Supplementary Material

In Sec. 5.1 we describe the procedure used to train the UDF network ϕ_θ on MGN garments. In Sec. 5.2 we explain the metrics used in the experiment section. In Sec. 5.3 we explain in more details the gradients introduced in the main paper. In Sec. 5.4 we show experimental evidence that artifacts appearing at high resolution are caused by the UDF field being approximated. In Sec. 5.5 we demonstrate the benefits of the voting scheme for establishing pseudo-signs. In Sec. 5.6, we show the fitting to sparse pointclouds of Sec. 4.3 can be initialized from random latent codes.

5.1 Network Training

We train one auto-decoder [32] network ϕ_θ to approximate the UDF field of a garment collection. We use the dataset from MGN [6] consisting of 328 meshes from which we keep 300 instances for training and 28 for testing.

To generate UDF supervision sample points and values, meshes are scaled to fit a sphere of radius 0.8, and for each mesh i we generate N training samples $(\mathbf{p}_{i,j}, d_{i,j}) \in \mathbb{R}^3 \times \mathbb{R}^+$ where $d_{i,j}$ is the minimum between d_{max} and the distance from 3D point $\mathbf{p}_{i,j}$ to the i-th shape. We clamp UDF values at $d_{max} = 0.1$ to avoid wasting the network's capacity on learning a precise field away from the surface as in [32,11]. We pick $N = 30000$ and sample 6000 points uniformly on the surface, 12000 within a distance 0.05, 8000 within a distance 0.3 and 4000 within the bounding box of side length 2. As opposed to SDF values, unsigned distances $d_{i,j}$ can be computed directly from raw triangle soups with standard software [14], and do not require any pre-processing of the meshes.

ϕ_θ is implemented by a 9-layer MLP with 512 hidden dimensions and ReLU activation functions. It uses Fourier positional encoding of fifth order on the 3D coordinate inputs [35]. We jointly optimize the network's weights θ with one latent vector embedding $\mathbf{z}_i \in \mathbb{R}^{128}$ per training shape i by minimizing the L_1 loss between the predicted and target UDF values, with a regularization of strength $\lambda = 10^{-4}$ on the norm of the latent codes. With $\mathcal{T}$ as the training set, the full loss is

$$\mathcal{L} = \frac{1}{|\mathcal{T}| \cdot N} \sum_{i \in \mathcal{T}} \left[\sum_{j=1}^{N} |\phi_\theta(\mathbf{z}_i, \mathbf{p}_{i,j}) - d_{i,j}| + \lambda \|\mathbf{z}_i\|_2 \right]$$

and is minimized using Adam [22] for 2000 epochs.

5.2 Metrics

Given a reconstructed mesh $\widetilde{M}$ and a set A of points on its surface, along with a ground-truth mesh M and a set B of points on its surface, we

– **Chamfer distance.** We take it to be

$$\begin{aligned}\mathrm{CHD}(\widetilde{M}, M) = &\frac{1}{|A|}\sum_{a\in A}\min_{b\in B}\|a-b\|^2 \\ &+\frac{1}{|B|}\sum_{b\in B}\min_{a\in A}\|a-b\|^2\,,\end{aligned} \tag{12}$$

the sum of the average distance of each point in A to B and the average distance of each point in B to A.

– **Image consistency.** Let K be a set of 8 cameras located at the vertices of a cuboid encompassing the garments looking at its centroid. For each $k \in K$ we render the corresponding binary silhouette $S_k \in \{0,1\}^{256\times256}$ (respectively $\widetilde{S}_k$) and normal map $N_k \in \mathbb{R}^{256\times256\times3}$ (resp. $\widetilde{N}_k$) of mesh M (resp. $\widetilde{M}$). Then we define the image consistency between $\widetilde{M}$ and M as

$$\mathrm{IC}(\widetilde{M}, M) = \tfrac{1}{|K|}\sum_{k\in K}\mathrm{IoU}(\widetilde{S}_k, S_k) * \mathrm{COS}(\widetilde{N}_k, N_k)\,, \tag{13}$$

