieval and transform estimation. In these approaches, we perform recognition to find the closest shape to the query shape in the set of input shapes and then perform registration to estimate a similarity transform.²

The first approach uses Harris3D keypoints and the SHOT[37] descriptor to perform recognition and provide correspondences for registration. Registration is then carried out using [47]. Specifically, for recognition, we describe each shape in the dataset using the descriptor and keypoint detecting

²We are aware of other state-of-the-art SDF based methods for jointly estimating shape and similarity transformation [5]. However, we could not find a publicly available implementation to use for testing.

method stated above. During test time, we compare the keypoints and descriptors of the query shape to those in the dataset. This method is referred to as Harris3D+SHOT.

The second approach replaces the keypoint detector and descriptor with the shape descriptor developed in [2]. For registration, points are sampled from the mesh and then applied to an Iterative Closest Point (ICP)[6] variant that makes use of [47] for transform estimation. Correspondences are assigned by solving an optimal assignment problem. We find that this significantly improves the result. This method is referred to as OURCVFH+ICP.

The last approach uses PointNet[33] as described in [31] for recognition and the ICP method described above for registration. This method is referred to as PointNet+ICP.

We report the average F@5% score of these methods in Table 2, which shows our algorithm significantly outperforms the three alternative approaches. We also provide qualitative results of randomly sampled test cases in Fig. 3, which shows that the shapes predicted by our method are qualitatively better than the alternative methods, with much less misalignment. Alternative methods using a two-step pipeline of recognition and registration can propagate errors in the recognition stage that cannot be corrected by just a similarity transformation in the registration stage. In addition, our method uses a richer search space for shapes by using the latent space as its dataset, rather than just the set of input shapes provided. Moreover, finding good correspondences can be difficult for other two-step approaches as clean CAD models and noisy real scans have differing local surface properties. Finally, the need to sample the surface to allow tractable registration may introduce errors in the sense that true corresponding pairs may not belong to the sampled sets. Our method on the other hand does not make explicit use of correspondences.

Table 2: Quantitative results on Redwood dataset. Average F-scores for shape retrieval and similarity transformation methods compared to ours. Our algorithm’s score more than doubles the others.

	Harris3D + SHOT	OURCVFH + ICP	PointNet + ICP	Ours
F@5%	0.273	0.367	0.452	0.902

4.3 3D Shape Compression

Finally, we conduct an experiment to explore the viability of our approach as a form of 3D data compression. Given the neural network parameters of \hat{f} , we can choose to store only the latent vector representing a shape and the associated similarity transform. With access to \hat{f} and the transform parameters, one can then decode the latent vector and convert it back to a mesh model. In our experiment, the network size is approximately 7MB. This is a constant cost shared across many uses and is consequently amortized.

For this experiment, we save each mesh in our Redwood test set under two mesh simplification schemes, mesh simplification using Quadric Error Metrics[14, 51] and Vertex Clustering [51]. We have varied the parameters of each method. For mesh simplification using Quadric Error Metrics, we set the target number of faces to be reduced by a factor of 100 and 1000 from the number of faces in the original mesh. We refer to these experiments as QE-100, QE-1000 respectively. For Vertex Clustering, we vary the size of the grid cells used for clustering to be a factor of 0.1 and 0.2 of the radius of the bounding sphere for the shape. We refer to these as VC-0.1, VC-0.2 respectively. We save all meshes and compute the average size of the files as well as the mean F-score (compared to the original mesh) to give an indication of the representation power at each simplification parameter. We compare these scores against our mean F-score and the cost of saving the latent vector and transformation parameters as a 4×4 matrix and report the results in Table 3. The result shows that our method provides an impressive tradeoff between reconstruction quality and storage size.

Table 3: Comparison of storage size and representation power for various mesh simplification algorithms. Our approach provides an excellent trade-off between accuracy and storage space, saving over 20x space as the most accurate method with less than a 10% drop in accuracy.

	Ours	QE-100	QE-1000	VC-0.1	VC-0.2
Average F-score	0.902	0.977	0.778	0.978	0.839
Average size (KB)	1.088	20.120	3.454	34.833	9.073

