Neural Convolutional Surfaces

Luca Morreale[1*] Noam Aigerman[2] Paul Guerrero[2] Vladimir G. Kim[2] Niloy J. Mitra[1,2]
[1]University College London [2]Adobe Research

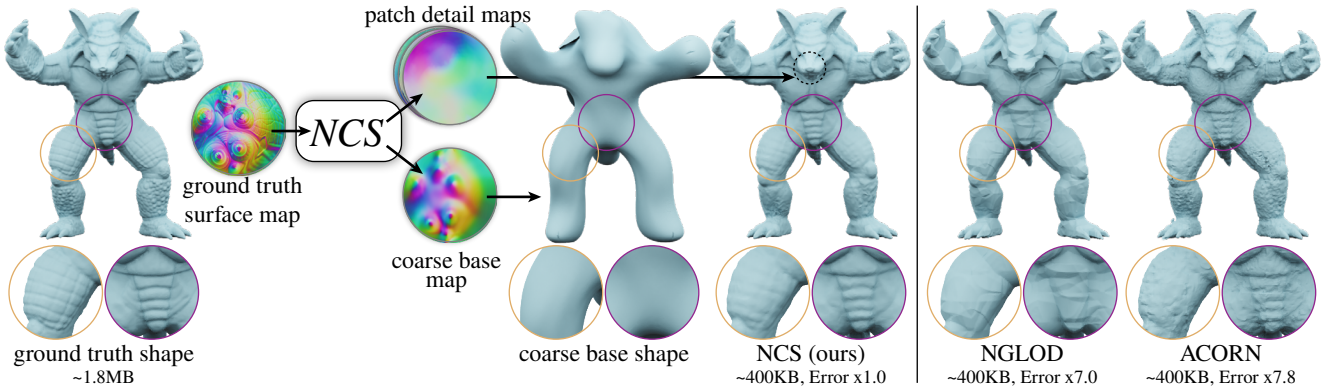


Figure 1. Neural Convolutional Surfaces (NCS) can faithfully represent a given ground-truth shape while disentangling coarse geometry from fine details, leading to a highly-accurate representation of the shape. Compared to other state-of-the-art methods neuralLOD [34] and Acorn [23], NCS achieves significantly more accurate results for the same memory footprint.

Abstract

This work is concerned with a representation of shapes that disentangles fine, local and possibly repeating geometry, from global, coarse structures. Achieving such disentanglement leads to two unrelated advantages: i) a significant compression in the number of parameters required to represent a given geometry; ii) the ability to manipulate either global geometry, or local details, without harming the other. At the core of our approach lies a novel pipeline and neural architecture, which are optimized to represent one specific atlas, representing one 3D surface. Our pipeline and architecture are designed so that disentanglement of global geometry from local details is accomplished through optimization, in a completely unsupervised manner. We show that this approach achieves better neural shape compression than the state of the art, as well as enabling manipulation and transfer of shape details. Project page http://geometry.cs.ucl.ac.uk/projects/2022/cnnmaps/.

1. Introduction

Triangle meshes have been the most popular representation across much of geometry processing since its early stages, however research has been devoted to devising novel representations of geometry to circumvent many of the shortcoming of triangular meshes. Lately, the rising prominence of deep learning has lead researchers to investigate ways to represent shapes via neural networks. While the immediate use of neural networks in this context is to represent entire shape spaces by using the same set of weights to decode any shape from a shared latent space, other methods use a shape-specific set of weights to represent a specific instance. This approach captures geometric detail efficiently and accurately and creates outputs that are on par with existing 3D models, while holding novel properties not attainable with surface meshes, such as differentiability. These neural representations for shape instances were demonstrated to be useful in geometry processing applications such as efficient rendering [34], level of details [23], surface parameterization, and inter-surface mapping [26].

