Neural Vector Fields for Implicit Surface Representation and Inference

Edoardo Mello Rella \(^1\) Ajad Chhatkuli \(^1\) Ender Konukoglu \(^1\) Luc Van Gool \(^{1,2}\)
\(^1\) Computer Vision Lab, ETH Zurich \(^2\) VISICS, KU Leuven

Abstract

Implicit fields have recently shown increasing success in representing and learning 3D shapes accurately. Signed distance fields and occupancy fields are decades old and still the preferred representations, both with well-studied properties, despite their restriction to closed surfaces. With neural networks, several other variations and training principles have been proposed with the goal to represent all classes of shapes. In this paper, we develop a novel and yet a fundamental representation considering unit vectors in 3D space and call it Vector Field (VF): at each point in \(\mathbb{R}^3\), VF is directed at the closest point on the surface. We theoretically demonstrate that VF can be easily transformed to surface density by computing the flux density. Unlike other standard representations, VF directly encodes an important physical property of the surface, its normal. We further show the advantages of VF representation, in learning open, closed, or multi-layered as well as piecewise planar surfaces. We compare our method on several datasets including ShapeNet where the proposed new neural implicit field shows superior accuracy in representing any type of shape, outperforming other standard methods. Code is available at https://github.com/edomel/ImplicitVF.

1. Introduction

Representing 3D surfaces efficiently and conveniently has long been a challenge in computer graphics and 3D vision. A representation for 3D surfaces should be accurate, while being suitable for downstream tasks. Shape analysis, such as 3D shape correspondences \([65, 21, 30]\), 3D registration and deformations \([62, 40, 17]\), or generation \([44, 7, 37, 22, 25]\) rely on having the right representations. Recently, some of these questions were answered with implicit neural representations (INRs) \([41, 33, 46, 10]\), which use the classical signed distance \([39, 38]\) or binary occupancy fields, learned with neural networks. Implicit fields are also used in Neural Radiance Fields (NeRFs) \([34, 60, 63]\), surface representation from point clouds \([2, 4]\), image synthesis and processing \([47, 1, 9, 16]\), and many others.

Two different types of implicit representations are commonly used. Distance based representations \([41, 12]\) associate to each point in space the distance from the closest surface of the object. This is either the signed distance field (SDF) \([41, 46, 10]\) or the unsigned one (UDF) \([12]\), with the former that gives a negative sign to points inside the object and positive outside. The second type of representation is binary occupancy \([33]\), where each point in space is classified to be either inside or outside the object. In both cases, a scalar field \(f(x)\) is learned to represent the surface at a specific level \(\lambda\). In general, SDF is the prevalent representation used in shape correspondences \([65]\), NeRFs \([60]\), geometric regularization \([2, 3, 4]\), etc. Recent work has proposed to supplement SDF with gradient information \([50]\), which adds regularization to surface reconstruction using the normals. Similar work also uses binary occupancy fields \([36, 28, 35]\). UDF was recently used to learn open surfaces \([12]\), with the downside being a much more complex meshing process. Alternative strategies, such as ray-based sampling methods \([18]\) and pairwise point sampling \([61]\) solve open surface representation using a very different approach \([18]\) or added training/architectural complexity \([61]\).

We introduce a novel field for shape representation anal-
ogous to SDF. Inspired by image boundary representations in [42], we map each point in the $\mathbb{R}^3$ 3D space to a unit vector pointing towards the closest surface point. We call such representation Vector Field (VF). We mathematically demonstrate that the surface points can be recovered at the $-1$ level set of the flux density of VF. The proposed vector field allows us to represent both open and closed shapes and can be used for surface regularization, by directly encoding the normals. Instead, in standard implicit representations, normals can only be obtained through a differentiation step [24, 4]. We show the potential application of normals to the representation of piecewise planar surfaces through VF, thus improving the accuracy for certain object classes. Moreover, we perform experiments and show comparisons with VF on the standard ShapeNet [8] subsets with additional classes as well as on open clothes surfaces from the MGN dataset [5]. We obtain state-of-the-art results for surface representation on point accuracy. In summary our contributions are threefold:

- We propose a vector field, VF, for implicit representation of 3D shapes with mathematical proofs and empirical evidence for its representation capability.
- We show that the surface normals directly encoded in VF can be used to efficiently represent piecewise planar objects.
- We perform extensive tests for standard and general shapes, showing the strong performance of our method, achieving superior accuracy and generalization in multiple testing set-ups.

2. Related Work

We briefly review modern implicit functions for shape representation and analysis, followed by discussions on different modifications of the original approaches.

Standard representations. When comparing to traditional voxel-based representations for 3D shapes, the INR [41, 33, 11, 46] have been able to solve, to a large extent, the high memory issues and limitations on downstream tasks [65, 58, 60]. Recent methods in INRs generally either use SDF [41, 10, 65] or binary occupancy [33, 36, 28], with very little consensus on where and how they differ. SDF and binary occupancy have the advantage that they represent closed surfaces and generate watertight meshes with the fast Marching Cubes (MC) algorithm. However, they cannot represent general shapes as open surfaces, non-manifold structures, or multi-layered objects cannot be defined using the inside-outside separation. An equally fundamental representation, the distance field [39], was proposed as a solution for INR of open surfaces [12]. On the other hand, [12] achieves it by forgoing quick and simple mesh inference by MC. A recent work [61] tackles open surface representation through classifying whether point-pairs are on the same side of the surface; it enables fast inference but at the cost of increased training complexity. In some cases, the standard INRs are augmented with positional encoding [34, 52] or sinusoidal activations [48, 4] replacing ReLU layers in the neural architecture.

INR in NeRF and 3D Reconstructions. Implicit fields are also used to construct NeRFs [34, 60], where the 3D surface is defined by a term called volume density $\sigma$. However, these methods are trained for accurate view synthesis rather than scene representation. Several works [60, 24, 54] have proposed the use of a proper SDF in NeRF in order to improve the implicitly represented 3D scene. On the other hand, a ray-based implicit representation [18] was recently proposed, inspired from the so-called light field networks [49]. Another recent work [50] proposes to use SDF along with its gradient for the task of 3D reconstruction. A very similar approach is presented in [55]. Even though in previous work [4, 3] the gradient information is used for surface regularization, it does not contribute to localize the surface, contrary to what we propose.

Grid-based representations and modifications. An important class of 3D shape representations uses the voxel
grid, which is essentially the discretized binary occupancy. [26, 27] directly extended the pixel-like image structure to volumetric data and were used with straightforward extensions to 3 dimensions of image processing techniques, such as convolutions. However, traditional voxel-based methods scale cubically with the resolution in terms of memory usage and computation. Therefore, the first methods proposed [13, 53, 57] could only work with $32^3$ voxel grids. It was later improved to $128^3$ [56, 64] with drawbacks in terms of network sizes and training speed. Recently, voxel-grids are used in implicit representation, particularly in NeRF, by using octree-based sparse voxels [31], sparse spherical harmonic functions on the grid [63], voxel occupancy and hashing [35] or simply interpolating voxel occupancy [51]. Such grid-based interpolation and proxy functions may help improve the standard representations discussed above.

