Local Deep Implicit Functions for 3D Shape

Kyle Genova\textsuperscript{1,2} Forrester Cole\textsuperscript{2} Avneesh Sud\textsuperscript{2} Aaron Sarna\textsuperscript{2} Thomas Funkhouser\textsuperscript{1,2}

\textsuperscript{1}Princeton University \quad \textsuperscript{2}Google Research

Figure 1. This paper introduces Local Deep Implicit Functions, a 3D shape representation that decomposes an input shape (mesh on left in every triplet) into a structured set of shape elements (colored ellipses on right) whose contributions to an implicit surface reconstruction (middle) are represented by latent vectors decoded by a deep network. Project video and website at ldif.cs.princeton.edu.

Abstract

The goal of this project is to learn a 3D shape representation that enables accurate surface reconstruction, compact storage, efficient computation, consistency for similar shapes, generalization across diverse shape categories, and inference from depth camera observations. Towards this end, we introduce Local Deep Implicit Functions (LDIF), a 3D shape representation that decomposes space into a structured set of learned implicit functions. We provide networks that infer the space decomposition and local deep implicit functions from a 3D mesh or posed depth image. During experiments, we find that it provides 10.3 points higher surface reconstruction accuracy (F-Score) than the state-of-the-art (OccNet), while requiring fewer than 1% of the network parameters. Experiments on posed depth image completion and generalization to unseen classes show 15.8 and 17.8 point improvements over the state-of-the-art, while producing a structured 3D representation for each input with consistency across diverse shape collections.

1. Introduction

Representing 3D shape is a fundamental problem with many applications, including surface reconstruction, analysis, compression, matching, interpolation, manipulation, and visualization. For most vision applications, a 3D representation should support: (a) reconstruction with accurate surface details, (b) scalability to complex shapes, (c) support for arbitrary topologies, (d) generalizability to unseen shape classes, (e) independence from any particular application domain, (f) encoding of shape priors, (g) compact storage, and (h) computational efficiency.

No current representation has all of these desirable properties. Traditional explicit 3D representations (voxels, meshes, point clouds, etc.) provide properties (a-e) above. They can represent arbitrary shapes and any desired detail, but they don’t encode shape priors helpful for efficient storage, 3D completion, and reconstruction tasks. In contrast, learned representations (latent vectors and deep network decoders) excel at representing shapes compactly with low-dimensional latent vectors and encoding shape priors in network weights, but they struggle to reconstruct details for complex shapes or generalize to novel shape classes.

Most recently, deep implicit functions (DIF) have been shown to be highly effective for reconstruction of individual objects [24, 7, 27, 45]. They represent an input observation as a latent vector $z$ and train a neural network to estimate the inside/outside or signed-distance function $f(x, z)$ given a query location $x$ in 3D space. This approach achieves state of the art results for several 3D shape reconstruction tasks. However, they use a single, fixed-length latent feature vector to represent the entirety of all shapes and they evaluate a complex deep network to evaluate the implicit function for every position $x$. As a result, they support limited shape complexity, generality, and computational efficiency.
Meanwhile, new methods are emerging for learning to infer structured decomposition of shapes [39, 13]. For example, [13] recently proposed a network to encode shapes into Structured Implicit Functions (SIF), which represents an implicit function as a mixture of local Gaussian functions. They showed that simple networks can be trained to decompose diverse collections of shapes consistently into SIFs, where the local shape elements inferred for one shape (e.g., the leg of a chair) correspond to semantically similar elements for others (e.g., the leg of a table). However, they did not use these structured decompositions for accurate shape reconstruction due to the limited shape expressivity of their local implicit functions (Gaussians).

The key idea of this paper is to develop a pipeline that can learn to infer Local Deep Implicit Functions, (“LDIF”, Figure 1). An LDIF is a set of local DIFs that are arranged and blended according to a SIF template. The representation is similar to SIF in that it decomposes a shape into a set of overlapping local regions represented by Gaussians. However, it also associates a latent vector with each local region that can be decoded with a DIF to produce finer geometric detail. Alternately, LDIF is similar to a DIF in that it encodes a shape as a latent vector that can be evaluated with a neural network to estimate the inside/outside function \( f(x, z) \) for any location \( x \). However, the LDIF latent vector is decomposed into parts associated with local regions of space (SIF Gaussians), which makes it more scalable, generalizable, and computationally efficient.