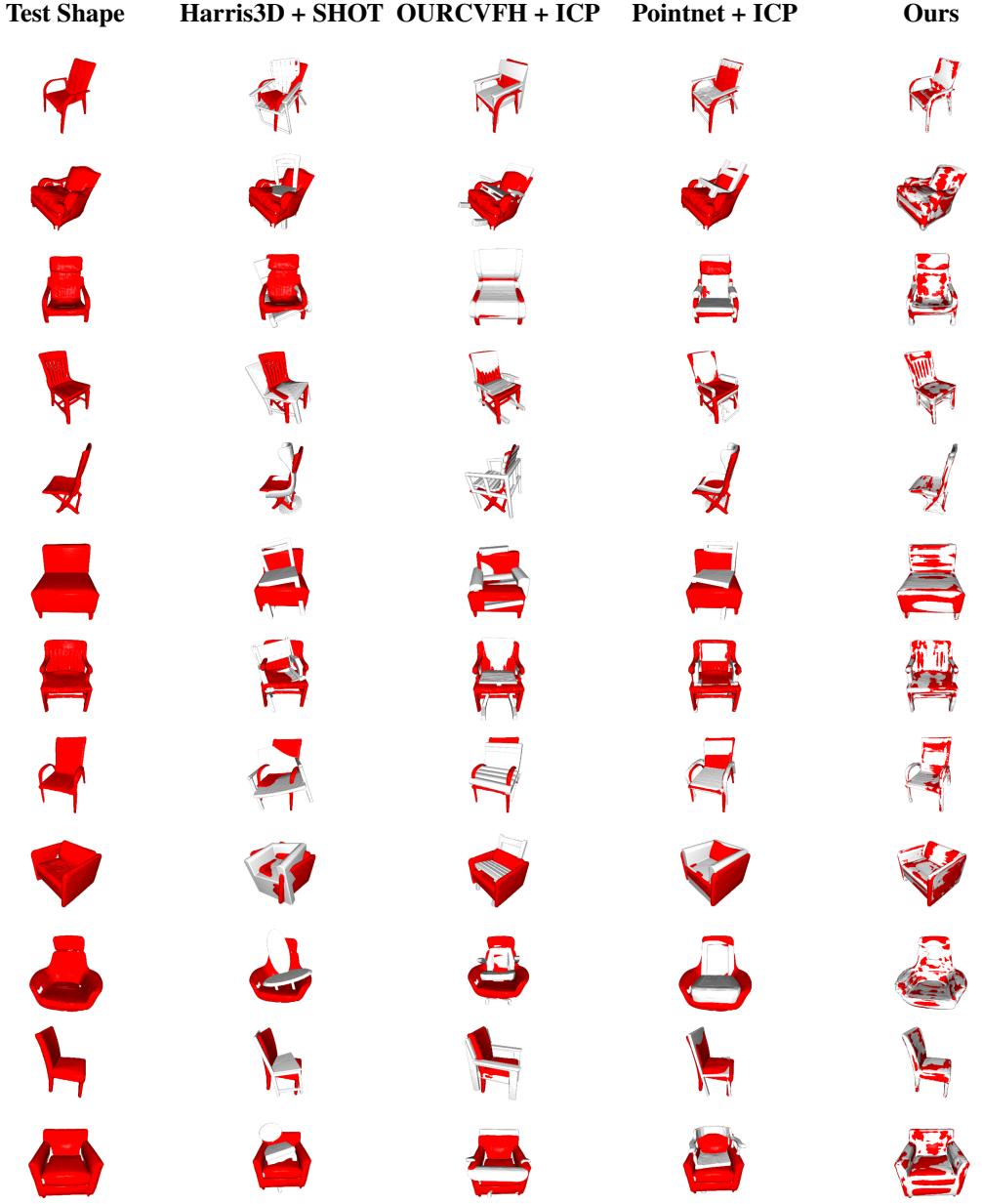


Figure 3: Qualitative result on Redwood dataset. Left to Right, test shape, results using Harris3D and SHOT[37] for shape retrieval, results using OURCVFH [2] and ICP, results using PointNet [33] and ICP, our proposed method. We have shown each method’s result in gray overlaid with the input mesh in red. Our algorithm provides significant qualitative improvement with almost perfect retrieval.

5 Conclusion

In this paper we have presented a formulation for shape and similarity transform estimation from SDFs as an optimization problem. We have obtained good results with a gradient-descent optimization scheme and good initialization of the parameters. We have also shown results on a dataset of real scans and obtained superior performance to benchmarks. Our method currently relies on good segmentation of the scene and knowledge of the class of objects. In addition, it may struggle to represent fine details and may sometimes lead to disconnected meshes. As future work, we plan to tackle these problems as well as incorporating this method as a tool for 3D indoor scene understanding and representation.

Broader Impact

The work presented in the paper has the potential to be applied to a wide range of Computer Vision tasks. We focus on concrete application domains of Mixed Reality and 3D scene understanding.

We envision that our work can be used as a part of Mixed Reality applications that require automatic content generation from real world scenes. Such applications include telepresence and automatic 3D modeling. While these applications are yet to reach full maturity we believe our work can contribute to the development of these applications. In addition, when coupled with other tools to help predict what is in a scene, our work can be used to help answer questions regarding "where is the object in the scene and how can we manipulate it?". These questions lie in the domain of 3D scene understanding with potential use cases in robotics.

