DOI: 10.1111/cgf.14916

Eurographics Symposium on Geometry Processing 2023
P. Memari and J. Solomon
(Guest Editors)

COMPUTER GRAPHICS *forum*
Volume 42 (2023), Number 5
e14916 ***pp. 1 - 13***

Neural Representation of Open Surfaces

T. V. Christiansen[1], J. A. Bærentzen[1], R. R. Paulsen[1] and M. R. Hannemose[1]

[1]Technical University of Denmark, Kgs. Lyngby, Denmark

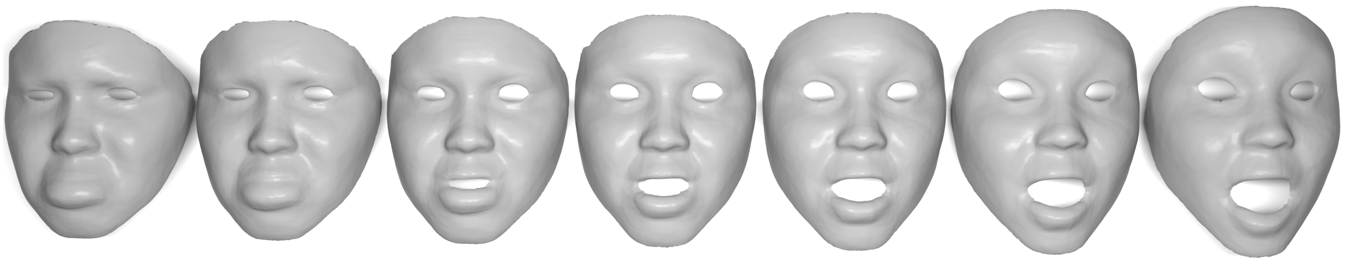

Figure 1: *Example of interpolating between two open surfaces with different number of boundary curves. The interpolation is linear in the latent space of the neural network and goes from a sad facial expression to a surprised facial expression from the* ***BU-3DFE*** *dataset [YWS*06]. The surfaces are generated using the SSDF based network.*

Abstract

Neural implicit surfaces have emerged as an effective, learnable representation for shapes of arbitrary topology. However, representing open surfaces remains a challenge. Different methods, such as unsigned distance fields (UDF), have been proposed to tackle this issue, but a general solution remains elusive. The generalized winding number (GWN), which is often used to distinguish interior points from exterior points of 3D shapes, is arguably the most promising approach. The GWN changes smoothly in regions where there is a hole in the surface, but it is discontinuous at points on the surface. Effectively, this means that it can be used in lieu of an implicit surface representation while providing information about holes, but, unfortunately, it does not provide information about the distance to the surface necessary for e.g. ray tracing, and special care must be taken when implementing surface reconstruction. Therefore, we introduce the semi-signed distance field (SSDF) representation which comprises both the GWN and the surface distance. We compare the GWN and SSDF representations for the applications of surface reconstruction, interpolation, reconstruction from partial data, and latent vector analysis using two very different data sets. We find that both the GWN and SSDF are well suited for neural representation of open surfaces.

CCS Concepts

• ***Neural Implicit Surface Representations of open surfaces*** $\rightarrow$ *Generalized Winding Number Field, Semi-Signed Distance Fields;*

1. Introduction

Neural implicit surfaces have emerged as an effective, learnable representation for shapes of arbitrary topology. However, open surfaces remain a challenge. This is hardly surprising since an implicit surface is the level set, $S = \{\mathbf{x} | f(\mathbf{x}) = \tau\}$, for a level, τ, and function, f, whose gradient, ∇f, must be non-zero at all points of S. Consequently, we can define an interior where $f > \tau$ and an exterior where $f < \tau$ (or vice versa). Any curve from the (thus defined) interior to the exterior must pass S where $f = \tau$. In other words, implicit surfaces are closed by definition, and we need to either modify or add an element to the definition of implicit surfaces to achieve open surfaces.

Perhaps, the most obvious modification is to define the surfaces in terms of extrema, e.g. using unsigned distance fields (UDF), rather than level sets. Unfortunately, this makes surface reconstruction significantly harder. Another approach would be to introduce explicit boundary curves as Palmer *et al.* [PSW*22]. However, this precludes having a varying number of boundary curves in the same model.

A completely different solution is to represent an original surface in terms of its generalized winding number (GWN) which is 1 inside the (solid) object and 0 outside. Thus, the surface can be defined as the set of points where the GWN changes discontinuously. In cases where the original surface has holes, the GWN changes smoothly along curves that pass through these holes, and we can obtain a closed surface as the 0.5-level set of the GWN. Ironically, this means that the GWN is an implicit representation of the holes in open surfaces. Conversely, wherever the surface is defined, the GWN changes discontinuously, but if the level set is extracted robustly, e.g. using bisection, we would also obtain the original surface as part of the level set. At surface points, the GWN gradient is, strictly speaking, not defined, but numerically it simply becomes very large. Based on these considerations, we can - effectively - treat the GWN as an implicit surface representation, and holes are simply regions where the gradient is comparatively small.

In this paper, we examine the use of the GWN as a neural shape representation for open surfaces and compare it to a novel representation, which we call semi-signed distance fields (SSDF). Semi-signed distance fields are a dual representation where both a signed and an unsigned distance field are learned. The signed distance field is obtained as the product of the GWN shifted by $\frac{1}{2}$ and the UDF. The GWN can later be restored as their ratio since it was used to sign the distance field in the first place. The benefit of SSDFs are twofold: we no longer have to learn the discontinuous GWN, and the learned representation is an implicit surface (with bounded gradients), which simplifies polygonization and enables faster rendering through ray casting.

Contrary to our initial expectations, we find that the GWN is a precise representation for open surfaces and as easy for the network to learn as the semi-signed distance field. However, much greater attention must be paid to surface reconstruction for the GWN representation. Moreover, the GWN has no information about the proximity of the surface, which would impact the efficiency of ray-tracing such surfaces. We also find that both the GWN and SSDF based representations are very similar for reconstruction from partial data. Both methods are able to interpolate between shapes with different numbers of holes, an example using SSDF is in Figure 1.

2. Contributions

We show that a GWN based neural implicit surface representation is able to accurately represent families of shapes with a varying number of holes, interpolate smoothly between shapes, do shape completion, and create a latent space that clusters similar shapes. All of this works even though GWN is technically a volumetric representation and not an implicit surface representation due to the unbounded gradients at the surface. Our proposed *semi-signed distance field* (SSDF) for neural implicit surface representation performs similarly if not better than the GWN based method.

In summary, our main contributions are:

- We introduce the semi-signed distance field (SSDF) for neural surface representation of shapes with holes.
- We perform the first quantitative evaluation of the GWN as a neural representation of open surfaces, and compare it to an SSDF based representations.
- We introduce a method for recovering an open surface from a neural GWN or SSDF representation, which includes automatically determining a threshold for which parts are holes.
- Finally, we introduce a method for quantitatively evaluating an interpolation between a pair of shapes.

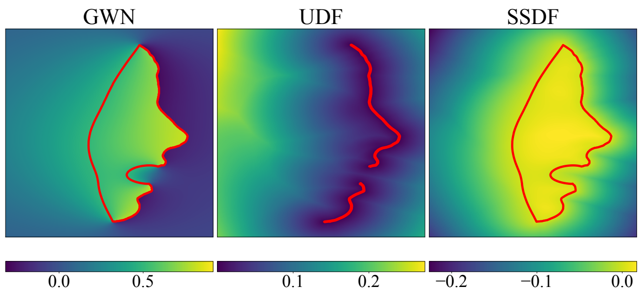

Figure 2: *A cross sectional view of a 3D scan of a human face from the* ***BU-3DFE*** *dataset [YWS*06]. The red lines are isocontours of* $\frac{1}{2}$, 0.001, 0 *for the GWN, UDF and SSDF respectively. The mouth and eyes are holes and the back of the head is also a hole.*

3. Generalized Winding Number

The term *winding number* refers to the number of times that a closed curve winds around a point in the plane. This is clearly an integer, and for a simple closed curve the value is 1 if the point is in the region bounded by the curve and 0 otherwise.

For a compact surface in 3D that does not intersect itself, we can likewise assign a winding number of 0 to any point outside the surface and 1 to any point inside. More generally, Jacobson *et al.* proposed the GWN as a method for robustly determining whether a point in 3D is inside or outside a shape [JKS13]. The GWN for a surface, S, at a given point, $\mathbf{x}$, is the integral over the surface of the signed area projected onto the unit sphere,

$$w(\mathbf{x}) = \frac{1}{4\pi}\int_S \frac{\mathbf{n}(\mathbf{s})\cdot(\mathbf{s}-\mathbf{x})}{\|\mathbf{s}-\mathbf{x}\|^3}\mathrm{d}\mathbf{s}, \tag{1}$$

where $\mathbf{n}(\mathbf{s})$ is the outward facing normal of the surface at $\mathbf{s}$ [BDS*18]. Thus, the sign is positive for surface elements whose normal points away from $\mathbf{x}$ and negative where the normal points toward $\mathbf{x}$.