In this paper, we not only propose the LDIF representation, but we also provide a common system design that works effectively for 3D autoencoding, depth image completion, and partial surface completion. First, we propose to use DIF to predict local functions that are residuals with respect to the Gaussian functions predicted by SIF – this choice simplifies the task of the DIF, as it must predict only fine details rather than the overall shape within each shape element. Second, we propose to use the SIF decomposition of space to focus the DIF encoder on local regions by gathering input 3D points within each predicted shape element and encoding them with PointNet [30]. Finally, we investigate several significant improvements to SIF (rotational degrees of freedom, symmetry constraints, etc.) and simplifications to DIF (fewer layers, smaller latent codes, etc.) to improve the LDIF representation. Results of ablation studies show that each of these design choices provides significant performance improvements over alternatives. In all, LDIF achieves 10-15 points better F-Score performance on shape reconstruction benchmarks than the state-of-the-art [24], with fewer than 1% of the network parameters.

2. Related Work

Traditional Shape Representations: There are many existing approaches for representing shape. In computer graphics, some of the foundational representations are meshes [2], point clouds [11], voxel grids [11], and implicit surfaces [31, 3, 4, 25, 26, 44]. These representations are popular for their simplicity and ability to operate efficiently with specialized hardware. However, they lack two important properties: they do not leverage a shape prior, and they can be inefficient in their expressiveness. Thus, traditional surface reconstruction pipelines based on them, such as Poisson Surface Reconstruction [19], require a substantial amount of memory and computation and are not good for completing unobserved regions.

Learned Shape Representations: To leverage shape priors, shape reconstruction methods began representing shape as a learned feature vector, with a trained decoder to a mesh [36, 14, 41, 15, 18], point cloud [10, 21, 46], voxel grid [9, 43, 5, 42], or octree [37, 33, 32]. Most recently, representing shape as a vector with an implicit surface function decoder has become popular, with methods such as OcNet [24], ImNet [7], DeepSDF [27], and DISN [45]. These methods have substantially improved the state of the art in shape reconstruction and completion. However, they do not scale or generalize very well because the fundamental representation is a single fixed-length feature vector representing a shape globally.

Structured Shape Representations: To improve scalability and efficiency, researchers have introduced structured representations that encode the repeated and hierarchical nature of shapes. Traditional structured representations include scene graphs [11], CSG trees [11], and partition of unity implicit [26], all of which represent complex shapes as the composition of simpler ones. Learned representations include SDM-Net [12], GRASS [20], CSGNet [35], Volumetric Primitives [39], Superquadrics [28], and SIF [13]. These methods can decompose shapes into simpler ones, usually with high consistency across shapes in a collection. However, they have been used primarily for shape analysis (e.g. part decomposition, part-aware correspondence), not for accurate surface reconstruction or completion. Concurrent work [8, 17] adds a voxel grid structure to deep implicit functions. This helps preserve local detail but does not take advantage of consistent shape decomposition.

3. Local Deep Implicit Functions

In this paper, we propose a new 3D shape representation, Local Deep Implicit Functions (LDIF). The LDIF is a function that can be used to classify whether a query point \( x \) is inside or outside a shape. It is represented by a set of \( N \) shape elements, each parameterized by 10 analytic shape variables \( \theta_i \) and \( M \) latent shape variables \( z_i \):

\[
LDIF(x, \Theta, Z) = \sum_{i \in [N]} g(x, \theta_i)(1 + f(x, z_i))
\] (1)
where \( g(x, \theta_i) \) is a local analytic implicit function and \( f(x, z_i) \) is a deep implicit function. Intuitively, \( g \) provides a density function that defines a coarse shape and region of influence for each shape element, and \( f \) provides the shape details that cannot be represented by \( g \).