The choice of the shape representation, and of the neural network's architecture, plays a critical role in how efficiently the capacity of the network is utilized. Existing

*Partially worked on the project during internship at Adobe Research.

representations usually use MLPs to model the shape as a function that maps points either from a 2D atlas to the surface [26] or points in a 3D volume to an implicit function such as a distance field [27]. The disadvantage of these architectures is that they entangle geometric details and overall shape structure, and do not have a natural mechanism to reuse the network weights to represent repeating local details, as Convolution Neural Networks (CNNs) achieve on images. Some methods indeed opt to use 2D images to represent geometry [31], however those exhibit finite resolution and hence cannot model surfaces with details in subpixel resolution. Alternatively, instead of a single global MLP, some prior techniques leverage repetitions by breaking the shape into smaller 3D voxels, each represented by an SDF function [23], however, these representations do not account for the fact that surface details are usually aligned with the surface, and thus, are less effective at representing local geometric textures that flow with the shape.

In this paper, we set to define a novel representation that achieves separation of local geometric details ("texture") from the global coarse geometry of the model, and thus leads to the reuse of network weights for repeating patterns that change their orientation with the surface. We achieve this by considering the standard atlas-based representation as in [15,26], but encode a surface as combination of a *coarse surface*, defining the general, coarse structure of the shape, represented via an MLP, along with an associated fine *detail map*, which adds geometric texture on top, represented via a CNN, which defines a continuous map of offsets. The geometric details are added to the coarse geometry either along its normal directions, or as general displacement vectors. Since the local displacement details are expressed with convolutional kernels, they can effectively be reused across similar regions of the surface. We call this hybrid representation *neural convolutional surfaces*.

This novel architecture enables the network to disentangle the fine CNN representation from the coarse MLP representation, in a completely *unsupervised* manner, i.e., without the need to supervise the split explicitly during fitting. We show that the inductive bias in our designed architecture leads to automatic separation of the shapes into coarse base shapes and reusable convolutional details, see Figure 1.

We evaluate our method on a range of complex surfaces and explore the associated tradeoff between representation quality and model complexity. We compare against a set of state-of-the-art alternatives (e.g., NeuralLod [34], ACORN [23], Neural Surface Maps [26]) and demonstrate that our model achieves better accuracy at a fraction of the model-complexity – between 1% to 10% parameters. Additionally, we demonstrate that the convolutional aspect of the representation makes it *interpretable*, leading to applications including detail modification within individual shapes and details transfer across different models – see Figure 7.

2. Related Works

Our method follows a long line of works on using neural shape representation, as well as the more specific recent trend of representing a single shape via a neural network. We also incorporate concepts on level-of-detail representation of geometry. Next, we review the existing literature on all these fields.

Neural representation of shape spaces. A large number of generative neural representations for 3D shapes have been proposed in recent years, such as voxel grids [5, 7, 12, 18], point clouds [1, 33], meshes [6], or unions of deformable primitives [11]. With these representations, the number or scale of the discrete elements has a great effect on the method's ability to represent fine details. To tackle this challenge, some methods model shapes as neural functions, allowing the network to optimize how to allocate its capacity: implicit models represent the shape as a map from a volume to a signed distance field [27] or occupancy [24] values. These can be further improved by enforcing satisfaction of the Eikonal equation [2, 3, 14] or using an intermediate meta-network for faster reconstruction [17]. Since most traditional computer-graphics pipelines require surface models, atlas-based representations offer another prominent alternative. These techniques model a shape as an atlas, i.e., a function that maps 2D points to positions in 3D [15, 35]. Improved versions of these methods include adding optimizing for low-distortion atlases [4], learning task-specific geometry of 2D domain [9], or forcing the surface to agree with an implicit function [28]. [31] use Geometry Images [16] and encode geometry as 2D images in order to perform deep learning tasks such as segmentation.

Fitting networks to shapes. While the previously-mentioned works are concerned with training networks to represent an arbitrary shape out of a collection, similar techniques can be employed to represent one, specific shape via one network, fitted to that specific shape, by optimizing the network's weights on that single example. This has several advantages; for example, Neural Surface Maps [26] use a fully connected neural network to represent a parameterized shape as an $R^2 \rightarrow R^3$ map and show that the achieved reconstruction is much more accurate and can represent fine details than a network trained to reconstruct multiple shapes. Furthermore, one can concatenate these maps and optimize them to achieve low-distortion surface-to-surface maps. Representing a shape via a neural network enables compression, of, e.g., implicit surfaces [8,34]. One can also optimize from indirect observations, such as multi-view appearance [25, 32].

