Gradient Distance Function

Hieu Le[1]   Federico Stella[2]   Benoit Guillard[2]   Pascal Fua[2]
[1]UNC-Charlotte   [2]EPFL

hle40@charlotte.edu   first-name.last-name@epfl.ch

Abstract

Unsigned Distance Functions (UDFs) can be used to represent non-watertight surfaces in a deep learning framework. However, UDFs tend to be brittle and difficult to learn, in part because the surface is located exactly where the UDF is non-differentiable. In this work, we show that Gradient Distance Functions (GDFs) can remedy this by being differentiable at the surface while still being able to represent open surfaces. This is done by associating to each 3D point a 3D vector whose norm is taken to be the unsigned distance to the surface and whose orientation is taken to be the direction towards the closest surface point. We demonstrate the effectiveness of GDFs on ShapeNet Car, Multi-Garment, and 3D-Scene datasets with both single-shape reconstruction networks or categorical auto-decoders.

1. Introduction

Signed Distance Functions (SDFs) excel at representing deep implicit surfaces [4, 14]. However, they are best suited for representing watertight surfaces. They can also be used to represent non-watertight ones but that usually involves inflating the surface by fitting an implicit surface around it, which incurs a loss of accuracy. Unsigned Distance Functions (UDFs) can remedy this [5, 7, 10] but pose their own challenges: they are strictly positive everywhere except for zero-values at surface points, which means that the surface is located exactly at the non-differentiable part of the distance field. Consequently, optimizing a UDF is numerically unstable, resulting the function never reaching the zero value and creating holes in the resulting surface, or reaching zero in a volume and creating double surfaces. Hence, UDF-based representations are more difficult to learn than their SDF-based counterparts.

In this work, we show that instead of learning the distance directly, learning the gradient of the distance function can overcome these limitations. Given a surface we wish to represent, we associate to each point in a volume containing it the vector pointing to the closest surface point. Its norm is the unsigned distance to the surface, while its direction is the
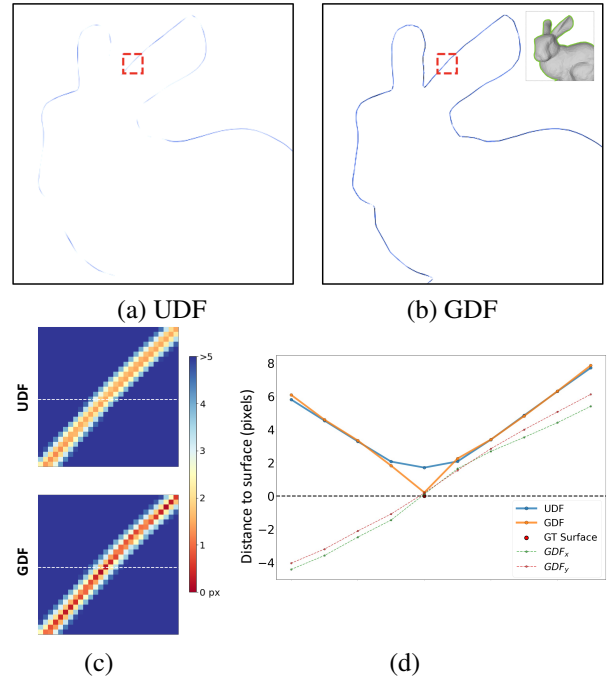

(a) UDF   (b) GDF

(c)   (d)

Figure 1. **UDF vs GDF.** (a) We trained a network to output a UDF representing an open 2D contour for a portion of the Stanford Bunny. The pixels whose distances to the resulting contours is smaller than 2 pixels are shown in blue. Note that many parts of the contour are missing. (b) We repeated the experiment but using our GDF instead and there are no more holes. (c) Estimated distances to the contour in the small patch denotes by the red box in (a,b). (d) Value of the UDF and GDF along the dotted line in (c). Note that the UDF never reaches 0, resulting in a hole in the contour. This problem goes away when using the GDF, hence the much better defined contour in (b).

negative gradient of the unsigned distance. When crossing the surface, each vector component switches signs while its magnitude goes to zero and increases again afterwards. In essence, we represent a non-watertight surface by three surface differentiable sub functions, as depicted in Fig. 1 for a 2D case.

We show that this gradient distance function, namely GDF, is easy to learn through deep networks. The recon-

structed surface is smooth and without staircase artefacts that often exhibit in the UDF counterparts. Additionally, the GDF representation inherently includes gradient direction, a critical factor for meshing non-watertight surfaces [7]. Like UDFs, GDFs can also be parameterized by latent vectors, allowing a pre-trained network to reconstruct 3D meshes from partial observations, such as 3D points on the surface of a target object. We conduct various experiments on the Multi-Garment, ShapeNet-Car, and 3DScenes datasets to demonstrate the effectiveness of GDFs, where they outperform UDFs with both single-shape reconstruction networks or categorical auto-decoders.

2. Related Work

Deep implicit surfaces [4, 13, 14] excel at modeling watertight surfaces using occupancy grids and SDFs. They can also be used to represent open surfaces by inflating an SDF around them [6, 7, 16]. This involves wrapping a watertight surface around them. If needed, a triangulated surface can then be obtained using the Marching Cubes algorithm [11]. However, this method is resolution-dependent and the resulting surface will be some distance away from the target surface, resulting in a loss of accuracy.

Unsigned Distance Functions (UDFs). Thus, an attractive alternative is to use UDFs [3, 5, 6, 9, 12, 17, 18] because they do not suffer from this inaccuracy issue. Furthermore, triangulations can be obtained from them by reasoning on their gradients [7, 21]. However, a UDF is non-differentiable at its zero level set, which is precisely where the surface is supposed to be. This is not easy to learn for a deep network and can result in the UDF values never reaching zero or remaining zero in a whole volume, leading to the surface disappearing or appearing as a being doubled.