The robustness of this formulation is due to the fact that minor holes in the surface or self-intersections only have a limited influence on the value of w.

Another important property is the fact that w changes discontinuously on the surface since the sign of the numerator in Equation (1) changes as $\mathbf{x}$ passes through the surface. However, if we cut a hole around the point where $\mathbf{x}$ impinges on the surface, the change becomes smooth as shown in Figure 2.

4. Related work

The interest in neural implicit surface representations was in large part sparked by the work of Park *et al.* [PFS*19] which demonstrated that it is possible to encode geometric objects using an auto-decoder neural network trained on a class of shapes and an associated latent vector encoding for each specific shape. Besides being

Table 1: *An overview of how DeepSDF [PFS*19], NDF [CP*20], and UDF [JMdB*21] compare to GWN and our proposed SSDF.*

	DeepSDF	NDF	UDF	GWN	SSDF
Latent space	✓	✗	✓	✓	✓
Open surfaces	✗	✓	✓	✓	✓
Surface recovery	easy	hard	hard	easy	easy
Distance information	✓	✓	✓	✗	✓
In-/outside information	✓	✗	✗	✓	✓

able to neurally represent shapes, it is also possible to cluster and classify shapes according to their latent vectors as shown by e.g. Juhl *et al.* [JMdB*21]

As mentioned, implicit surface methods, such as e.g. signed distance fields [JBS06], generally represent the surface as a level set of a scalar field, $f : \mathbb{R}^3 \to \mathbb{R}$, and this limits the methods' applicability to watertight surfaces.

Recent works [CP*20; JMdB*21; AL20; GSF22; VKS*21; RCKV22; PSW*22] alleviate this problem by learning functions that can represent open surfaces. One example is the Unsigned Distance Function (UDF) [CP*20; JMdB*21; AL20]. Unfortunately, surface points are minima of the UDF, and this makes it considerably harder to reconstruct the surface from a UDF than from an SDF. Chibane *et al.* [CP*20] therefore use the gradients of the UDF but this is problematic, as the gradient of the UDF is discontinuous at the surface and very sensitive to noise and sampling artifacts. However, recently, convincing results using a modified marching cubes algorithm have been shown by Guillard *et al.* [GSF22].

Venkatesh *et al.* [VKS*21] adopt an approach in which they return the closest point on the surface for a given query point, whereas Rella *et al.* [RCKV22] find a vector field and extract the surface as the set of points where the vector field is zero. Palmer *et al.* [PSW*22] employ a method rooted in geometric measure theory. Essentially, their approach is also an implicit surface representation, but the boundary curves are explicitly represented. The SSDF is different in this respect since our boundary is also implicit. Importantly, this makes it straight forward to learn a class of shapes in which instances might have different numbers of boundary curves. This does not appear to be possible using the scheme due to Palmer *et al.*

Other works on volume rendering also address the problem of object representation. Mildenhall *et al.* [MST*20] use a light field representation and can accurately represent scenes, but this comes at the expense of poorer geometry representation. Yariv *et al.* [YGKL21] and Azinovic *et al.* [AMG*22] build on this by learning both a radiance field and a SDF. However, neither is able to represent open surfaces. Another approach involving parameterization of a surface using a tetrahedral grid is presented by Gao *et al.*[GWM*22]. They encode shapes within a tetrahedral grid and then use a discriminator to essentially learn an inside/outside classification of the tetrahedrons in order to represent shapes. They are also not capable of representing open surfaces.

We compute the sign of our semi-signed distance fields using the generalized winding number method proposed by Jacobson *et al.*

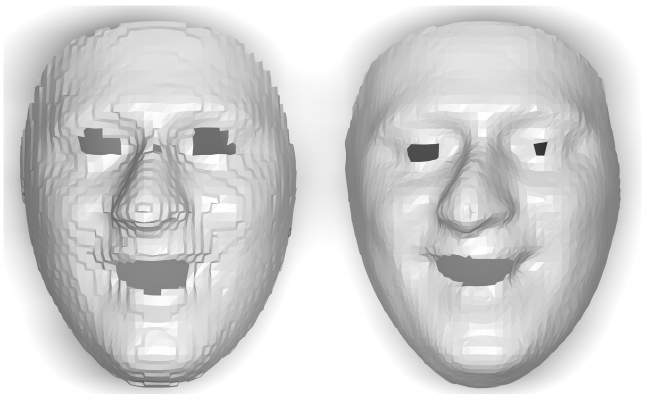

Figure 3: *A comparison of how a face from the* ***BU-3DFE*** *dataset [YWS*06] is reconstructed with the method by Chen et al. [CTFZ22]. Left: reconstructed from learned UDF, right: reconstructed from the ground truth UDF.*

[JKS13]. Specifically, we employ the fast winding number method [BDS*18]. Prior to this work, the winding number has been learned directly by Chi and Song [CS21] who used it in a system for garment reconstruction. However, their work targeted pose estimation in which the surface reconstruction was used as a method of estimating the pose. In this work, we specifically investigate surface reconstruction and the shape representation abilities. Unfortunately, the discontinuous nature of the GWN provides further challenges when doing surface reconstruction and does not contain global shape information. For this reason, we propose to learn the GWN indirectly as the ratio of the semi-signed distance and the unsigned distance. In Table 1 we compare our proposed SSDF method to the most similar competing methods.

5. Unsigned distance fields

As mentioned, many recent works use the unsigned distance field for neural representation of open surfaces [CP*20; JMdB*21; AL20; GSF22]. The unsigned distance field is easy to compute for open surfaces and seems like an obvious choice to represent them. However, in the process of *learning* the unsigned distance with a neural network, small errors are introduced, mostly smoothing, which make them crucially different from the ground truth unsigned distance field. Reconstructing the surface from a learned UDF is therefore a non-trivial task. This is very apparent when comparing the reconstruction obtained with a *learned* UDF (a network with the architecture presented in Section 6.1) to the reconstruction from the *ground truth* UDF, as we do in Figure 3. Here, we use the recent method by Chen *et al.* [CTFZ22] for the surface reconstruction which was trained on reconstructing ground truth UDFs. We also experimented with MeshUDF [GSF22], but it performed worse than the method by Chen *et al.* on our learned UDF.

One could potentially fine-tune the method by Chen *et al.* on our specific type of learned UDF, but this is more complicated than our presented approaches. Additionally, the UDF has the inherent weakness that it has no notion of which parts are inside and outside, making interpolation between shapes that are far apart very difficult. This is because the model has to smoothly move the iso-contour of zero between the locations of the shapes and if it moves slightly above or below zero, the surface will disappear or be doubled, respectively. Based on these considerations we focus on GWN and SSDF based representations in this paper.

6. Method

In this section we present the implementation details. We compare two different types of networks based on the GWN and SSDF. While all these are applicable to any task in surface representation with neural networks, in this paper we focus on the auto-decoder style networks introduced by Park *et al.* [PFS*19] that are able to represent a multitude of shapes with a single network, by having a unique latent vector for each shape.

For simplicity in the sequel, we shift the GWN such that 0 is contained in the discontinuous jump.

Consequently, the shifted GWN is defined as:

$$w^s(\mathbf{x}) = w(\mathbf{x}) - \frac{1}{2} \tag{2}$$

Our proposed SSDF representation requires predicting two continuous functions; the semi-signed distance, and the unsigned distance. Given the unsigned distance $d^u(\mathbf{x})$ the semi-signed distance is then given as:

$$d^{ss}(\mathbf{x}) := w^s(\mathbf{x}) d^u(\mathbf{x}) \tag{3}$$

The goal of the neural network f_θ is to, for every shape i and latent vector $\mathbf{h}_i$ associated with shape i, be a good function approximator of the signal in consideration for all points $\mathbf{x}$ in the target domain Ω_i. For the network learning the GWN, the network should predict the shifted generalized winding number for shape i, $w_i^s(\mathbf{x})$,

$$f_\theta^{w_s}(\mathbf{h}_i, \mathbf{x}) \approx w_i^s(\mathbf{x}), \forall\, \mathbf{x} \in \Omega_i \tag{4}$$

As opposed to the previous network configurations, the SSDF based network should predict both the semi-signed distance d_i^{ss} and the unsigned distance d_i^u, for all points $\mathbf{x}$ in the target domain Ω_i, i.e.,

$$f_\theta^u(\mathbf{h}_i, \mathbf{x}), f_\theta^{ss}(\mathbf{h}_i, \mathbf{x}) \approx \{d_i^u(\mathbf{x}), d_i^{ss}(\mathbf{x})\}, \forall\, \mathbf{x} \in \Omega_i \tag{5}$$

We compute d^u using PyGEL3D and $w(\cdot)$ using libigl [JP*18; BDS*18]

6.1. Network architecture and latent vectors

We use the same network architecture as Park *et al.* [PFS*19], i.e. eight fully connected 512-dimensional layers and a 256-dimensional latent space. To train the network we use the L1 distance, which for the SSDF based network is:

$$\mathcal{L}(\theta, \mathbf{h}_i, \mathbf{x}) = \left|d_i^u(\mathbf{x}) - f_{\mathbf{h}_i}^u(\mathbf{x})\right| + \left|d_i^{ss}(\mathbf{x}) - f_{\mathbf{h}_i}^{ss}(\mathbf{x})\right| \tag{6}$$

The losses are similar for the GWN network.