Many of these works are concerned with preserving details. ACORN [23] solves an integer-program to optimally

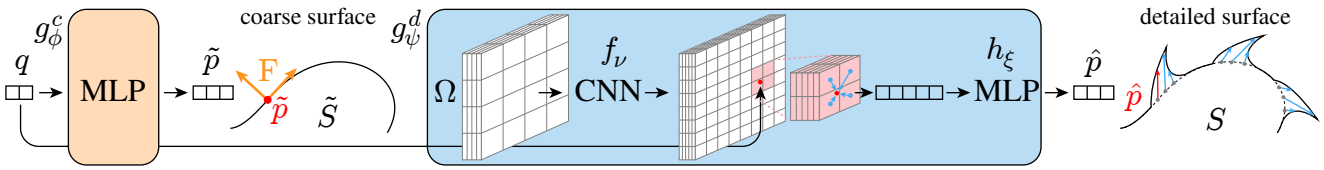


Figure 2. **Overview of Neural Convolutional Surfaces.** Surfaces S are represented by two models, a coarse model g^c_ϕ that encodes a coarse version $\tilde{S}$ of the surface and allows computing a local reference frame $\mathbf{F}$, and fine model g^d_ψ that encodes geometric detail as offsets $\hat{p}$ from the coarse surface, in coordinates of the local reference frame.

subdivide an SDF volume to allocate more parameters to the regions holding more details. Yifan *et al.* propose siren-based implicit displacement fields [36] restricted to the normal direction, which separate geometric details from the coarse shape structure. In our work, we use convolutional filters applied on the 2D domain to model displacement from the atlas-based surface representation. Using convolutional filters enables us to rely on repeating signals (which could be encoded in those filters applied throughout the surface). By modeling our displacements relative to the parameterized surface atlas, we can learn also learn geometric textures that are aligned with the surface.

Level-of-detail in geometry processing. Many prior optimization techniques focus on simplifying meshes with high polygon count. Different simplification operators [10, 20, 30] with geometry-based and appearance-based [21] objective functions (see [22] for a thorough overview). In this work, instead of simplifying an existing mesh, we propose to use a neural network to represent the surface. One can control the amount of geometric detail by simply changing the capacity of the neural network. Our representation can also effectively represent geometric texture by using neural convolutional surfaces that utilize shared kernels for repeating patterns.

3. Neural Convolutional Surfaces

Our goal is to represent a 3D surface $S \subset \mathbb{R}^3$ with a neural network g_θ, so that the parameters θ compactly encode the surface. We assume to have computed a bijective parameterization of the surface into the unit circle in 2D (we use SLIM [29] in all our experiments). We consider this parameterization as a map from the unit disk to the surface $D : \{\mathbf{x} \in \mathbb{R}^2 : \|\mathbf{x}\| \leq 1\} \rightarrow S$, mapping a 2D point q to a 3D point $p \in S$ on the surface. Following [26], we fit the network g_θ to approximate this function: $g_\theta(q) \approx s(q)$.

We propose *Neural Convolutional Surfaces* (NCS) as an accurate and compact neural approach to model s. The NCS g_θ consists of two modules: (i) a coarse module, g^c_ϕ, based on a standard MLP-based model, which aims to approximate the coarse shape of the surface; and (ii) a fine module, g^d_ψ, which adds detailed displacements to the coarse surface. Internally, this fine module is comprised of a CNN component generating a grid of codes, which are interpolated and fed to an MLP in order to get the final fine offset vector. These two modules are then added to get the final map from 2D to 3D,

$$g_\theta(q) := g^c_\phi(q) \; + \; \mathbf{F}_{g^c_\phi}(q) \; g^d_\psi(q), \tag{1}$$

where $\mathbf{F}_{g^c_\phi}$ is a rotation to the local reference frame at a coarse surface location and $\theta = (\phi, \psi)$. Please refer to Figure 2 for an overview. These two models are trained *jointly*, with only the target mapping s as supervision. Intuitively, the coarse-fine separation allows the fine model g^d_ψ to expend capacity only on the high-frequency geometric texture of the surface. Repeating structures in this geometric texture can be modelled efficiently by the shared weights of the convolution kernels.