Several approaches to mitigating these drawbacks have been proposed. In [23], the level-sets close to the surface are constrained to be parallel to it to learn UDFs from sparse point clouds. In [22], 3D position features extracted via a set of anchor points on the surface are used as additional inputs for predicting UDF values. This approach improves single-view garment reconstruction. Some methods involve learning additional information. In [18], an additional network is used to predict a pseudo-sign for each spatial point, defined via the surface gradient of the closest surface point while the algorithm of [17] predicts surface orientation together with the UDF for improved modeling of local geometric details. However, none of these methods addresses the challenge inherent to training deep networks to predict UDF values accurate enough to prevent the creation of holes and non-smooth surfaces.

Open Surface Representations. Instead of improving UDF-based schemes, one can also propose different representation schemes for open surfaces. In 3PSDF [3], they are
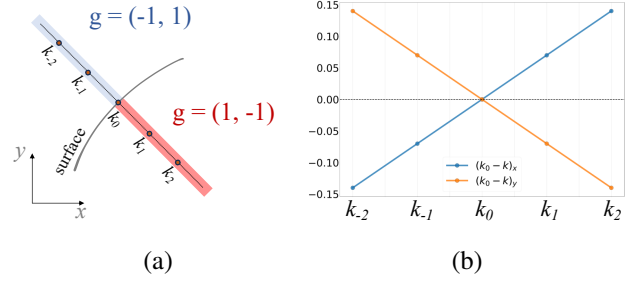

(a)　　　　　　　　　(b)

Figure 2. **GDF's surface differentiability.** (a) Points on different sides of the surface have gradients with opposite signs. (b) GDF is the product of normalized gradient and unsigned distance. It monotonically changes from one sign to the other when crossing the surface, making it differentiable.

modeled by using occupancy grids, by masking out empty regions, and by classifying unmasked grid cells as either positive or negative. These pseudo signs provide a notion of "inside" or "outside" the local surface based on the surface gradient of the nearest surface point, as in [18]. However, this method requires pre-orienting the training data, and does not provide the continuous distances required for precise meshing. Instead of directly learning the UDF, the CSP algorithm of [16] predicts the closest surface point. This representation is differentiable at the surface, but the scalar field does not exhibit distance-like continuity and is challenging to learn. In contrast, our proposed GDF predicts the vector pointing to the nearest surface point. When crossing the surface: like SDFs, GDFs scalar field changes from positive to negative or vice versa, while CSP yields constant values. As a result, CSPs are harder to train and yield less precise reconstructions, as we will demonstrate. One approach that is related to our method and CSP is NVF [20], which predicts the gradient vector for each query point using codebooks to memorize UDF patterns that match the input point cloud. Our method, whereas, shows that directly learning gradient distance functions yields a general-purpose, surface-differentiable representation—well beyond codebook-style point-cloud modeling—while remaining simple to train with a standard auto-decoder.

3. Method

In this section we describe how the Gradient Distance Functions (GDF) is differentiable at the surface and more robust than the original Unsigned Distance Function (UDF), which can effectively represent non-watertight shapes.

3.1. Definition

We define a GDF as the function

$$\text{GDF} : \mathbb{R}^3 \to \mathbb{R}^3,$$
$$\mathbf{x} \to \mathbf{v} = \hat{\mathbf{x}} - \mathbf{x}, \quad (1)$$

which associates to each 3D point $\mathbf{x}$ the vector $\mathbf{v}$ pointing to the closest surface point $\hat{\mathbf{x}}$. The unsigned distance u and the unit-norm gradient $\mathbf{g}$ can be estimated from $\mathbf{v}$ as

$$u = \|\mathbf{v}\|, \quad (2)$$
$$\mathbf{g} = \frac{1}{u}\mathbf{v}. \quad (3)$$

As with UDFs, the surface is defined implicitly as the location where $\mathbf{v} = [0, 0, 0]$ and, consequently, u is zero and $\mathbf{g}$ is taken to be the null vector. While a UDF is non-differentiable at a surface point, our GDF is. Each component of $\mathbf{v}$ switches sign when crossing the surface while its magnitude goes to zero, which makes the behavior of GDF similar to that of SDF in this respect. Fig. 2 illustrates this for a 2D case: Let us consider a set of points $\{k_i, i \in [-N, N]\}$ sampled along a line perpendicular to the surface and crossing it at k_0, which makes it the surface point closest to all of them. When crossing the surface, one element of $\mathbf{v}$ goes from positive to negative while the other goes from negative to positive.

3.2. GDFs as Output of a Deep Network

Deep networks can be trained to regress GDFs, instead of SDFs or UDFs, in essentially the same manner.

In a fully supervised setup, we would be given a set S of N training pairs

$$S = \{S_i = (\mathbf{x}^i, \mathbf{v}^i), 1 \le i \le N\},$$
$$\mathbf{x}^i = \{\mathbf{x}_1^i \dots \mathbf{x}_k^i\}, \quad (4)$$
$$\mathbf{v}^i = \{\mathbf{v}_1^i \dots \mathbf{v}_k^i\},$$

where the $\mathbf{x}_{1\dots k}^i$ are sampled points for a particular shape S_i and the $\mathbf{v}^i$ represent the corresponding GDF vectors. Similar to deepSDF [15], we sample more aggressively near the surface and a small fraction of points (5%) is uniformly sampled throughout the entire volume.

Let $f_\Phi(\cdot, \mathbf{C})$ be a function implemented by a network with weights Φ that takes as input a 3D point and a latent code $\mathbf{C}_i$ associated to each S_i. We want to guarantee that

$$\forall i, j, \quad f_\Phi(\mathbf{x}_j^i, \mathbf{C}_i) \approx \mathbf{v}_j^i. \quad (5)$$

To this end, we minimize with respect to the weights Φ and the latent vectors $\mathbf{C}_i$ the loss function

$$\mathcal{L}(\Phi, \mathbf{C}_1, \dots, \mathbf{C}_N) = \sum_{i,j} |\hat{\mathbf{v}}_j^i - \mathbf{v}_j^i|, \quad (6)$$
$$\hat{\mathbf{v}}_j^i = f_\Phi(\mathbf{x}_j^i, \mathbf{C}_i).$$

4. Experiments

We now demonstrate the representational abilities of GDFs in two different scenarios. In the first, we train a network to represent a single 3D scene. In the second. we use an auto-decoding approach to learning a latent vector representation for a class of shapes. We show that, in both cases, GDFs are easy to learn and capture fine geometric details. In the second scenario, they also generalize well to unseen shapes of the same category.

