Neural Feature Matching in Implicit 3D Representations

Yunlu Chen [1] **Basura Fernando** [2] **Hakan Bilen** [3] **Thomas Mensink** [4 1] **Efstratios Gavves** [1]

Abstract

Recently, neural implicit functions have achieved impressive results for encoding 3D shapes. Conditioning on low-dimensional latent codes generalises a single implicit function to learn shared representation space for a variety of shapes, with the advantage of smooth interpolation. While the benefits from the global latent space do not correspond to explicit points at local level, we propose to track the continuous point trajectory by matching implicit features with the latent code interpolating between shapes, from which we corroborate the hierarchical functionality of the deep implicit functions, where early layers map the latent code to fitting the coarse shape structure, and deeper layers further refine the shape details. Furthermore, the structured representation space of implicit functions enables to apply feature matching for shape deformation, with the benefits to handle topology and semantics inconsistency, such as from an armchair to a chair with no arms, without explicit flow functions or manual annotations.

1. Introduction

In recent years neural implicit functions for 3D representations (Park et al., 2019; Mescheder et al., 2019; Chen & Zhang, 2019; Michalkiewicz et al., 2019) have gained popularity with benefits including continuous resolution-free representation and handling complicated topologies. Moreover, a single implicit function can be generalised to encode a variety of shapes, by representing each shape with a low-dimensional latent code that conditions the function. This is advantageous in terms of the better reconstruction performance as well as better smoothness and reasonable interpolations between shapes (Park et al., 2019; Chen & Zhang, 2019; Chen et al., 2019b) compared to explicit representations (Yang et al., 2018; Fan et al., 2017; Groueix et al., 2018b; Chen et al., 2020).

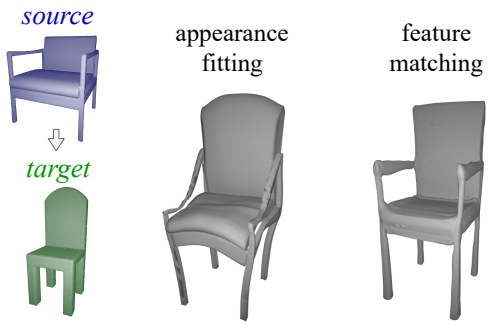


Figure 1. Feature Matching applied to mesh deformation. *Appearance-fitting* methods overfit to the raw geometry of the target shape and fail with inconsistent semantics or topologies easily. Showing MeshODE (Huang et al., 2020) as an example when deforming an armchair to a no-arm chair. Instead, *feature matching* helps to resolve the semantic inconsistency issue and generate meaningful shapes with no extra annotations.

Intuitively, implicit functions enjoy smooth interpolations because they rely on continuous coordinates as query point inputs. However, due to the nature of the encoding, the learned representations are global and do not correspond to any explicit local points. Thus, the advantage of smooth interpolations cannot be applied to shape manipulation of explicit 3D representations like meshes or CAD models.

We propose to extract point-level transformations from the deep features and the gradients with regard to the coordinate input in implicit functions. We track the continuous point trajectory that matches with the minimum change in point-wise features from implicit function with interpolated latent code. These can be considered as "generalised" correspondences, in the sense that the point does not necessarily lie on the interpolated shape surface but can be at any spatial location for the implicit function input.

The resulting transformed shapes will reflect the characteristics of the layer where we collect features. This provides insights of what each layer learns, which was not possible before. By analyzing the *hierarchy* in standard deep implicit functions, we find that early layers gradually map the latent code to coarse shapes, while deeper layers refine finer details. Mid-layer features are semantically distinctive that

[1]Informatics Institute, University of Amsterdam, the Netherlands [2]AI3, IHPC, A*STAR, Singapore [3]School of Informatics, University of Edinburgh, Scotland [4]Google Research, Amsterdam, the Netherlands. Correspondence to: Yunlu Chen <y.chen3@uva.nl>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

encodes high-level information. We postulate that the hierarchical nature of implicit functions with latent codes is what facilitates generalisation over various styles of geometry.

The inherent structure in the implicit functions allows us to apply our method on mesh deformation, which requires to fit a source to a target shape while preserving the local structure (the edge connectivities) from the source shape. Existing learning-based deformation methods (Jiang et al., 2020a; Huang et al., 2020; Wang et al., 2019) follow the *appearance-fitting* paradigm, where the deformed source shape is enforced to fit the target shape as the training objective. Instead, we rather rely on a pre-trained shape autoencoding implicit function, where it learns to fit single shapes with a standard implicit function. The point transformation is later extracted via *feature matching* at inference time. As illustrated in Figure 1, feature matching can be favourable even though appearance fitting looks a more natural choice for such a task. The reason is that by having to optimally fit to different target shapes featuring their own fine details, appearance fitting can be harmful and lead to inconsistent semantics or topologies. Differently, feature matching does not enforce fitting too strictly to the target geometry. We rather match the implicit features with high-level information, helping to resolve the semantic inconsistency issue and generate meaningful shapes without any external part segmentation annotations.

In this work we make the following contributions.

- We propose a way to extend latent-coded implicit functions, so that they can be used for matching boundary points between a pair of examples with minimum feature difference at different scales.
- We find out that features at different scales capture hierarchically different characteristics, with earlier layers capturing the coarser shape outlines and later layers encoding finer shape details.
- We propose a novel shape deformation method that matches point features. The proposed method handles the challenging inconsistencies in topology and semantics, as our approach benefits from the structured feature space from implicit functions.

2. Method

2.1. Preliminaries on implicit functions

A neural implicit representation for a single 3D shape is a function $f_\theta : \mathbb{R}^3 \to \mathbb{R}$ which takes as input a 3D coordinate of any query point from the Euclidean space $x \in \mathbb{R}^3$ and predicts a scalar value indicating if x is inside or outside the shape. A latent-coded implicit function $F_\theta : \mathbb{R}^3 \times \mathbb{R}^k \to \mathbb{R}$ further generalises the function to representing a variety of shapes by conditioning the network on a k-dimensional latent code $z \in \mathbb{R}^k$ as shape identity, which is

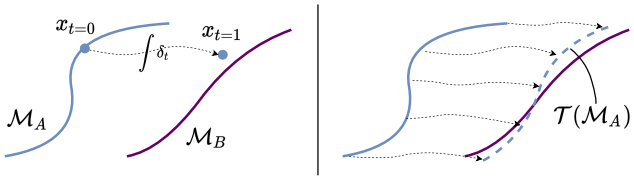


Figure 2. Illustration of the method. *left:* for source shape surface $\mathcal{M}_A$ (blue) and target shape surface $\mathcal{M}_B$ (purple), we sample point $x_{t=0} \sim \mathcal{M}_A$ and solve the trajectory with equation (3) and (4). Note that $x_{t=1}$ does not necessarily lie on $\mathcal{M}_B$. *right:* Sampling dense points from $\mathcal{M}_A$ for feature matching returns the transformed shape $\mathcal{T}(\mathcal{M}_A)$ (blue dashed curve).

either regressed by an encoder E_ψ or jointly optimised in the auto-decoder framework (Park et al., 2019).

For a shape M with the latent code z given, $F_\theta(\cdot, z)$ is a scalar field being either a signed distance field (SDF) (Park et al., 2019) or a $[0, 1]$ occupancy probability field (Mescheder et al., 2019; Chen & Zhang, 2019) that represents the shape. The explicit shape surface $\mathcal{M} := \{x \in \mathbb{R}^3 | F_\theta(x; z) = \tau\}$ is then extracted by marching-cubes (Lorensen & Cline, 1987), with τ the decision boundary whether the query point is inside or outside the shape. Typically $\tau = .5$ for occupancy fields and $\tau = 0$ for SDFs.