To improve the interpolations obtained with our network we also use the Lipschitz loss introduced by Liu *et al.* [LWJ*22]:

$$\underset{\theta, \{\mathbf{h}_i\}_{i=1}^N}{\operatorname{argmin}} \sum_{i=1}^{N} \sum_{\mathbf{x} \in \Omega_i} \left(\mathcal{L}(\theta, \mathbf{h}_i, \mathbf{x}) + \alpha \cdot \Pi_j^l \operatorname{softplus}(c_j)\right). \tag{7}$$

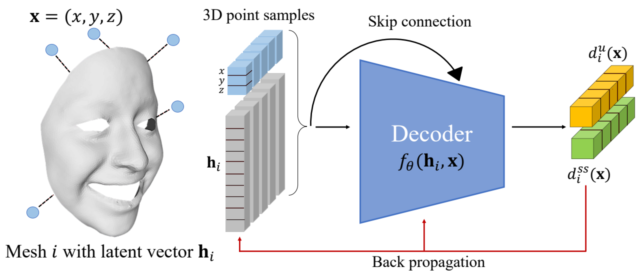

Figure 4: *Illustration of the training process for the network predicting the semi-signed distance field and the unsigned distance field. Adapted with permission from [JMdB*21].*

Here N is the number of shapes in the data set, $\alpha = 10^{-6}$ is a scaling factor which is used to "scale each row of W_j to have the absolute value row-sum less than or equal to softplus(c_j)" [LWJ*22], where W_j is the weights for layer j.

During inference we freeze the network parameters θ and obtain the latent vector for an unseen shape by solving:

$$\tilde{\mathbf{h}} = \underset{\mathbf{h}}{\operatorname{argmin}} \sum_{\mathbf{x} \in \Omega} \mathcal{L}(\theta, \mathbf{h}, \mathbf{x}) \tag{8}$$

The Adam optimizer [KB14] was used to optimize both the network parameters and the latent space vectors with learning rate 0.0001 and 0.001 respectively. The neural networks were implemented using PyTorch [PGM*19]. The network training is illustrated in Figure 4.

The latent space vectors are generated from a multivariate normal distribution with zero mean, variance 0.01 and zero covariance.

6.2. Point sampling

We uniformly sample points on the surface of the object and offset them along a normally distributed vector. Additionally, we uniformly sample points within the bounding box from $[-1, -1, -1]$ to $[1, 1, 1]$. This is how the target domain Ω_i is defined.

6.2.1. SSDF surface recovery

Our method for recovering the surface from our semi-signed distance is a two step process. First, we recover the closed surface from the 0-level set of the semi-signed distance field. Second, we remove the parts of this surface that do not correspond to the actual surface. This two-step approach enables us to use any method that can operate on a regular signed distance field to recover the closed surface such as dual-contouring or ray casting, followed by our hole detection step.

For the experiments in this paper we have used dual-contouring [BGAA12]. This is done by creating a voxel grid of the volume in the bounding box and evaluating $f_{\mathbf{h}}^{ss}$ in every voxel. Then, a closed mesh can be extracted as the 0-level. Afterwards, the mesh is smoothed using simple average smoothing and every vertex in the mesh is then projected down on the exact 0-level set by running eight iterations of

$$\mathbf{x} \leftarrow \mathbf{x} - f_{\mathbf{h}}^{ss}(\mathbf{x}) \frac{\nabla f_{\mathbf{h}}^{ss}(\mathbf{x})}{||\nabla f_{\mathbf{h}}^{ss}(\mathbf{x})||_2^2}, \tag{9}$$

where $\mathbf{x}$ is the position of a vertex in the mesh.

After the vertices have been projected to the 0-level isocontour, the quad faces are split into triangles. Then the mesh is optimized by edge flipping in order to minimize the dihedral angle and maximize the minimum angle. Lastly, the resolution of the reconstructed mesh is increased by applying root3 subdivision by Kobbelt [Kob00] to the reconstructed mesh, and then all vertices are projected to the 0-level set by running eight iterations of Equation (9) again. Finally we convert the closed surface to the open one using the method in Section 6.3.

6.2.2. GWN surface recovery

The method we use to recover a surface from the GWN is very similar to the method for the SSDF, but uses bisection instead. The same voxel grid is created and $f_{\mathbf{h}}^{w_s}$ is evaluated in every voxel. A closed surface is then recovered by using dual contouring on the voxel grid. The closed surface is cubic due to the nature of the GWN field. Therefore, the mesh is smoothed using simple average smoothing before the vertices are placed on the 0-level isocontour. As the GWN field is discontinuous, the vertices cannot be projected down on the 0-level isocontour using the gradients. Therefore, a bisection algorithm is used instead. For each vertex, two points are offset in the negative and positive direction of the vertex normal, until the two points are inside and outside the surface respectively, determined by the sign of $f_{\mathbf{h}}^{w_s}$. The mid-point between the two point offsets is found and depending on the sign of $f_{\mathbf{h}}^{w_s}$ of this mid-point, the midpoint replaces either the point inside or outside the shape. This process is repeated eight times and afterwards the vertex position is set to the point inside the shape. If the two offset points have the same sign, the vertex position is left unchanged. Similar to the SSDF reconstruction process, the quad faces are split into triangles and the mesh is optimized in the same way as the SSDF. Finally we convert the closed surface to the open one using the method in Section 6.3.

6.3. Detecting holes in the surface

When the closed surface has been obtained, the remaining step is to figure out which parts of it should be removed to obtain the final surface.

To detect the location of the holes in the closed surfaces, we use the GWN. For the SSDF network, the GWN is approximated from the output of the neural network as follows:

$$\widetilde{w}_{\mathbf{h}}^{s}(\mathbf{x}) = \frac{f_{\mathbf{h}}^{ss}(\mathbf{x})}{f_{\mathbf{h}}^{u}(\mathbf{x}) + \varepsilon}, \tag{10}$$

where $\varepsilon = 0.5 \cdot 10^{-5}$ to avoid numerical issues when $f_{\mathbf{h}}^{u}(\mathbf{x})$ is close to zero.

For the GWN network, $\widetilde{w}_{\mathbf{h}}^{s}(\mathbf{x})$ is simply the output of the network. From these, the spatial gradients $\nabla\widetilde{w}_{\mathbf{h}}^{s}(\mathbf{x})$ for each network can be computed using automatic differentiation.

As the generalized winding number is discontinuous at the surface, but smooth everywhere else, the approximated winding number will have a large gradient at the surface. Thus, we apply a threshold k to its gradient magnitude to obtain an indicator function

$$\left\|\nabla\widetilde{w}_{\mathbf{h}}^{s}(\mathbf{x})\right\| > k \tag{11}$$

This indicator function can be evaluated everywhere on the closed surface to determine which parts should be removed because they do not correspond to the real surface. For a mesh we evaluate the gradient magnitude at each vertex and perform a cut along the isocurve where the interpolated gradient magnitude is k. This is done as follows: If $\|\nabla\widetilde{w}_{\mathbf{h}}^{s}(\mathbf{x})\| \leq k$ at all vertices in a triangle face, the triangle face is removed. If one or two of the vertices in a triangle face are on the surface, linear interpolation is used to find the points on the edges at which the triangle face should be cut.

In the case of ray casting, if the indicator function indicates that a zero-crossing of the semi-signed distance field is a hole, the ray is re-cast from the surface intersection to see if there is another intersection along the ray.

6.3.1. Selecting k

Intuitively, varying k corresponds to making the holes in the mesh smaller or larger. Both holes that are too small and holes that are too large compared to ground truth are penalized by the chamfer distance, and we seek the value of k that minimizes this metric.

We reconstruct the closed learned mesh for each surface in a subset consisting of approximately 10% of the training samples (in order to reduce run-time) evenly selected from all classes. We then uniformly sample 30,000 points on the surface of both the original and the reconstructed meshes once. We calculate $\|\nabla\widetilde{w}_{\mathbf{h}}^{s}(\mathbf{x})\|$ at all points on the reconstructed meshes. An initial guess for k is then chosen as the average of the median of these values. For a given k we discard points where $\|\nabla\widetilde{w}_{\mathbf{h}}^{s}(\mathbf{x})\| \leq k$ and approximate the chamfer distance based on the remaining points. We minimize the approximated chamfer distance with basinhopping [WD97] followed by Nelder-Mead [GH12] to determine the optimal k.

7. Experiments

In this work we have used two different data sets. These data sets were chosen, as the data in the data sets differ in topology and have holes as opposed to other closed data sets such as ShapeNet [CFG*15], which has been widely used elsewhere [PFS*19; CP*20; VKS*21].