3. Method

We first formally define VF and list the properties which make it suitable for the representation task. We outline how it can be used in practice and compare its properties to other fundamental representations. Finally, we show how using VF allows us to impose a planar prior on the predicted shapes, which is useful for several object/scene classes.

3.1. Vector Field Representation

Suppose we have a 3D object or scene in a subset of 3D space, $\Omega \subseteq \mathbb{R}^3$, embedding the set of 2D surfaces, $\Pi \subset \Omega$, with $x_S \in \Pi$ a surface point. To define the VF representation, we map each point $x \in \Omega$ to the normalized direction $v \in \Gamma \subset \mathbb{R}^3$ towards the closest surface point $x_S$. Given the definition, the points $\{x_S\}$ are discontinuities in the field where the direction of $v$ converges and flips. At the surface points, the field $v$ is mapped arbitrarily to either of the two opposite directional normals of the surface. Generalizing to non-differentiable surface points where no surface normal is defined, this is equivalent to treating the point $x_S$ as a point $x' = x_S + \epsilon$ s.t. $x' \notin \Pi$ and $\|\epsilon\|_2 \rightarrow 0$ and computing the field values for point $x'$.

The VF representation is thus the mapping from positions in space to directions $f : \Omega \rightarrow \Gamma$. Figure 2 visualizes our representation for an example object. We define the VF representation formally as follows:

$$f(x) = v \quad \text{with} \quad x \in \Omega, \quad v \in \Gamma$$

$$v = \frac{x - x_S}{\|x - x_S\|} \quad \text{and} \quad \hat{x}_S = \text{argmin}_{x_S \in \Pi} \|x - x_S\|$$

$$\text{if} \quad x \notin \Pi, \text{otherwise:}$$

$$v = \frac{x' - x_S}{\|x' - x_S\|} \quad \text{with} \quad x' = \lim_{\|r\| \rightarrow 0} \frac{x + \epsilon; x' \notin \Pi \text{and } \epsilon > 0.}$$

Here, $\|\cdot\|$ is the $\ell_2$ norm operator. Eq. (1) defines the representation that maps each point in the subset $\Omega$ of space to a normalized direction vector $v \in \Gamma$. $\epsilon \in \mathbb{R}^3$ is an infinitesimal displacement. We note that when multiple points $x_S$ satisfy the argmin condition (for example at equal distance between surfaces), one of them can be chosen arbitrarily. Similarly, the point $x'$ is chosen as a point with higher coordinate value than $x_S$, with the only constraint that $x' \notin \Pi$. This particular choice is also an arbitrary one.

We now need to ensure that VF can properly represent surfaces embedded in $\mathbb{R}^3$. Specifically, with the three following properties, we establish a one-to-one map between surface points and the $−1$ level set of flux density computed on VF. Here and throughout the paper, we define flux density as the flux through a spherical surface divided by its cross-sectional area when the radius of such sphere tends to zero.

**Property 3.1** The vector field $v = f(x)$ is equal to the negative gradient of the unsigned distance field, i.e., $-\nabla u(x)$, except at the discontinuities at the surface points $\{x_S\}$ and at points at equal distance from multiple surface points.

**Property 3.2** The vector field $v = f(x)$ is equal to the surface normal as it approaches the surface.

**Property 3.3** Consider the VF representation $v = f(x)$ of a piecewise smooth surface as defined in Eq. (1), and the following transform:

$$g(x) = D_\Phi f(x).$$

Here, $D_\Phi$ is the operator for flux density, defined as flux per unit cross-sectional area [32], measured using an infinitesimal spherical surface. A point $x \in \mathbb{R}^3$ then is a surface point, $x \in \Pi$, if and only if it belongs to the zero level set of $g(x) + 1$, i.e.,

$$\Pi = \{x \mid g(x) = 0\}.$$

The properties listed above hold only for piecewise smooth surfaces [38]. However, in practice, the flux density goes slightly lower than $−1$ at discontinuous surface points. Nonetheless, a simple analysis of the field reveals that elsewhere, the flux density is either around 0 or positive, making the thresholding operation for surface inference simple and robust. Following these properties, we show how VF can be used for INR to represent any type of surface.

3.1.1 Field Creation and Training

VF can be used on top of any INR architecture without significant changes. We follow standard practice [41, 65] and use the auto-decoder architecture for training the INR with VF. The field prediction $\hat{v} \in \Gamma$ at each point $x$ is conditioned
on a high dimensional embedding vector $c \in \mathbb{R}^M$. Mathematically, we parameterize the neural VF prediction, with the network parameters $\theta$, as follows:

$$f(x; \theta, c) = \hat{v}.$$  \hfill（4）

At training time, the embedding vector $c$ is optimized together with the network parameters $\theta$ to a shape class; at test time, only the embedding vector is optimized on a small set of observed points. This allows one to test the reconstruction accuracy, while also considering the generalization capabilities. Our INR architecture only differs from [41] in that the output dimension is now 3 instead of 1. Strictly speaking, $D_\theta f(x) + 1$ is the scalar-valued implicit function that defines the surface. However, a discrete flux density measure can be easily performed on $f$ at the post-processing step, and therefore we use the term INR more broadly to discuss the original function $f$ itself.

For training and optimization, the $\ell_1$ loss\(^1\) is computed between the predicted $\hat{v}$ and ground truth vector $v^{gt}$:

$$\ell_{VF} = \|v^{gt} - \hat{v}\|_1.$$  \hfill（5）

More specifically, for each object, the loss is computed on a set of coordinates randomly sampled in space with a higher density around the surface [41]. For inference, surfaces can be defined as points in space with flux density lower than a fixed threshold $\alpha$. Given the theoretical value of $-1$ for surface points, we set $\alpha = -0.7$ to account for small prediction inconsistencies. The flux density computation allows us to adapt the marching cubes (MC) algorithm to run on a VF voxel grid. With this, similarly to an independent work [23] which uses the gradient of UDF, it is then possible to obtain accurate meshes. The $\alpha$ threshold on the flux density identifies the voxels containing a surface and the VF values are used to assign on which side of the surface the vertices lie. For more details about our implementation, refer to the Supplementary Material.

Figure 3: **Representation comparison**: VF is compared to SDF and binary occupancy on thin surface reconstruction. Thanks to its ability to represent open surfaces, VF accurately preserves complete thin surfaces.

\(^1\)\(\ell_2\) or cosine loss may also be used with a similar outcome.