where IoU is the intersection-over-union of two binary silhouettes and COS is the average cosine-similarity between two normal maps. Both can be written as

$$\mathrm{IoU}(\widetilde{S}, S) = \sum_{u=1}^{H}\sum_{v=1}^{W}\widetilde{S}_{u,v}S_{u,v} \tag{14}$$

$$\cdot\left[\sum_{u=1}^{H}\sum_{v=1}^{W}max(\widetilde{S}_{u,v}+S_{u,v},1)\right]^{-1}\,, \tag{15}$$

$$\mathrm{COS}(\widetilde{N}, N) = \tfrac{1}{HW}\sum_{u=1}^{H}\sum_{v=1}^{W}\tfrac{\widetilde{N}_{u,v}\cdot N_{u,v}}{\|\widetilde{N}_{u,v}\|\|N_{u,v}\|}\,,$$

with H and W the image height and width, $S_{u,v} \in \{0,1\}$ the binary pixel value at coordinate (u,v) of S and $N_{u,v} \in \mathbb{R}^3$ the color pixel value at coordinate (u,v) of N.

– **Normal consistency.** We take it to be

$$\begin{aligned}\mathrm{NC}(\widetilde{M}, M) = &\tfrac{1}{|A|}\sum_{a\in A}\left|\cos[\widetilde{\mathbf{n}}(a), \mathbf{n}(\operatorname*{argmin}_{b\in B}\|a-b\|^2)]\right| \\ &+\tfrac{1}{|B|}\sum_{b\in B}\left|\cos[\mathbf{n}(b), \widetilde{\mathbf{n}}(\operatorname*{argmin}_{a\in A}\|a-b\|^2)]\right|,\end{aligned} \tag{16}$$

the average *unsigned* cosine-similarity between the normals of pairs of closest point in A and B, where $\widetilde{\mathbf{n}}(x)$ denotes the normal at point x.

5.3 Differentiating through Iso-Surface Extraction

In Sec. 3.2, we derived gradients for surface points with respect to the latent code $\mathbf{z}$. We here expand on the underlying assumptions and justify our choices.

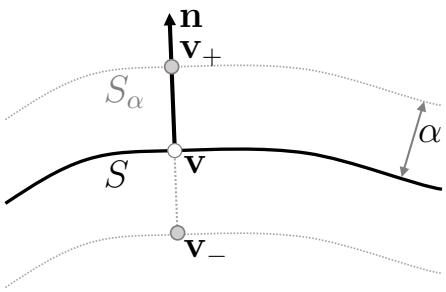


Fig. 10: **Iso-surface differentiation**: S is the minimum-levelset of UDF $\phi(\mathbf{z}, \cdot)$ and S_α its α-levelset ($\alpha > 0$). By using already established differentiability results on $\mathbf{v}_+ \in S_\alpha$ and $\mathbf{v}_- \in S_\alpha$, we derive new derivatives for $\mathbf{v} \in S$.

Using the α-isolevel. Let $\alpha > 0$ be a small scalar and $\mathbf{z} \in \mathbb{R}^C$ a latent code parametrizing the UDF field $\phi(\mathbf{z}, \cdot)$. We consider $\mathbf{v} \in \mathbb{R}^3$, a surface point lying within a facet of the mesh $M_\mathbf{z}$, and $\mathbf{n}$ its surface normal –defined up to its orientation. $\mathbf{v}$ lies on the 0-levelset of the field, and we choose to formulate it as the following linear combination

$$\mathbf{v} = \tfrac{1}{2}(\mathbf{v}_- + \mathbf{v}_+) \,, \tag{17}$$

where

$$\mathbf{v}_+ = \mathbf{v} + \alpha\mathbf{n} \quad \text{and} \quad \mathbf{v}_- = \mathbf{v} - \alpha\mathbf{n} \,.$$

The arrangement of $\mathbf{v}$, $\mathbf{v}_+$ and $\mathbf{v}_-$ is depicted in Fig. 10.