Architecture. In a simple form, F_θ is an L-layer multilayer perceptron (MLP) network

$$F_\theta(x; z) = W_L \circ \sigma \circ \cdots \circ \sigma \circ W_1 \, (x \oplus z), \qquad (1)$$

where σ is a piecewise linear activation function (Arora et al., 2018) (*e.g.* ReLU), $\oplus$ denotes concatenation, and for $l = 1, \ldots, L$, $W_l : \mathbb{R}^{w_{l-1}} \to \mathbb{R}^{w_l}$ is the affine mapping corresponding to the l-th layer weights, and w_l is the width of the l-th layer. Note that $w_0 = k + 3$ for the input layer with the concatenation of x and z, and $w_L = 1$ for the output layer. Moreover, for $l = 1, \ldots, l - 1$, we denote $\Phi_\theta^{(l)} : \mathbb{R}^3 \times \mathbb{R}^k \to \mathbb{R}^{w_l}$ to be the first l layers of F_θ

$$\Phi^{(l)}(x; z) := \sigma \circ W_l \circ \sigma \circ \cdots \circ \sigma \circ W_1 \, (x \oplus z). \qquad (2)$$

We refer to the output of $\Phi_\theta^{(l)}(x; z)$ (or simply $\Phi_\theta(x; z)$) as the *implicit features* at x with latent code z.

Training. Given a set of training examples $\{M\}$. The network parameter is usually optimised with a supervised regression loss, or a classification loss in (Mescheder et al., 2019).. With an encoder-decoder design, it is $\mathcal{L}(\theta, \psi) = \sum_{M,x} |F_\theta(x; E_\psi(M)) - s_{M,x}|$, with $z = E_\psi(M)$ and $s_{M,x}$ is the ground-truth signed distance or occupancy probability value. We omit the term θ in the following of the section for simplicity as θ is fixed after training.

2.2. Implicit Feature Matching

The literature on unsupervised correspondence on the image domain (Aberman et al., 2018; Choy et al., 2016), consider as corresponding points those pixels with similar deep features. As inspired, we track the continuous point trajectory that minimises the change in the pointwise implicit features with the latent code interpolation, yielding matching features iteratively with small steps of interpolated latent code.

Given a source shape A, a target shape B and their latent codes z_A and z_B in association with a trained implicit function F, $\mathcal{M}_A = \{x \in \mathbb{R}^3 | F(x; z_A) = \tau\}$ is the source shape surface represented by F. Our objective is to find a shape transformation $\mathcal{T}$ which can be decomposed into point transformations $\mathcal{T}_\mathcal{X}$, such that the collection of local transformations $\mathcal{T}(\mathcal{M}_A) = \{\mathcal{T}_\mathcal{X}(x) | x \sim \mathcal{M}_A\}$ yields a reasonable shape in accordance with shape B.

Inspired by the smooth interpolated shapes with implicit functions, we linearly interpolate the latent code as inspired by the smooth shape interpolations from implicit functions. $z_t := (1-t) \cdot z_A + t \cdot z_B$ yields the interpolation path of z, with the interpolation rate $t \in [0, 1]$. Note that $z_0 = z_A$ and $z_1 = z_B$. At $t = 0$, the initial point coordinate is sampled from the source shape surface $x_0 \sim \mathcal{M}_A$. For a continuous point trajectory initiated at x_0, for any infinitesimal time step between t and $t + \mathrm{d}t$, we define the displacement $\delta_t = x_{t+\mathrm{d}t} - x_t$ required to achieve minimum feature difference:

$$\delta_t = \underset{\|\delta'_t\| < \sigma}{\operatorname{argmin}} \|\Phi(x_t + \delta'_t, z_{t+\mathrm{d}t}) - \Phi(x_t, z_t)\|. \quad (3)$$

σ is a small positive value defining a ball search region $x_{t+\mathrm{d}t}$ around x_t. Our assumption is that the point trajectory is smooth and continuous, because F or Φ, modelled as *piecewise linear functions*, are continuous with the inputs x and z (Arora et al., 2018; Atzmon et al., 2019). While an analytical solution to the path x_t is intractable, we resort to numerical integration

$$x_t = x_0 + \int_{t'=0}^{t'=t} \delta_{t'}. \quad (4)$$

over small displacements δ_t iteratively achieved from Equation (3) within a small time step $\mathrm{d}t$, until reaching $t = 1$ for the desired point transformation to be $\mathcal{T}_\mathcal{X}(x_0) = x_0 + \int_0^1 \delta_t$.

We illustrate the main idea of our method in Figure 2.

Gauss-Newton solution. The numerical integration in equation (4) rests upon an efficient and robust optimization of equation (3). Viewing equation (3) as an over-constrained ($w_l \gg 3$) nonlinear equation system $\Phi(x_t + \delta'_t, z_{t+\mathrm{d}t}) - \Phi(x_t, z_t) = 0$, we resort to Gauss-Newton algorithm for a least square solution that takes into account all first-order partial derivatives when computing local updates and effectively performs an approximation of the second-order derivatives. The Newton's optimisation has quadratic convergence and empirically requires less hyperparameter tuning compared to gradient descent.

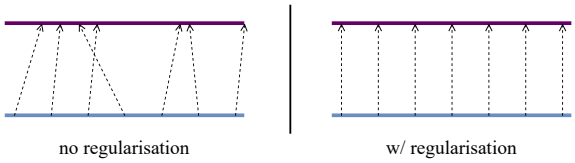


Figure 3. Regularisation. *left:* Nearby points on flat surface have similar features which could cause mismatch. *right:* Adding small penalty on displacement resolves the issue.

The numerical solution of equation (3) is obtained by performing N Gauss-Newton iterations. We denote the displacement at the n-th iteration as $[\delta_t]_n$ initialised at $[\delta_t]_0 = [0, 0, 0]^T$,

$$[\delta_t]_n = [\delta_t]_{n-1} - (J^T J)^{-1} J^T [\mathrm{d}\Phi]_n, \quad (5)$$

where $J = \nabla_{\delta_t} \Phi(x_t + \delta_t, z_{t+\mathrm{d}t}) \approx \nabla_x \Phi(x_t; z_t) \in \mathbb{R}^{w_l \times 3}$ is the partial derivative of Φ with respect to the input coordinates and $[\mathrm{d}\Phi]_{n-1}$ is the feature difference with an infinitesimal change in the input at current estimation of $[\delta_t]_{n-1}$,

$$[\mathrm{d}\Phi]_{n-1} = \Phi(x_t + [\delta_t]_{n-1}, z_{t+\mathrm{d}t}) - \Phi(x_t, z_t). \quad (6)$$

Optimization is run for N iterations such that $\delta_t \leftarrow [\delta_t]_N$. We emphasize again that all above optimizations in equations (3), (4) and (5) are per point coordinate x.

Regularisation on displacement. Due to that the implicit feature field $\Phi(\cdot; z)$ is highly nonconvex, especially in deeper layers, the gradient fields $\nabla_x \Phi(\cdot; z)$ show erratic changes under small disturbances in input x. This happens more often on flat surfaces where adjacent points share similar features, and therefore more sensitive to noise and easily drift aside during feature matching, as illustrated in Figure 3.

To address this, we propose to add a regularisation penalty on the norm of the displacement δ. In our iterative optimizer, this boils down to adding the regularisation that minimises $\|[\delta_t]_n\|$, given $\|[\delta_t]_0\| = 0$. Note the difference from Levenberg–Marquardt algorithm, which minimises $\|[\delta_t]_n - [\delta_t]_{n-1}\|$.

In the iterative scheme, equation (5) is in the form of the generalised Newton's iteration (Ben-Israel, 1966), where we understand $([\delta_t]_n - [\delta_t]_{n-1})$ as the actual target variable, with equation (5) reformed as $[\delta_t]_n - [\delta_t]_{n-1} = -(J^T J)^{-1} J^T [\mathrm{d}\Phi]_{n-1}$. Therefore, minimising is equivalent to optimising for $([\delta_t]_n - [\delta_t]_{n-1}) \rightarrow (-[\delta_t]_{n-1})$. Then we solve it together within the same iteration by modifying equation (5) with:

$$J \leftarrow J \oplus \lambda \cdot \mathrm{diag}(1, 1, 1) \text{ , and} \quad (7)$$

$$[\mathrm{d}\Phi]_{n-1} \leftarrow [\mathrm{d}\Phi]_{n-1} \oplus \lambda \cdot (-[\delta_t]_{n-1}), \quad (8)$$

where λ is the weighting factor for the regularisation and $\mathrm{diag}(1,1,1)$ is indeed the counterpart of the Jacobian.