### 3.2. Piecewise Planar Prior on VF

Applying surface priors on an INR requires discovery and regularization of the surface normals, which are conventionally obtained in SDF and UDF through differentiation. In contrast, VF representation allows one to enforce priors on surface normals directly on the prediction. We show this advantage of VF by using it to represent piecewise planar objects. Piecewise planar surface representation or reconstruction is an application well explored in the literature [19, 43, 59, 15] as large low-textured planar parts are common in man-made environment. The Manhattan-world assumption [14] was recently used in INR[24] to obtain multi-view 3D reconstruction of Manhattan worlds. We show that, using VF, it is straightforward to extend the Manhattan assumption to arbitrary directions and numbers of planes which can be specific to the object in a category.

Due to the explicit normals in the VF implicit representation, it is possible to jointly predict a set of bases and select between them at each point in space to obtain the final VF. The direction bases represent a very strong prior on the dominant directions that are present in the scene. The new architecture is described by the following:

$$f(x; \theta, c, b) = s_b$$

$$h(c_i; \theta_b) = b$$

$$\hat{v} = s_b^Tb.$$  \hfill（6）

We use a similar auto-decoder $f$ with the latent embedding $c$, and an additional small network branch $h$\(^2\); its input is a different latent vector $c_i$, and the output is the set of bases $b \in \mathbb{R}^{3 \times k}$, where $k$ is the number of bases. We fix $k = 10$ for our implementation. The main branch $f$ takes as input the latent vector $c$ and the predicted bases $b$, together with the query position $x$, and selects which of the bases to use at that queried position $x$. The selection is done using the binary one-hot vector output $s_b$. In this way, it is possible to learn shape-specific bases, which are then selected to form the VF prediction.

The main branch of the network is trained as a classification network - with the number of classes equal to $k$ - to predict the best matching basis $b_i$ using the Cross Entropy loss. Basis prediction, instead, is trained with an $\ell_1$ loss applied to the basis that best matches the ground truth VF. The final loss is given by:

$$\ell_{pl} = -\log \frac{\exp(s_{b_i})}{\sum_{n=1}^k \exp(s_{b_n})} + w_b \|b_i - \hat{v}\|_1$$

$$i = \arg\min_{n \in [1, k]} \|b_n - \hat{v}\|_1$$  \hfill（7）

Here, we fix the scalar hyperparameter $w_b = 0.1$.

\(^2\)hidden layers with 128 hidden size.
3.3. Discussion on Implicit Representations

Unlike SDF and UDF [39], the VF as an implicit function is not discussed in the literature. Here we provide more insights into the main differences with existing representations.

VF, SDF and binary occupancy. The properties in Section 3.1 show that VF can represent any piecewise smooth surface, including open and non-orientable surfaces, proving its expressiveness. Besides the stronger representation power, we show in Section 4 that VF is able to outperform SDF [41] and binary occupancy [33] also on watertight meshes. The difference can still be partially attributed to the more general representation power. A very thin object, even though represented as watertight, can be better modeled as a thin-shelled surface. When modeled as watertight, instead, small prediction errors may lead to holes in the surface. We show this effect on the inference of a lamp from ShapeNet [8] in Figure 3. Only VF can fully preserve the lampshade and the wire structure.

Additionally, the different performance can be attributed to the effect of extracting the surfaces by thresholding the flux density of VF rather than the predicted level in itself as in SDF and binary occupancy. We conduct a small experiment with a very simple shape shown in Figure 4. We use a small 4-layered network to represent a cube. As it is visible from the edges, VF can best preserve the sharpness of the shape. Furthermore, in Figure 4e we show the effect of artificially adding Gaussian noise on the field predictions. We plot the reconstruction error as rescaled Chamfer distance while increasing the noise standard deviation. As it can be seen, VF and binary occupancy are less sensitive to noise, which makes them less affected by small errors in the prediction. The worse performance of VF compared to binary occupancy at high noise levels is expected, and is due to the addition of noise on three components, which makes the likelihood of flipping at least one direction far higher; at lower and more realistic levels, no discernible difference can be noted. However, it is not clear how the lack of surface context in binary occupancy affects the overall learning process in comparison to others. At each point, away from the surface, the field value contains information regarding the surface distance (in SDF) and direction (in VF), no such information are encoded in binary occupancy.

VF and open surface representations. While considering distance based representations, the UDF, used by [12], can represent any type of surface, similar to VF. At a first glance, one may expect VF and UDF to perform similarly, after all they are related by Property 3.1. However, as shown by [61], predicting UDF [12] suffers from slow inference times and is prone to producing results with holes in the reconstructed shapes due to prediction noise, a major weakness in UDF.

There are some major differences when comparing VF and UDF in practice. Inferring a surface with UDF requires computing its gradient [12], which makes the process more time-consuming and memory-intensive than SDF/VF. Furthermore, by evaluating normal consistency at the surface, we show that VF predictions are more accurate than differentiating UDF. By computing the cosine similarity to the ground truth at the surface across all the categories used in Section 4, VF performs over 8% better than the directions obtained by differentiating UDF. Distinct results for all the categories are available in the Supplementary Material. The same is observed on the cube reconstruction in Fig. 4, where the $\ell_1$ distance from the ground truth of VF is 0.125, better than 0.149 obtained by UDF. This performance gap results in lower accuracy at the edges in Figure 4a and makes VF prediction more suitable than UDF to be applied with MC.

Additionally, as UDF is not symmetric around 0, it suffers from a bias towards the middle of the prediction range between 0 and the maximum distance threshold. This is a factor which leads to high prediction noise as shown in [42] for 2D boundary detection. We observe the same when representing a cube (Fig. 4) with predicted UDF values at the surface that fluctuate between 0 and 0.048. We provide further analysis on this aspect in the Supplementary Material.

Note that the following reference is cited in the text:

Better than cosine similarity for highlighting differences in the simple toy experiment.
Point-pair training for surface detection, proposed by [61], can represent any type of surface as VF. However, it requires a more complex network as it predicts UDF together with the surface detection. Furthermore, for each candidate cube at inference, it requires computing the boundary presence for each 2 vertices couples. The prediction with point couples does not allow to trivially apply prior knowledge on the surface properties, something that is straightforward with VF. Finally, as shown in the Section 4, it achieves quantitatively lower performance compared to competitor methods. A hypothesis for this difference, might be the higher complexity of learning a function that relates point couples, when compared to single point predictions as SDF, binary occupancy, UDF, and VF.

### 4. Experiments

First, we describe the experimental details for network structure and the training set-up, followed by the tasks and the metrics used. We then provide the results on each task comparing the performance of VF to other representations. Finally, we show more qualitative results and provide a discussion on the performance.