Like a typical deep implicit function, our LDIF represents a 3D shape as an isocontour of an implicit function decoded with a deep network conditioned on predicted latent variables. However, LDIF replaces the (possibly long) single latent code of a typical DIF with the concatenation of \( N \) pairs of analytic parameters \( \theta_i \) and short latent codes \( z_i \) – i.e., the global implicit function is decomposed into the sum of \( N \) local implicit functions. This key difference helps it to be more accurate, efficient, consistent, scalable, and generalizable (see Section 6).

**Analytic shape function.** The analytic shape function \( g \) defines a coarse density function and region of influence for each shape element. Any simple analytic implicit function with local support would do. We use an oriented, anisotropic, 3D Gaussian:

\[
g(x, \theta_i) = c_i e^{-\frac{||T_i x||^2}{4}}
\]  

(2)

where the parameter vector \( \theta_i \) consists of ten variables: one for a scale constant \( c_i \), three for a center point \( p_i \), three radii \( r_i \), and three Euler angles \( e_i \) (this is the same parameterization as [13], except with 3 additional DoFs for rotation).

The last 9 variables imply an affine transformation matrix \( T_i \) that takes a point \( x \) from object space coordinates to the local isotropic, oriented, centered coordinate frame of the shape element.

**Deep shape function.** The deep implicit function \( f \) defines local shape details within a shape element by modulating \( g \) (one \( f \) function is shared by all shape elements). To compute \( f \), we use a network architecture based on Occupancy Networks [24]. As in the original OccNet, ours is organized as a fully-connected network conditioned on the latent code \( z_i \), and trained using conditional batch normalization. However, one critical difference is that we transform the point \( x \) by \( T_i \) before feeding it to the network. Another critical difference is that \( f_i \) only modulates the local implicit function \( g_i \), rather than predicting an entire, global function. As a result, our local decoder has fewer network layers (9 vs. 33), shorter latent codes (32 vs. 256), and many fewer network parameters (8.6K vs 2M) than the original OccNet, and still achieves higher overall accuracy (see Section 6).

**Symmetry constraints.** For shape collections with man-made objects, we constrain a subset of the shape elements (half) to be symmetric with respect to a selected set of transformations (reflection across a right/left bisecting plane). These “symmetric” shape elements are evaluated twice for every point query, once for \( x \) and once for \( Sx \), where \( S \) is the symmetry transformation. In doing so, we effectively increase the number of shape elements without having to compute/store extra parameters for them. Adding partial symmetry encourages the shape decomposition to match global shape properties common in many shape collections and gives a boost to accuracy (Table 4).

### 4. Processing Pipeline

The processing pipeline for computing an LDIF is shown in Figure 2. All steps of the pipeline are differentiable and trained end-to-end. At inference time, the input to the system is a 3D surface or depth image, and the output is a set of shape element parameters \( \Theta \) and latent codes \( Z \) for each of \( N \) overlapping local regions, which can be decoded to predict inside/outside for any query location \( x \). Complete surfaces can be reconstructed for visualization by evaluating LDIF(\( x \)) at points on a regular grid and running Marching

![Diagram](image-url)
Cubes [23].

The exact configuration of the encoder architecture varies with input data type. We encode a 3D mesh by first rendering a stack of 20 depth images at 137 x 137 resolution from a fixed set of equally spaced views surrounding the object. We then give the depth images to an early-fusion ResNet50 [16] to regress the shape element parameters $\Theta$. Meanwhile, we generate a set of 10K points with normals covering the whole shape by estimating normals from the depth image(s) and unprojecting randomly selected pixels to points in object space using the known camera parameters.

Then, for each shape element, we select a sampling of 1K points with normals within the region of influence defined by the predicted analytic shape function, and pass them to a PointNet [30] to generate the latent code $z_i$. Alternatively, we could have encoded 3D input surfaces with CNNs based on mesh, point, or voxel convolutions, but found this processing pipeline to provide a good balance between detail, attention, efficiency, and memory. In particular, since the local geometry of every shape element is encoded independently with a PointNet, it is difficult for the network to “memorize” global shapes and it therefore generalizes better.