We describe the coarse model next in Section 3.1, the fine model in Section 3.2, the local reference frame in Section 3.3, and the overall training setup in Section 3.4.

3.1. Coarse Model

The coarse model represents a smooth, coarse version of the ground truth surface S, which serves as a basis, by providing a well defined local coordinate frame at any surface point, for applying the detailed geometric texture predicted by the fine model.

Our coarse model approximates a coarse version of s with a low-capacity MLP, which takes as input a 2D coordinate q in the unit disk and outputs the corresponding point $\tilde{p}$ on the coarse surface: $\tilde{p} = g^c_\phi(q)$. The limited capacity of the MLP ensures that only a coarse approximation of the surface is modelled, while the MLP's architecture enforces smoothness of the coarse surface, thereby delegating reconstruction of sharp, fine, and repeating structures to the fine model. Note that we do *not* use intermediate supervision for the coarse surface $\tilde{p}$.

3.2. Fine Model

The fine model represents a detailed geometric texture that is applied to the coarse surface g^c_ϕ. The fine model is designed to first generate a high-resolution 2D grid of codes, and then interpolate the codes and map the interpolated code

to 3D displacement vectors via a small MLP. This is facilitated by three components: (i) we keep a low-resolution input grid of learned features (i.e., a low-resolution 2D image with multiple channels) $\Omega \in \mathbb{R}^{D_0 \times H_0 \times W_0}$; (ii) a CNN f_ν transforms and up-samples(2x per layer) the feature map into a high-resolution 2D grid of codes; and (iii) for a given 2D query point q that falls into a grid cell, the 4 grid codes at that cell's corners are interpolated, and a small 2-layer MLP h_ξ finally maps the interpolated code into a local displacement vector:

$$g^d_\psi(q) = h_\xi\big(f_\nu(\Omega)|_q\big), \tag{2}$$

where $X|_q$ denotes bilinear interpolation of the image X at q (assuming pixel coordinates in $[-1,1]^2$) and $\psi = (\Omega, \nu, \xi)$. Intuitively, the feature map Ω stores coarse information about the geometric surface details that is refined by the CNN f_ν, introducing learned priors stored in the shared CNN kernels. The interpolation for a given sample q is performed in feature space as opposed to 3D space, and followed by a small MLP to allow for complex non-linear interpolating surfaces between the pixels of the CNN output. For details of the model architectures, please refer to the supplementary material.

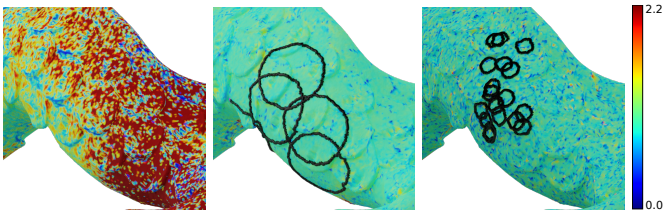
Patches. Directly discretizing the full parameter domain of s on a grid has two drawbacks: (i) The resolution of the pixel grid processed by the CNN would need to be very large to accurately model small geometric detail; and (ii) the initial mapping s may exhibit significant area distortion, i.e., there can be a large difference between scale factors in different regions of the mapping, making a single global resolution inefficient.

To avoid these problems, we split the surface S into small overlapping patches $R_0, \ldots, R_m$ with each having a separate local parameterization $r_i : [-1,1]^2 \to R_i$. Since each patch only covers a small region of the surface, the distortion within each individual mapping is small, and the resolution of the pixel grid in each patch can be lower, without compromising geometric detail. Further, since we assume our fine model is CNN-based, we can learn and reuse the same CNN kernels for each one of our patches without harming our goal of training the fine model to represent repeated geometry, now simply split into different patches.