#### 4.1. Network Structure and Methods

In order to fairly evaluate our method against others, we follow standard architecture and training methods. For shape reconstruction, we use the popular architecture proposed in [41] in all methods, together with its training and sampling protocol, unless a change is required for the compared method. It is a fully connected auto-decoder network that takes as input a latent vector together with the queried position to predict the field. Both the network and the latent code are optimized at training time, while only the latent vector is optimized at test time. The network is trained for 2000 epochs with samples from 64 scenes in each batch and 16384 points per scene. Each method uses a latent vector of size 256 which is optimized for 800 iterations on 800 points per scene. Each method uses a latent vector of size 256 which is optimized for 800 iterations on 800 points per scene.

The auto-decoder network structure is used together with the state-of-the-art representations currently available, namely the binary occupancy (OccNet [33]), the thresholded signed and unsigned distance field (DeepSDF [41] and NDF [12]), the point-pair method of surface detection (GIFS [61]), and the proposed VF. Each method is trained with its proposed loss; specifically, OccNet [33] uses the binary cross-entropy loss, DeepSDF [41], NDF [12], and GIFS [61] the $l_1$ loss, which we use for VF as well. The threshold value for the signed and unsigned distance field is also set based on previous work [12, 41]. Additionally, we compare to PRIF [18] by taking the results reported on their work. No qualitative comparison was possible for PRIF [18] as no code was provided for reproducing the results.

At inference, we first sample the fields on a $256^3$ voxel grid and then apply marching cubes (MC) algorithm on it. OccNet [33] and DeepSDF [41] can be directly used with the standard version of MC [29] to provide the resulting mesh. GIFS [61] uses an adapted MC algorithm available

### Table 1: Evaluation on watertight surfaces

| Method     | chairs | lamps | planes | sofas | cars |
|------------|--------|-------|--------|-------|------|
|            | Chamfer F1-Score | Chamfer F1-Score | Chamfer F1-Score | Chamfer F1-Score | Chamfer F1-Score |
|            | mean/median | 0.01 | mean/median | 0.01 | mean/median | 0.01 |
| OccNet [33] | 0.420/0.173 | 87.10 | 3.002/0.448 | 76.83 | 0.286/0.027 | 84.21 |
| DeepSDF [41] | 0.337/0.078 | 89.31 | 0.194/0.166 | 85.34 | 0.120/0.073 | 87.84 |
| NDF [12] | 0.418/0.275 | 82.47 | 0.640/0.235 | 75.34 | 0.177/0.085 | 86.49 |
| GIFS [61] | 0.629/0.347 | 78.59 | 0.844/0.278 | 70.05 | 0.347/0.161 | 84.90 |
| PRIF [18] | 0.982/0.267 | 91.21 | 3.276/0.534 | 76.83 | 0.389/0.125 | 84.90 |
| GIFS [61] | 0.612/0.252 | 83.32 | 0.749/0.296 | 68.84 | 0.295/0.041 | 88.84 |

**VF** | 0.193/0.092 | 87.57 | 0.299/0.163 | 79.26 | 0.074/0.024 | 90.11 |

Table 1: Evaluation on watertight surfaces. Reconstruction results on watertight ShapeNet [8] categories. The latent vector used to represent the object has size 256 and is optimized for 800 iterations.

### Table 2: Evaluation on open surfaces

| Method     | cars | bases | lamps | clothes |
|------------|------|-------|-------|---------|
|            | Chamfer F1-Score | Chamfer F1-Score | Chamfer F1-Score | Chamfer F1-Score |
|            | mean/median | 0.01 | mean/median | 0.01 | mean/median | 0.01 |
| NDF [12] | 0.194/0.166 | 83.82 | 0.178/0.157 | 87.82 | 0.573/0.287 | 67.12 |
| MC NDF | 0.483/0.319 | 75.04 | 0.632/0.457 | 84.77 | 0.893/0.309 | 66.96 |
| GIFS [61] | 0.397/0.289 | 76.60 | 0.288/0.125 | 89.43 | 0.729/0.285 | 67.82 |
| VF | 0.137/0.106 | 86.10 | 0.153/0.091 | 90.48 | 0.312/0.144 | 78.46 |

**VF** | 0.137/0.106 | 86.10 | 0.153/0.091 | 90.48 | 0.312/0.144 | 78.46 |

Table 2: Evaluation on open surfaces. Reconstruction results on unprocessed open ShapeNet [8] categories and clothes from MGN dataset [6]. The latent vector used to represent the object has size 256 and is optimized for 800 iterations.
in their public code. NDF \cite{12} is evaluated with their proposed evaluation protocol which projects points in space to the surface based on the predicted NDF \cite{12}; additionally, for fair comparison with the other methods, we also evaluate NDF \cite{12} with the developed adaptation of MC and call this method MC NDF. The same MC variant is applied to our method, VF, to obtain a resulting mesh.

4.2. Tasks, Metrics, and Datasets

We apply the INR methods on three different tasks: watertight shape reconstruction, open shape reconstruction, and piecewise planar objects reconstruction. In each task, the network is first trained on a shape class, and then used to reconstruct a set of unseen objects belonging to the same class.

All reconstructions are evaluated using the symmetric Chamfer distance (CD) on 30,000 points with the results written as \( CD \times 1000 \). Specifically, 30,000 points are uniformly sampled on the ground truth and the predicted mesh, and the average distances from each point in the set to the closest in the other are summed. To reduce the random effect of points sampling, each result is obtained by averaging the results over three different samplings. Additionally, each method is evaluated also in terms of F1-score \cite{20}.

We use multiple classes of the popular ShapeNet \cite{8} dataset for evaluation. We measure watertight mesh reconstruction performance on the chairs, lamps, planes, sofas, and cars classes separately; chairs, lamps, planes, sofas are made watertight following the protocol in \cite{41} and, for cars, we use the popular \cite{13} split. Open shapes are evaluated on the unprocessed mesh from the ShapeNet \cite{8} classes, lamps, cars, buses and the clothes of the MGN dataset \cite{6}. For every task, each class is divided into a training, validation and testing set, with respectively 70\%, 10\%, and 20\% of the data.

4.3. Shape Reconstruction

First, in Table 1, we evaluate the different representations on watertight shape reconstruction from different ShapeNet \cite{8} categories. Second, in Table 2, inference is run on non-watertight meshes from ShapeNet \cite{8} and MGN \cite{6} dataset.

Table 1 and 2 show that VF is overall the best performing method, outperforming any other representation in most cases. The differences can be attributed to the higher capability of VF to preserve sharp details and its robustness as discussed in Section 3.3. Furthermore, the better performance of VF compared to NDF \cite{12}, together with the worse results achieved when applying MC to it (MC NDF), suggests that directly predicting directions is more effective than deriving them from a distance function.