We encode a depth image with known camera parameters by first converting it into a 3 channel stack of 224 x 224 images representing the XYZ position of every pixel in object coordinates. We then feed those channels into a ResNet50 to regress the shape element parameters $\Theta$, and we regress the latent codes $Z$ for each shape element using the same process as for 3D meshes.

4.1. Training Losses

The pipeline is trained with the following loss $L$:

$$L(\Theta, Z) = w_P L_P(\Theta, Z) + w_C L_C(\Theta)$$

Point Sample Loss $L_P$. The first loss $L_P$ measures how accurately the LDIF($x$) predicts inside/outside of the ground-truth shape. To compute it, we sample 1024 points near the ground truth surface (set $S$) and 1024 points uniformly at random in the bounding box of the shape (set $U$). We combine them with weights $w_i \in \{w_S, w_U\}$ to form set $C = U \cup S$. The near-surface points are computed using the sampling algorithm of [13]. We scale by a hyperparameter $\alpha$, apply a sigmoid to the decoded value LDIF($x_i$), and then compute an $L_2$ loss to the ground truth indicator function $I(x)$ (see [13] for details):

$$L_P(\Theta, Z) = \frac{1}{|C|} \sum_{x_i \in C} w_i |\text{sig}(\alpha \text{LDIF}(x_i, \Theta, Z)) - I(x_i)|$$

Shape Element Center Loss $L_C$. The second loss $L_C$ encourages the center of every shape element to reside within the target shape. To compute it, we estimate a signed distance function on a low-res 32x32x32 grid $G$ for each training shape. The following loss is applied based on the grid value $G(p_i)$ at the center $p_i$ of each shape element:

$$L_C(\Theta) = \left\{ \begin{array}{ll}
\sum_{\theta_i \in \Theta} G(p_i)^2 & G(p_i) > \beta \\
0 & G(p_i) \leq \beta
\end{array} \right.$$  

Here, $\beta$ is a threshold chosen to account for the fact that $G$ is coarse. It is set to half the width of a voxel cell in $G$. This setting makes it a conservative loss: it says that when $p_i$ is definitely outside the ground truth shape, $p_i$ should be moved inside. $L_C$ never penalizes a center that is within the ground truth shape.

It is also possible for the predicted center to lie outside the bounding box of $G$. In this case, there is no gradient for $L_C$, so we instead apply the inside-bounding-box loss from [13] using the object-space bounds of $G$.

5. Experimental Setup

We execute a series of experiments to evaluate the proposed LDIF shape representation, compare it to alternatives, study the effects of its novel components, and test it in applications. Except where otherwise noted, we use $N = 32$ shape elements and $M = 32$ dimensional latent vectors during all experiments.

Datasets. When not otherwise specified, experiments are run on the ShapeNet dataset [6]. We use the train and test splits from 3D-R2N2 [9]. We additionally subdivide the train split to create an 85%, 5%, 10% train, validation, and test split. We pre-process the shapes to make them watertight using the depth fusion pipeline from Occupancy Networks [24]. We train models multi-class (all 13 classes together) and show examples only from the test split.

Metrics. We evaluate shape reconstruction results with mean intersection-over-union (IoU) [24], mean Chamfer distance [24], and mean F-Score [38] at $\tau = 0.01$. As suggested in [38], we find that IoU is difficult to interpret for low values, and Chamfer distance is outlier sensitive, so we focus our discussions mainly on F-Scores.

Baselines. We compare most of our results to the two most related prior works: Occupancy Networks [24] (OccNet), the state-of-the-art in deep implicit functions, and Structured Implicit Functions [13] (SIF), the state-of-the-art in structural decomposition. We also compare to the AtlasNet autoencoder [15], which predicts meshes explicitly.