2.3. Application to mesh deformation

Implicit feature matching can be applied to deforming one source mesh to a target shape. With the source mesh as a set $(\mathcal{V},\mathcal{E},\mathcal{F})$ of vertices $\mathcal{V}$ and the edges $\mathcal{E}$ or faces $\mathcal{F}$, we transform vertices $\mathcal{V}$ with feature matching but not the edges or faces, yeilding the deformed mesh as $(\mathcal{T}(\mathcal{V}),\mathcal{E},\mathcal{F})$.

Freedom of self-intersections. Notably, an important advantage for application to mesh deformation is that our method naturally prevents self-intersections, which implies the existence of at least two point trajectories u_t and v_t intersects at a certain intermediate time t^*.At this moment, $u_{t^*} = v_{t^*}$ are at the same spatial location. Then for any $t' > t^*$, $u_{t'} = v_{t'}$ holds. So in the worst case there can be some mesh vertices that collapse to be at the same position, however, self-intersections are naturally avoided.

We clarify that may still exist due to discrete processing. However, when the set of hyperparameters are properly chosen in our experiments, self-intersection hardly happens with discrete tracking even in the presence of noise,, probably due to the implicit regularisation (e.g. spectral bias (Rahaman et al., 2019)) of the ReLU-MLP network with coordinate inputs.

3. Related Work

Neural implicit 3D representations. Deep implicit functions for 3D shapes have been shown to be highly effective for reconstruction (Park et al., 2019; Mescheder et al., 2019; Chen & Zhang, 2019; Michalkiewicz et al., 2019). Compared to other methods working on explicit representations (Wu et al., 2016; Fan et al., 2017; Groueix et al., 2018b; Yang et al., 2019; 2018), they incorporate 3D coordinates as input to the network which enables resolution-free representation for all topologies.The applications include reconstructing shapes from single images (Saito et al., 2019; Xu et al., 2019; 2020), raw point clouds (Atzmon et al., 2019; Atzmon & Lipman, 2020a;b), 4D reconstruction (Niemeyer et al., 2019), and view synthesis (Sitzmann et al., 2019; Mildenhall et al., 2020). Some recent advances enable to include structural and hierarchical designs (Genova et al., 2019; Jiang et al., 2020b; Chibane et al., 2020; Peng et al., 2020) with implicit function models in order to be aware of information from the local neighbourhood (Chen et al., 2019a) of the query point. We show the inherent hierarchy emerges in a simple latent-coded implicit function.

Learning-based shape deformation. Deformation between shapes with varying topology is a challenging problem. Recent learning-based solutions include learning the offset of each mesh vertices (Wang et al., 2019), free-form deformation grids (Kurenkov et al., 2018) and deformation cages (Yifan et al., 2020). Current state-of-the-art methods use invertible flows (Rezende & Mohamed, 2015; Kingma & Dhariwal, 2018; Dinh et al., 2014; 2016; Chen et al., 2018) to model the shape deformation field (Huang et al., 2020; Jiang et al., 2020a), which enables bijective transformation and is free of intersection by nature.

All above methods follow the appearance-fitting paradigm which could be harmful with inconsistent topology or semantics, unless the segmentation ground truth is available, as in Huang et al. 2020; Gao et al. 2019. By contrast, we rely on the generalisable latent space and implicit features to resolve the problem with no extra annotations.

4. Experiments and Evaluations

Implementation details. For implicit function architecture, we follow IM-Net (Chen & Zhang, 2019) and use a 7-layer MLP with Leaky-ReLU activations, taking as input a 3D coordinate and a 256-dim latent code, and outputs an occupancy probability. The latent code is obtained from a voxel encoder. we train one network per shape category with the same architecture, following the coarse-to-fine progressive training scheme from Chen & Zhang 2019. [1]

We use objects from the ShapeNet dataset (Chang et al., 2015). The optimisation uses the following settings: we use $\mathrm{t} = 0.02$ for a total of 50 intermediate steps with latent code interpolation. For the number of Newton's iterations at each time step we use $N = 3$. The regularisation factor λ is set as 0.01. See supplementary material for more details.

4.1. Analysis and ablation

We first study the effect of matching features from different layers. This can help us obtain insights and better understanding in per layer features in the implicit function.

We densely sample points from the reconstructed shape surface represented by implicit functions and solve the point trajectory of the set of surface points. Then we observe what the set of transformed points forms.

Matching features in different layers We test the matching performance of implicit features for each of the six intermediate layers in the network, as the example in Figure 4. For deeper layers, the transformation gets closer to the target. While for early until mid-layers, the fine geometrical details of the source shape are preserved. In contrast, when deeper layer features are used, the model fits the target shape more in detail. Here some points fail to reach the target surface, because of the largely varied geometry such

[1] We use the improved implementation from the authors at https://github.com/czq142857/IM-NET-pytorch, which has some subtle differences from the original paper.

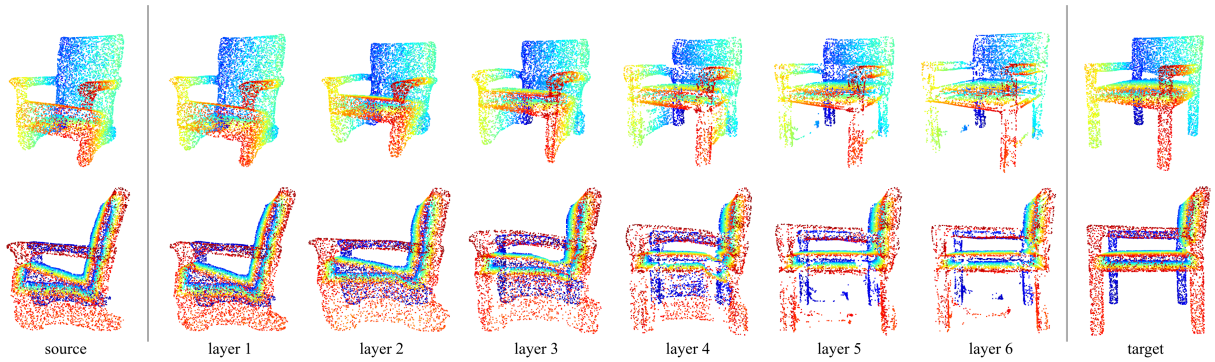


Figure 4. Implicit feature matching applied to different layers. From the source shape transformed to match the target shape in two viewpoints, one per row. *This is not interpolation over time.* We observe that matching with implicit features in earlier layers focuses on fitting the outline, while matching with implicit layers in later layers focuses on fine-grained details. See for example how from layer 1 to layer 4 the model deforms to the coarse geometry of the target in a rigid way, while in layer 6 the model matches almost the exact surface of the target. Empirically matching features from early and mid-level layers leads to no breaking the topology or local details so to be applicable for mesh deformation.

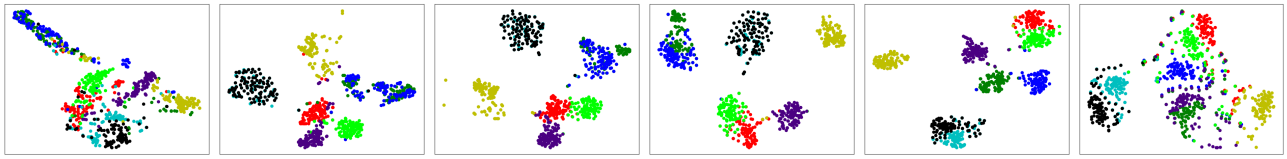
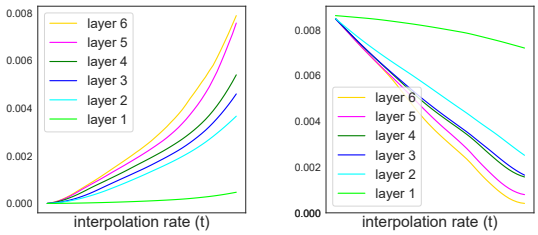


Figure 5. t-SNE visualisation of corresponding point features at different levels, from level 1 (*left*) to level 6 (*right*). Colours represent different groups of corresponding points across 100 chairs. We observe, from early to mid-layers, implicit features become more distinctive, however, the last layers, as closed to the final output that maps all surface points to the same value (τ), hence surface point features become more uniform distributed. We conclude that the mid-level layers encode more high-level semantics, while the last layers, focusing on fitting local shape details, lose global information.