Both $\mathbf{v}_+$ and $\mathbf{v}_-$ are at a distance α from $\mathbf{v}$. Assuming such points to belong to the α-levelset, the outwards pointing normal of $\mathbf{v}_+$ on the α-levelset is $\mathbf{n}$, and the one of $\mathbf{v}_-$ is $-\mathbf{n}$, and we can use [1,34] to write

$$\frac{\partial \mathbf{v}_+}{\partial \mathbf{z}} = -\mathbf{n}\frac{\partial \phi}{\partial \mathbf{z}}(\mathbf{z}, \mathbf{v}_+) \qquad \text{and} \qquad \frac{\partial \mathbf{v}_-}{\partial \mathbf{z}} = \mathbf{n}\frac{\partial \phi}{\partial \mathbf{z}}(\mathbf{z}, \mathbf{v}_-) \,. \tag{18}$$

and differentiating the mapping of Eq. 17 –which we consider as fixed– yields

$$\frac{\partial \mathbf{v}}{\partial \mathbf{z}} = \frac{\mathbf{n}}{2}\left[\frac{\partial \phi}{\partial \mathbf{z}}(\mathbf{z}, \mathbf{v} - \alpha\mathbf{n}) - \frac{\partial \phi}{\partial \mathbf{z}}(\mathbf{z}, \mathbf{v} + \alpha\mathbf{n})\right] \,. \tag{19}$$

Approximate gradients. In practice however, $\mathbf{v}_+$ and $\mathbf{v}_-$ are not guaranteed to lie on the α-levelset, but can be on a β-levelset with $\beta < \alpha$, in which case their normals differ from $\mathbf{n}$ and $-\mathbf{n}$. For our assumption to hold, $\mathbf{v}$ needs to be the closest point to $\mathbf{v}_+$ on the 0-levelset, and similarly for $\mathbf{v}_-$, which is true when α is small compared to the surface curvature. We thus use Eqs. 18, 19 as approximations only.

Eq. 19 is only flawed for points with high curvature, and still holds true for most of the points lying on unwrinkled regions of the surface. Since gradients backpropagated to the latent code are averaged over the entire surface (as in [30]), a minority of them being noisy is not an issue. Sec. 4.3 empirically shows that using $\alpha = 0.01$ works in practice for a wide range of shapes.

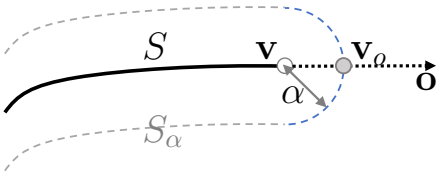


Fig. 11: **Iso-surface differentiation at borders**: S is the minimum-levelset of UDF $\phi(\mathbf{z}, \cdot)$ and S_α its α-levelset ($\alpha > 0$). By using already established differentiability results on $\mathbf{v}_o \in S_\alpha$, we derive a new derivative for $\mathbf{v} \in S$.

Uniqueness of the mapping. Eq. 17 is an arbitrary choice of a mapping. It is not unique, and one could instead pair $\mathbf{v}$ to other points on the α-levelset, leading to a different result in Eq. 19. We deliberately chose the 2 closest points to naturally surround $\mathbf{v}$ with its closest neighbors.

Minimizing a downstream loss. Eq. 19 can be used to minimize downstream loss functions directly defined on mesh vertices with gradient descent. Given such a loss function $\mathcal{L}$, we use the chain rule to write

$$\frac{\partial \mathcal{L}}{\partial \mathbf{z}} = \sum_{(\mathbf{v},\mathbf{n}) \in M_\mathbf{z}} \frac{\partial \mathcal{L}}{\partial \mathbf{v}} \frac{\mathbf{n}}{2} \left[\frac{\partial \phi}{\partial \mathbf{z}}(\mathbf{z}, \mathbf{v} - \alpha \mathbf{n}) - \frac{\partial \phi}{\partial \mathbf{z}}(\mathbf{z}, \mathbf{v} + \alpha \mathbf{n}) \right] \quad .$$

We rely on the field being an UDF and move its zero level set. This is in practice enforced by freezing the network weights, which is thus acting as a strong prior on the field, and only optimize the latent code.