7.1. BU-3DFE

The Binghamton University 3D Facial Expression [YWS*06] dataset contains 2500 3D scans of facial expressions from 56 women and 44 men ranging in age from 18 years to 70 years. There are 25 scans of each person, covering six different expressions (happiness, disgust, fear, angry, surprise and sadness) with varying intensity and one neutral facial expression. Before using the data set, 31 scans were discarded due to scan and landmark annotation issues. The remaining meshes were rigidly aligned in the

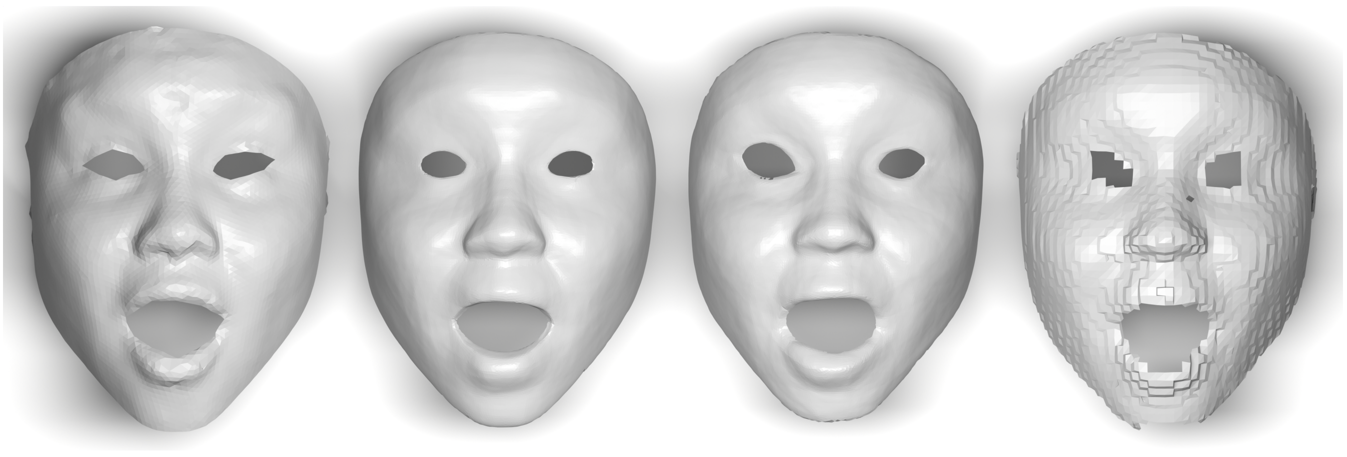

Figure 5: *Comparison of the three different representation methods for surface reconstruction of a partial facial expression from the* ***BU-3DFE*** *dataset [YWS*06]. From left to right: Ground truth, GWN reconstruction, SSDF reconstruction, and UDF reconstruction using the method by Chen et al. [CTFZ22].*

same way using the annotated landmarks. Afterwards, the meshes were converted into triangle meshes, centered around the origin and scaled using the same scaling factor such that they all would fit inside a sphere with diameter 0.99. This ensured that the relative scale between the meshes was preserved. Different issues with the meshes such as multiple vertices located at the same position, duplicate edges and disconnected components were also resolved. As a last processing step, the triangle faces within the left and right eye regions of the facial scans were removed, if the eyes were open. Moreover, the triangle faces in the mouth region were removed, if the facial scan belonged to one of the categories: happiness, disgust, fear, surprised. This processing step was carried out in order to introduce more holes into the surfaces. The dataset was split on a person-basis into a training set containing facial expressions from 45 women and 35 men (around 80% of the data) and a test set containing 11 women and 9 men (remaining 20% of the data set) such that the distribution of men and women in both the training set and test set was similar. The triangle faces within the eye regions and within the mouth region of the meshes in the training set were also removed.

7.2. MGN

The MGN (Multi-Garment Net) dataset [BTTP19] including the additional Part-2 data consists of 3D scans of garments and body poses. In our work, we only used the 3D scans of garments, which entails 328 3D scans of pants, shorts, T-shirts and coats. The meshes were already pre-aligned before use. They were also centered around the origin and scaled using the same scaling factor, such that they would all fit inside a sphere with diameter 0.99. This ensured that the relative scale between the meshes was preserved. Each category of garments was split into a training set and a test set such that the distribution of each type of garment in both the training set and test was similar. Consequently, the training set consists of 264 types of garments (around 80% of the data set) and the test consists of the remaining 64 types of garments.

8. Results

We compare two different neural networks based on the GWN and SSDF for both the **BU-3DFE** and **MGN** data set. The networks use the same architectures and only differ in what they are trained to predict. We have evaluated the networks on surface reconstruction accuracy, smoothness of interpolations from one shape to another, shape completion and how the latent vectors cluster according to labels. The evaluation of the networks are based on shapes, which have not been used to train the network parameters θ.

8.1. Surface reconstruction

To assess the quality of the surface reconstructions, we use the meshes in the test set. The reconstructed meshes are compared to the original meshes using the chamfer distance and mesh completion as metrics. These metrics were chosen, as they provide an overall assessment of how different a reconstructed mesh is to an original mesh. We left out mesh accuracy, as this metric is only applicable to closed surfaces. The chamfer distance and mesh completion are calculated the same way as in [PFS*19] with 30.000 points used for the chamfer distance and 1000 points for mesh completion. When we sample the points to calculate the metric, we weight the samples by the triangle face areas. The evaluation of the surface reconstructions for our method (SSDF) using the above mentioned metrics is in Table 2. The GWN and SSDF surface reconstructions are very similar in accuracy with the GWN being slightly better on the **BU-3DFE** dataset and SSDF for the **MGN** data set. Examples of different methods' surface reconstruction ability is seen in Figure 5.

8.2. Interpolation in latent space

Using the vectors in the latent space associated with the learned shapes, it is possible to interpolate between different shapes with varying number of boundary curves. In Figure 1, an interpolation between two different facial expressions (angry and surprised) from

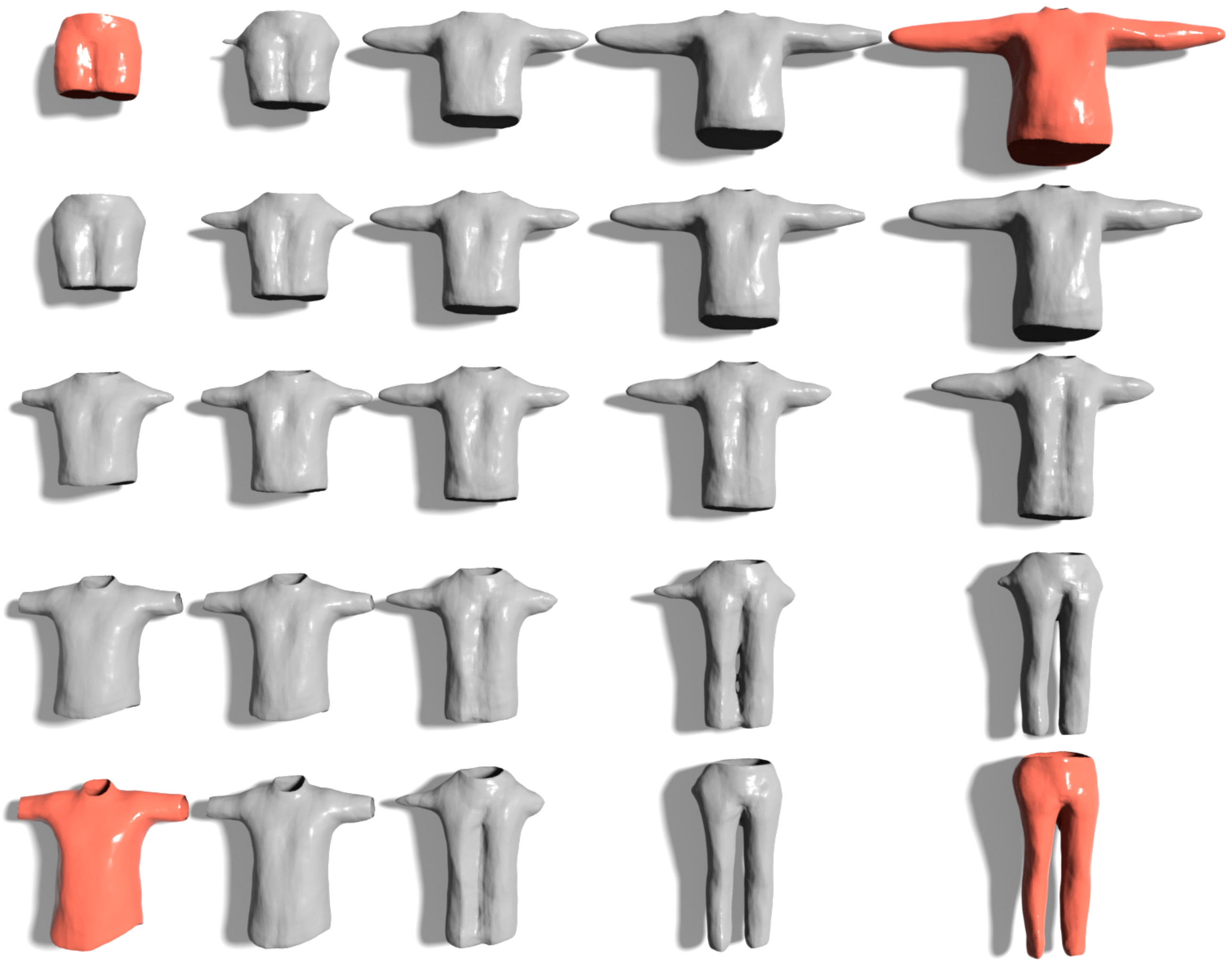

Figure 6: *Samples from a linear interpolation between latent vectors from the GWN based network for a pair of shorts, a T-shirt, a coat, and a pair of pants from the* ***MGN*** *dataset [BTTP19].*