6. Experimental Evaluation

In this section, we report results of experiments that compare LDIF and baselines with respect to how well they satisfy desirable properties of a 3D shape representation.
<table>
<thead>
<tr>
<th>Category</th>
<th>IoU (%)</th>
<th>Chamfer (%)</th>
<th>F-Score (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Occ.</td>
<td>SIF</td>
<td>Ours</td>
</tr>
<tr>
<td>airplane</td>
<td>77.0</td>
<td>66.2</td>
<td><strong>91.2</strong></td>
</tr>
<tr>
<td>bench</td>
<td>71.3</td>
<td>53.3</td>
<td><strong>85.6</strong></td>
</tr>
<tr>
<td>cabinet</td>
<td>86.2</td>
<td>78.3</td>
<td><strong>93.2</strong></td>
</tr>
<tr>
<td>car</td>
<td>83.9</td>
<td>77.2</td>
<td><strong>90.2</strong></td>
</tr>
<tr>
<td>chair</td>
<td>73.9</td>
<td>57.2</td>
<td><strong>87.5</strong></td>
</tr>
<tr>
<td>display</td>
<td>81.8</td>
<td>69.3</td>
<td><strong>94.2</strong></td>
</tr>
<tr>
<td>lamp</td>
<td>56.5</td>
<td>41.7</td>
<td><strong>77.9</strong></td>
</tr>
<tr>
<td>rifle</td>
<td>69.5</td>
<td>60.4</td>
<td><strong>89.9</strong></td>
</tr>
<tr>
<td>sofa</td>
<td>87.2</td>
<td>76.0</td>
<td><strong>94.1</strong></td>
</tr>
<tr>
<td>speaker</td>
<td>82.4</td>
<td>74.2</td>
<td><strong>90.3</strong></td>
</tr>
<tr>
<td>table</td>
<td>75.6</td>
<td>57.2</td>
<td><strong>88.2</strong></td>
</tr>
<tr>
<td>telephone</td>
<td>90.9</td>
<td>83.1</td>
<td><strong>97.6</strong></td>
</tr>
<tr>
<td>watercraft</td>
<td>74.7</td>
<td>64.3</td>
<td><strong>90.1</strong></td>
</tr>
<tr>
<td>mean</td>
<td>77.8</td>
<td>66.0</td>
<td><strong>90.0</strong></td>
</tr>
</tbody>
</table>

Table 1. Autoencoder results. Comparison of 3D-R2N2 test set reconstruction errors for OccNet ("Occ."), SIF ("SIF") [13], AtlasNet ("Atl."), [15], and LDIF ("Ours") autoencoders.

Figure 3. Autoencoder examples. F-scores for the test set (8746 shapes) are shown ordered by the LDIF F-score, with examples marked with their position on the curve. Our reconstructions (blue curve) are most accurate for 93% of shapes (exact scores shown faded). The scores of OccNet and SIF follow roughly the same curve as LDIF (rolling means shown bold), indicating shapes are similarly difficult for all methods. Solid shapes such as the rifle are relatively easy to represent, while shapes with irregular, thin structures such as the lamp are more difficult.

Accuracy. Our first experiment compares 3D shape representations in terms of how accurately they can encode/decode shapes. For each representation, we compare a 3D→3D autoencoder trained on the multiclass training data, use it to reconstruct shapes in the test set, and then evaluate how well the reconstructions match the originals (Table 1). LDIF’s mean F-Score is 92.2, 10.3 points higher than OccNet, 20.0 points higher than AtlasNet, and 33.2 points higher than SIF. A more detailed breakdown of the results appears in Figure 3, which shows the F-scores for all models in the test set – LDIF improves on OccNet’s score for 93% of test shapes. The increase in accuracy translates into a large qualitative improvement in results (shown above in Figure 3). For example, LDIF often reproduces better geometric details (e.g., back of the bench) and handles unusual part placements more robustly (e.g., handles on the rifle).

Efficiency. Our second experiment compares the efficiency of 3D shape representations in terms of accuracy vs. storage/computation costs. Since LDIF can be trained with different numbers of shape elements (N) and latent feature sizes (M), a family of LDIF representations is possible, each with a different trade-off between storage/computation and accuracy. Figure 4 investigates these trade-offs for several combinations of N and M and compares the accuracy of their autoencoders to baselines. Looking at the plot on the top, we see that LDIF provides more accurate reconstructions than baselines at every decoder size – our decoder with N = 32 and M = 32 is 0.004× the size of OccNet and provides 1.13× better F-Score. On the bottom, we see that LDIF performs comparably to OccNet and outperforms SIF.
at the same number of bytes, despite having both deep and analytic parameters, and that it scales to larger embeddings.