(a) Distance to source (b) Distance to target

Figure 6. Chamfer distance to source and target shapes when transforming shapes using different layers for matching features. Averaged on transformation between 100 random pairs of chairs. We observe that the feature matching in the earlier layers preserves to not deviate from the source while in deeper layers it matches better the target. This is consistent with the observations from Figure 4.

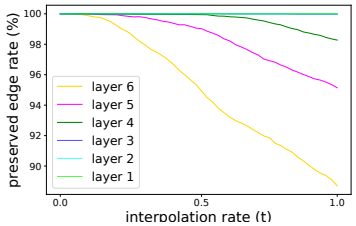


Figure 7. Edge preserveness. Higher value means less edges are broken from the source mesh, which implies that the finer detail is more preserved. Averaged on transformation between 100 random pairs of chairs. Earlier layers (1-3) preserve the detail of the source shape with all 100% edges preserved, and the details are much changed in deeper layers. This is consistent with the observations from Figure 4.

that the model perceives that the corresponding point does not exist on the target at detail-level.

We further corroborate the results quantitatively using a large number of pairs. Figure 6 shows the Chamfer distance from the transformed point set to the source and the target shapes. The transformed shapes from deeper layers are closer to the target and far away from the source. In Figure 7, we evaluate how much the finer detail is preserved by measuring the edge preserveness using mesh data with known edges $\epsilon \in \mathcal{E}$ as the source shape (See §2.3). $\mathcal{T}_{\mathcal{E}}(\epsilon)$ is the transformed edge of source edge ϵ. Edge preserveness is defined as the percentage (%) of $\epsilon \in \mathcal{E}$ subject to $\frac{1}{5} \cdot |\epsilon| \leq |\mathcal{T}_{\mathcal{E}}(\epsilon)| \leq 5 \cdot |\epsilon|$, *i.e.*, when the length of the edge

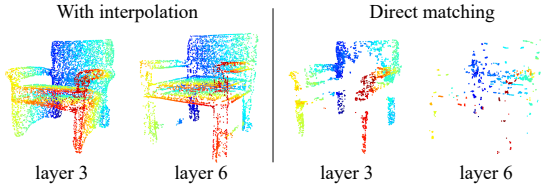


Figure 8. Necessity of matching with interpolation. Direct matching from source and target implicit fields with no intermediate steps from latent code interpolation fails.

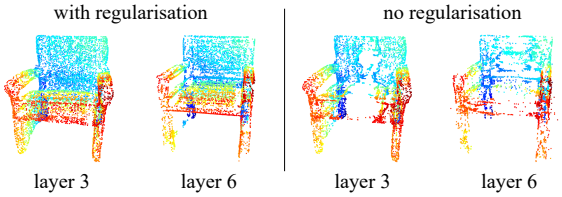


Figure 9. Effect of regularisation. Without regularisation, some points (especially those on the planar regions) do not move to the correct position. Since on the planar region the adjacent points are expected to have similar features, δ_t can easily drift away until the points reach the non-flat part of the shape, where the gradients of the point features has stronger values, and therefore less influenced from the noise. Layer 6 features suffer more from the absence of regularisation because the gradient field is more complex and non-smooth.

does not change more than five times longer or shorter . From Figure 7 we conclude that the transformed shape from early and mid layers preserves well the finer details with regard to the source shape, while in deeper layers the details of local connectivity details are changed significantly. These conclusions are all consistent with the observations from Figure 4.

Hierarchy in implicit function We interpret Figure 4 as reflecting the hierarchy of the layers in the network. The early layers learn gradually the outline of the rough shape geometry, while the final layers learn the finer details. The observation is in the context of shape transformations, but we expect it also holds for reconstruction of single shapes.

We claim that this observation is significant with MLP architecture used for implicit functions. compared with the literature that shows emerging hierarchies in ConvNets for image domain (Zeiler & Fergus, 2014; Bau et al., 2017). ConvNets are inherently more structured with local convolution operations, while similar structures are less expected in MLPs as more general universal function approximators.

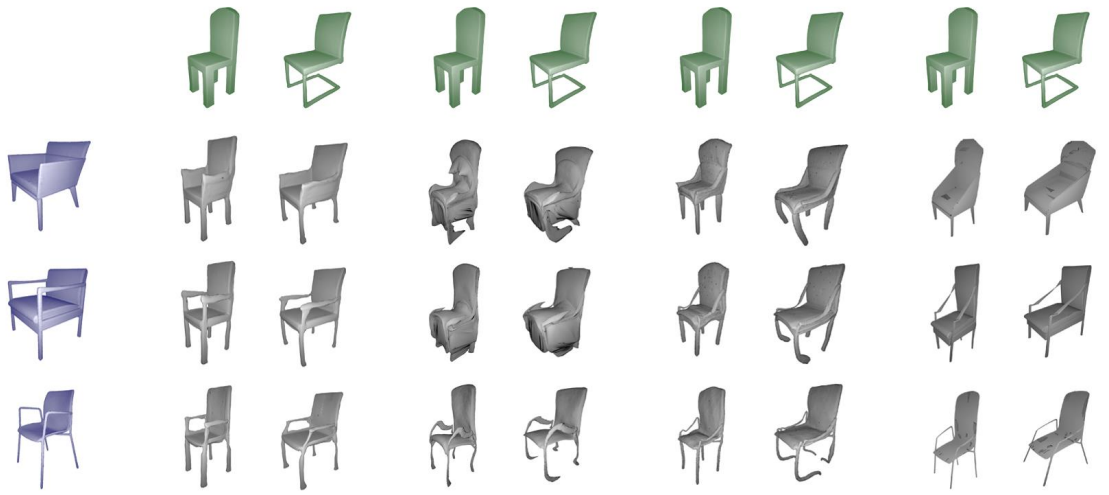
Mid-level features are the most distinctive. To better understand the difference of the features from different layers in the network, we check whether the network can distinguish the features from a set of different corresponding points obtained by feature matching. We take eight points

Figure 10. Effectiveness of transformation with features from combined layers. Deformations shown from matching features from layer 3 (c) and from layer 3 and layer 6 (d). Using feature from multi-levels refines the local geometry to the target (e.g. the legs become thinner) without breaking the topology of the source shape.

from one chair with farthest point sampling (fps). These points are matched by our method to 100 randomly selected chairs and taking the closest point on the target surface. and we mark those from each of the eight source points as a group of correspondents. We visually evaluate if clusters are formed using t-SNE (Maaten & Hinton, 2008), showing the points from each group of correspondents with different colours in Figure 5. We observe that the first layer feature as a linear projection from the input can hardly learn the discrimination, while the mid-level features show good clustering. In the last hidden layer the corresponding points mix, which implies that the features from the layer cannot differentiate between the groups. The reason is that the last layer features are to be mapped to the output implicit field, which is the same value τ to all surface points. Thus, the final layers focus on the local detail and lose global information of the shape.

Our observations give an explanation to the fact that some implicit methods require a *shallow* network design with restricted expressivity in order to aim for either part-level (Chen et al., 2019b) or point-level (Liu & Liu, 2020) correspondence among the shapes with varying topology, while their achievements cannot generalise to more standard deep architectures. With a shallow architecture that limits the expressive power, the output layer feature is less likely to lose too much of the global information.