Table 2: *Analysis of the quality of surface reconstructions test set meshes. ↓ indicates lower is better, ↑ indicates higher is better. Best results are marked in bold.*

	BU-3DFE		**MGN**	
	mean	median	mean	median
Chamfer distance, ↓ multiplied by 10^4				
GWN	**0.151**	**0.143**	0.354	0.201
SSDF	0.195	0.189	**0.271**	**0.199**
Mesh completion, ↑				
GWN	**0.996**	**0.997**	0.967	**0.980**
SSDF	0.987	0.989	**0.969**	0.979

a person in the **BU-3DFE** test set is seen. Figures 6 to 8 show interpolations between a pair of shorts, a T-shirt, a pair of pants, and a coat from the **MGN** test set for the GWN, SSDF and UDF based methods respectively. There is a smoother transition between the SSDF based interpolations. Further interpolations with more in-between steps are provided in the supplementary material as MP4-files.

Interpolations are usually only qualitatively evaluated as there is rarely ground truth for interpolation. We introduce a method to qualitatively evaluate interpolations of shapes. We do this by measuring how much a metric varies when interpolating between two shapes. We measure the following four different metrics: surface area, number of connected components, number of boundary curves and the length of all the boundary curves.

We employ an approximation of two times the Dirichlet energy to measure the variability. More specifically, we calculate it as the average of the squared difference of the specific metric between interpolation i and interpolation $i+1$ across 30 interpolations between two pairs of shapes. This is averaged over five pairs of ob-

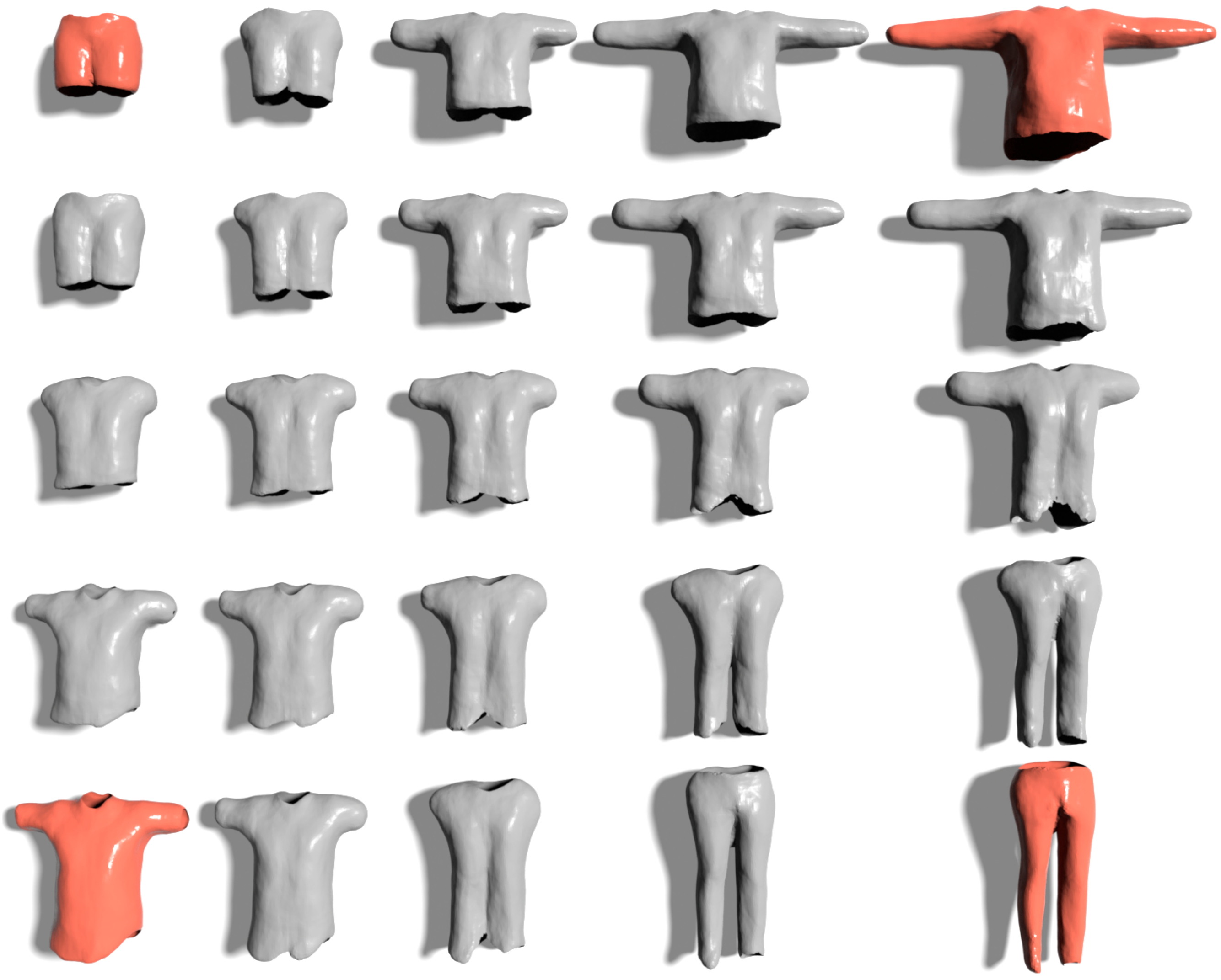

Figure 7: *Samples from a linear interpolation between latent vectors from the SSDF based network for the same pair of shorts, T-shirt, coat, and pair of pants as in Figure 6.*

jects with varying boundary curves. The results are in table Table 3, which shows how the SSDF based method is better at interpolation for both data sets across almost all metrics, as the metrics are lower, which indicates that the interpolations are smoother.

8.3. Shape Completion

We also test the ability of the networks to do shape completion. We generate data sets of partial shapes from the test sets of **BU-3DFE** and **MGN**. For each test mesh, we sample a random vertex on the mesh, then we obtain its normal vector and afterwards we remove those vertices in the mesh, where the dot product between the sampled normal vector and the normal vector of the other vertices is negative. By constructing the partial shapes in this way, we approximate the process of only scanning the objects from one direction. Moreover, by sampling the initial vertex at random, we avoid introducing bias in this process. However, this also means that some shapes will be reduced a lot with respect to the surface area while other shapes will only experience a slight reduction in triangle count.

Similar to inference for shapes in the test data set, we find the latent vectors using Equation (8). However, to avoid the missing parts of the shapes being perceived as holes, the target domain Ω_i for partial shape i is constrained to only include points sampled on the remaining parts of the surface of the partial shape. As we know on beforehand that we want to infer the surface from a partial shape, we can leave out points sampled in the bounding box of the partial shape. Despite this change of point sampling strategy, we unfortunately cannot avoid that the network perceives some of the missing surface as being holes. We evaluate the accuracy of the surface reconstructions of the partial shapes against the ground truth shapes using the same metrics as for evaluating the full sur-