4.1. Datasets, Metrics, Baselines, and Settings

We use three main datasets for our experiments: MGN [1], ShapeNet-Car [2], and 3D-Scene [24]. MGN comprises 320 garments. They are mainly made of single-layer with a smooth 3D shape. We use two versions of the ShapeNet-Car dataset, which we denote by *ShapeNet-Car(R)*, *ShapeNet-Car(P)*. The first comprises 3091 cars with open surfaces and inner structures. The second is a cleaned-up version with 3D car models pre-processed to yield closed surfaces without interiors [19]. Last, the 3D-Scene dataset comprises seven samples of 3D scenes with highly complex geometry and many details.

Metrics. To evaluate performance, we use the Chamfer distance (CD) and the Normal consistency (NC). The former measures the distance between 3D points sampled from the surfaces, the lower the better. The latter quantifies the surface normal similarity between the predicted mesh and ground-truth mesh, the higher the better.

Baselines. We compare our method against three other implicit-surface methods, NDF [5] that predicts UDF values directly, CSP [18] that regresses the closest surface point, and 3PSDF [3] that labels points as either "−1","1", or "*null*". To ensure fair comparison in each experiment, we train different networks with the same architecture, training data, and optimization settings. The only difference is how the output is represented. For each training instance, we extract 400000 points near the surface and 20000 points uniformly sampled throughout the entire volume. We use Adam optimizer [8] with a starting learning rate of 10^{-4} that decreases by 25% after each quater of the training. We use MeshUDF [7] as the meshing algorithm for all methods except for the 3PSDF representation that can be meshed using Marching Cubes. MeshUDF reasons on the gradient of the UDF for meshing. For NDF, we compute this gradient by running a back-propagating pass through the network, as in deepSDF [14]. For CSP and our method, the gradient can be computed directly from the output representation. In our case, it is given by Eq. 3, without the need to back-propagate. We never perform any post-processing step.
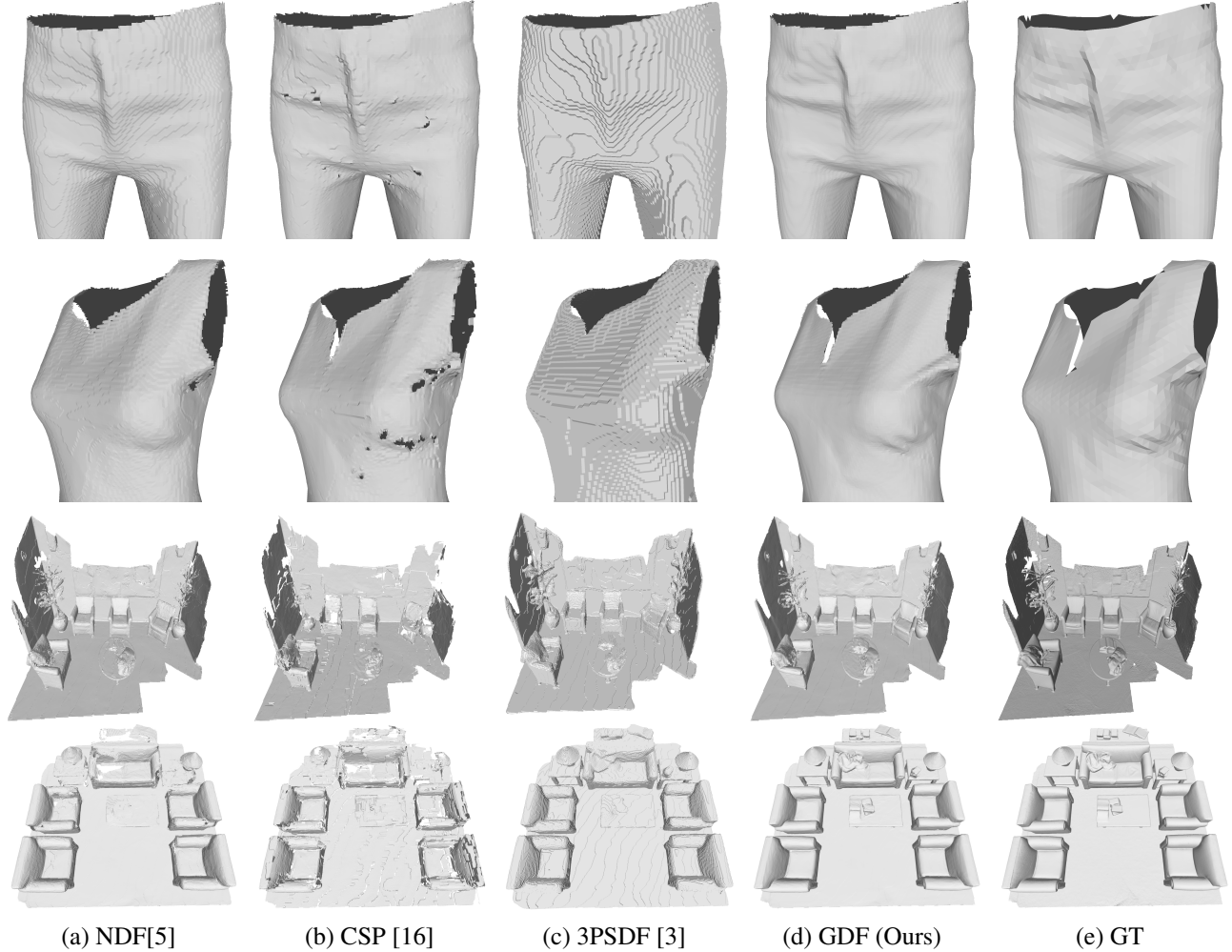
Figure 3. **Single 3D Scene Reconstruction.** We train identical MLPs to reconstruct a single 3D surface using different implicit representations. The top two rows depict garments from the MGN dataset [1]. The bottom two rows are examples from the 3D-Scene dataset [24].