Patch-based model. Once we decompose the input into multiple patches, Equation 2 can be generalized as:

$$g^d_\psi(q) = \frac{1}{\sum_i w_i(q)} \sum_i w_i(q)\; h_\xi\big(f_\nu(\Omega_i)|_{l_i(q)}\big) \tag{3}$$
$$\text{with } w_i(q) = \max\big(0, -d^b_i\big(l_i(q)\big)\big),$$

where $l_i(q)$ maps the global parameters q to the local patch parameterization r_i, and the contribution of overlapping patches to a point q is weighed as a function of the signed distance d^b_i from the boundary of the patch to $l_i(q)$ in the parameter domain of the patch. We train the same MLP and CNN (with shared weights ν, ξ) over all patches, thereby encouraging the CNN to reuse filters across patches; the only different parameter between different patches is the coarse input feature grid map Ω_i that is assigned to each patch.



(a) 1 patch **(b)** 700 patches **(c)** >2000 patches

Figure 3. **Effect of number of patches.** Representing a shape with a single patch, results in high distortion in the underlying paramaterization (red in (a)). Decomposing the base domain into many small patches (c) leads to much lower per-patch distortion. However, this comes at the cost of a much increased memory budget to represent the shape. Medium size patches (b) strikes a balance by reducing the per-patch distortion while still being light in the final memory requirement.

3.3. Local Reference Frame

The output of the fine model $g^d_\psi(q)$ is a displacement vector $\hat{p}$ of the coarse surface at $\tilde{p} = g^c_\phi(q)$. Naively adding $\tilde{p} + \hat{p}$ would render the displacements sensitive to the local orientation of the coarse surface. Hence, to encourage consistency between the displacements, we define them in a local coordinate frame $\mathbf{F}_{g^c_\phi}$ aligned to the tangent space of the coarse surface, measured via the Jacobian of the coarse surface mapping, as:

$$\mathbf{J}^c := [\mathbf{J}^c_u, \mathbf{J}^c_v] = \left[\frac{\partial g^c_\phi}{\partial q_u}, \frac{\partial g^c_\phi}{\partial q_v}\right], \tag{4}$$

where q_u and q_v are the two coordinates of the global parameterization. The local coordinate frame as a function of q is then defined as:

$$\mathbf{F}_{g^c_\phi} := [n,\ \mathbf{J}^c_u,\ n \wedge \mathbf{J}^c_u] \text{ with } n = \mathbf{J}^c_u \wedge \mathbf{J}^c_v, \tag{5}$$

where n returns the normal of the coarse surface and $\wedge$ denotes the cross product. Note that, although not shown in the expressions above, each of the axis vectors are normalized to be of unit length.

3.4. Training

As discussed earlier, the coarse and fine modules together define a neural network mapping 2D points to 3D.

Table 1. Comparison of shape representations with 100K parameters wrt Chamfer distance ↓. Numbers are multiplied by 10^3.

	[34]	Sparse [34]	[36]	Ours
Armadillo	1.95	1.34	1.06	**0.54**
Bimba	2.30	2.07	2.09	**1.04**
Dino	1.70	1.55	2.55	**1.48**
Dragon	1.57	1.12	0.62	**0.57**
Grog	2.06	1.06	**0.81**	1.28
Seahorse	1.26	1.15	-	**0.44**
Elephant	4.06	**2.24**	3.93	2.49
Gargoyl	6.30	-	8.51	**2.29**

We fit this combined map to the ground truth surface mapping s via an L2 loss:

$$\mathcal{L}_{\text{joint}} = \int_{Q_S} \|g_\theta(q) - s(q)\|_2^2 \, dq, \tag{6}$$

where Q_S is the subregion of the global parameter domain Q that maps to the surface S.

4. Experiments

We now detail the various experiments we performed. We compare our reconstruction quality to other methods, move on to ablations, and finish with a additional applications enabled by our representation.