Direct matching vs. matching with interpolation Alternative to feature matching iteratively with interpolated latent, one can also attempt direct matching between source and target latent without iterations. As shown in Figure 8, points transformed by direct matching fail to form a reasonable shape. Using layer 3, all points move to shape edges, which are expected to have more significant feature gradients. Using layer 6, the points become random, which is consistent with the above analysis that deep layer features lose global information. we hypothesize that the reason for this poor behaviour could be the existence of a smooth non-Euclidean data manifold in the shared latent-interpolated feature space encoded by implicit features. Direct matching would fail



	Ours	ShapeFlow	MeshODE	NeuralCage
fitting paradigm	feature matching	appearance-fitting	appearance-fitting	appearance-fitting
rigidity constraint	✘	✘	✓ (moderate)	✓ (heavy)

Figure 11. Deforming from *arm* (source, blue) to *no-arm* (target, green) chairs. Appearance-fitting methods fail with overfitting to the target shapes with no chairs. Adding rigidity constraints eases the problem, but not a fundamental solution. By contrast, our method resolves the problem without enforcing rigidity, benefited from implicit features.

in case of such a non-Euclidean manifold, since the feature difference is measured using distance in Euclidean space. However, more investigation would be needed for this claim.

In Figure 9 we show qualitatively the ablation experiment from transformation via feature matching without such a regularisation. We show layer 3 and layer 6 as examples with the same source and target shapes as in Figure 4 of the main paper. We see that without regularisation, some points are not at the ideal position to compose the shape, especially on planar regions such as seat and back. Layer 6 features suffer more from no regularisation because it is more complex and non-smooth. Those points more easily goes to the direction of the edges of the shape where the feature gradients are expected to be more significant. This is probably due to that the change of latent code is not ideally infinitesimal, so the resulting δ_t is not accurate enough. Since on the planar region the adjacent points are expected to have similar features, δ_t can easily drift aside.

4.2. Mesh Deformation

Based on the above observations and analysis, matching mid-level features are able to fit the target shape outline without breaking the local detail of the source. In addition, mid-level layers encode the most high-level global information. We apply feature matching for mesh deformation, which requires to preserve the local edge connectivity.

Choice of layers to match. We mostly rely on layer 3 features for matching each vertex from the source mesh. However, we are not restricted to using only single-level features. Empirically we find that using mid-layer features jointly with finer level features downweighted by a small factor $0 < \eta < 1$ helps to fit the local geometry of the target shape better, as illustrated in Figure 10. We take a combination of layer 3 and layer 6 features where the layer 6 feature is weighted by $\eta = 0.1$.

Qualitative results and analysis. We compare the quality of mesh deformation to ShapeFlow (Jiang et al., 2020a) MeshODE (Huang et al., 2020) and NeuralCage (Yifan et al., 2020). We focus on an armchair to no-arm chair transition as a challenging senario, as in Figure 11. We conclude that our method produces much more plausible results, while other methods, all following the appearance-fitting paradigm, suffer from some degree of overfitting to the target shape.

Though not a fundamental solution, appearance-fitting methods mitigates the problem of overfitting to the target by imposing explicit constraints on rigidity (Sorkine & Alexa), measuring how much the source edge is preserved by the deformed mesh. Both ShapeFlow and MeshODE learn flexible deformable flow fields, yet introduce unnatural distortions. The difference is that MeshODE constrains deformation with a rigidity loss such that the local connectivity is preserved better than ShapeFlow, and the distortion is less

Table 1. Evaluation between deformed shape and the target. Numbers are CD($\times 0.001$) / EMD($\times 0.01$) in each cell. Note that the whole shape measurements is biased towards overfitting to the target, *e.g.* distorting the chair arms in Figure 11 is considered as better performance, hence tailored for appearance-fitting methods, and not always a good measure. Our method is consistently better at part-level metrics, indicating better handling the semantics inconsistency.

Shape category	chair		airplane		table	
Part-level evaluation	×	✓	×	✓	×	✓
ShapeFlow (Jiang et al., 2020a)	1.365 / **6.750**	4.285 / 5.794	**0.378** / **5.194**	5.551 / 5.229	- / -	- / -
MeshODE (Huang et al., 2020)	**1.187** / 7.281	4.148 / 5.315	- / -	- / -	**2.564** / **8.298**	14.859 / 7.578
NeuralCage (Yifan et al., 2020)	4.372 / 8.563	6.477 / 6.319	- / -	- / -	11.367 / 11.116	21.676 / 9.378
This paper	1.744 / 7.143	**3.772** / **3.256**	0.935 / 5.601	**5.458** / **4.193**	4.998 / 8.387	**14.748** / **4.174**

Figure 12. Examples of transforming chairs in a variety of styles from source (blue) to target (green). Our method is able to handle all variations meaningfully.

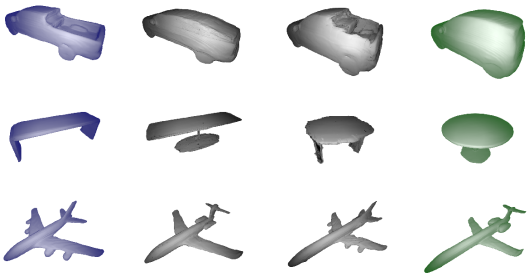


Figure 13. Deformations of car, table and airplane. From left to right: source shape, target to source, source to target, and target shape. Our method generalise to shapes from different categories.

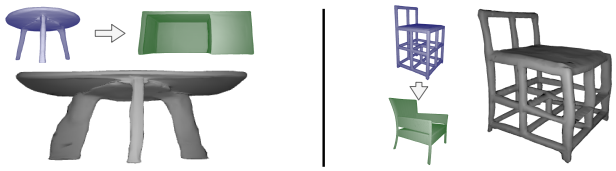


Figure 14. Failure cases. From source (blue) to target (green). Our method is able to handle all variations meaningfully. *Left*: two back cylinder legs are deformed to flakes to match the thin-plate table stand on the target; *Right*: the seating part is not well-recognized and stays at arms height on the target.

heavy. In NeuralCage the rigidity constraint is even stronger by design,where the cage structure is used to bound an area of vertices, and deformation is solved per cage rather than per vertex by morphing the vertices accordingly. However, NeuralCage still suffers occasionally. As seen in the figure, the method bends down the arms and even the seats as a whole, which indicates that the rigidity constraints do not suffice. By contrast, our method proposes a more fundamental solution, which relies on the hierarchy and the high-level information encoded by mid-level implicit features. We resolve the overfitting to such semantics or topology inconsistencies without explicit rigidity constraints.

More results are available in Figure 12, where we see that our method handles different styles of shapes. In Figure 13, a few examples on three other categories of airplanes, tables and cars. In Figure 14, we show some examples of failure cases. We also include some results on the continuous interpolation of the deformation process in supplement material.

Quantitative evaluations. We evaluate the bidirectional Chamfer distance (CD) and Earth Mover's distance (EMD) between the transformed shape $\mathcal{T}(A)$ and the target B to quantify the matching quality. Both metrics measuring the matching quality of the shapes as a whole are biased towards the aforementioned overfitting issue, *e.g.* the unnatural distortions in Figure 11 are regarded as better performance. See supplementary material for more discussion on the limitation of the global metrics. For this reason, we also evaluate the part-averaged distances which better reflect the ability to handle semantic inconsistency. Formally

$\text{CD}_{\text{part}}(\mathcal{T}(A), B) = \frac{1}{N_c}\sum_{c=1}^{N_c} \text{CD}(\mathcal{T}(A_c), B_c)$. $A_c \subseteq A$ is the part segment that belongs to the c-th of all N_c part category for shape A , and so is B_c. EMD_{part} is defined similarly. We evaluate on three representative shape categories, chair, table and airplane. For each category, we randomly select 500 pairs of source and target shapes from the test split of ShapeNetPart (Yi et al., 2016) with the data preprocessed by Chen et al. 2019b. The results are in Table 1.