4.2. Representing a Single Complex Shape

We first evaluate all methods on a basic test aimed at reconstructing a single 3D shape via a deep network. In this test, we use an 8-layer multi-layer perceptron (MLP) with 512 feature maps per layer and train the network to regress the target representations for sampled query points (identical in all cases). For the MGN dataset, we train all networks for 30000 iterations. For 3D-Scene, which is significantly more complex, we train them for 100000 iterations. All outputs are meshed at 512^3 resolutions for evaluation.

In Table 1, we report the Chamfer Distances $\times 10^4$ (CD) and the Normal Consistency of the resulting surfaces from different methods. Our proposed GDF produces smooth meshes that closely match the ground-truth ones in all cases, as shown in Fig. 3. This is because surfaces in our GDF representation locate in continuous, differentiable parts of the scalar field, enabling the deep network to easily memorize

them. On the contrary, the results of NDF exhibit staircase artifacts and surface holes, typical issues associated with the direct learning of UDFs. CSP creates smoother surfaces. However, they have many holes, despite being trained for an identical number of iterations as other methods. Compared to ours, this representation is more difficult to learn as it does not exhibit distance-like scalar field. The 3PSDF representation makes network training easier with fast convergence and no holes in the reconstructed surfaces, even for challenging cases such as the "reading room" of the 3D-Scene dataset shown in bottom row of the figure. However, this efficiency comes with a cost of discretizing distance values into either "-1" or "1," resulting in less accurate meshes with noticeable staircase artifacts.

Table 1. **Fitting to a single complex mesh.** Average L2 Chamfer Distance $\times 10^{-4}$ (CD) and Normal Consistency (NC) for the MGN and 3D-Scenes datasets. We use the same MLP for each scene and only change the output representation.

	Garments		3D Scenes	
	CD ($\downarrow$)	NC($\uparrow$)	CD ($\downarrow$)	NC($\uparrow$)
NDF[5]	0.22	99.10	0.27	85.7
CSP[16]	0.22	98.38	3.01	71.5
3PSDF[3]	0.32	96.73	0.60	83.2
GDF (Ours)	**0.21**	**99.16**	**0.20**	**87.1**

Table 2. **Representing known shapes using an auto-decoder.** Average L2 Chamfer Distance $\times 10^{-4}$ (CD) and normal consistency (NC) for the MGN and ShapeNet-Car datasets.

	Garments		SN-Car (P)		SN-Car (R)	
	CD ($\downarrow$)	NC($\uparrow$)	CD ($\downarrow$)	NC($\uparrow$)	CD ($\downarrow$)	NC($\uparrow$)
NDF[5]	0.47	95.42	0.88	93.79	0.76	58.73
CSP[16]	1.07	92.72	0.95	90.17	0.85	55.37
3PSDF[3]	0.41	91.52	0.69	91.89	0.54	57.35
GDF (Ours)	**0.32**	**97.15**	**0.67**	**93.87**	**0.34**	**62.66**

4.3. Auto-Decoders to Model a Category of Shapes

Auto-decoders were introduced in [14] and are now an important 3D shape modeling tool. They make it possible to represent multiple 3D shapes from a given category using a single network, each one being characterized by a latent vector. The auto-decoder learns a shape prior for the category, enabling the reconstruction of unseen instances from partial input observations. We evaluate two aspects of using auto-decoder in conjunction with our proposed implicit representation: the capacity of the network to memorize training instances and the modeling of previously unseen data given 3D point clouds.

Representing Known Shapes. We train auto-decoders on three datasets of increasing complexity: MGN, ShapeNet-Car(P), and ShapeNet-Car(R). We use 12-layer MLPs for MGN and ShapeNet-Car(P) and 18-layer MLPs for ShapeNet-Car(R) with 1024 feature maps per layer. They are trained by minimizing the reconstruction loss of Eq. 6. The code length is set to 512 and the resolution of the output meshes is 256^3 in all cases.

We report quantitative comparative results in Table 2. GDF yields significantly more accurate meshes than the other methods in terms of both metrics. Fig. 4 features four reconstructed cars. The GDF reconstructions are smooth with accurate details, while the others exhibit some of the same issues as in the single-scene examples of Fig. 3.

Table 3. **Representing previously unseen shapes using an auto-decoder.** Average L2 Chamfer Distance $\times 10^{-4}$ (CD) and normal consistency (NC) for the MGN and closed ShapeNet-Car datasets.

	SN-Car (P)		SN-Car (R)	
Input Points	3K	10K	3K	10K
NDF[5]	7.54	1.53	1.65	0.78
CSP[16]	5.91	1.99	1.12	0.87
GDF (Ours)	**1.32**	**0.86**	**0.98**	**0.53**

Generalizing to New Shapes. Given a trained auto-decoder, we evaluate its ability to model previously unseen test instances by optimizing a randomly initialized latent code to fit partial observations in the form of a sparse cloud of surface points. To this end, given an initial latent code, we sample a set of query points and find the nearest points among the point cloud to compute the gradient distance vectors. These gradient distance vectors serve as pseudo ground-truth for optimizing the latent code. Similarly, we compute the pseudo unsigned distances for optimizing NDF and pseudo closest surface points for CSP. We do not include 3PSDF in this experiment because it requires knowledge of the surface gradients to assign pseudo labels to each query point, which is not trivial to obtain from the point cloud. Note that in the original paper, 3PSDF relies on a separate 3D point encoder for this task. However, the code for this module is not publicly available. We report our comparative results in Table 3. Again, our approach delivers significantly better accuracy. We provide qualitative results in Fig. 5. Unlike those of the other two methods, our results are free of holes in irregular parts of the surfaces.