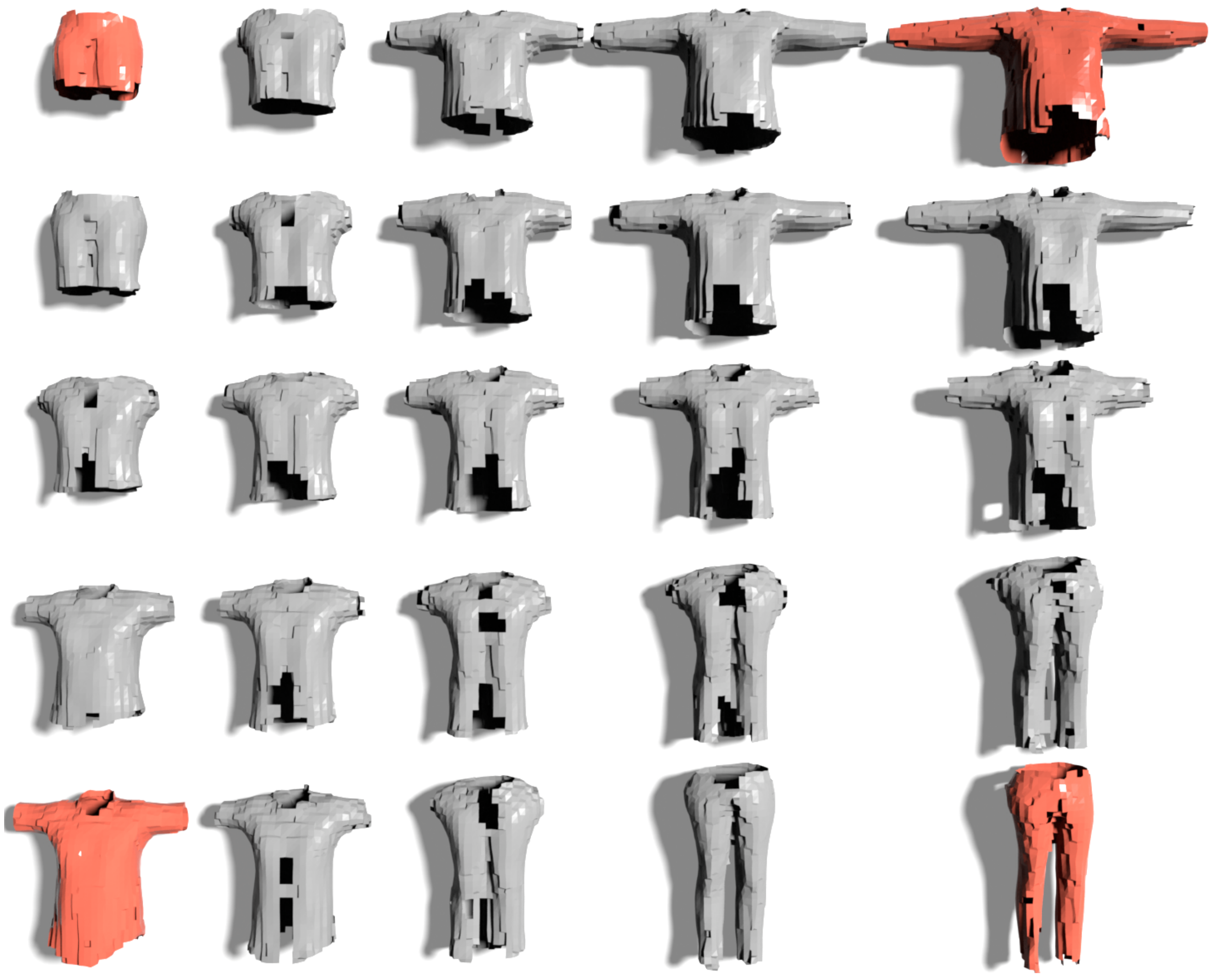

Figure 8: *Samples from a linear interpolation between latent vectors from the UDF based network for the same pair of shorts, T-shirt, coat, and pair of pants as in Figure 6. Reconstructed using the method by Chen et al. [CTFZ22].*

face reconstructions. The results are in table Table 4 and examples of partial reconstructions from the **BU-3DFE** dataset are seen in Figure 10. It is clear that the partial scan reconstructions do not capture the topology of the original surface, as both reconstructions have an open mouth. These results are further discussed in Section 9.

We observe that the SSDF-based network is better at reconstructing partial shapes for the **BU-3DFE** data set, whereas the GWN based network is better at reconstructing partial shapes for the **MGN** based network.

8.4. Latent vector analysis

By using the Lipschitz loss proposed by Liu *et al.* [LWJ*22] we ensure that the output from the network is smooth with respect to the latent vectors. Put differently, the Lipschitz loss penalizes latent vectors which are very close but correspond to very different shapes. Consequently, we expect similar shapes to have similar latent vectors and this is borne out by our experiments. Figure 9 shows a two-dimensional visualization of the inferred latent vectors of the test shapes in **BU-3DFE** and **MGN** for both the GWN and the SSDF based methods. For **BU-3DFE**, it is clear that there are certain areas with similar shapes, even though the points are not linearly separable into their classes. This is in line with similar experiments by Juhl *et al.* [JMdB*21], which also show that gender cannot easily be classified on the basis of the geometry of the facial shapes. For **MGN** the latent vectors for the pants are clearly separated into one distinct group, except a single pair of pants in the GWN based latent space. For both methods, the shirts and coats blend together. For the GWN method the shorts are spread out, whereas the shorts are more grouped together for the SSDF based network. Overall both the GWN and SSDF based methods perform

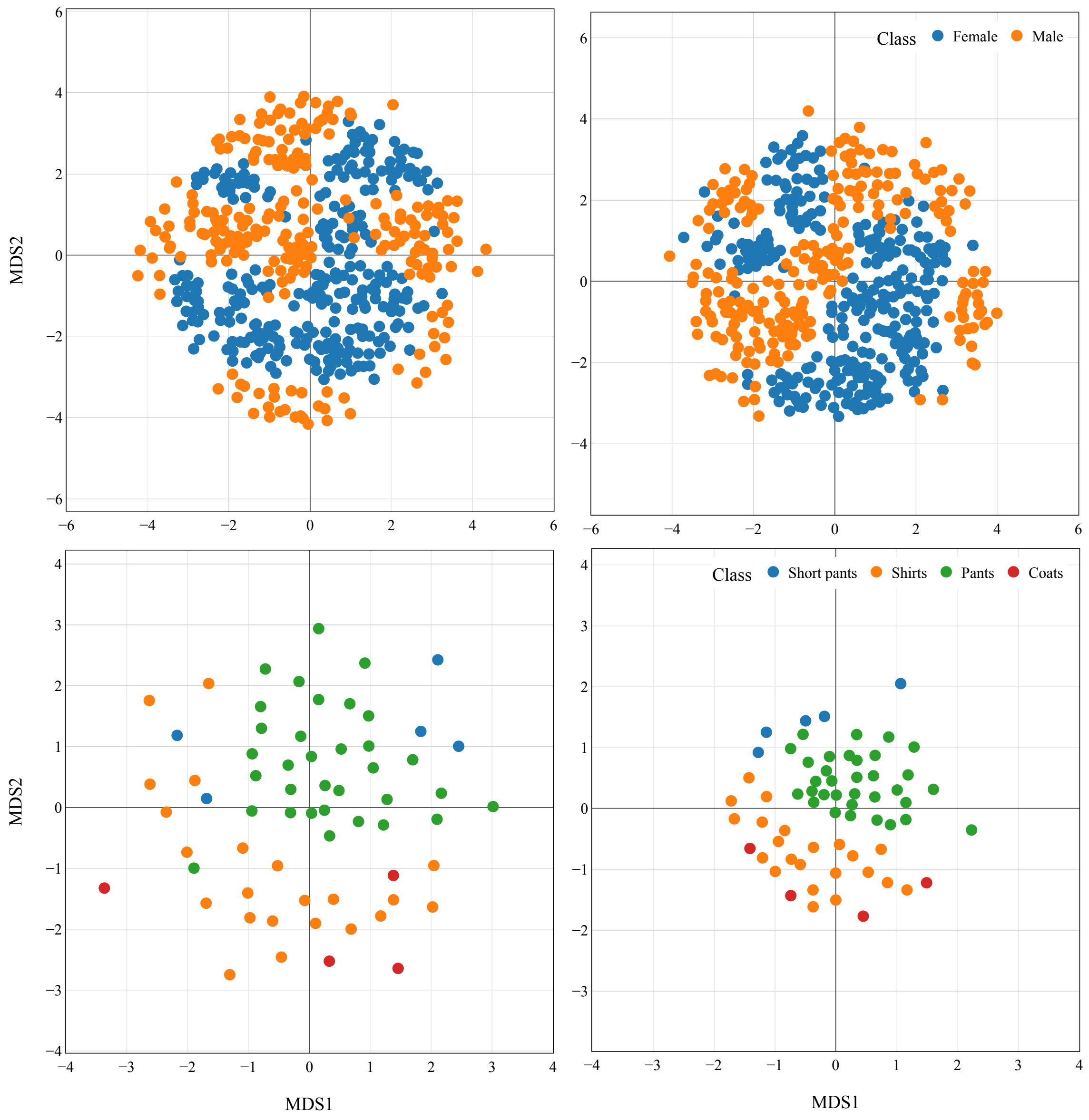

Figure 9: *Two-dimensional visualization of the obtained latent vectors from the test set using multidimensional scaling. Left: GWN based method, right: SSDF based method. Top:* ***BU-3DFE*** *data set, bottom:* ***MGN*** *data set.*

very similarly with respect to clustering similar classes together in the latent space.

9. Discussion

At the outset, we believed that simply using the generalized winding number in a neural implicit representation scheme would lead to very poor results since the GWN is discontinuous on the boundary of 3D shapes and these discontinuities would likely be hard to learn. This turned out to be incorrect: the GWN is not harder to learn than the SSDF and, in fact, the reconstruction error for whole and partial shapes is sometimes slightly smaller for the GWN than the SSDF; both are far better representations than *unsigned* distance fields. A reason for the occasional (slight) superiority of GWN could be that the SSDF representation requires us to also learn the unsigned distance which is gradient discontinuous on the surface and therefore slightly smoothed by the network. Dividing by the reconstructed UDF introduces some error, particularly in the gradient magnitude.

When it comes to the application of shape interpolation, the two methods differ a bit more. It seems that the SSDF representation is generally superior for interpolation, but it depends on the dataset and the specific measure.