Our method is consistently better at handling semantic part consistency. ShapeFlow and MeshODE are better at matching the global shapes, although as motivated above they often overfit. NeuralCage is not competitive in either of the metrics, due to trading shape flexibility with higher rigidity by over-constraining.

4.3. Evaluation of point correspondence

We further evaluate the correspondences obtained by feature matching. This can be achieved by matching a source point to the target, and find the nearest point on target shape surface. The goal is not to achieve state-of-the-art performance, since correspondences is not our main focus. Rather, we want to show that implicit features inherently encode the correspondences across shapes to some extent, without explicit training or specialized network designs.

Setup and baseline method. We compare with Occupancy Flow (OccFlow) (Niemeyer et al., 2019). OccFlow reconstructs 4D human motion from D-FAUST dataset (Bogo et al., 2017) with 3D human motions such as punching and jumping jacks, preprocessed into sequences of 17 (3D) frames. OccFlow consists of two networks, an implicit 3D function *occupancy network* (OccNet) (Mescheder et al., 2019) for encoding the shape at the initial frame, and a *velocity network* to predict the flow or the correspondences over time, similar to that in ShapeFlow or MeshODE. By contrast, we extract correspondences only from the 3D implicit function of OccNet without a specific flow network, see Figure 15.

We use the official release of the code from OccFlow (Niemeyer et al., 2019) for the evaluation of the ℓ_2 loss of the correspondence as well as the implementation of OccNet. We do not use correspondence ground truth, nor do we need the velocity network with our method, which means we only need half of the components compared with OccFlow. Following Niemeyer et al. 2019, dense point cloud is used as input of the source shape (the first frame) and the target shape (the last frame) to discover correspondences between them. We match the implicit feature from OccNet, and find the nearest point on the shape for correspondence. See supplement material for more implementation details.

Results. We show the evaluation results in Table 2. When no supervision is available, the proposed method performs favourably. we are competitive to OccFlow with supervision, although we do not really focusing on correspondence. Our method and OccFlow cannot catch the performance of 3D-Coded (Groueix et al., 2018a) as a state-of-the-art method specifically for the task, trained with large amount of data and heavy augmentation. We conclude that feature in a standard implicit function encodes correspondence and can be extracted with feature matching, even without adopting a specific architecture design.

Table 2. Correspondences. The proposed method recovers inherent correspondence from the implicit features without explicit flow or correspondence functions or supervisions.

	Supervised	Cor. ℓ_2
Nearest Neighbour	×	0.374
Ours	×	**0.169**
OccFlow (Niemeyer et al., 2019)	✓	0.167
3D-Coded (Groueix et al., 2018a)	✓	**0.096**

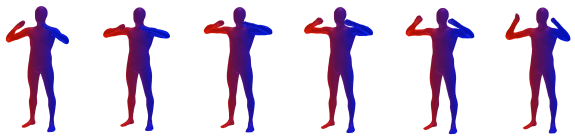


Figure 15. Joint reconstruction and correspondence of human motion sequence. Correspondence (in colour) is usually considered as unavailable with a single implicit network OccNet for 3D shape, as noted by Niemeyer et al. 2019. Our method extracts correspondence from matching features.

5. Conclusion

In this work, we propose to extend deep implicit functions, which normally give global representations, so that they are amenable to local feature matching. To do so, we start from a self-fitting learning paradigm for learning good shared representation space, upon which we can condition implicit functions. Then, to achieve local feature matching, we propose generalized correspondences by casting them as the trajectories from one shape to another where we have minimum change in the feature with interpolated latent code. By introducing locality to implicit functions, we can analyze what each layer in the implicit function learns, with earlier layers encoding coarse shapes and higher layers encoding finer details. What is more, locality enables shape deformations, where the resulting shapes can handle complex topologies and semantics inconsistencies.

Acknowledgement This research was supported in part by the SAVI/MediFor project and the EPSRC programme grant Visual AI EP/T028572/1. We thank the anonymous reviewers for helpful comments and suggestions.

References

Aberman, K., Liao, J., Shi, M., Lischinski, D., Chen, B., and Cohen-Or, D. Neural best-buddies: Sparse cross-domain correspondence. *ACM Transactions on Graphics (TOG)*, 37(4):1–14, 2018.

Arora, R., Basu, A., Mianjy, P., and Mukherjee, A. Understanding deep neural networks with rectified linear units. In *International Conference on Learning Representations*, 2018.

Atzmon, M. and Lipman, Y. Sal: Sign agnostic learning of shapes from raw data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2565–2574, 2020a.

Atzmon, M. and Lipman, Y. Sal++: Sign agnostic learning with derivatives. *arXiv preprint arXiv:2006.05400*, 2020b.

Atzmon, M., Haim, N., Yariv, L., Israelov, O., Maron, H., and Lipman, Y. Controlling neural level sets. *arXiv preprint arXiv:1905.11911*, 2019.

Bau, D., Zhou, B., Khosla, A., Oliva, A., and Torralba, A. Network dissection: Quantifying interpretability of deep visual representations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 6541–6549, 2017.

Ben-Israel, A. A newton-raphson method for the solution of systems of equations. *Journal of Mathematical analysis and applications*, 15(2):243–252, 1966.

Bogo, F., Romero, J., Pons-Moll, G., and Black, M. J. Dynamic faust: Registering human bodies in motion. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 6233–6242, 2017.

Chang, A. X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

Chen, R. T., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. Neural ordinary differential equations. In *Advances in neural information processing systems*, pp. 6571–6583, 2018.

Chen, Y., Mensink, T. E. J., and Gavves, E. 3d neighborhood convolution: Learning depth-aware features for rgb-d and rgb semantic segmentation. In *International Conference on 3D Vision*, 2019a.

Chen, Y., Hu, V. T., Gavves, E., Mensink, T., Mettes, P., Yang, P., and Snoek, C. G. Pointmixup: Augmentation for point clouds. *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

Chen, Z. and Zhang, H. Learning implicit fields for generative shape modeling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5939–5948, 2019.

Chen, Z., Yin, K., Fisher, M., Chaudhuri, S., and Zhang, H. Bae-net: Branched autoencoder for shape co-segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 8490–8499, 2019b.

Chibane, J., Alldieck, T., and Pons-Moll, G. Implicit functions in feature space for 3d shape reconstruction and completion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6970–6981, 2020.

Choy, C. B., Gwak, J., Savarese, S., and Chandraker, M. Universal correspondence network. *arXiv preprint arXiv:1606.03558*, 2016.

Dinh, L., Krueger, D., and Bengio, Y. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

Fan, H., Su, H., and Guibas, L. J. A point set generation network for 3d object reconstruction from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 605–613, 2017.

Gao, L., Yang, J., Wu, T., Yuan, Y.-J., Fu, H., Lai, Y.-K., and Zhang, H. Sdm-net: Deep generative network for structured deformable mesh. *ACM Transactions on Graphics (TOG)*, 38(6):1–15, 2019.

Genova, K., Cole, F., Sud, A., Sarna, A., and Funkhouser, T. Deep structured implicit functions. *arXiv preprint arXiv:1912.06126*, 2019.

Groueix, T., Fisher, M., Kim, V. G., Russell, B. C., and Aubry, M. 3d-coded: 3d correspondences by deep deformation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 230–246, 2018a.

Groueix, T., Fisher, M., Kim, V. G., Russell, B. C., and Aubry, M. A papier-mâché approach to learning 3d surface generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 216–224, 2018b.

Huang, J., Jiang, C. M., Leng, B., Wang, B., and Guibas, L. Meshode: A robust and scalable framework for mesh deformation. *arXiv preprint arXiv:2005.11617*, 2020.

Jiang, C., Huang, J., Tagliasacchi, A., Guibas, L., et al. Shapeflow: Learnable deformations among 3d shapes. In *Advances in Neural Information Processing Systems*, 2020a.

Jiang, C., Sud, A., Makadia, A., Huang, J., Nießner, M., and Funkhouser, T. Local implicit grid representations for 3d scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020b.