5. Analysis and Ablations

5.1. Convergence Rate

Using the same metrics as above, we now look at the convergence rates while training the same network architecture but using different surface representations: UDF, GDF and CSP. As can be seen in Fig. 6, GDF not only yields better final results but also converges much faster. The corresponding meshes are shown Fig. 7. After only 3000 iterations, the output from our GDF is already smooth and almost without holes, which is not true for either CSP or NDF.

5.2. Additional Loss Terms

Unlike for SDFs, meshing a UDF often requires reasoning about its gradient [7, 21]. Our proposed GDF can be easily decomposed into the unsigned distance and its gradient via Eqs. 2 and 3. This gives us direct access to the distance and its gradient for meshing. We show that we can further supervise the training for each term separately. This additional

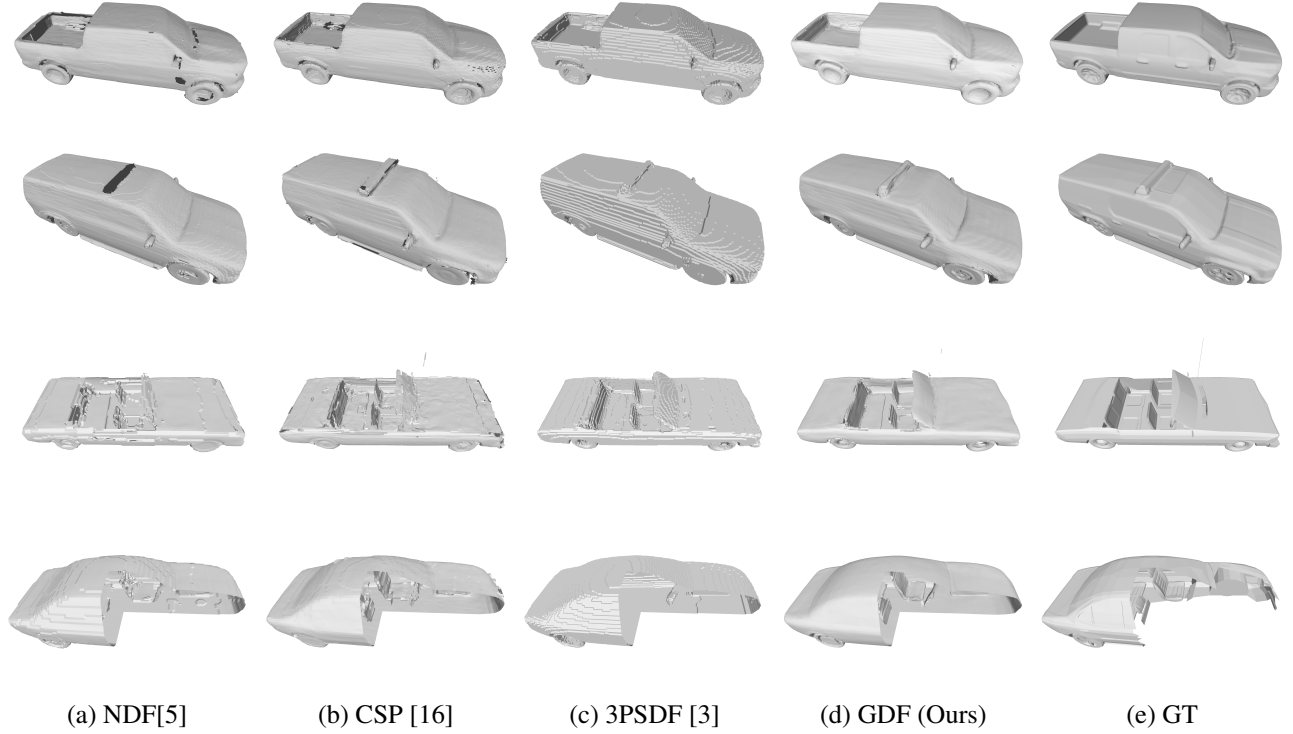|				
(a) NDF[5]	(b) CSP [16]	(c) 3PSDF [3]	(d) GDF (Ours)	(e) GT

Figure 4. **Representing known shapes using an auto-decoder.** We use the same MLPs trained using different implicit representations. **Top rows.** Car from ShapeNet-Car(P). **Bottom rows.** Car from the ShapeNet-Car(R), which contains the inside of the car. We remove a quarter of the reconstructed models in the last row to make it visible.

supervision helps preserve the underlying structural integrity of the gradient field and results in more robust estimation of the UDF and its gradient. In this scenario, we minimize the composite loss function:

$$\mathcal{L}(\Phi, \mathbf{C}_1, \ldots, \mathbf{C}_N) = \sum_{i,j} \Big[\lambda_{adf} |\hat{\mathbf{v}}_j^i - \mathbf{v}_j^i| \quad (7)$$

$$+ \lambda_{grad} \left| \frac{\hat{\mathbf{v}}_j^i}{\|\hat{\mathbf{v}}_j^i\|} - \mathbf{g}_j^i \right|$$

$$+ \lambda_{udf} \left| \|\hat{\mathbf{v}}_j^i\| - \mathbf{u} \right| \Big] ,$$

where $\lambda_{adf}, \lambda_{grad}, \lambda_{udf}$ are scalar weights chosen to balance the loss terms.

To demonstrate the effectiveness of this, we trained a network to reconstruct garments from the MGN dataset with and without the separate loss terms. The values of $\lambda_{adf}, \lambda_{grad}, \lambda_{udf}$ are empirically set to $(100, 4, 50)$. We then measured the distances between the reconstructed meshes and the ground-truth meshes for both unsigned distances and gradients. To this end, we sampled evenly points in the volume to obtain a 3D tensor of 512^3 query points and then select from these only the points with close proximity (<1 pixel) to the ground-truth surface. We measured the L1 differences between the unsigned distances and gradients at

Table 4. **Influence of the Additional Loss Terms.** The first two columns indicate which components of the loss function of Eq. 7 when training the network the reconstruct MGN garments. The L2 Chamfer Distances are multiplied by 10^{-5}. The first rows is the same as the results in Table 1.