On the other hand, mixed reality applications require privacy preserving algorithms. As discussed in [41], and illustrated in [32], Mixed Reality applications may require cloud services for data storage, transmission and processing. Adversarial attacks on cloud infrastructure could then prove catastrophic since mixed reality applications tend to take place in private spaces such as homes. While the work described in this paper is amenable to cloud applications due to its compression properties, it is not naturally privacy preserving. Adversarial access to neural network parameters could reveal information about the 3D geometry and spatial content of private spaces. Hence, extra measures such as the use of encryption must be put in place to protect privacy. Likewise, applications, such as robotic manipulation tasks, that may depend on our method for estimating the pose of objects may need to put measures in place to mitigate catastrophic results in failure cases. Wrong pose estimation could lead to damaging object while trying to manipulate them or even fatal injuries to humans in the vicinity.

Taking all this into account, we see the opportunity for more research on ways to apply shape retrieval methods such as ours in Mixed Reality and robotic applications while preserving privacy and detecting failure modes. We hope that this work encourages more discussion among researcher, engineers and consumers on the impact of such technologies and ways to mitigate negative consequences.

Acknowledgement

This work is supported in part by the Office of Naval Research (ONR) under grant N00014-19-1-2055.

Supplementary

A Overview

We provide additional quantitative and qualitative results on the algorithm presented in the main paper. We also provide more implementation details and descriptions of the metrics used.

B Metrics

We have made use of the F-score as recommended in [43] to provide a quantitative evaluation of our work. Given a groundtruth shape S and an estimate shape \hat{S} , as well as samples of points on their surfaces $\partial\Omega_S$ and $\partial\Omega_{\hat{S}}$,³ the F-score is the harmonic mean of precision and recall metrics. Precision and recall are defined as follows:

To compute the precision, for each point y in the estimate shape, we first compute its distance to the groundtruth as:

$$e_y = \min_{x \in \partial\Omega_S} \|y - x\|_2.$$

The precision is then computed as the percentage of points within distances less than ϵ_p :

$$P(\epsilon_p) = \frac{100}{|\partial\Omega_{\hat{S}}|} \sum_{y \in \partial\Omega_{\hat{S}}} \mathbf{1}_{\{e_y < \epsilon_p\}}. \quad (7)$$

Recall is computed in the opposite direction. Again, the error is first computed for each point x on the groundtruth shape as:

$$e_x = \min_{y \in \partial\Omega_S} \|x - y\|_2, \quad (8)$$

and recall as:

$$R(\epsilon_r) = \frac{100}{|\partial\Omega_S|} \sum_{x \in \partial\Omega_S} \mathbf{1}_{\{e_x < \epsilon_r\}}. \quad (9)$$

Finally, the F-score is computed as:

$$F(\epsilon) = \frac{2P(\epsilon)R(\epsilon)}{P(\epsilon) + R(\epsilon)} \quad (10)$$

The F-score ranges from 0 to 1, with higher values indicating a better match between the estimate and the groundtruth. The F-score is very sensitive to deviations in shape and thus makes an excellent metric. To obtain pointclouds from the meshes, we have sampled 3000 points from the surface of each mesh. We have also set $\epsilon = 0.05r$, where r is the radius of the bounding sphere of the shape S .

An alternative metric may have been computing the distance between the estimated parameters and the groundtruth parameters, however, we find that solutions that are close in 3D space may not be close in the parameter space. For example, we find that the learned shapes can have different centers from their groundtruth counterparts, as such a good result may have a different translation value than the groundtruth.

C Additional Results

In this section, we provide additional results to support those presented in the main paper.

C.1 Synthetic data

We perform additional experiments making use of synthetic data to observe the behavior of our algorithm under varying initialization conditions. Following the same procedure described in Section

Table 4: Parameter ranges for the difference between initialization points and groundtruth values.

Scenario	Δs	$\Delta\psi$	$\Delta\rho$	$\Delta\theta$	$ \Delta t (m)$
Known rotation axis	[0, 0.5)	N/A	N/A	$[-\frac{2\pi}{9}, \frac{2\pi}{9})$	[0, 0.20)
Unknown rotation axis	[0, 0.5)	$[-\frac{\pi}{36}, \frac{\pi}{36})$	$[-\frac{\pi}{36}, \frac{\pi}{36})$	$[-\frac{2\pi}{9}, \frac{2\pi}{9})$	[0, 0.20)

4.1 of the main paper, we conduct two additional experiments that vary the initialization parameters as follows:

In the first scenario in Table 4, we assume a known axis a rotation, but increase the range of values that the initialization parameters can take on compared to those presented in the main paper. In the second scenario, we relax the assumption of a known axis of rotation and introduce some uncertainty. We retain the same increase in the ranges of the other parameters. The purpose of this experiment is to see how the performance degrades when we introduce more uncertainty.

As shown in Fig. 4, we observe that both scenarios perform worse than those presented in the main paper. We also observe that for most of the experiments the median F-score is still greater than 0.8. The difference here is that we observe significantly more F-scores in the [0.5, 0.7] range.