The case of border points. Border do not only have 2 closest neighbors on the α-levelset, but an entire semi-circle as depicted in blue on Fig. 11. In this case, we pair $\mathbf{v}$ with the outmost point on the α-level set with

$$\mathbf{v} = \mathbf{v}_o - \alpha \mathbf{o} \ , \tag{20}$$

and follow the same reasoning as above. We consider $\mathbf{o}$ as a mapping direction, and thus locally fixed.

Constructing the $\mathbf{o}$ *vectors.* Fig. 12 depicts the outwards pointing vectors $\mathbf{o}$ for one reconstructed garment. They are computed as follows. Let $\mathbf{v}$ be a vertex lying on the border, $\mathbf{n}$ be the normal vector of the facet it belongs to, and $\mathbf{e}$ be the border edge it is on. We take $\mathbf{o}$ to be

$$\mathbf{o} = \omega \frac{\mathbf{n} \times \mathbf{e}}{\|\mathbf{n} \times \mathbf{e}\|} \quad \text{with} \quad \omega = \pm 1 \ , \tag{21}$$

the unit vector colinear to the cross product of $\mathbf{n}$ and $\mathbf{e}$. This way, $\mathbf{o}$ is both in the tangent plane of the surface and perpendicular to the border. We choose the sign ω to orient $\mathbf{o}$ outwards. We write

$$\omega = \underset{\{-1,1\}}{\text{argmax}} \ u(\mathbf{v} + \omega \frac{\mathbf{n} \times \mathbf{e}}{\|\mathbf{n} \times \mathbf{e}\|}) \ , \tag{22}$$

that is, we evaluate the UDF in both directions and pick the one that yields the highest value.

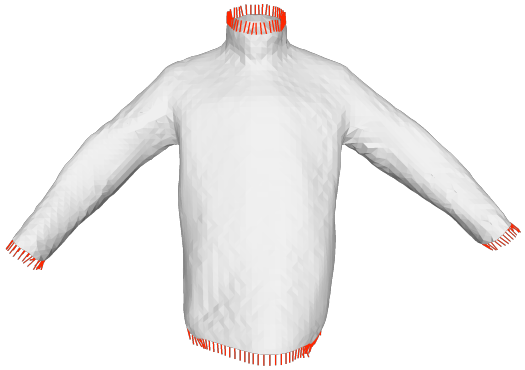


Fig. 12: **Outwards pointing vectors**: for border vertices we define outwards pointing vectors **o** to construct derivatives allowing the surface to shrink or extend along them.

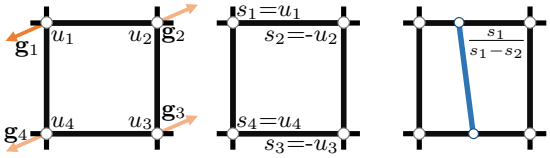
(a) (b) (c)

Fig. 13: **Meshing UDFs: (a)** Ground truth mesh; **(b)** Our meshing procedure applied to a shallow UDF neural network yields staircase artifacts at a very high resolution (512); **(c)** Our method applied to the exact UDF at the same resolution reconstructs a smooth surface.

5.4 Meshing approximate or real UDFs

In Sec. 4.6 and Fig. 6 of the main paper, we mention artifacts of our meshing procedure when applied to approximate UDFs and at a high resolution. This is depicted in Fig. 13**(b)**, where meshing a UDF represented by a shallow network (4 layers) with a grid resolution of 512 yields a mesh that is not smooth.

We hypothesized that this is due to the 0-levelset of the field being slightly inflated into a volume, with many grid locations evaluating to a 0 distance near the surface. This impedes Marching Cube's interpolation step and produces this staircase artifact. To validate this hypothesis, in Fig. 13**(c)** we apply our meshing procedure to the exact UDF grid, numerically computed from the ground truth mesh of Fig. 13**(a)**. This results in a smooth surface, thus indicating that the staircase artifact is indeed a consequence of meshing approximate UDFs.