4.4. Qualitative Results

Overall, we can see that every method is able to accurately predict the target shapes with smooth results as shown in Figure 5. Furthermore, VF, together with NDF \cite{12} and GIFS \cite{61} are able to generalize to open and more complex shapes, as in Figure 6. Looking at Fig. 5 more closely, it can be seen that DeepSDF \cite{41} and OccNet \cite{33} struggle in predicting thin surfaces as in the plane reconstruction example. They also struggle to predict sharp angles in the chair.
Figure 6: **General shape representation**: VF can represent both open and multi-layered shapes. VF shows very accurate reconstructions even at the fine details outperforming recent methods in the reconstruction quality.

Figure 7: **Piecewise planar shape representation**: Planar VF represent highly planar objects with high accuracy, preserving straight angles at a detail level even better than VF.

and sofa examples; highlighted in blue. In comparison to NDF [12], VF can predict smoother surfaces and does not suffer from small holes or artifacts, with a possible reason being the higher level of noise on the UDF gradient with respect to VF as described in Section 3.3. Despite smoother than NDF [12], GIFS [61] still suffers from inconsistencies in the surface details and struggles to accurately reconstruct tubular parts as in chair and lamp examples. We provide additional visualizations in the supplementary material.

### 4.5. Piecewise Planar Shape Representation

![Table 3: Reconstruction results on the mainly planar ShapeNet [8] categories of bookshelf, cabinet and laptop.](image)

| Method   | Bookshelves F1-Score | Cabinet F1-Score | Laptop F1-Score |
|----------|----------------------|------------------|-----------------|
| VF       | 1.116/0.491          | 0.298/0.192      | 0.081/0.063     |
| Planar VF| 0.558/0.313          | 0.282/0.176      | 0.074/0.060     |

Table 3: Reconstruction results on the mainly planar ShapeNet [8] categories of bookshelf, cabinet and laptop. While the VF already shows good performance on piecewise planar shapes, we can observe even better performance with Planar VF.

Here, we evaluate qualitatively and quantitatively the performance of applying the planar prior on VF - called Planar VF - on piecewise planar shapes and compare it to VF. This is done on the raw meshes from the bookshelves, cabinets and laptops classes of Shapenet dataset [8].

In Figure 7, we qualitatively compare the meshes generated with VF and Planar VF. Even though both methods are able to represent shapes accurately, Planar VF preserves even small planar regions better and can represent square angles sharply, showing the potential surface regularization capability of VF normals. This can be seen for example in the legs of the cabinet or in the details of the edges. The higher qualitative accuracy is also supported by the quantitative evaluation in Table 3. These results show an additional advantage of being able to explicitly model directions, which allows one to adapt to the desired target shapes effectively.

**Limitations.** Despite the significant results, VF does not ensure watertight meshes even when required and necessitates a non-standard MC algorithm for mesh inference. Furthermore, compared to the decades old binary occupancy and half a century old SDF, many of its properties and potential benefits/drawbacks are yet to be understood, which may be important in many downstream tasks.

### 5. Conclusion

In this paper, we revisit the INRs for representing 3D shapes and analyze the drawbacks of currently available representations. Popular representations, such as the SDF or binary occupancy, lack the expressive power to represent open or non-watertight surfaces. On the other hand, previous INR methods for open surfaces have problems with fast inference, accuracy, or training complexity. We propose to solve open surface representation by considering the novel VF and proved its theoretical suitability. We further showed the possibility to easily apply a shape prior on VF, allowing us to effectively fine-tune the method for piecewise planar objects. Our INR method using VF on standard architecture outperforms existing representations by a significant margin and opens up several new avenues in surface regularization and reconstruction.

**Acknowledgements.** This research was funded by Align Technology Switzerland GmbH (project AlignTech-ETH).
Research was also funded by the EU Horizon 2020 grant agreement No. 820434.

References

[1] Ivan Anokhin, Kirill Demochkin, Taras Khakhulin, Gleb Sterkin, Victor Lemptipsky, and Denis Korzenkov. Image generators with conditionally-independent pixel synthesis. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 14278–14287, 2021.

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

[3] Matan Atzmon and Yaron Lipman. SALD: sign agnostic learning with derivatives. In 9th International Conference on Learning Representations, ICLR 2021, 2021.

[4] Yizhak Ben-Shabat, Chamin Hewa Koneputugodage, and Stephen Gould. DigS: Divergence guided shape implicit neural representation for unoriented point clouds. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 19323–19332, 2022.

[5] Bharat Lal Bhatnagar, Garvita Tiwari, Christian Theobalt, and Gerard Pons-Moll. Multi-garment net: Learning to dress 3d people from images. In Proceedings of the IEEE/CVF international conference on computer vision, pages 5420–5430, 2019.

[6] Bharat Lal Bhatnagar, Garvita Tiwari, Christian Theobalt, and Gerard Pons-Moll. Multi-garment net: Learning to dress 3d people from images. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), October 2019.

[7] Eric R Chan, Connor Z Lin, Matthew A Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas J Guibas, Jonathan Tremblay, Sameh Khamis, et al. Efficient geometry-aware 3d generative adversarial networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 16123–16133, 2022.

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

[9] Yinbo Chen, Sifei Liu, and Xiaolong Wang. Learning continuous image representation with local implicit image function. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 8628–8638, 2021.

[10] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In CVPR, 2019.

[11] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 5939–5948, 2019.

[12] Julian Chibane, Mohamad Aymen mir, and Gerard Pons-Moll. Neural unsigned distance fields for implicit function learning. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, Advances in Neural Information Processing Systems, volume 33, pages 21638–21652. Curran Associates, Inc., 2020.

[13] Christopher B. Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, Computer Vision – ECCV 2016, pages 628–644, Cham, 2016. Springer International Publishing.

[14] James M Coughlan and Alan L Yuille. Manhattan world: Compass direction from a single image by bayesian inference. In Proceedings of the seventh IEEE international conference on computer vision, volume 2, pages 941–947. IEEE, 1999.

[15] Boyang Deng, Kyle Genova, Soroosh Yazdani, Sofien Bouaziz, Geoffrey Hinton, and Andrea Tagliasacchi. Cvxnet: Learnable convex decomposition. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 31–44, 2020.

[16] Emilien Dupont, Adam Goliński, Milad Alizadeh, Yee Whye Teh, and Arnaud Doucet. Coin: Compression with implicit neural representations. arXiv preprint arXiv:2103.03123, 2021.

[17] Marvin Eisenberger, Zorah Lahner, and Daniel Cremers. Smooth shells: Multi-scale shape registration with functional maps. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 12265–12274, 2020.

[18] Brandon Y. Feng, Yinda Zhang, Danhang Tang, Ruofei Du, and Aimitabh Varshney. Prif: Primary ray-based implicit function. In Proceedings of the European Conference on Computer Vision (ECCV), October 2022.