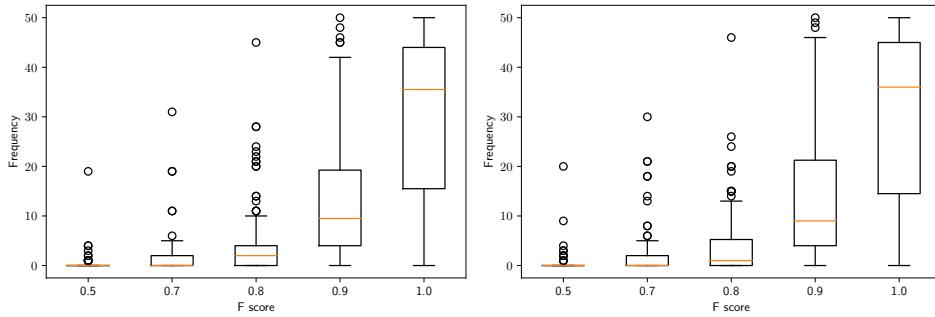


Figure 4: Box plot of F@5% from experiments described in Table 4. **Left:** Scenario with a known axis of rotation; **Right:** Scenario with an unknown axis of rotation. The median for each bin is shown as an orange line in the box, with outliers shown as circles. Each box is placed on the right edge of its corresponding bin, the left edge of each bin is the right edge of the preceding bin. We still observe most median F-scores greater than 0.80, although we now see more cases of F-scores in the range [0.5, 0.7].

C.2 Real Data

Figures 5 - 10 provide qualitative results from all the experiments carried out on the Redwood dataset [9] detailed in Section 4.2 of the main paper. We have displayed comparisons to the benchmark making use of PointNet [33] and a variant of ICP [6] described in the main paper. We report a similar level of performance as that presented in the main paper. Our algorithm consistently provides qualitatively better results with fewer shape mismatch and misalignments. The figures also indicate that our algorithm sometimes struggles with fine details and star shaped legs. This is partly due to poor convergence of the SDF model during training and also because in reality the star-shaped legs are free to rotate separately from the main body of the chair. These additional degrees of freedom are not well captured by our assumptions.

D Implementation Details

D.1 SDF Sampling

For experiments using synthetic data (Section 4.1 of the main paper), we follow the same sampling procedure described in [29]. For experiments using the Redwood dataset [9] (Section 4.2 of the main

³For simplicity of notation, we have retained the same notation as that used for the set of all points on the surface. To compute the F-score, we can only use a sampling of the set of points on the surface.

paper), we first preprocess the data by manually segmenting the floor and main object in each scene. We then make use of the plane fitting algorithm provided by [51] to segment the main object and the floor. We also obtain the axis of rotation in this fashion as the normal to the floor plane. Furthermore, we make use of the hidden point removal algorithm described in [22] and implemented in [51] to remove artifacts. Other artifacts and noisy points are also removed using the statistical outlier removal algorithm provided by [51].

Finally, for each point in the preprocessed mesh, we sample new points at a distance of $\pm 0.01\text{m}$ in the direction of the normal to the surface at each point. We estimate the SDF at these new points as ± 0.01 respectively. To obtain SDF samples representing the freespace, we take each point on the mesh and randomly and uniformly sample a new point between 0.07m and 0.20m away from the point on the mesh, in the direction of the outward pointing normal. We compute the SDF at these new points by finding the approximate distance to the closest point on the surface of the mesh. The sign is then determined by whether each new point is in the direction of the outward pointing normal at its nearest neighbor on the mesh. We use a k-d tree and the set of vertices on the densely sampled meshes to approximate the nearest neighbors. We obtain 25000 freespace SDF samples in this manner.

The SDF samples used during optimization are composed of the freespace SDF as well as the SDF sampled at distances $\pm 0.01\text{m}$ from points on the mesh.

D.2 Optimization Parameters

To solve the proposed optimization problem, we employ Adam [23] as implemented in [30]. We follow a stochastic gradient-descent approach and make use of 8000 SDF samples at each iteration. The learning rate is set to 0.05 for all parameters except for those assumed to be known. The learning rate is decreased by a factor of 5 every 400 iterations. We solve the optimization problem for 800 iterations.

D.3 Real data benchmarks

We report some implementation details for the benchmarks used for experiments on the Redwood dataset [9] presented in the main paper.

D.3.1 Harris3D + SHOT

For this benchmark we make use of Harris3D keypoint detector and the SHOT[45] descriptor as implemented in [36].

D.3.2 OURCVFH + ICP

For this benchmark, we make use of OURCVFH [2] as implemented in [36] as a shape descriptor. We use the same subset of ShapeNet [8] chair models used to train our SDF model and compute a shape descriptor for each chair model in the subset. During test time we compute a OURCVFH[2] descriptor for the test model and use it to find the nearest neighbor from the subset of ShapeNet[8] chair models.