**Consistency.** Our third experiment investigates the ability of LDIF to decompose shapes consistently into shape elements. This property was explored at length in [13] and shown to be useful for structure-aware correspondences, interpolations, and segmentations. While not the focus of this paper, we find qualitatively that the consistency of the LDIF representation is slightly superior to SIF, because the shape element symmetries and rotations introduced in this paper provide the DoFs needed to decompose shapes with fewer elements. On the other hand, the local DIFs are able to compensate for imperfect decompositions during reconstruction, which puts less pressure on consistency. Figure 5 shows qualitative results of the decompositions computed for LDIF. Please note the consistency of the colors (indicating the index of the shape element) across a broad range of shapes.

**Generalizability.** Our third experiment studies how well trained autoencoders generalize to handle unseen shape classes. To test this, we used the auto-encoders trained on 3D-R2N2 classes and tested them without fine-tuning on a random sampling of meshes from 10 ShapeNet classes that were not seen during training. Table 2 shows that the mean F-Score for LDIF on these novel classes is 84.4, which is 17.8 points higher than OccNet and 41.4 points higher than SIF. Looking at the F-Score for every example in the bottom of Figure 6, we see that LDIF is better on 91% of examples. We conjecture this is because LDIF learns to produce consistent decompositions for a broad range of input shapes when trained multiclass, and because the local encoder network learns to predict shape details only for local regions. This two-level factoring of structure and detail seems to help LDIF generalize.

**Domain-independence.** Our fifth experiment investigates whether LDIF can be used in application domains beyond the man-made shapes found in ShapeNet. As one example, we trained LDIF without any changes to autoencode meshes of human bodies in a wide variety of poses sampled from [40]. Specifically, we generated 5M meshes by randomly sampling SMPL parameters (CAESAR fits for shape, mocap sequence fits for pose). We used an 80%, 5%, 15% split for the train, val, and test sets, similar to [40], and measured the error of the learned autoencoder on the held-out test set. The challenge for this dataset is quite dif-

<table>
<thead>
<tr>
<th>Category</th>
<th>Chamfer (↓)</th>
<th>F-Score (↑, %)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SIF</td>
<td>OccNet</td>
</tr>
<tr>
<td>bed</td>
<td>2.24</td>
<td>1.30</td>
</tr>
<tr>
<td>birdhouse</td>
<td>1.92</td>
<td>1.25</td>
</tr>
<tr>
<td>bookshelf</td>
<td>1.21</td>
<td>0.83</td>
</tr>
<tr>
<td>camera</td>
<td>1.91</td>
<td>1.17</td>
</tr>
<tr>
<td>file</td>
<td>0.71</td>
<td>0.41</td>
</tr>
<tr>
<td>mailbox</td>
<td>1.46</td>
<td>0.60</td>
</tr>
<tr>
<td>piano</td>
<td>1.81</td>
<td>1.07</td>
</tr>
<tr>
<td>printer</td>
<td>1.44</td>
<td>0.85</td>
</tr>
<tr>
<td>stove</td>
<td>1.04</td>
<td>0.49</td>
</tr>
<tr>
<td>tower</td>
<td>1.05</td>
<td>0.50</td>
</tr>
<tr>
<td>mean</td>
<td>1.48</td>
<td>0.85</td>
</tr>
</tbody>
</table>

Table 2. Generalization to unseen classes. Comparison of 3D reconstruction accuracy when 3D autoencoders are tested directly on ShapeNet classes not seen during training. Note that our method (LDIF) has a higher F-Score by 17.8 points.
Human body modeling. Surface reconstructions and decompositions for 4 random SMPL [22] human meshes from the SURREAL [40] dataset. For each triple, from left to right: SMPL mesh, our reconstruction, our shape decomposition. These results demonstrate unsupervised correspondence between people in different poses as well as accurate reconstructions of organic shapes.

ferent than for ShapeNet – the autoencoder must be able to represent large-scale, non-rigid deformations in addition to shape variations. Our reconstructions achieve 93% mIOU compared to 85% mIOU for SIF. The results of LDIF reconstructions and the underlying SIF templates are shown in Figure 7. Despite a lack of supervision on pose or subject alignment, our approach reconstructs a surface close to the original and establishes coarse correspondences.