Kingma, D. P. and Dhariwal, P. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in neural information processing systems*, pp. 10215–10224, 2018.

Kurenkov, A., Ji, J., Garg, A., Mehta, V., Gwak, J., Choy, C., and Savarese, S. Deformnet: Free-form deformation network for 3d shape reconstruction from a single image. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 858–866. IEEE, 2018.

Liu, F. and Liu, X. Learning implicit functions for topology-varying dense 3d shape correspondence. *arXiv preprint arXiv:2010.12320*, 2020.

Lorensen, W. E. and Cline, H. E. Marching cubes: A high resolution 3d surface construction algorithm. *ACM siggraph computer graphics*, 21(4):163–169, 1987.

Maaten, L. v. d. and Hinton, G. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov): 2579–2605, 2008.

Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S., and Geiger, A. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4460–4470, 2019.

Michalkiewicz, M., Pontes, J. K., Jack, D., Baktashmotlagh, M., and Eriksson, A. Deep level sets: Implicit surface representations for 3d shape inference. *arXiv preprint arXiv:1901.06802*, 2019.

Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., and Ng, R. Nerf: Representing scenes as neural radiance fields for view synthesis. *arXiv preprint arXiv:2003.08934*, 2020.

Niemeyer, M., Mescheder, L., Oechsle, M., and Geiger, A. Occupancy flow: 4d reconstruction by learning particle dynamics. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 5379–5389, 2019.

Park, J. J., Florence, P., Straub, J., Newcombe, R., and Lovegrove, S. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 165–174, 2019.

Peng, S., Niemeyer, M., Mescheder, L., Pollefeys, M., and Geiger, A. Convolutional occupancy networks. *arXiv preprint arXiv:2003.04618*, 2020.

Rahaman, N., Baratin, A., Arpit, D., Draxler, F., Lin, M., Hamprecht, F., Bengio, Y., and Courville, A. On the spectral bias of neural networks. In *International Conference on Machine Learning*, pp. 5301–5310. PMLR, 2019.

Rezende, D. J. and Mohamed, S. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*, 2015.

Saito, S., Huang, Z., Natsume, R., Morishima, S., Kanazawa, A., and Li, H. Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2304–2314, 2019.

Sitzmann, V., Zollhöfer, M., and Wetzstein, G. Scene representation networks: Continuous 3d-structure-aware neural scene representations. In *Advances in Neural Information Processing Systems*, pp. 1121–1132, 2019.

Sorkine, O. and Alexa, M. As-rigid-as-possible surface modeling.

Wang, W., Ceylan, D., Mech, R., and Neumann, U. 3dn: 3d deformation network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1038–1046, 2019.

Wu, J., Zhang, C., Xue, T., Freeman, B., and Tenenbaum, J. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *Advances in neural information processing systems*, pp. 82–90, 2016.

Xu, Q., Wang, W., Ceylan, D., Mech, R., and Neumann, U. Disn: Deep implicit surface network for high-quality single-view 3d reconstruction. In *Advances in Neural Information Processing Systems*, pp. 492–502, 2019.

Xu, Y., Fan, T., Yuan, Y., and Singh, G. Ladybird: Quasi-monte carlo sampling for deep implicit field based 3d reconstruction with symmetry. In *European Conference on Computer Vision*, pp. 248–263. Springer, 2020.

Yang, G., Huang, X., Hao, Z., Liu, M.-Y., Belongie, S., and Hariharan, B. Pointflow: 3d point cloud generation with continuous normalizing flows. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 4541–4550, 2019.

Yang, Y., Feng, C., Shen, Y., and Tian, D. Foldingnet: Point cloud auto-encoder via deep grid deformation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 206–215, 2018.

Yi, L., Kim, V. G., Ceylan, D., Shen, I.-C., Yan, M., Su, H., Lu, C., Huang, Q., Sheffer, A., and Guibas, L. A scalable active framework for region annotation in 3d shape collections. *ACM Transactions on Graphics (ToG)*, 35(6):1–12, 2016.

Yifan, W., Aigerman, N., Kim, V. G., Chaudhuri, S., and Sorkine-Hornung, O. Neural cages for detail-preserving 3d deformations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 75–83, 2020.

Zeiler, M. D. and Fergus, R. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pp. 818–833. Springer, 2014.

Neural Feature Matching in Implicit 3D Representations: *Supplementary Material*

Yunlu Chen [1] **Basura Fernando** [2] **Hakan Bilen** [3] **Thomas Mensink** [4 1] **Efstratios Gavves** [1]

A. Limitation of global shape matching error

We further clarify the inherent limitation of the global shape distance metric for measuring the shape deformation quality in the presence of inconsistencies in topology or semantics between the source and the target shapes.

The global metrics assign a low error, when the two shapes overlap significantly, even if this implies an unnatural fitting. Figure 1 is a characteristic example. With our feature matching, the source arms can only find the closest points on the seat or the back of the target chair, leading to a larger global fitting error; while, with cross-fitting, the arms are forced very close to the seat and the back in an unnatural and distorted manner, which, however, reduces the whole shape error. By contrast, part-level metrics do not count such errors with inconsistent semantics, which makes more sense when shapes differ significantly.

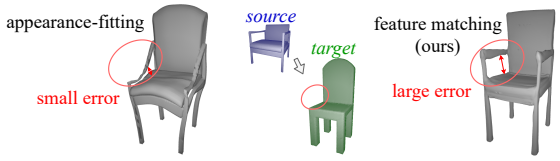


Figure 1. Limitation of global shape error metrics. Mesh deformed from source (blue) to target (green). The appearance-fitting result, generated with MeshODE (Huang et al., 2020), has a lower global matching error from the target shape at the arms (*e.g.* Chamfer distance), with an unnatural fitting.

B. Interpolating mesh deformation

Some additional visual results are provided in Figure 2 for chairs and Figure 3 for shapes in other categories, with deformed shapes from intermediate time steps. We show smooth and meaningful interpolated shapes as our method transfers the vertices of the shape mesh continuously.

[1]Informatics Institute, University of Amsterdam, the Netherlands [2]AI3, IHPC, A*STAR, Singapore [3]School of Informatics, University of Edinburgh, Scotland [4]Google Research, Amsterdam, the Netherlands. Correspondence to: Yunlu Chen <y.chen3@uva.nl>.

C. Implementation Details

Analysis and mesh deformation (§4.1 and §4.2). Our implementation is based on IM-Net or IM-AE from Chen & Zhang 2019 with the codebase available at `https://github.com/czq142857/IM-NET-pytorch`, which is an improved implementation from the authors. We use the preprocessed ShapeNet dataset (Chang et al., 2015) available with the codebase. For evaluation of part-aware measures in Table 1 in the main paper, we take semantic part segmentation annotation from ShapeNetPart (Yi et al., 2016) dataset preprocessed by Chen et al. 2019 (available at `https://github.com/czq142857/BAE-NET`). For each of the shape categories we take the first 200 shapes from the test split, and deform the first shape to the second, the third to the fourth, until the 199th shape to the 200th.

The implicit decoder is 7-layer MLP with Leaky-Relu activation except that the last layer to the output is linear. The negative slope set for Leaky-Relu is -0.02. There are no batch normalization or other normalization layers. The widths of each layers $\{w_l\}$ from input to output are 259-1024-1024-1024-512-256-128-1. Note, $w_0 = 259$ is for the 3-dim input coordinates concatenated with 256-dim latent code. The encoder is a 5-layer 3D ConvNet that takes voxels of shapes as input. Each conv layer is followed by an instance normalization and Leaky-Relu activation (with negative slope -0.02). The widths are 1-32-64-128-256-256 and so the output is a 256-dim latent code.

We train one network per shape category with the same architecture, following the coarse-to-fine progressive training scheme from Chen & Zhang 2019 with the resolutions at 16^3, 32^3, 64^3 respectively for 100, 200 and 800 epochs. The batch size is 32. Adam optimizer is used with learning rate 0.00005. The supervised ℓ_1 loss is used for training. Training of each model takes around 30 hours on one single Nvidia Geforce 1080 Ti.