For registration, we first obtain a rough approximation of the optimal transform parameters. We estimate the scale as the radius of the bounding sphere for the test shape divided by the radius of the bounding sphere of its closest match from the subset of ShapeNet[8] chairs. We estimate the translation as the difference between the centers of their bounding spheres. Since we know the axis of rotation, we estimate the rotation by searching the space of rotation angles in increments of 20° . We select the rotation angle that minimizes the maximum difference between points on the test shape and their nearest neighbor on the closest matching shape.

We refine this estimate using an algorithm inspired by the Iterative Closest Point (ICP) algorithm [6]. We first sample the test shape and its closest matching shape uniformly to obtain sets of points representing each shape. Then for each iteration of the ICP algorithm, we find correspondences by solving an optimal assignment problem. We then estimate the optimal similarity transformation given the optimal correspondences. The optimal assignment problem gives us a one-to-one mapping between correspondences. This prevents us from obtaining a degenerate solution to the registration

problem where the scale becomes almost zero. We make use of [48] to solve the optimal assignment problem and [47] as implemented in [51] to solve the registration problem.

D.3.3 PointNet + ICP

For this benchmark, we make use of the PointNet [33] classifier as implemented in [21]. We train the classifier to predict the id of the closest ShapeNet [8] model from our subset of chair models given an input pointcloud. During training, for each model we randomly sample 1024 point from the mesh and feed it to classifier for prediction. We add random noise to each of the points, and the set of points are re-sampled each epoch. We run it for 60 epochs after which the training loss does not significantly improve. We validate the model by making use of 30 models from the Redwood dataset [9]. We do not train on the Redwood dataset [9] models since they provide only a small number of models. Instead, we train on mesh models provided by our subset of ShapeNet [8] chair mesh models. During test time we sample 1024 points from the test shape and use it for predicting the closest matching model.

Registration is performed using the same algorithm described in the subsection above.

References

- [1] P. Achlioptas, O. Diamanti, I. Mitliagkas, and L. Guibas. Learning representations and generative models for 3d point clouds. *arXiv preprint arXiv:1707.02392*, 2017.
- [2] A. Aldoma, F. Tombari, R. B. Rusu, and M. Vincze. Our-cvfh-oriented, unique and repeatable clustered viewpoint feature histogram for object recognition and 6dof pose estimation. In *Joint DAGM (German Association for Pattern Recognition) and OAGM Symposium*, pages 113–122. Springer, 2012.
- [3] K. S. Arun, T. S. Huang, and S. D. Blostein. Least-squares fitting of two 3-d point sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-9(5):698–700, 1987.
- [4] A. Avetisyan, M. Dahnert, A. Dai, M. Savva, A. X. Chang, and M. Nießner. Scan2cad: Learning cad model alignment in rgb-d scans. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2614–2623, 2019.
- [5] A. Avetisyan, A. Dai, and M. Nießner. End-to-end cad model retrieval and 9dof alignment in 3d scans. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2551–2560, 2019.
- [6] P. J. Besl and N. D. McKay. Method for registration of 3-d shapes. In *Sensor fusion IV: control paradigms and data structures*, volume 1611, pages 586–606. International Society for Optics and Photonics, 1992.
- [7] E. Bylow, J. Sturm, C. Kerl, F. Kahl, and D. Cremers. Real-time camera tracking and 3d reconstruction using signed distance functions. In *Robotics: Science and Systems*, volume 2, page 2, 2013.
- [8] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.
- [9] S. Choi, Q.-Y. Zhou, S. Miller, and V. Koltun. A large dataset of object scans. *arXiv preprint arXiv:1602.02481*, 2016.
- [10] Z. Cui, S. Mahmoodi, and M. Bennett. A robust and high-performance shape registration technique using characteristic functions. In *2018 IEEE International Conference on Image Processing, Applications and Systems (IPAS)*, pages 1–6. IEEE, 2018.
- [11] F. Dellaert, M. Kaess, et al. Factor graphs for robot perception. *Foundations and Trends® in Robotics*, 6(1-2):82–95, 2017.
- [12] H. Deng, T. Birdal, and S. Ilic. 3d local features for direct pairwise registration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3244–3253, 2019.
- [13] Y. Duan, H. Zhu, H. Wang, L. Yi, R. Nevatia, and L. J. Guibas. Curriculum deepsdf. *arXiv preprint arXiv:2003.08593*, 2020.