(a) (b) (c)

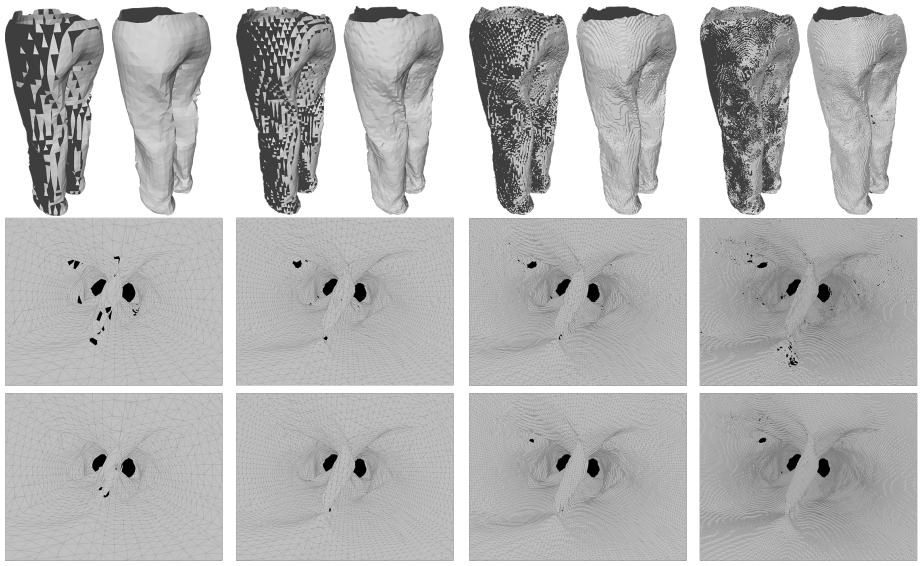
Fig. 14: **Detecting surface crossings**: **(a)** all corners of the grid's cell are annotated with unsigned distance values u_i and gradients $\mathbf{g}_i$; **(b)** we locally approximate signed distances with $s_i{=}\text{sgn}(\mathbf{g}_1 \cdot \mathbf{g}_i)u_i$; **(c)** marching cubes processes these pseudo-signed distances and produces a surface element accordingly.

5.5 Ablation study: pseudo-sign and breadth-first exploration

In Sec. 3 we described a way to locally compute the pseudo-signed distance using gradient orientations (*PSD*), that is described in more details in Fig. 14. The *PSD* method has two shortcomings. First, the choice of the anchor corner implies that the anchor will have a positive pseudo-sign, and thus choosing a different anchor might invert all the signs of the cell. Since the choice is arbitrary, adjacent cells might have opposing sign choices: they will produce meaningful facets, but with opposing orientations. This can be partially fixed in a post-processing step that scans the mesh trying to consistently reorient the facets, but this proved to be a time-consuming operation and it does not always find a consistent orientation. Second, if the surface in the cell or in the immediate proximity is not smooth enough, the gradients of the field can have ambiguous orientations (i.e. they do not clearly oppose each other, for example at a 45° angle). In this setting, two different anchors can produce different pseudo-signs for the corners of the cell, and thus nearby cells that use a different anchor can assign different pseudo-signs to the same corner. This inconsistency creates an unwanted hole in the mesh and happens especially with learned UDF fields, which have noisy gradients.

The breadth-first exploration (*BFE*) method with a voting scheme that we propose has the purpose of improving these shortcomings: produce consistent normal orientations in adjacent facets and increase the robustness of the method on learned UDF fields with noisy gradients. The first objective is reached thanks to the breadth-first exploration itself, which is implemented using queues: following the surface makes it possible to store values of previously computed pseudo-signs, ensuring that corners have the same pseudo-sign in adjacent cells. This also reduces the number of dot products required to complete the meshing procedure, since corners are only computed once instead of being recomputed in every cell. However, simply plugging the pseudo-signed distance computation in this breadth-first exploration can cause even more artifacts due to anchor choice, as they can propagate in nearby cells since the cells are not treated independently anymore.