[19] David Gallup, Jan-Michael Frahm, and Marc Pollefeys. Piecewise planar and non-planar stereo for urban scene reconstruction. In 2010 IEEE computer society conference on computer vision and pattern recognition, pages 1418–1425. IEEE, 2010.

[20] Kyle Genova, Forrester Cole, Avneesh Sud, Aaron Sarna, and Thomas Funkhouser. Local deep implicit functions for 3d shape. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2020.

[21] Thibault Groueix, Matthew Fisher, Vladimir G Kim, Bryan C Russell, and Mathieu Aubry. 3d-coded: 3d correspondences by deep deformation. In ECCV, 2018.

[22] Thibault Groueix, Matthew Fisher, Vladimir G Kim, Bryan C Russell, and Mathieu Aubry. A papier-mâché approach to learning 3d surface generation. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 216–224, 2018.

[23] Benoît Guillard, Federico Stella, and Pascal Fua. Meshudf: Fast and differentiable meshing of unsigned distance field networks. CoRR, abs/2111.14549, 2021.

[24] Haoyu Guo, Sida Peng, Haotong Lin, Qianqian Wang, Guofeng Zhang, Hujun Bao, and Xiaowei Zhou. Neural 3d scene reconstruction with the manhattan-world assumption.
F. Proofs of VF Properties

In this Section, we provide the proofs of Properties stated in the main text in Section 3.1. For clarity, we restate the representation definition and the properties followed by the actual proofs.

We define VF (see Section 3.1) as follows.

\[ f(x) = v = \begin{cases} \frac{x - \hat{x}_S}{\|x - \hat{x}_S\|} & \text{with } x \in \Omega, \quad v \in \Gamma \\ -\frac{x - \hat{x}_S}{\|x - \hat{x}_S\|} & \text{and} \\ \hat{x}_S = \text{argmin}_{x \in \Pi} \|x - x_S\| & \text{if } x \notin \Pi, \text{ otherwise:} \\ v = -\frac{x' - \hat{x}_S}{\|x' - \hat{x}_S\|} & \text{with} \\ x' = \lim_{\|\epsilon\| \to 0} x + \epsilon; \quad x' \notin \Pi \text{ and } \epsilon > 0. \end{cases} \]

The arbitrary constraint \( \epsilon > 0 \) implies that each component is greater than 0. Based on the definition of VF, we prove the following properties, listed in Section 3.1.

Property 3.1 The vector field \( v = f(x) \) is equal to the negative gradient of the unsigned distance field, i.e., \( -\nabla u(x) \), except at the discontinuities at the surface points \( \{x_S\} \) and at points at equal distance from multiple surface points.

Proof Consider \( u(x) = \|x - \hat{x}_S\| \). Except at the discontinuities on the surface and at points at equal distance from multiple surfaces, \( \nabla u(x) \) is defined as,

\[ u(x) = \|x - \hat{x}_S\|. \tag{8} \]

The gradient of the distance field is given by, \( \nabla u(x) = \frac{x - \hat{x}_S}{\|x - \hat{x}_S\|} \), thus \( \nabla u(x) = -v \) for points outside the surface. Please refer to [38], page 9 regarding a discussion on the gradient of the distance field. The gradient points to the direction of increasing distance. The result is valid for all
smooth surfaces. For piece-wise smooth surfaces, additional discontinuities may arise, specifically where the vector field $v$ diverges. Note that at the discontinuities, we can still consider the property to be true provided that we use the same one hand-side limit, thanks to the definitions 4.1 and 5.1 in Rudin [45].

**Property 3.2** The vector field $v = f(x)$ is equal to the surface normal as it approaches the surface.

**Proof** For any implicit function $F(x) = 0$, the gradient is equal to the surface normal as $x$ approaches the surface (please refer to [38], pages 9 - 12). Note that at the surface, there are two opposite directions depending on the limit chosen, and both are surface normals, equally valid for our purpose.

**Property 3.3** Consider the VF representation $v = f(x)$ of a piecewise smooth surface as defined in Eq. (1), and the following transform:

$$g(x) = D_{\phi} f(x).$$

Here, $D_{\phi}$ is the operator for flux density, defined as flux per unit cross-sectional area [32], measured using an infinitesimal spherical surface. A point $x \in \mathbb{R}^3$ then is a surface point, $x \in \Pi$, if and only if it belongs to the zero level set of $g(x) + 1$, i.e.,

$$\Pi = L_0(g + 1).$$

Before proving Prop. 3.3, we clarify the meaning of flux density and how it is computed in practice. The flux $\Phi$ of a vector field through an oriented surface is the surface integral of the inner product between the vector field and the unit vector normal to the surface. We now give three examples.

- **i:** Given a flat square surface of side $L$ and a constant vector field $z$ with angle $\theta$ between it and the surface normal, the flux is $\Phi = L^2 \cos \theta \cdot ||z||$.
- **ii:** If we consider a sphere with radius $R$ in the same constant vector field $z$, then the flux is inward in half the sphere and outward in the other half. By computing the surface integral, we find that both the inward and outward components have magnitude $\Phi = \pi R^2 \cdot ||z||$ but opposite sign so that the resulting flux is 0.
- **iii:** As a third example, we take the same sphere with radius $R$ and place it in a radial vector field $z$ pointing outward with constant magnitude $||z|| = \zeta$ and centered on the sphere center. Here, the field vectors are perpendicular to the sphere on its surface so that the overall flux is $\Phi = 4\pi R^2 \cdot ||z||$. Starting from the flux, flux density is the amount of flux divided by the unit area. Similarly to the flux, we define the area $A$ as the surface integral of the absolute value of the inner product between the unit vector normal to the sphere parallel to the vector field in each point and the unit vector normal to the surface. By applying this to the three previous examples, we obtain: i $A = L^2 \cos \theta$, ii $A = 2\pi R^2$, iii $A = 4\pi R^2$. We note that here (ii) we do not have components with opposite sign and in all the three examples the area is not proportional to the magnitude of the field. From the definitions of flux and area it is then possible to compute the flux density $D_{\phi}$ in the three examples as the ratio between the flux $\Phi$ and the area $A$. In practice, we obtain $i D_{\phi} = ||z||$, $ii D_{\phi} = 0$, $iii D_{\phi} = ||z||$. We note that, given the vector field properties and the sphere positioning chosen, $D_{\phi}$ does not depend on the size of the surface or the relative angle $\theta$ (in example i). Thus, the value does not change when the size of the spherical surface tends to zero as is the case here. Furthermore, given the properties of VF and the smoothness assumption, we can consider that at any point in space (except at equal distance from multiple surface points, a case that we handle in the proof) the VF is constant in direction in an infinitesimal sphere. Therefore, in all these points, the area is $A = 2\pi R^2$, which is the cross-sectional area of the two sides. Having clarified the flux density as the flux through a spherical surface divided by its cross-sectional area when the radius of such sphere tends to zero, we can prove Property 3.3.