- [14] M. Garland and P. S. Heckbert. Surface simplification using quadric error metrics. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 209–216, 1997.
- [15] K. Genova, F. Cole, D. Vlasic, A. Sarna, W. T. Freeman, and T. Funkhouser. Learning shape templates with structured implicit functions. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7154–7164, 2019.
- [16] Y. Guo, M. Bennamoun, F. Sohel, M. Lu, and J. Wan. 3d object recognition in cluttered scenes with local surface features: a survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(11):2270–2287, 2014.
- [17] B. K. Horn. Closed-form solution of absolute orientation using unit quaternions. *Josa a*, 4(4):629–642, 1987.
- [18] B. K. Horn, H. M. Hilden, and S. Negahdaripour. Closed-form solution of absolute orientation using orthonormal matrices. *JOSA A*, 5(7):1127–1135, 1988.
- [19] X. Huang, N. Paragios, and D. N. Metaxas. Shape registration in implicit spaces using information theory and free form deformations. *IEEE transactions on pattern analysis and machine intelligence*, 28(8):1303–1318, 2006.
- [20] H. Izadinia, Q. Shan, and S. M. Seitz. Im2cad. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5134–5143, 2017.
- [21] K. J., E. Smith, J.-F. Lafleche, C. Fuji Tsang, A. Rozantsev, W. Chen, T. Xiang, R. Lebare-dian, and S. Fidler. Kaolin: A pytorch library for accelerating 3d deep learning research. *arXiv:1911.05063*, 2019.
- [22] S. Katz, A. Tal, and R. Basri. Direct visibility of point sets. In *ACM SIGGRAPH 2007 Papers*, SIGGRAPH ’07, page 24–es, New York, NY, USA, 2007. Association for Computing Machinery.
- [23] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [24] Y. Li, A. Dai, L. Guibas, and M. Nießner. Database-assisted object retrieval for real-time 3d reconstruction. In *Computer Graphics Forum*, volume 34, pages 435–446. Wiley Online Library, 2015.
- [25] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *ACM siggraph computer graphics*, 21(4):163–169, 1987.
- [26] Y. Ma, S. Soatto, J. Kosecka, and S. S. Sastry. *An invitation to 3-d vision: from images to geometric models*, volume 26. Springer Science & Business Media, 2012.
- [27] S. Mahmoodi, M. S. Al-Huseiny, and M. S. Nixon. Similarity registration for shapes based on signed distance functions. In *International Symposium on Visual Computing*, pages 599–609. Springer, 2012.
- [28] N. Paragios, M. Rousson, and V. Ramesh. Non-rigid registration using distance functions. *Computer Vision and Image Understanding*, 89(2-3):142–165, 2003.
- [29] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove. Deep sdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 165–174, 2019.
- [30] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.
- [31] Q.-H. Pham, M.-K. Tran, W. Li, S. Xiang, H. Zhou, W. Nie, A. Liu, Y. Su, M.-T. Tran, N.-M. Bui, et al. Shrec’18: Rgb-d object-to-cad retrieval. *Proc. 3DOR*, 2, 2018.
- [32] F. Pittaluga, S. J. Koppal, S. B. Kang, and S. N. Sinha. Revealing scenes by inverting structure from motion reconstructions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 145–154, 2019.

- [33] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.
- [34] O. Rodrigues. *De l’attraction des sphéroïdes, Correspondance sur l’École Impériale Polytechnique*. PhD thesis, PhD thesis, Thesis for the Faculty of Science of the University of Paris, 1816.
- [35] R. B. Rusu, N. Blodow, and M. Beetz. Fast point feature histograms (fpfh) for 3d registration. In *2009 IEEE international conference on robotics and automation*, pages 3212–3217. IEEE, 2009.
- [36] R. B. Rusu and S. Cousins. 3d is here: Point cloud library (pcl). In *2011 IEEE international conference on robotics and automation*, pages 1–4. IEEE, 2011.
- [37] S. Salti, F. Tombari, and L. Di Stefano. Shot: Unique signatures of histograms for surface and texture description. *Computer Vision and Image Understanding*, 125:251–264, 2014.
- [38] Y. Shan, B. Matei, H. S. Sawhney, R. Kumar, D. Huber, and M. Hebert. Linear model hashing and batch ransac for rapid and accurate object recognition. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 2, pages II–II. IEEE, 2004.
- [39] M. Slavcheva. *Signed Distance Fields for Rigid and Deformable 3D Reconstruction*. PhD thesis, Technische Universität München, 2018.
- [40] M. Slavcheva, W. Kehl, N. Navab, and S. Ilic. Sdf-2-sdf: Highly accurate 3d object reconstruction. In *European Conference on Computer Vision*, pages 680–696. Springer, 2016.
- [41] P. Speciale, J. L. Schonberger, S. B. Kang, S. N. Sinha, and M. Pollefeys. Privacy preserving image-based localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5493–5503, 2019.
- [42] M. Steiner and M. Urschler. Statistical shape models with signed distance functions. 2013.
- [43] M. Tatarchenko, S. R. Richter, R. Ranftl, Z. Li, V. Koltun, and T. Brox. What do single-view 3d reconstruction networks learn? In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3405–3414, 2019.
- [44] F. Tombari and L. Di Stefano. Object recognition in 3d scenes with occlusions and clutter by hough voting. In *2010 Fourth Pacific-Rim Symposium on Image and Video Technology*, pages 349–355. IEEE, 2010.
- [45] F. Tombari, S. Salti, and L. Di Stefano. Unique signatures of histograms for local surface description. In *European conference on computer vision*, pages 356–369. Springer, 2010.
- [46] S. Tulsiani, S. Gupta, D. F. Fouhey, A. A. Efros, and J. Malik. Factoring shape, pose, and layout from the 2d image of a 3d scene. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 302–310, 2018.
- [47] S. Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(4):376–380, 1991.
- [48] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. Jarrod Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and S. . . Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.
- [49] M. W. Walker, L. Shao, and R. A. Volz. Estimating 3-d location parameters using dual number quaternions. *CVGIP: image understanding*, 54(3):358–367, 1991.
- [50] A. Zeng, S. Song, M. Nießner, M. Fisher, J. Xiao, and T. Funkhouser. 3dmatch: Learning local geometric descriptors from rgb-d reconstructions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1802–1811, 2017.
- [51] Q.-Y. Zhou, J. Park, and V. Koltun. Open3D: A modern library for 3D data processing. *arXiv:1801.09847*, 2018.