Additionally, there are some practical reasons to prefer the SSDF. Since the GWN is discontinuous at the surface, it is not possible to sample it first and then reconstruct from an interpo-

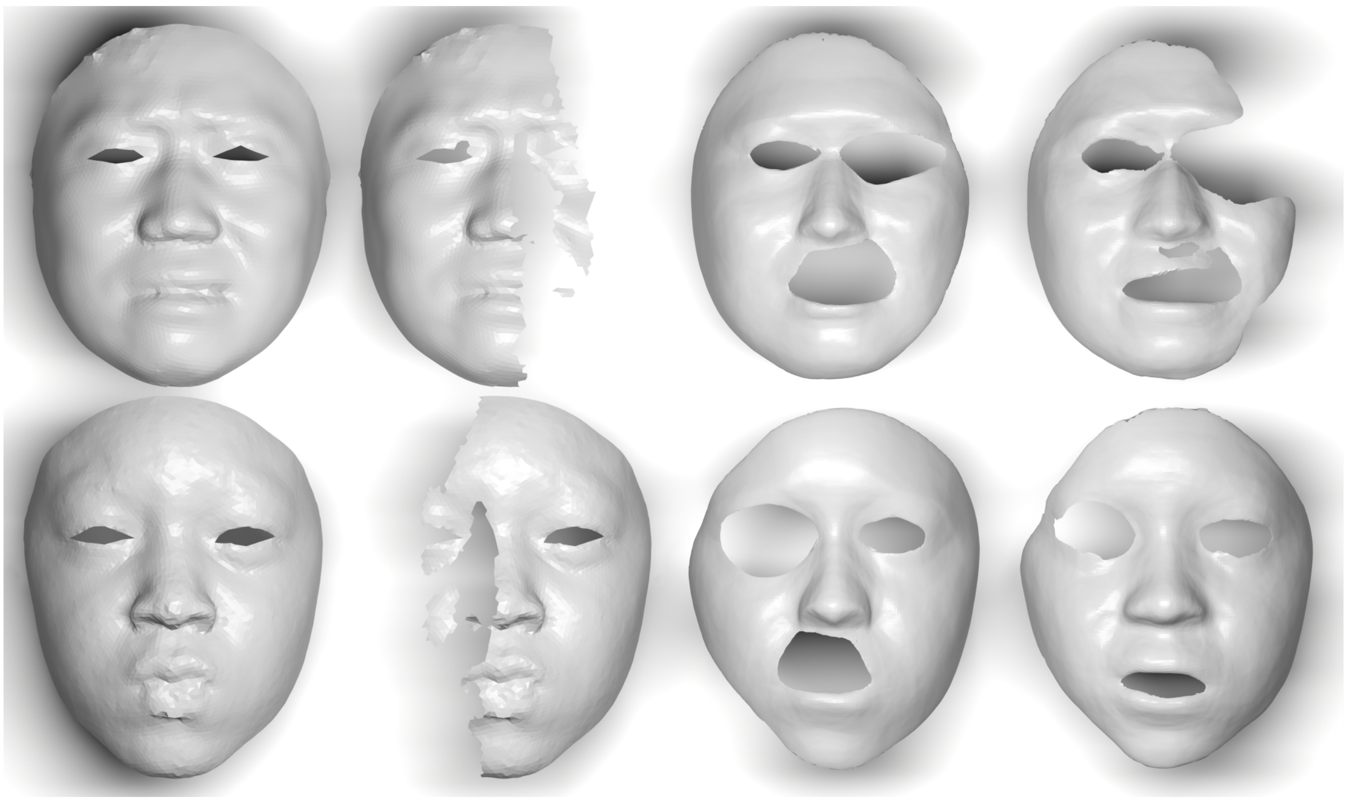

Figure 10: *Partial shape reconstruction from the* ***BU-3DFE*** *dataset [YWS*06]. From left to right: ground truth, simulated partial shape, GWN reconstruction, SSDF reconstruction. The simulated partial scans contain* 61% *and* 59% *(top and bottom) of the original surface.*

lated version of the GWN. This is tantamount to sampling a binary function and would lead to rampant aliasing artifacts. Good reconstruction quality requires the bisection approach discussed above. For similarity in the methods, we used eight iterations of bisection and gradient descent for GWN and SSDF respectively. However, as the SSDF can utilize distance and gradient information to perform Newton-Raphson, it needs much fewer iterations in practice. Moreover, for ray tracing GWN offers no way to estimate the distance to the surface — unlike for a true implicit representation such as the SSDF. This means that schemes such as sphere tracing [Har96] or the method proposed by Sharp and Jacobson [SJ22] are not applicable to the GWN.

We note that the mesh optimization process described in Sections 6.2.1 and 6.2.2 was only carried out in order to enhance the mesh resolution, so the reconstructions could be compared quantitatively. This procedure is not applicable to UDFs where neither bisection nor Newton-Raphson methods can be used to project points onto the surface.

We observed with shape completion that the inferred latent vectors were further apart from the latent vectors from the training set. Using the lipschitz loss does not prevent the inferred latent vectors from converging to a point, where the latent vector matches the partial shape arbitrarily well but does not meaningfully capture the missing parts. Therefore, further work should investigate imposing additional losses during training and inference to avoid the inferred latent vectors to deviate from the training set.

In these experiments the input to the networks has been triangle meshes, but point clouds with normals could also have been used, if these came from an already learned class of shapes, as we only need to estimate the distance from the points to the surface.

9.1. Limitations and future work

The partial reconstructions of the shapes in the **MGN** dataset are of lower quality than the partial reconstructions of the **BU-3DFE** data set. Figure 11 shows the completion of a partial T-shirt mesh where neither method reconstructs the back of the shirt.

We suspect that the poor performance with regards to shape completion can be attributed to our choice of using Lipschitz regularization [LWJ*22]. Lipschitz regularization ensures that the output of the network is smooth as a function of the latent vectors. *Partial* reconstruction is tantamount to finding the latent vector that corresponds to the partial input based on the original (unregularized) loss. Nothing prevents this optimization from moving the latent vector very far away from the latent vectors of the training data if this reduces the loss. That this occurs in practice is indicated by the fact that for multiple inferences of the same partial shape, the standard deviation of the obtained latent vectors were around twice the standard deviation across latent vectors of different shape

Table 3: *Analysis of how four measures vary as the shape is interpolated between two different shapes. The numbers are means over five pairs from the respective test set. ↓ indicates lower is better. Best results marked in bold.*

	BU-3DFE	MGN
Variability, surface area ↓ multiplied by 10^5		
GWN	0.338	0.849
SSDF	**0.323**	**0.609**
Variability, number of connected components, ↓		
GWN	1.45	**0.013**
SSDF	**0.480**	0.020
Variability, number of boundary curves, ↓		
GWN	4.43	4.81
SSDF	**1.23**	**0.373**
Variability, boundary curve length, ↓ multiplied by 10^3		
GWN	6.07	3.34
SSDF	**0.933**	**0.720**

Table 4: *Analysis of the surface reconstructions of the partial shapes. The partial surfaces are grouped after whether they contain more or less than 50% of the original surface area. ↓ indicates lower is better, ↑ indicates higher is better. Best results marked in bold.*

	BU-3DFE		MGN	
Fraction of original surface	≤50%	>50%	≤50%	>50%
Chamfer distance, mean ↓ multiplied by 10^3				
GWN	5.11	1.31	**1.92**	**1.77**
SSDF	**2.90**	**0.772**	2.62	1.84
Mesh completion, mean ↑				
GWN	0.315	0.643	**0.335**	**0.381**
SSDF	**0.387**	**0.749**	0.200	0.277

in the training set, see Figure 12. Note that this issue does not affect interpolation where the latent vectors are affine combinations of latent vectors which correspond to fully specified shapes. Future work should investigate constraining the latent vectors during training and inference.

10. Conclusions

After investigating the GWN and our proposed SSDF for neural representation of open surfaces, we find that, despite minor differences in performance, both the GWN and SSDF are well suited for neural implicit surface representations. They can both represent surfaces with a high degree of accuracy and generate a latent space where latent vectors belonging to the same class cluster together, and reconstruct from partial shapes. Additionally, both methods are able to perform applaudable interpolations, even between shapes that have a varying number of holes, but we find that our SSDF performs slightly smoother interpolations than the GWN.

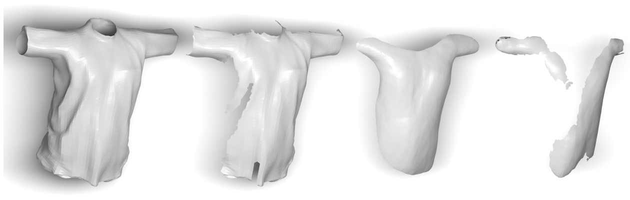
Figure 11: *Partial shape reconstructions of a T-shirt from the* ***MGN*** *dataset [BTTP19]. From left to right: Ground truth shape, simulated partial shape, GWN reconstruction, SSDF reconstruction.*

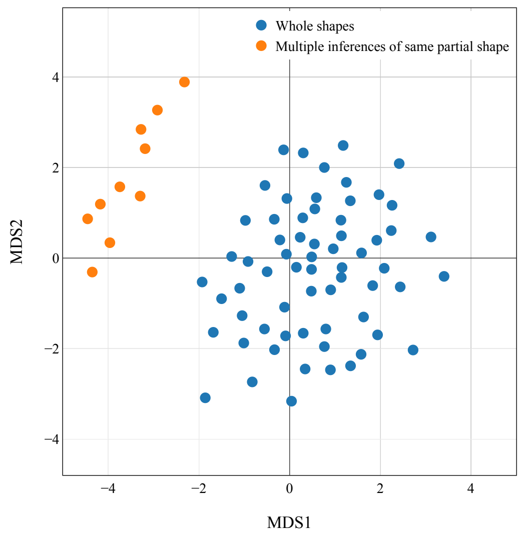

Figure 12: *The latent vectors obtained for the same partial shape with different initializations, along with the latent vectors for all shapes in the test set (whole shapes).*

References

[AL20] Atzmon, Matan and Lipman, Yaron. "Sal: Sign agnostic learning of shapes from raw data". *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, 2565–2574 3.