Comparison. We compare our method against three state-of-the-art methods for neural shape representations: (i) ACORN [23]; (ii) NGLOD [34], and (iii) Neural Surface Maps (NSM) [26]. In the supplemental, we provide additional comparisons. We compare the methods on a variety of shapes with different amount and type of geometric details (see Table 2). In the main paper we provide comparisons on 5 different shapes, with additional shapes in the supplementary material. Note that all models are scaled to a unit sphere. We evaluate performance along two main axes: (i) The representation *accuracy* is measured by the Bidirectional Chamfer distance, which computes the distance between output and ground truth surfaces. Lower values indicate more accurate representations. (ii) The *memory cost* is measured by the number of parameters required by a representation. Typically parameters are represented as 32 bit floats, so multiplying by 4 gives the number of bytes.

As can be seen in Figure 6, for the same number of parameters, our method achieves significantly higher accuracy than the state of the art. Furthermore, in all cases but one (Lucy), our method's accuracy exceeds all others methods even when using 10 times less parameters. As shown in Figure 4, our method preserves detail such as the dragon's scales and Bimba's braids much more accurately than both ACORN and NGLOD. Furthermore, both competing methods exhibit discretization artifacts or noise, while our method provides a smooth, artifact-free surface.

Table 2. Evaluation and distribution of the network parameters between the different modules in our architecture, for various models tested in the paper, of size 100K. The latent codes use the majority of parameters, while the other components are self-contained.

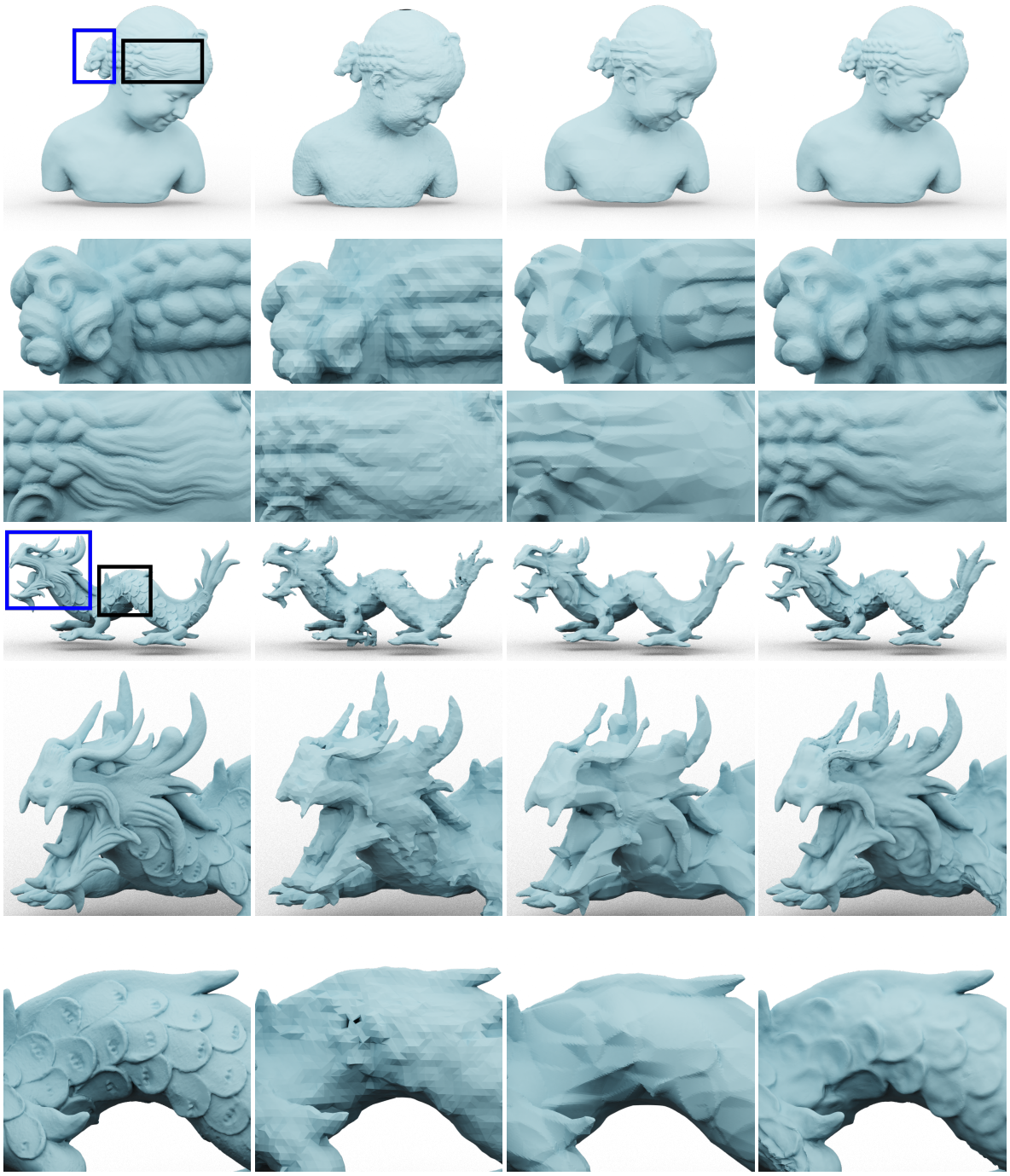
			Coarse	Fine			TOT.
	#V	#F	MLP	Code	CNN	MLP	params
Armadillo	172K	346K	13K	93K	6K	467	113K
Bimba	50K	100K	9K	115K	6K	467	130K
Lucy	877K	1753K	13K	96K	4K	435	114K
Dino	26K	51K	4K	120K	6K	467	130K
Dragon	451K	902K	13K	99K	6K	467	119K