Dist.	Grad.	CD ($\downarrow$)	NC($\uparrow$)	$Dist.E.$ ($\downarrow$)	$Grad.\ E.$($\downarrow$)
-	-	2.13	99.16	5×10^{-4}	0.0438
✓	-	2.09	99.53	3×10^{-4}	0.0367
-	✓	2.09	99.67	5×10^{-4}	0.0277
✓	✓	2.07	99.67	3×10^{-4}	0.0262

those points between the ground-truth and the learned GDFs. We report the results in Table 4. The additional loss terms result in more accurate results. Nevertheless, even without them, GDF still outperforms the other methods.

In general, GDF allows the application of direct supervision or, potentially, regularization terms on both the unsigned distance and its gradient. Thus, previous methods [22, 23] designed for UDFs can be directly applicable to GDFs.

5.3. Meshing at High Resolution

In Fig. 8, we show the reconstruction results of a shape at 512^3 and 768^3 resolution. Higher resolution mesh re-

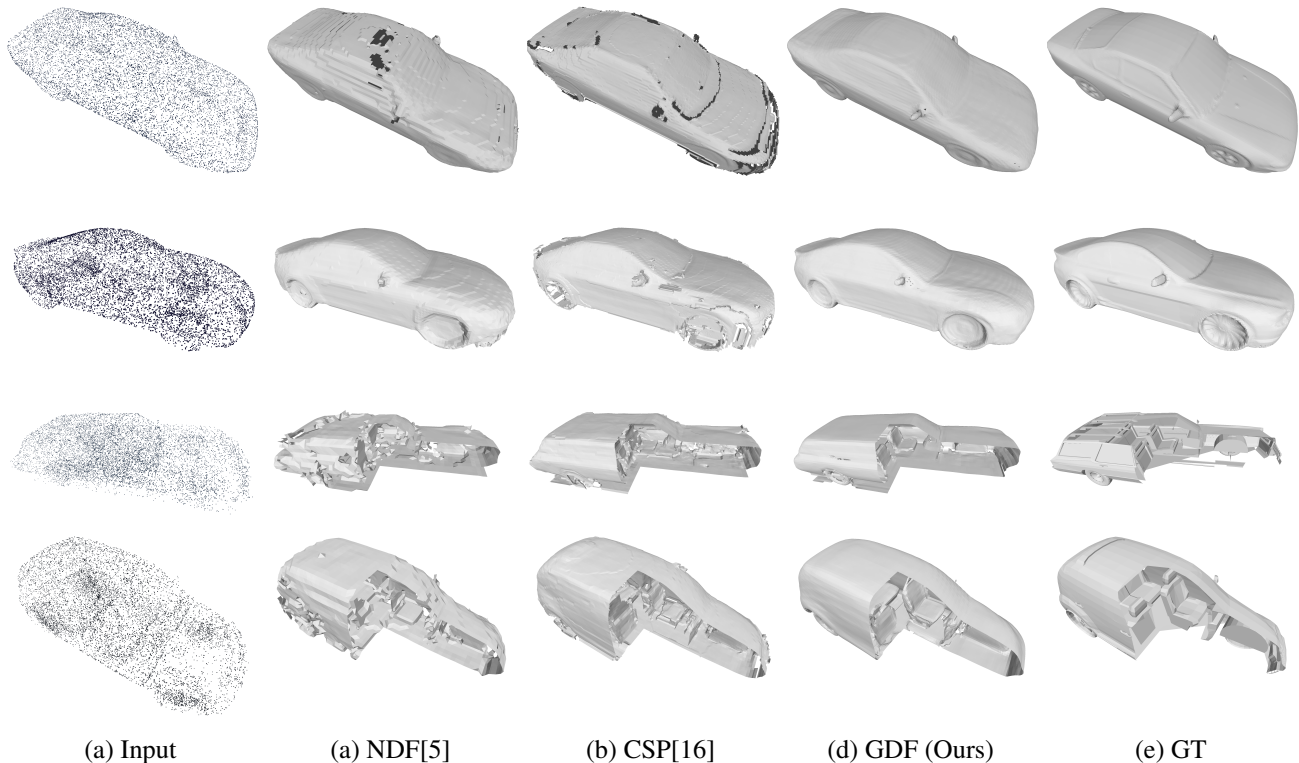| (a) Input | (a) NDF[5] | (b) CSP[16] | (d) GDF (Ours) | (e) GT |

Figure 5. **Representing previously unseen shapes using an auto-decoder.** Meshes reconstructed by fitting a latent variable model to 10000 surface points. **Top rows.** Car from ShapeNet-Car(P). **Bottom rows**. Cars from the ShapeNet-Car(R), which contains the inside of the car. We remove a quarter of the reconstructed models to make it visible.
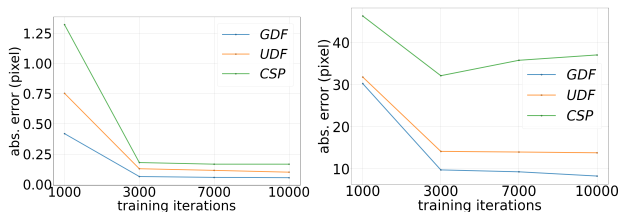