[AMG*22] Azinovic, Dejan, Martin-Brualla, Ricardo, Goldman, Dan B., et al. "Neural RGB-D Surface Reconstruction". eng. *Proceedings of the Ieee Computer Society Conference on Computer Vision and Pattern Recognition* 2022- (2022), 6280–6291. issn: 2332564x, 10636919, 25757075. doi: 10.1109/CVPR52688.2022.00619 3.

[BDS*18] Barill, Gavin, Dickson, Neil, Schmidt, Ryan, et al. "Fast Winding Numbers for Soups and Clouds". *ACM Transactions on Graphics* (2018) 2–4.

[BGAA12] Bærentzen, Jakob Andreas, Gravesen, Jens, Anton, François, and Aanæs, Henrik. *Guide to Computational Geometry Processing: Foundations, Algorithms, and Methods*. eng. Springer, 2012. isbn: 1447140745, 1447140753, 9781447140740, 9781447140757. doi: 10.1007/978-1-4471-4075-7 4.

[BTTP19] Bhatnagar, Bharat Lal, Tiwari, Garvita, Theobalt, Christian, and Pons-Moll, Gerard. "Multi-garment net: Learning to dress 3d people from images". *proceedings of the IEEE/CVF international conference on computer vision*. 2019, 5420–5430 6, 7, 12.

[CFG*15] Chang, Angel X., Funkhouser, Thomas, Guibas, Leonidas, et al. "ShapeNet: an information-rich 3D model repository [arXiv]". eng. *Arxiv* (2015), 11 pp. 5.

[CP*20] Chibane, Julian, Pons-Moll, Gerard, et al. "Neural unsigned distance fields for implicit function learning". *Advances in Neural Information Processing Systems* 33 (2020), 21638–21652 3, 5.

[CS21] Chi, Cheng and Song, Shuran. "GarmentNets: Category-Level Pose Estimation for Garments via Canonical Space Shape Completion". eng. *Proceedings of the Ieee International Conference on Computer Vision* (2021), 3304–3313. ISSN: 23807504, 15505499. DOI: 10.1109/ICCV48922.2021.00331 3.

[CTFZ22] Chen, Zhiqin, Tagliasacchi, Andrea, Funkhouser, Thomas, and Zhang, Hao. "Neural dual contouring". eng. *Acm Transactions on Graphics* 41.4 (2022), 3530108. ISSN: 15577368, 07300301. DOI: 10.1145/3528223.3530108 3, 6, 9.

[GH12] Gao, Fuchang and Han, Lixing. "Implementing the Nelder-Mead simplex algorithm with adaptive parameters". eng. *Computational Optimization and Applications* 51.1 (2012), 259–277. ISSN: 15732894, 09266003. DOI: 10.1007/s10589-010-9329-3 5.

[GSF22] Guillard, Benoit, Stella, Federico, and Fua, Pascal. "MeshUDF: Fast and Differentiable Meshing of Unsigned Distance Field Networks". *European Conference on Computer Vision*. 2022 3.

[GWM*22] Gao, William, Wang, April, Metzer, Gal, et al. "TetGAN: A Convolutional Neural Network for Tetrahedral Mesh Generation [arXiv]". eng. *Arxiv* (2022) 3.

[Har96] Hart, John C. "Sphere tracing: A geometric method for the antialiased ray tracing of implicit surfaces". *The Visual Computer* 12.10 (1996), 527–545 11.

[JBS06] Jones, Mark, Bærentzen, Jakob Andreas, and Sramek, Milos. "3D Distance Fields: A Survey of Techniques and Applications". English. *IEEE Transactions on Visualization and Computer Graphics* 12.4 (2006). ISSN: 1077-2626 3.

[JKS13] Jacobson, Alec, Kavan, Ladislav, and Sorkine-Hornung, Olga. "Robust inside-outside segmentation using generalized winding numbers". eng. *Acm Transactions on Graphics* 32.4 (2013), 33. ISSN: 15577368, 07300301. DOI: 10.1145/2461912.2461916 2, 3.

[JMdB*21] Juhl, Kristine Aavild, Morales, Xabier, de Backer, Ole, et al. "Implicit Neural Distance Representation for Unsupervised and Supervised Classification of Complex Anatomies". eng. *Lecture Notes in Computer Science* 12902 (2021). Ed. by de Bruijne, Marleen, Cattin, Philippe C., Cotin, Stéphane, et al., 405–415. ISSN: 16113349, 03029743. DOI: 10.1007/978-3-030-87196-3_38 3, 4, 9.

[JP*18] Jacobson, Alec, Panozzo, Daniele, et al. *libigl: A simple C++ geometry processing library*. https://libigl.github.io/. 2018 4.

[KB14] Kingma, Diederik P and Ba, Jimmy. "Adam: A method for stochastic optimization". *arXiv preprint arXiv:1412.6980* (2014) 4.

[Kob00] Kobbelt, Leif. "Sqrt(3) subdivision". eng. *SIGGRAPH'00: 27th International Conference on Computer Graphics and Interactive Techniques Conference* (2000), 103–12 5.

[LWJ*22] Liu, Hsueh-Ti Derek, Williams, Francis, Jacobson, Alec, et al. "Learning Smooth Neural Functions via Lipschitz Regularization". eng. *Acm Siggraph 2022 Conference Proceedings* (2022), 1–13. DOI: 10.1145/3528233.3530713 4, 9, 11.

[MST*20] Mildenhall, Ben, Srinivasan, Pratul P., Tancik, Matthew, et al. "NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis". eng. *Lecture Notes in Computer Science (including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 12346 (2020), 405–421. ISSN: 16113349, 03029743. DOI: 10.1007/978-3-030-58452-8_24 3.

[PFS*19] Park, Jeong Joon, Florence, Peter, Straub, Julian, et al. "Deepsdf: Learning continuous signed distance functions for shape representation". eng. *Proceedings of the Ieee Computer Society Conference on Computer Vision and Pattern Recognition* 2019- (2019), 165–174. ISSN: 25757075, 2332564x, 10636919. DOI: 10.1109/CVPR.2019.00025 2–6.

[PGM*19] Paszke, Adam, Gross, Sam, Massa, Francisco, et al. "PyTorch: An imperative style, high-performance deep learning library". eng. *Advances in Neural Information Processing Systems* 32 (2019). ISSN: 10495258 4.

[PSW*22] Palmer, David, Smirnov, Dmitriy, Wang, Stephanie, et al. "DeepCurrents: Learning Implicit Representations of Shapes with Boundaries". *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, 18665–18675 1, 3.

[RCKV22] Rella, Edoardo Mello, Chhatkuli, Ajad, Konukoglu, Ender, and Van Gool, Luc. "Neural Vector Fields for Surface Representation and Inference". *arXiv preprint arXiv:2204.06552* (2022) 3.

[SJ22] Sharp, Nicholas and Jacobson, Alec. "Spelunking the Deep: Guaranteed Queries on General Neural Implicit Surfaces via Range Analysis". eng. *Acm Transactions on Graphics* 41.4 (2022), 107. ISSN: 15577368, 07300301. DOI: 10.1145/3528223.3530155 11.

[VKS*21] Venkatesh, Rahul, Karmali, Tejan, Sharma, Sarthak, et al. "Deep implicit surface point prediction networks". *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, 12653–12662 3, 5.

[WD97] Wales, David J. and Doye, Jonathan P.K. "Global optimization by basin-hopping and the lowest energy structures of Lennard-Jones clusters containing up to 110 atoms". eng. *Journal of Physical Chemistry a* 101.28 (1997), 5111–5116. ISSN: 10895639, 15205215. DOI: 10.1021/jp970984n 5.

[YGKL21] Yariv, Lior, Gu, Jiatao, Kasten, Yoni, and Lipman, Yaron. "Volume Rendering of Neural Implicit Surfaces". eng. *Advances in Neural Information Processing Systems* 6 (2021), 4805–4815. ISSN: 10495258 3.

[YWS*06] Yin, Lijun, Wei, Xiaozhou, Sun, Yi, et al. "A 3D facial expression database for facial behavior research". eng. *Fgr 2006: Proceedings of the 7th International Conference on Automatic Face and Gesture Recognition* 2006 (2006), 1613022. DOI: 10.1109/FGR.2006.6 1–3, 5, 6, 11.