7. Applications

In this section, we investigate how the proposed LDIF representation can be used in applications. Although SIF (and similarly LDIF) has previously been shown useful for 3D shape analysis applications like structure-aware shape interpolation, surface correspondence, and image segmentation [13], we focus our study here on 3D surface reconstruction from partial observations.

7.1. 3D Completion from a Single Depth Image

Task. Reconstructing a complete 3D surface from a single depth image is an important vision task with applications in AR, robotics, etc. To investigate how LDIF performs on this task, we modified our network to take a single depth image as input (rather than a stack of 20) and trained it from scratch on depth images generated synthetically from random views of the 3D-R2N2 split of shapes. The depth images were 512 x 512 to approximate the resolution of real depth sensors (though all CNN inputs are 224 x 224 due to memory restrictions). The depth images were rendered from view points sampled from all view directions and at variable distances to mimic the variety of scan poses. Each depth image was then converted to a three channel XYZ image using the known camera parameters.

Baseline. For comparison, we trained an OccNet network from scratch on the same data. Because the OccNet takes a point cloud rather than depth images, we train an XYZ image encoder network to regress the 256-D OccNet embedding. This OccNet* model provides an apples-to-apples baseline that isolates differences due only to the representation decoding part of the pipeline.

Results. Table 3 shows results of this 3D depth completion experiment. We find that the F-Score of LDIF is 15.8 points higher than OccNet* (78.2 vs. 62.4). Figure 8 highlights the difference in the methods qualitatively. As in the 3D case, we observe that LDIF’s local part encoders result in substantially better performance on hard examples.

Ablation study. To further understand the behavior of LDIF during depth completion, we ablate three components of our pipeline (Table 4). First, we verify that having local

Table 3. Depth completion accuracy. Our method (LDIF) provides better 3D surface completions than an OccNet* trained on our XYZ image inputs.

<table>
<thead>
<tr>
<th>Category</th>
<th>IoU (↑)</th>
<th>Chamfer (↓)</th>
<th>F-Score (↑, %)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>OccNet*</td>
<td>Ours</td>
<td>OccNet*</td>
</tr>
<tr>
<td>airplane</td>
<td>-</td>
<td>80.2</td>
<td>0.47</td>
</tr>
<tr>
<td>bench</td>
<td>-</td>
<td>70.9</td>
<td>0.70</td>
</tr>
<tr>
<td>cabinet</td>
<td>-</td>
<td>82.8</td>
<td>1.13</td>
</tr>
<tr>
<td>car</td>
<td>-</td>
<td>81.4</td>
<td>0.99</td>
</tr>
<tr>
<td>chair</td>
<td>-</td>
<td>70.6</td>
<td>2.34</td>
</tr>
<tr>
<td>display</td>
<td>-</td>
<td>82.4</td>
<td>0.95</td>
</tr>
<tr>
<td>lamp</td>
<td>-</td>
<td>62.1</td>
<td>9.91</td>
</tr>
<tr>
<td>rifle</td>
<td>-</td>
<td>81.5</td>
<td>0.49</td>
</tr>
<tr>
<td>sofa</td>
<td>-</td>
<td>81.4</td>
<td>1.08</td>
</tr>
<tr>
<td>speaker</td>
<td>-</td>
<td>80.2</td>
<td>3.50</td>
</tr>
<tr>
<td>table</td>
<td>-</td>
<td>73.5</td>
<td>2.49</td>
</tr>
<tr>
<td>telephone</td>
<td>-</td>
<td>92.3</td>
<td>0.35</td>
</tr>
<tr>
<td>watercraft</td>
<td>-</td>
<td>76.0</td>
<td>1.15</td>
</tr>
</tbody>
</table>

mean     | -       | 78.1        | 1.97           | 0.76          | 62.4          | 78.2          |
Table 4. Depth completion ablation study. Local PointNet encoders, camera transformations, and partial symmetry all improve performance. Independently and locally encoding the input improves the Z with PointNet is particularly good for generalization (see Section 6).