Figure 5: Additional qualitative result on Redwood dataset. Left to Right, test shape, results using PointNet [33] and ICP, our proposed method. We have shown each method's result in gray overlaid with the input mesh in red. Our algorithm provides significant qualitative improvement on par with those displayed in the main paper.



Figure 6: Additional qualitative result on Redwood dataset. Left to Right, test shape, results using PointNet [33] and ICP, our proposed method. We have shown each method's result in gray overlaid with the input mesh in red. Our algorithm provides significant qualitative improvement on par with those displayed in the main paper.

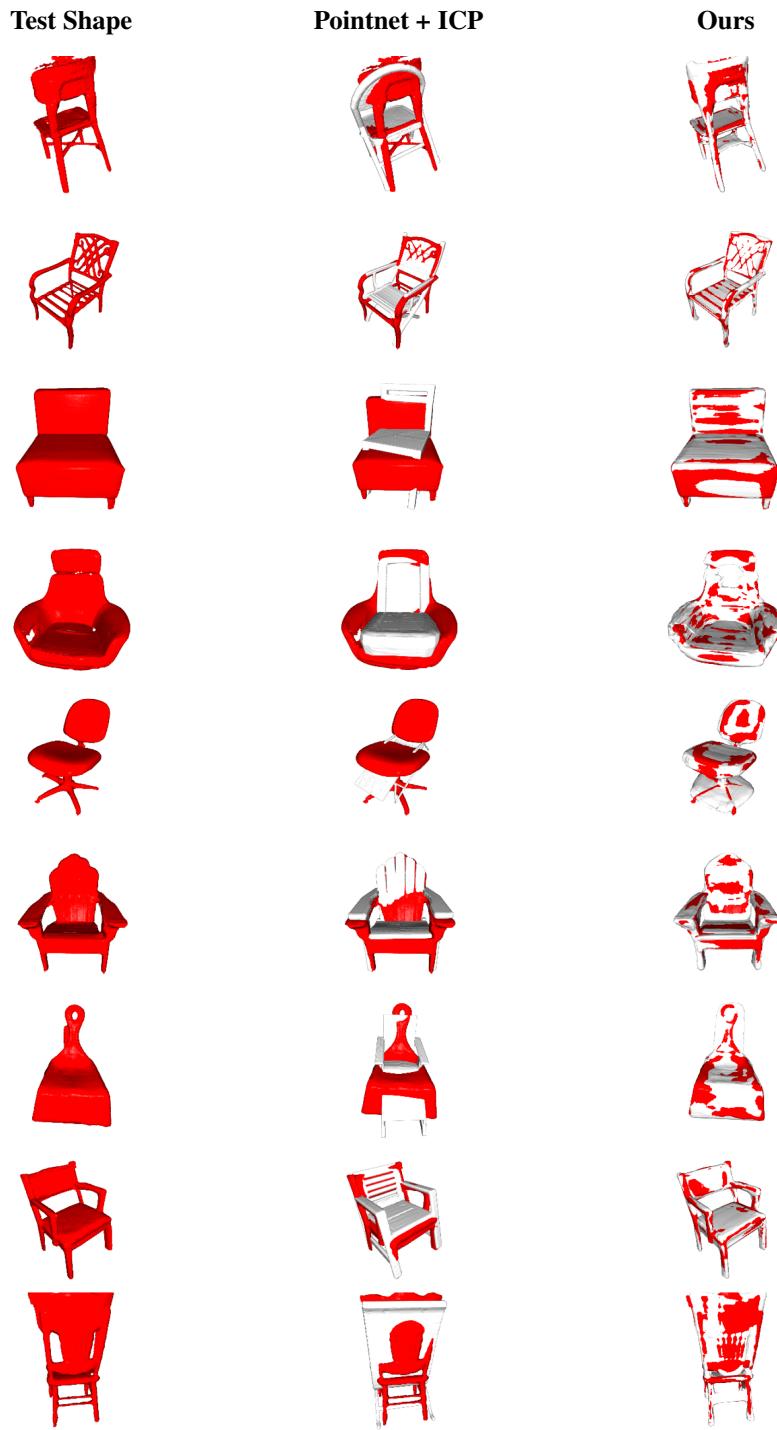


Figure 7: Additional qualitative result on Redwood dataset. Left to Right, test shape, results using PointNet [33] and ICP, our proposed method. We have shown each method's result in gray overlaid with the input mesh in red. Our algorithm provides significant qualitative improvement on par with those displayed in the main paper.