To solve this problem and at the same time address the second objective, we use the voting scheme described in Sec. 3. This voting scheme has been experimentally inferred by looking at artifacts of the previous procedure, and has three motivations. First, it avoids an explicit and arbirary anchor choice,



(a) 64 (b) 128 (c) 256 (d) 512

Fig. 15: **Comparing qualitative results of PSD and BFE.** Each of the 4 columns corresponds to a meshing resolution, as indicated in the labels. In each column, top row left is the result of PSD, top row right is the result of BFE. Center and bottom rows show an above view of the same mesh, with holes colored in black. The two bigger holes correspond to the legs. Center row is PSD, bottom row is BFE.

which is the main cause of inconsistencies, and it increases the robustness by making multiple neighbours vote for a single corner. Second, it prevents votes to be computed along diagonals in a cell, because the underlying interpolation algorithm of marching cubes does not create vertices along cell diagonals. Third, it prevents gradients facing each other along an edge to vote for having an opposing sign –when they indicate a local maximum of the field instead.

Moreover, we notice that in corners with possible ambiguities the absolute value of the sum of received votes will be low. Some neighbours will vote positively and some others negatively, and the weight itself of the votes can be low when gradients are not clearly facing or opposing each other. We detect these cases that get a sum of votes below a threshold, fixed to $\cos(\pi/4)$, and we put them into a separate queue with a lower priority, to be re-evaluated later. The threshold has been set by noticing that, in a single-vote scenario, gradients at a $[45°, 135°]$ angle have a high ambiguity, since a $45°$ variation in the angle would flip the sign of their dot product. This queue is explored when the main exploration is over, thus increasing the number of neighbours that can vote and making the sign decision more robust. We also employ a third queue, which is explored with the lowest priority, that contains cells with multiple non-adjacent facets. These cells can potentially start the exploration of a non-contiguous surface, and are thus explored at the very end.

Table 5: **Comparing UDF meshing methods: pseudo sign (*PSD*) versus breadth-first exploration with voting strategy (*BFE*).** Average Chamfer distance (CHD), average number of excess holes (EH) and average processing time on 300 garments. We use a single UDF network and only change the meshing procedure.

Resolution	**64**		**128**		**256**		**512**	
Meshing procedure	*PSD*	*BFE*	*PSD*	*BFE*	*PSD*	*BFE*	*PSD*	*BFE*
CHD (↓)	**1.63**	1.66	**1.51**	**1.51**	1.52	**1.51**	1.61	**1.53**
EH (↓)	21	**1.6**	153	**7.8**	1566	**38**	11526	**478**
Time (↓)	0.35s	**0.24s**	1.4s	**1.2s**	10.0s	**9.1s**	105s	**69s**

To validate this algorithm, we compare *BFE* with the simple application of *PSD* using the same garment network and dataset described in Sec. 4.2 (and Tab. 1 left), at different resolutions. The post-processing steps (Fig. 3) applied to the two methods are the same except for the parameters used. Since *PSD* produces slightly less precise borders, we apply a coarser filtering of spurious facets and remove those whose UDF value is larger than 1/6 of the side-length of a cubic cell instead of half. In *PSD* we also apply 5 steps of laplacian smoothing on the borders instead of 1 for *BFE*.

In Fig 15 we see that *BFE* produces consistent facets orientations, while *PSD* does not. Moreover, one can notice small holes in the garments reconstructed with *PSD* (center row), which tend to increase in number and decrease in dimension as the resolution increases, whereas *BFE* is able to close most of them (bottom row), proving to be more robust. Tab. 5 shows that the *BFE* method produces meshes with a slightly lower Chamfer distance, except at resolution 64. Since the size of the holes produced by *PSD* is very small, they do not significatively impact the CHD of this method. They however produce artifacts that are detrimental to the quality of the reconstructed mesh. To have a quantitative measure of this, given a ground-truth mesh M and a reconstructed mesh $\widetilde{M}$, we define the *number of excess holes* as:

$$EH(\widetilde{M}, M) = ||\widetilde{H}| - |H|| \,, \tag{23}$$

where H and $\widetilde{H}$ are the sets of holes of M and $\widetilde{M}$, computed as closed loops of edges that belong to a single triangle. This amounts to computing the number of holes in excess that are in $\widetilde{M}$ compared to M, or viceversa.