The optimisation for feature matching uses the following settings: we use $\mathrm{d}t = 0.02$ for a total of 50 intermediate steps with latent code interpolation. We use $N = 3$ Newton's iterations at each time step. The regularisation factor λ is set as 0.01. The entire feature matching process from one source mesh with 3000 vertices to a target shape takes

around 60 seconds. The bottleneck of runtime is mostly at the calculation of Jacobian, which requires iterating over the dimension in the hidden layer feature w_l in modern deep learning frameworks PyTorch or TensorFlow.

Inherent correspondence evaluation (§4.3). We use the released code from OccFlow (Niemeyer et al., 2019) available at `https://github.com/autonomousvision/occupancy_flow` for the preprocessed D-FAUST dataset (Bogo et al., 2017), the evaluation of the ℓ_2 error of the correspondence as well as the implementation of OccNet (Mescheder et al., 2019). The velocity network component is not used.

The implicit function OccNet contains sequentially 5 residual blocks. Each block is with two fully-connected layers followed by ReLU activation, with residual connection from the input to the output of the block. In total, the implicit decoder as 10 fully-connected layers. All hidden layer widths are 256. The input has 259 dimensions and the output is a scalar occupancy probability. The encoder is a PointNet (Qi et al., 2017) that takes point coordinate inputs and output a 256-dim latent code. Following the original setup from Niemeyer et al. 2019 in this evaluation, all vertices of the human shape are taken as the point inputs.

We train the OccNet to reconstruct human shapes with all poses from all training sequences, unlike Niemeyer et al. 2019 that trains the reconstruction network with only the poses in the first frame of each sequence. Other training details follow the original implementation. The batch size is 16. Adam optimizer is used with learning rate 0.0001. Training uses cross-entropy classification loss on the binary occupancy probability and takes around 5 days for 3000 epochs.

We match the last hidden layer implicit feature from OccNet as it outcomes points that are most close to the target shape surface, and find the nearest point on the target shape surface for correspondence. We use a total of 8 intermediate steps with latent code interpolation. The ℓ_2 error of the correspondence is evaluated on the test split with the author's code. For the number of Newton's iterations at each time step we use $N = 4$. The regularisation factor λ is set as 0.001.

More architectural and training details can be referred to the original implementations, since the architecture and training processes highly rely on the existing standard implicit function methods.

References

Bogo, F., Romero, J., Pons-Moll, G., and Black, M. J. Dynamic faust: Registering human bodies in motion. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 6233–6242, 2017.

Chang, A. X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

Chen, Z. and Zhang, H. Learning implicit fields for generative shape modeling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5939–5948, 2019.

Chen, Z., Yin, K., Fisher, M., Chaudhuri, S., and Zhang, H. Bae-net: Branched autoencoder for shape co-segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 8490–8499, 2019.

Huang, J., Jiang, C. M., Leng, B., Wang, B., and Guibas, L. Meshode: A robust and scalable framework for mesh deformation. *arXiv preprint arXiv:2005.11617*, 2020.

Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S., and Geiger, A. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4460–4470, 2019.

Niemeyer, M., Mescheder, L., Oechsle, M., and Geiger, A. Occupancy flow: 4d reconstruction by learning particle dynamics. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 5379–5389, 2019.

Qi, C. R., Su, H., Mo, K., and Guibas, L. J. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 652–660, 2017.

Yi, L., Kim, V. G., Ceylan, D., Shen, I.-C., Yan, M., Su, H., Lu, C., Huang, Q., Sheffer, A., and Guibas, L. A scalable active framework for region annotation in 3d shape collections. *ACM Transactions on Graphics (ToG)*, 35(6):1–12, 2016.

Figure 2. Mesh deformation interpolation over time, chairs. Every two rows are a group of examples. The blue mesh is the source shape and the green mesh is the target shape. The odd row shows interpolation from source to target (*left* to *right*), and the even row from target to source (*right* to *left*).

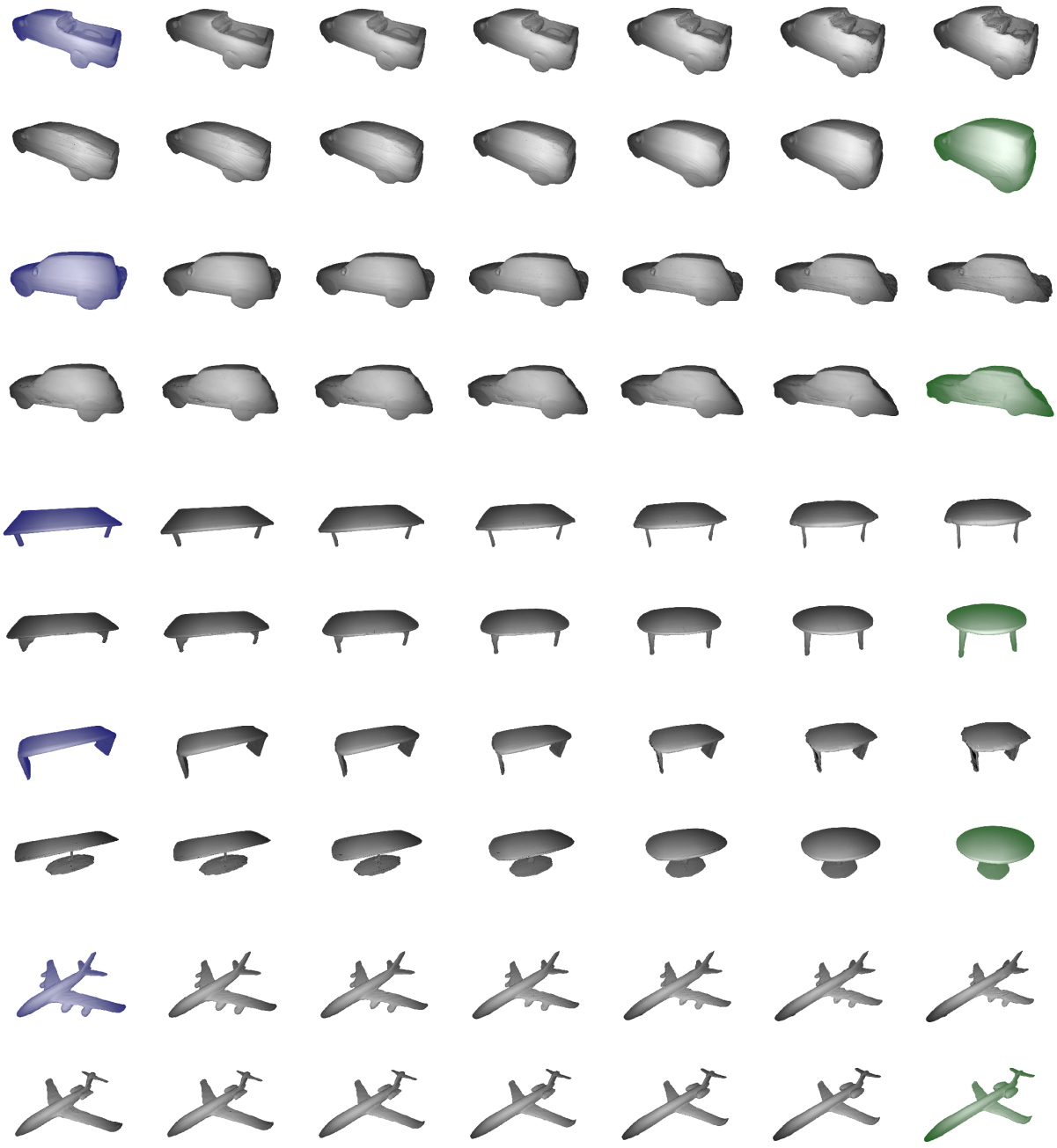


Figure 3. Mesh deformation interpolation over time, other categories. Every two rows are a group of examples. The blue mesh is the source shape and the green mesh is the target shape. The odd row shows interpolation from source to target (*left* to *right*), and the even row from target to source (*right* to *left*).