Figure 8: Additional qualitative result on Redwood dataset. Left to Right, test shape, results using PointNet [33] and ICP, our proposed method. We have shown each method's result in gray overlaid with the input mesh in red. Our algorithm provides significant qualitative improvement on par with those displayed in the main paper.

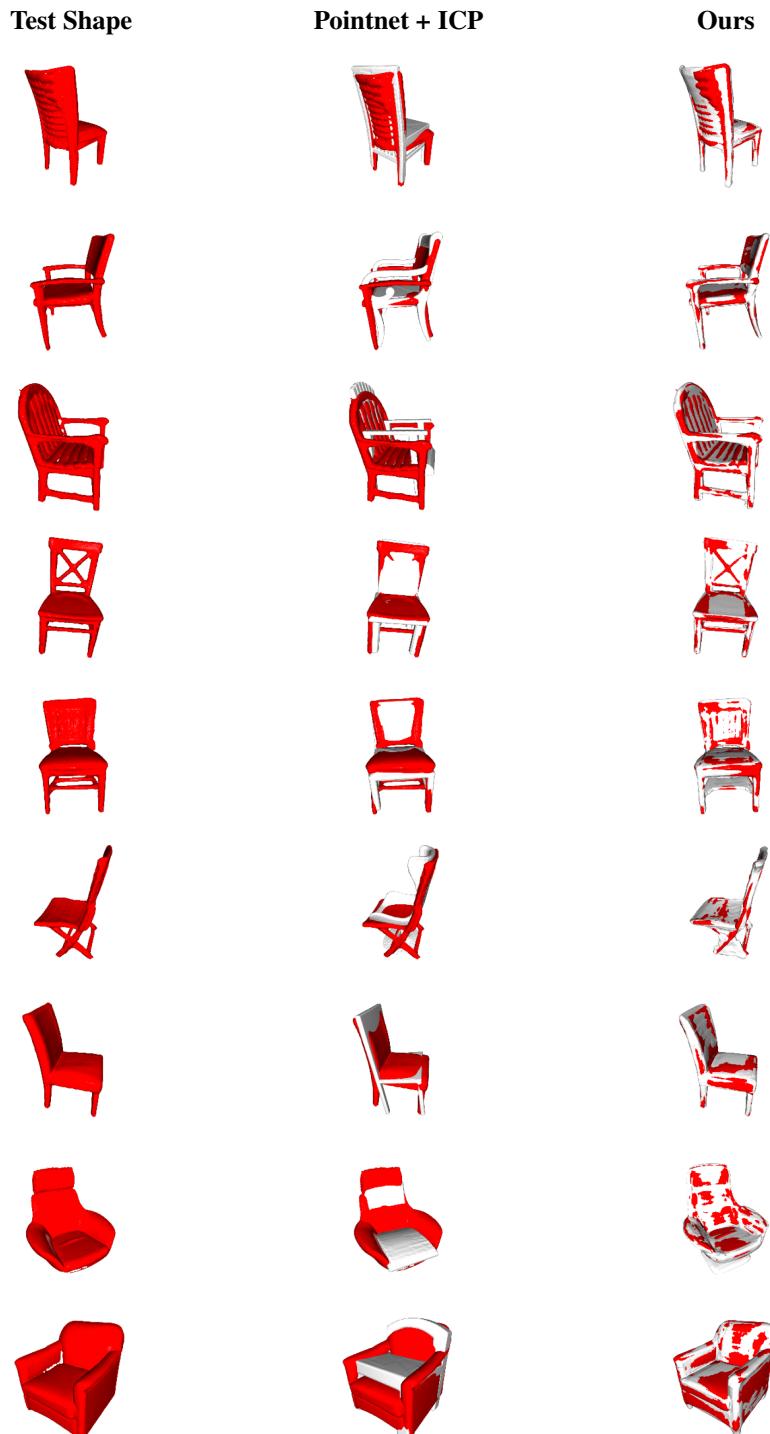


Figure 9: Additional qualitative result on Redwood dataset. Left to Right, test shape, results using PointNet [33] and ICP, our proposed method. We have shown each method's result in gray overlaid with the input mesh in red. Our algorithm provides significant qualitative improvement on par with those displayed in the main paper.



Figure 10: Additional qualitative result on Redwood dataset. Left to Right, test shape, results using PointNet [33] and ICP, our proposed method. We have shown each method's result in gray overlaid with the input mesh in red. Our algorithm provides significant qualitative improvement on par with those displayed in the main paper.