Tab. 5 shows that *BFE* has a consistent advantage over *PSD* in this metric across all tested resolutions. In both methods, the EH tends to increase with resolution, as the limits of the learned field are approached and the gradients become noisier. The same experiment with a network trained on only 4 garments yields better results on such garments, with the *BFE* producing no excess holes at all 64-512 resolutions, and *PSD* producing a similar amount to that shown in the table.

Finally, the *BFE* method is also slightly faster than *PSD*. This is mainly due to the reduced number of dot products computed. In *PSD* we compute 8 dot products per cell –which amounts to an average of 4 dot products per corner, since every corner belongs to 4 different cells. In *BFE* each corner receives

Table 6: **Fitting to sparse point clouds, with different latent code initializations:** either from a code of the same garment type (left), or from a random code (right). The table shows average Chamfer (CHD), image consistency (IC), and normal consistency (NC) wrt. ground truth test garments. We report metrics for un-optimized latent codes (*Init.*), after optimizing ($\mathcal{L}_{PC,mesh}$) using our method, and optimizing either $\mathcal{L}_{PC,UDF}$ or $\widetilde{\mathcal{L}}_{PC,UDF}$ in the implicit domain.

	Initialization: same class				**Initialization: random**			
	Init.	$\mathcal{L}_{PC,mesh}$	$\mathcal{L}_{PC,UDF}$	$\widetilde{\mathcal{L}}_{PC,UDF}$	*Init.*	$\mathcal{L}_{PC,mesh}$	$\mathcal{L}_{PC,UDF}$	$\widetilde{\mathcal{L}}_{PC,UDF}$
CHD (↓)	20.45	**3.54**	4.54	4.69	129.51	**3.64**	4.59	4.60
IC (%,↑)	69.54	**84.84**	82.80	82.31	49.08	**84.70**	83.22	82.94
NC (%,↑)	74.54	**86.85**	80.68	86.35	56.74	**86.96**	84.20	86.62

votes from a maximum of 6 neighbours with existing pseudo-signs. Since the exploration starts from one cell and proceeds breadth-first, for the vast majority of corners only a smaller number of neighbours will actually vote, decreasing the total number of dot products.

5.6 Optimization from random initial latent codes

In Tab. 6 we reproduce the experiment from Sec. 4.3 and fit latent codes using sparse point clouds, but start from random latent codes instead of codes from a similar semantic class. This shows that the latter is not even a requirement because, despite starting from much worse initializations, our approach still succeeds better than direct supervision on the UDF values. Starting with latent codes of the same object category remains a plausible scenario because such codes could be provided by a regressor.

5.7 Additional results

Comparison of UDF meshing methods: In light grey are our open surface reconstructions ; In dark grey we display the ϵ-inflated baseline that yields wrongly inflated meshes.

In Fig. 16 we show additional results of our method applied to mesh the UDF regressed from NDF [11] from sparse input point clouds. In Fig. 17, we mesh the UDF predicted by AnchorUDF [39] from input images. In Fig. 18, we compare our meshing method to the ϵ-inflation baseline for other garment samples reconstructed by an auto-decoder network.