<table>
<thead>
<tr>
<th>Method</th>
<th>IoU (↑)</th>
<th>Chamfer (↓)</th>
<th>F-Score (↑, %)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full (D)</td>
<td>77.2</td>
<td>0.78</td>
<td>77.6</td>
</tr>
<tr>
<td>No PointNet</td>
<td>69.1</td>
<td>0.98</td>
<td>66.2</td>
</tr>
<tr>
<td>No Transform</td>
<td>71.9</td>
<td>1.80</td>
<td>71.9</td>
</tr>
<tr>
<td>No Symmetry</td>
<td>76.7</td>
<td>0.76</td>
<td>76.6</td>
</tr>
</tbody>
</table>

7.2. Reconstruction of Partial Human Body Scans

Task. Acquisition of complete 3D surface scans for a diverse collection of human body shapes has numerous applications [1]. Unfortunately, many real world body scans have holes (Figure 9a), due to noise and occlusions in the scanning process. We address the task of learning to complete and beautify the partial 3D surfaces without any supervision or even a domain-specific template.

Dataset and baselines. The dataset for this experiment is CAESAR [34]. We use our proposed 3D autoencoder to learn to reconstruct an LDIF for every scan in the CAESAR dataset, and then we extract watertight surface from the LDIFs (using the splits from [29]). For comparisons, we do the same for SIF (another unsupervised method) and a non-rigid deformation fit of the S-SCAPE template [29].

Results. Figure 9 shows representative results. Note that LDIF captures high-frequency details missing in SIF reconstructions. Although the approach based on S-SCAPE provides better results, it requires a template designed specifically for human bodies as well as manual supervision (landmarks and bootstrapping), whereas LDIF is domain-independent and unsupervised. These results suggest that LDIF could be used for 3D reconstruction of other scan datasets where templates are not available.

8. Conclusion

Summary of research contributions: In this paper, we propose Local Deep Implicit Functions (LDIF), a new 3D representation that describes a shape implicitly as the sum of local 3D functions, each evaluated as the product of a Gaussian and a residual function predicted with a deep network. We describe a method for inferring an LDIF from a 3D surface or posed depth image by first predicting a structured decomposition into shape elements, encoding 3D points within each shape element using PointNet [30], and decoding them with a small residual decoder. This approach provides an end-to-end framework for encoding shapes in local regions arranged in a global structure.

We show that this LDIF representation improves both reconstruction accuracy and generalization behavior over previous work – its F-Score results are better than the state-of-the-art [24] by 10.3 points for 3D autoencoding of test models from trained classes and by 17.8 points for unseen classes. We show that it dramatically reduces network parameter count – its local decoder requires approximately 0.4% of the parameters used by [24]. We show that it can be used to complete posed depth images – its depth completion results are 15.8 percentage points higher than [24]. Finally, we show that it can be used without change to reconstruct complete 3D surfaces of human bodies from partial scans.

Limitations and future work: Though the results are encouraging, there are limitations that require further investigation. First, we decompose space into a flat set of local regions – it would be better to consider a multiresolution hierarchy. Second, we leverage known camera poses when reconstructing shapes from depth images – it would be better to estimate them. Third, we estimate a constant number of local regions – it would be better to derive a variable number dynamically during inference (e.g., with an LSTM). Finally, we just scratch the surface of how structured and implicit representations can be combined – this is an interesting topic for future research.

Acknowledgements: We thank Boyang Deng for sharing OccNet* training code and Max Jiang for creating single-view depth renders. We also thank Fangyin Wei and JP Lewis for feedback on the manuscript.
References