Figure 6. **Distance and gradient reconstruction errors during fitting.** We train identical networks to fit a single mesh using different implicit representations including GDF (Ours), CSP[16] and NDF[5]. At different iterations, we measure the L1 distance between the estimated unsigned distances and gradients via the deep networks and the ones computed from the ground-truth mesh. Our GDF can minimize the errors much faster and more accurately.
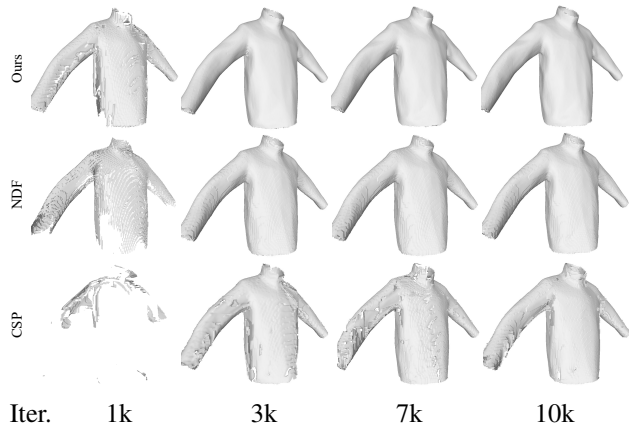

Figure 7. **The resulting surfaces at the fitting progresses.** We compare the output surfaces at different fitting iterations when using a single network to represent a single mesh via different representations including GDF (Ours), CSP[16] and NDF[5]. After only 3000 iterations, our proposed GDF is able to output a smooth surface with almost no unwanted holes.

construction requires more precise distance and gradient estimation to prevent holes on surfaces. Training a network to output the GDF representation results in smooth surface without holes. It is not the case when regressing unsigned distance values, as in NDF[5]. Here we train the two networks to represent a single shape for 50000 iterations with the batch size of 32000 query points.
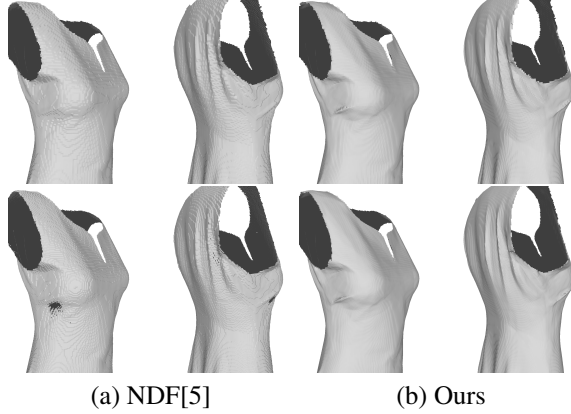
(a) NDF[5]   (b) Ours

Figure 8. **Reconstruction results of a shape at** 512^3 **(top) and** 768^3 **(bottom) resolution.** Higher resolution mesh reconstruction requires more precise distance and gradient estimation to prevent the appearance of holes. GDF representation results in smooth surface without holes when meshing at 768^3 resolution.

5.4. Gradient Distance Field Visualization

We visualize the values of the gradient distance field extracted directly from the ground-truth data and those estimated via a deep network in Fig. 9 for a 2D version of GDF. In this experiment, we train a network to output a GDF representation of an open contour for a portion of the Stanford Bunny (as shown in Fig.1a). The ground-truth GDF value at each query point is computed by finding the nearest point on the contour. We only show the values of the x-dimension. The top row visualizes the vectors while the bottom row visualizes the normalized gradient components extracted from them (via Eq.3). As can be seen, the contour is placed at smooth, continuous, and differentiable areas of the GDF scalar field. For a pair of example points, A and B, on the two sides of the contour, depicted as the end-point of the arrows in the first column, they have opposite gradient values: $\mathbf{g}_{A_x} = -1$ and $\mathbf{g}_{B_x} = 1$. The values of A and B in the first row have opposite signs: $\mathbf{v}_{A_x} < 0$ while $\mathbf{v}_{B_x} > 0$ and the contour in between is at exact zero-level set. In the second column, the deep network can effectively capture this continuity accurately.

Further, it can be seen that GDF is inherently not differentiable everywhere. Discontinuities occur at locations where there are multiple closest surface points, *i.e.*, medial axis of the shape. However, there points are often far away from the surface and do not affect the reconstruction accuracy. In rare instances where it happens near the surface, it typically affects a small number of points, such as the point C illustrated in the bottom-left image. In practice, we did not observe substantial issues arising from the discontinuity caused by the medial axis. In fact, it is a common practice to only use near surface points for training the network. Thus, the majority of the medial axis would not be used for training.

(a) GT distance   (b) Learned distance
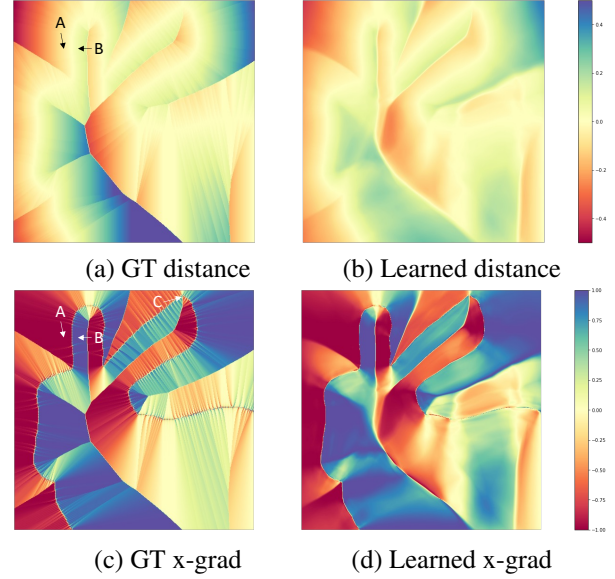
(c) GT x-grad   (d) Learned x-grad

Figure 9. **Ground-Truth vs Learned Gradient Distance Field.** We visualize GDF and the gradient component extracted from it for an open 2D contour for a portion of the Stanford Bunny (see Fig.1). The first two columns show the ground-truth and learned distance values. The last two columns show the corresponding gradient component for the x-dimension.

In this particular case shown in the figure, it can be seen that the network mainly models the near surface values. Note that even though the ground-truth gradients can be noisy at some surface points, the outputs of the deep network are not.