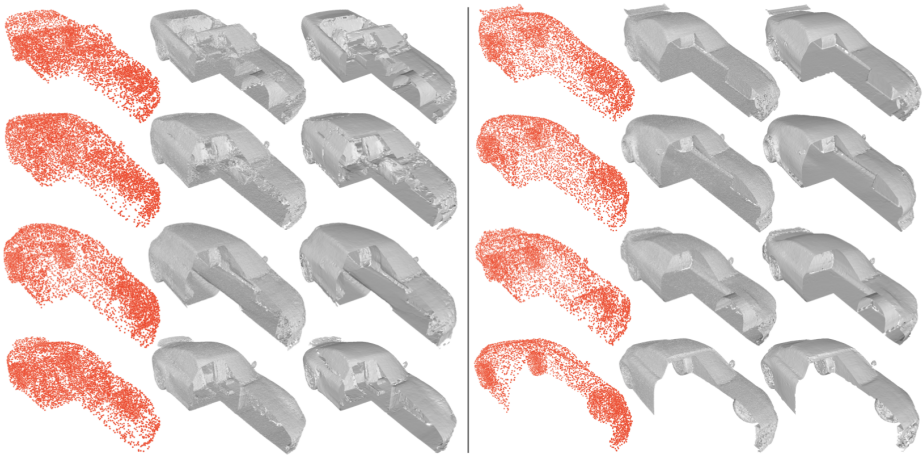


Fig. 16: **Using our approach to triangulate the outputs of NDF [11].** For 8 examples we display the input to the network (a sparse point cloud), a mesh of the predicted UDF mesh reconstructed by the ball pivoting method in more than 2 hours, and a triangulation of the UDF generated using our method in less than 10 seconds.

Fig. 17: **Using our approach to triangulate the outputs of AnchorUDF [39]**, For 4 examples we display the input to the network (a color image), a point cloud of the predicted UDF as originally provided by this network, and a triangulation of the UDF generated using our method.

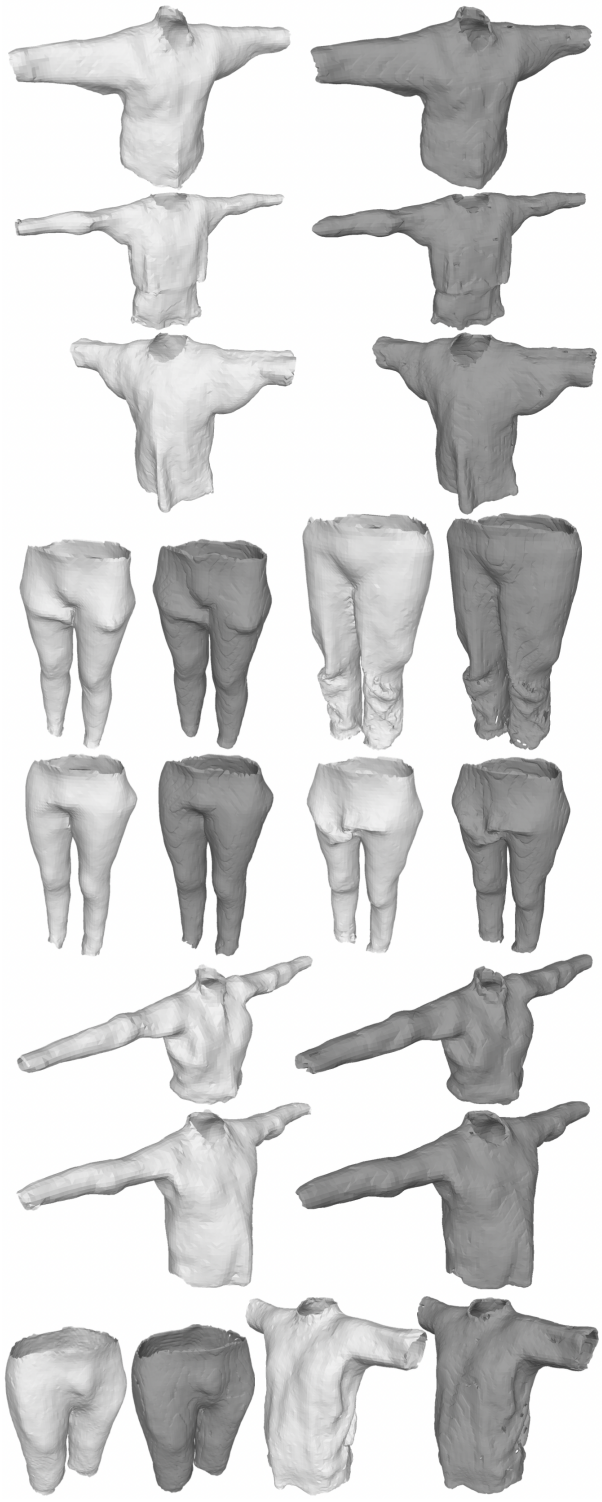


Fig. 18: **Comparison of UDF meshing methods:** Comparison of UDF meshing methods: In light grey are our open surface reconstructions ; In dark grey we display the ϵ-inflated baseline that yields wrongly inflated meshes.