We offer additional comparisons against the concurrent method IDF [36] in Table 1. Further visual comparison is offered in the supplementary material. IDF [36] fails to correctly describe the iso-surface of Dino and Elephant. Note that iso-surfaces produced by IDF include excess geometry that wraps around the shape. To offer a numerical comparison, we manually removed the additional excess surface. IDF completely fails to represent the Seahorse and hence we leave a '-' in the table.

Ablation. Figure 5 evaluates the necessity of various components in our framework: (i) *scalar displacements*: we restrict the fine network to only apply scalar displacements along normal directions, instead of displacement vectors as used in the full model. This significantly hinders the ability of the fine model to add details on top of the coarse model, leading to an oversmoothed result that resembles the coarse model; (ii) *PCA only*: we remove the coarse model and use only the fine model. For the local reference frame **F** required by the fine model, we pre-compute and store the PCA frame of the ground truth patch. This causes the fine model to spend its capacity on re-creating the coarse geometry, and as a result, artifacts and ripples appear in the reconstruction.

Feature enhancement. Since the CNN encodes local geometric detail, we can edit geometric detail by manipulating the CNN's feature maps. In Figure 8 we perform feature enhancement on a few models by scaling up the CNN output, before feeding it into the MLP of the fine model. Similarly, we can perform smoothing, by scaling down the same features.

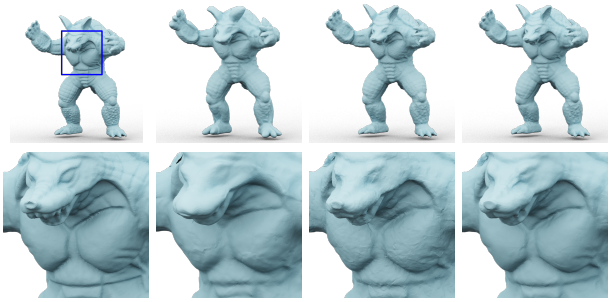
Detail transfer. Our disentanglement of coarse and fine detail enables us to perform detail transfer, similarly to IDF [36]. Figure 9 show results of our method, transferring creases from one model to the other. Similarly to IDF, we achieve this by training one coarse network for the source



(a) Ground truth **(b)** ACORN [23] **(c)** NGLOD [34] **(d)** NCS (ours)

Figure 4. **Surface representation.** The reconstruction quality of our method, compared with ACORN [23] and NeuralLOD [34] for two models, using the same number of network parameters on each method model size (100K parameters in this example). Our result exhibits higher accuracy and reconstruction of fine details, while not exhibiting artifacts such as artificial edges or aliasing.

shape and one for the target shape. The fine network is trained to accurately fit the source shape. We then transfer the details to the target shape with a forward pass using the source global and local parameterization.