**Proof** To avoid confusion throughout the proof, we refer to the surface of the object to represent as object-surface and to the surface of the sphere used to compute the flux density as sphere-surface. Let us start proving that, if a point...
belongs to the object-surface, then it is in the zero level set of Eq. (3). We measure the flux density as the flux through a spherical surface divided by its cross-sectional area when the radius of such sphere tends to zero. In the infinitesimal sphere around any object-surface point, using Property 3.2, the only component of VF present is normal to the object-surface, with tangential components approaching 0. Given the assumption of piecewise smooth object-surfaces, the field can be considered constant in an infinitesimal neighborhood, except from the orientation flip at the object-surface. By taking as positive the orientation outward of the sphere-surface, the flux through such infinitesimal sphere-surface centered on an object-surface point is $-2\pi R^2$. Similarly, the cross-sectional area of the sphere-surface is $2\pi R^2$. By taking the ratio of the two values, we obtain the flux density of $\frac{-1}{1}$, which demonstrates that the point on the object-surface belongs to the zero level set $L_0$. 

In order to prove that only object-surface points are in the level set, we can identify two cases of points $\hat{x}$ not belonging to the object-surface. We consider case I when the given point $\hat{x}$ is equidistant to multiple object-surfaces points, multiple outward vectors stem from its infinitesimal neighborhood, consequently producing a positive flux density value. We consider case II when a point $\hat{x}$ has a single closest object-surface point. Given the smoothness assumption, the field can be considered constant inside the sphere-surface as its radius tends to zero. Therefore, in case II the flux density is zero because in each hemisphere the flux has magnitude $\pi R^2$ but opposite sign. Fig. 9a gives an intuition of the levels of flux density in the three types of points we explained.

Having proved property 3.3, we show intuitively that it is possible to extend it to discontinuities without differences in its practical application. There is always a sizable gap in flux density between points that lie on the object-surface and points outside. As shown in Fig. 9b, this can be understood by observing that when an infinitesimal sphere-surface is built around a point on the object-surface (even at a discontinuity), then the VF always points inward on such sphere-surface. Therefore, we always obtain flux density values of $\frac{-1}{1}$. On the other hand, when considering an infinitesimal sphere-surface centered on points outside the object-surface, there is always a component that is pointing outward of the sphere-surface which brings the flux density value around 0.

G. Qualitative Results Analysis

Here we provide additional visualizations of the results obtained with VF and compare to the other representations. Figure 8 shows the results achieved by VF on some challenging open and multi-layered examples to further demonstrate the ability of VF to generate accurate predictions in every shape class. In particular it is worth noting the accuracy in reconstructing the very thin parts in the lamp structure and the inside of the car and the mini-bus.

In Figure 10, we provide additional qualitative comparison to the other existing methods. Similarly to the results shown previously, this highlights the different properties of the methods. As volumetric representations, like DeepSDF [41] and OccNet [33], predict smooth watertight meshes, they achieve high-quality results in smooth classes such as chairs. On the other hand, this property shows to be a limitation when representing thin object parts, such as in the sofas or planes example. Overall, they also struggle in predicting sharp details as highlighted throughout every class.

More example comparison between VF, NDF [12], and GIFS [61] can be seen in Figure 11. Here again, we can notice the similar expressive power between methods, with VF that can outperform the other representations at the detail level and with sharper results. In particular, NDF [12] is significantly more affected by noise compared to VF, which...
leads to small holes and artifacts. As shown in the main paper and in Section I, a probable cause being the higher level of noise observed in the UDF\(^4\) and its gradient. GIFS \([61]\), on the other hand, is less affected by noise but is still less effective in representing details. This might be caused by the higher complexity of having 2 separate predictions and of learning a function that relates point pairs. Furthermore, it does not allow to easily model surface properties, such as planarity, restricting its possible future applications.

\(4\)The representation used by NDF [12]

**H. Implementation Details and Ablation**

In this section, we show the specifics of the network structure used for VF and Planar VF and its training. We further ablate the network structure showing that VF can benefit from larger architectures, such as the one proposed by [12] on the cars ShapeNet [8] category.

As described in the main text, the architecture structure used for the experiments with VF and every other method is an auto-decoder network [41] The specifics of its number of layers and size can be seen in Figure 12a. It is trained for 2000 epochs with samples from 64 scenes in each batch and 16384 points per scene. The optimizer used is Adam, with a starting learning rate of 0.001 which is decreased by a factor of 2 every 500 epochs. The same optimizer and scheduler are used also for the latent vector with a starting learning rate of 0.0005. In addition to the \(\ell_1\) loss used to train VF, a \(\ell_2\) regularization loss is applied to the latent vector to keep its norm small. Weight normalization and 0.2 dropout are applied to each network layer, as well as the ReLU activation except for the output layer.

During inference, only the latent vector is optimized for each shape. It is initialized as a zero-mean Gaussian random
Figure 11: **General shape representation:** VF can represent both open and multi-layered shapes. VF can outperform the other representations across every shape class. We note VF reconstructs with accuracy even the inside of the shown car.

![General shape representation](image)

In order to obtain the reconstructed mesh, the auto-decoder is queried using each position on a voxel grid, together with the optimized latent vector. The norm of VF is then taken and used as input to the MC algorithm, while the normalized VF is used to compute the discrete flux density.

Planar VF uses a similar architecture as shown in Figure 12b. It is trained and optimized in an equivalent manner as explained for VF with the same hyperparameters highlighted for the network and latent vector optimization. As an important training detail, we note that the basis vectors \( b \) are first shuffled in their order before being fed to the main network branch. This allows the network to learn to select the appropriate basis and not overfit on the most common set of bases.

Table 4 shows the results obtained using the larger network proposed in [12]. It is significantly larger compared to the standard auto-decoder architecture commonly used for the task, and requires significantly longer training - 2× increase in both memory and training time. From Table 4, it can be seen that VF benefits from the larger network, and keeps an edge over the compared methods in most metrics.

| Method | Chamfer mean | Chamfer median | F1-Score mean | F1-Score median |
|--------|--------------|----------------|---------------|-----------------|
| NDF [12] | 0.0126 | 0.0120 | 88.09 | 99.54 |
| GIFS [61] | 0.0128 | 0.0123 | 88.05 | 99.31 |
| VF | **0.0122** | **0.0116** | **88.23** | **99.47** |

Table 4: **Network ablation:** Reconstruction results on unprocessed ShapeNet [8] cars with the larger backbone used in [12] [61].