6. Conclusion

We have shown that the Gradient Distance Function is an effective representation for 3D open surfaces. Each point is associated to a vector pointing to the closest point on the object surface. The norm of this vector is the unsigned distance while the orientation is the negative gradient of the unsigned distance. This representation is differentiable at the surface and can easily be learned by a deep network. We have demonstrated that, compared to UDF and other open surface representations, GDF yields more accurate distances and gradients, resulting in better object reconstruction. Extensive evaluations on three datasets demonstrates the expressive capacity and generalizability of GDF in capturing complex surface geometries. There are several directions for future work. Checking the zero-crossing points across different dimensions of the vector fields can avoid the misalignment between them. Further, a sampling scheme prioritizing potentially errornous areas such as near-surface medial axis would be beneficial when modelling complex surfaces.

References

[1] Bhatnagar, B.L., Tiwari, G., Theobalt, C., Pons-Moll, G.: Multi-Garment Net: Learning to Dress 3D People from Images. In: International Conference on Computer Vision (2019) 3, 4

[2] Chang, A., Funkhouser, T., G., L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., Xiao, J., Yi, L., Yu, F.: Shapenet: An Information-Rich 3D Model Repository. In: arXiv Preprint (2015) 3

[3] Chen, W., Lin, C., Li, W., Yang, B.: 3psdf: Three-pole signed distance function for learning surfaces with arbitrary topologies. Conference on Computer Vision and Pattern Recognition (2022) 2, 3, 4, 5, 6

[4] Chen, Z., Zhang, H.: Learning Implicit Fields for Generative Shape Modeling. In: Conference on Computer Vision and Pattern Recognition (2019) 1, 2

[5] Chibane, J., Mir, A., Pons-Moll, G.: Neural Unsigned Distance Fields for Implicit Function Learning. In: Advances in Neural Information Processing Systems (2020) 1, 2, 3, 4, 5, 6, 7, 8

[6] Corona, E., Pumarola, A., Alenya, G., Pons-Moll, G., Moreno-Noguer, F.: Smplicit: Topology-Aware Generative Model for Clothed People. In: Conference on Computer Vision and Pattern Recognition (2021) 2

[7] Guillard, B., Stella, F., Fua, P.: MeshUDF: Fast and Differentiable Meshing of Unsigned Distance Field Networks. In: European Conference on Computer Vision (2022) 1, 2, 3, 5

[8] Kingma, D.P., Ba, J.: Adam: A Method for Stochastic Optimization. In: International Conference on Learning Representations (2015) 3

[9] Liu, Y.T., Wang, L., Yang, J., Chen, W., Meng, X., Yang, B., Gao, L.: Neudf: Leaning neural unsigned distance fields with volume rendering. In: Conference on Computer Vision and Pattern Recognition (2023) 2

[10] Long, X., Lin, C., Liu, L., Liu, Y., Wang, P., Theobalt, C., Komura, T., Wang, W.: NeuralUDF: Learning Unsigned Distance Fields for Multi-view Reconstruction of Surfaces with Arbitrary Topologies. In: Conference on Computer Vision and Pattern Recognition (2022) 1

[11] Lorensen, W., Cline, H.: Marching Cubes: A High Resolution 3D Surface Construction Algorithm. In: ACM SIGGRAPH. pp. 163–169 (1987) 2

[12] Meng, X., Chen, W., Yang, B.: Neat: Learning neural implicit surfaces with arbitrary topologies from multi-view images. Conference on Computer Vision and Pattern Recognition (2023) 2

[13] Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S., Geiger, A.: Occupancy Networks: Learning 3D Reconstruction in Function Space. In: Conference on Computer Vision and Pattern Recognition. pp. 4460–4470 (2019) 2

[14] Park, J.J., Florence, P., Straub, J., Newcombe, R.A., Lovegrove, S.: Deepsdf: Learning Continuous Signed Distance Functions for Shape Representation. In: Conference on Computer Vision and Pattern Recognition (2019) 1, 2, 3, 5

[15] Park, K., Patten, T., Vincze, M.: Pix2pose: Pixel-Wise Coordinate Regression of Objects for 6D Pose Estimation. In: International Conference on Computer Vision (2019) 3

[16] Venkatesh, R., Karmali, T., Sharma, S., Ghosh, A., Babu, R.V., Jeni, L.A., Singh, M.: Deep Implicit Surface Point Prediction Networks. In: International Conference on Computer Vision (2021) 2, 4, 5, 6, 7

[17] Venkatesh, R., Sharma, S., Ghosh, A., Jeni, L.A., Singh, M.K.: Dude: Deep unsigned distance embeddings for hi-fidelity representation of complex 3d surfaces. arXiv Preprint (2020) 2

[18] Wang, L., Yang, J., Chen, W., Meng, X., Yang, B., Li, J., Gao, L.: Hsdf: Hybrid sign and distance field for modeling surfaces with arbitrary topologies. In: Advances in Neural Information Processing Systems (2022) 2, 3

[19] Xu, Q., Wang, W., Ceylan, D., Mech, R., Neumann, U.: DISN: Deep Implicit Surface Network for High-Quality Single-View 3D Reconstruction. In: Advances in Neural Information Processing Systems (2019) 3

[20] Yang, X., Lin, G., Chen, Z., Zhou, L.: Neural vector fields: Implicit representation by explicit learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 16727–16738 (June 2023) 2

[21] Zhang, C., Lin, G., Yang, L., Li, X., Komura, T., Schaefer, S., Keyser, J., Wang, W.: Surface extraction from neural unsigned distance fields. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 0000–0000 (October 2023) 2, 5

[22] Zhao, F., Wang, W., Liao, S., Shao, L.: Learning Anchored Unsigned Distance Functions with Gradient Direction Alignment for Single-View Garment Reconstruction. In: International Conference on Computer Vision (2021) 2, 6

[23] Zhou, J., Ma, B., Liu, Y.S., Fang, Y., Han, Z.: Learning consistency-aware unsigned distance functions progressively from raw point clouds. In: Advances in Neural Information Processing Systems (2022) 2, 6

[24] Zhou, Q.Y., Koltun, V.: Dense scene reconstruction with points of interest. ACM Transactions on Graphics (TOG) (2013) 3, 4