**I. Shape Representation Properties**

In this section, we show experiments on simple shapes as in Section 3.3 to support the validity of VF and compare against UDF. To highlight the differences, we try to reconstruct with a small network two tetrahedron-based shapes and analyze the accuracy and noise levels on them. The first toy experiment uses a tetrahedron overlapped with its dual shape; the second, which can be considered a stress test of the representation’s capabilities, consists of trying to reconstruct a more complex shape, obtained by overlapping multiple rotated tetrahedrons (as explained in Fig. 14).

The network used, takes as input the 3D coordinates, has 5 fully connected (FC) layers with a hidden size of 512, and outputs the VF or UDF prediction.

Figure 13 shows the performance on the 2 overlapped tetrahedrons. We observe that on this slightly more complex shape, UDF struggles significantly more than VF in preserving all the flat surfaces and the edges accurately. This is reflected on the accuracy in predicting the correct directions towards the surface. VF achieves an average \( \ell_1 \) error.
Figure 13: Representation comparison on a double tetrahedron shape. Top row: Reconstruction results; VF better preserves planar sides compared to UDF and does not have holes. Bottom row: Effect of UDF bias on the prediction. It shows the number of UDF predictions in a small range divided by the number of ground truth (gt) UDF in the same range. Values above 1 at a certain distance implies that the number of predictions around that value is higher than the actual number of points at that distance; the opposite is true for values below 1. For a perfect UDF surface extraction, one desires the value at distance 0 to be 1 (necessary condition). Instead the value is much lower indicating an unknown bias.

0.1831, while the UDF gradients performs almost 3 times worse with an error of 0.4862. Figure 13d shows the effect of the bias towards the mid range of the predicted values that affects UDF, as noted in the main text. It is evident that the model predicts values in the middle of the range much more often than they actually appear. Instead, the number of predictions of values at the extremes of the range is significantly smaller than in the ground truth. This reduced capability to predict accurately, is a further reason for the difference in performance between VF and UDF.

Figure 14 shows the stress test on a complex toy shape. Qualitatively we can see a significant difference in the representation ability of VF when compared to UDF. This is reflected in the noise value, with the average $\ell_1$ error of VF at 0.3085, much lower than the one of the UDF gradient at 0.7952. Regarding Figure 14, we note that the high complexity is not just given by the exterior of the shape, but also by all the intersection of the planes under the surface; both these make the function representing the object extremely hard to learn. Similar results are observed on the palm tree representation in Fig. 14 with VF that reconstructs the thin leaves with higher fidelity and avoiding artifacts.

Table 5: Normal consistency evaluation computed as cosine similarity (higher is better). First line of results is on watertight meshes [8] and the second one on general objects [8, 6].

Table 5 further highlights the better performance of VF in comparison to UDF and SDF, and shows that explicitly supervising the normal prediction significantly improves normal consistency. This is evaluated by computing the cosine similarity between the normal at the surface and the normal predicted with VF or obtained by differentiating SDF or UDF. Across all categories, VF outperforms the distance based counterparts even in categories in which the Chamfer distance or the F1-Score were comparable.

$^5$Given that SDF does not flip direction at the surface, the direction is evaluated by always taking the orientation in agreement with the ground truth.
J. Adapted Marching Cubes

A challenging part of implicit 3D representation is an easy transformation to standard mesh representation. Mesh allows rendering and manipulation of 3D shapes using standard graphics tools. For the purpose of going from implicit to mesh representation, a traditionally successful algorithm has been Marching Cubes (MC). Current MC algorithms are developed to produce smooth surfaces without holes and with continuously changing normals. This produces visually appealing results but constitutes a challenge when adapting the algorithm for different types of representation.

The standard MC algorithm is applied to a scalar field to produce triangles at the positions where the scalar field crosses a predefined value. Considering SDF, the value used is 0 and the resulting polygonal surface approximates the zero level set of SDF. This is done by first voxelizing the space and evaluating the field values in these voxels. Furthermore, in order to choose between the multiple possible mesh configurations that arise from the vertex assignments, neighboring voxels are used to ensure surface continuity. To secure smoothly changing normals, the MC algorithms compute them based on a neighborhood around each surface position and locate the mesh faces in each voxel with a trilinear interpolation of the field.

Figure 15: MC comparison on VF with (left) and without (right) the use of its norm. VF norm constitutes an effective measure of the distance from the surface and helps smooth the reconstructed meshes.

In short, MC needs a way to define which voxels contain a surface, which vertices are inside and which outside the shape, and what is the distance of each vertex from the surface. For SDF all the three aspects are solved taking the distance: if there are vertices with different signs then there is a zero-crossing and a surface; the sign of the distance indicates whether a pixel is inside or outside; the absolute value of SDF is the distance from the surface. To adapt the MC algorithm to VF, we now assess the three aspects just highlighted:

- **Surface voxels:** as VF does not have a level set that identifies surface voxels that can be used in MC, we need to define a way to indicate such voxels. For this, we use Property 3.3 and identify the surface voxels as the ones that have a flux density smaller than or close to $-1$. In practice, we explicitly compute the flux density in each voxel and assign a flag to ones where the value is lower than $-0.7$\(^6\); those are surface voxels. From a computational standpoint, discrete flux density can be computed in a highly parallel manner inside the MC computation.

- **Inside-Outside clustering:** for each surface voxel, the directions of the vertices are then clustered into two groups, based on their cosine similarity. Then the two clusters are randomly selected to be considered inside or outside. In practice, the two clusters can be identified by taking the two vectors among the 8 of the voxel with the lowest cosine similarity between each other; these identify the two dominant opposite directions. The 6 remaining vectors are then associated to the cluster with which they share the highest cosine similarity. This is an effective clustering technique with very low computational cost.

- **Surface distance:** the final step to adapt is the use of distance values for the vertices to exploit the trilinear interpolation inside MC. In the case of VF, there is no exact representation of the distance to the surface as it does not explicitly encode it. However, we can use the continuity property of INRs; as the surface is defined by points where field directions flip, we observe that the norm of predicted VF is reduced around the surface points. We note that, even though this measure is not the actual distance, it monotonically changes close to the surface and hence can be used for our purpose. The same effect is also used when applying the traditional MC algorithm on the binary occupancy field [33]. Figure 15 shows that this can be effectively used for the purpose, as the resulting mesh is much smoother.

Our modified MC method is described in Algorithm 1.

Algorithm 1 MCvector \((\mathcal{X}, f_{VF}(x))\)

1. Sample point set \(\mathcal{X} \subset \mathbb{R}^3\) to form a voxel grid.
2. For each vertex \(x \in \mathcal{X}\), compute VF, compute the flux density, and flag voxels with a flux density value lower than $-0.7$. These are surface voxels.
3. For flagged voxels, cluster vertices into two groups using normals. One group of vertices is arbitrarily assigned to be inside and the other the outside the surface.
4. Assign the norm of the predicted VF to the vertices and generate the surface.

\(^6\)As explained in the main paper, we use $-0.7$ instead of $-1$ to account for small mis-predictions.