(a) GT **(b)** normal **(c)** PCA **(d)** Ours

Figure 5. Ablation study. (a) the ground truth model; (b) adding scalar displacement along the normal direction (to a learned base coarse model) yields smoothed-out results; (c) adding displacement vectors to a per-patch canonical coordinate frame (established using patch's PCA axes) yields artifacts and surface ripples; (d) our reconstruction is sharp and does not exhibit artifacts.

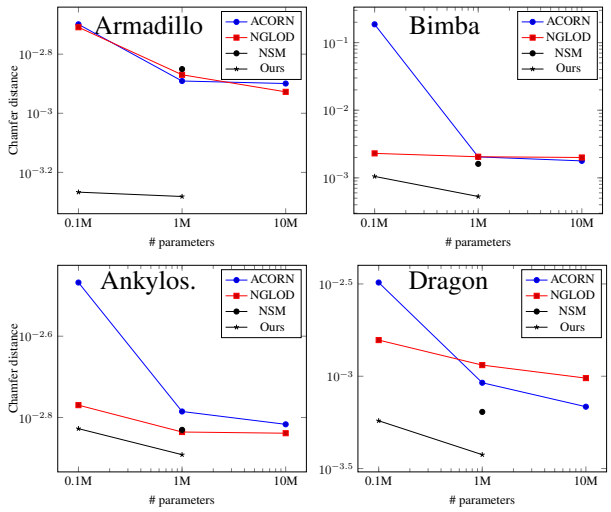


Figure 6. Reconstruction quality versus Model complexity for different models. Note that our method achieves better reconstruction quality with significantly lower memory footprint. Note that values are reported in logscale. Bottom row shows our results. Given our reconstructions, we do not use more than 1M parameters.

Interpretability of the kernels. The use of a CNN in the fine branch of our network's architecture leads to interpretable kernels, i.e., specific kernels react to specific details of the geometry. In Figure 7 we show an example in which we select a region on the model, find features that are activated strongly in that region, and then highlight other areas in which those features are activated. As we can see, the features associated with one of the Dino's spikes also affect all other spikes. This shows that our kernels are reused across the model, explaining our network's ability to represent detailed models with a smaller number of network parameters than previous methods. This may also lead to future work

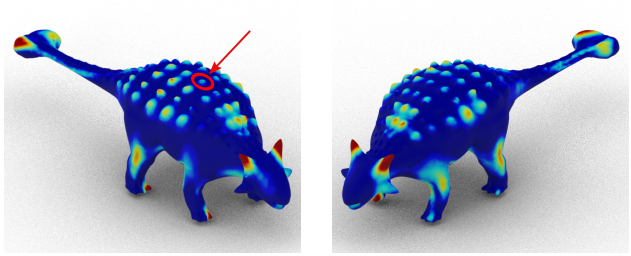
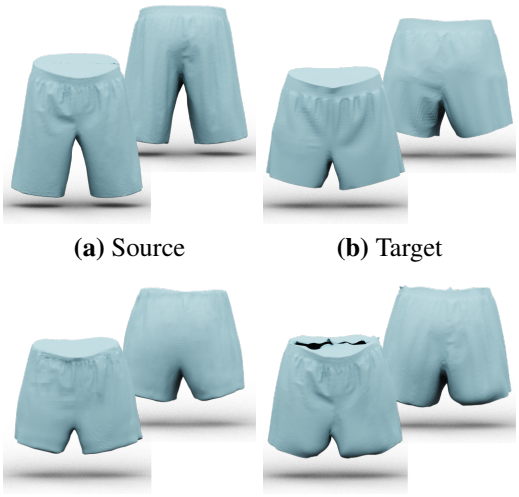


Figure 7. Our method yields interpretable convolutional kernels: we select one spike (highlighted) on the dino, identify the CNN features that are strongly active in its region, and then identify other regions where the same features are active. High correlation (hotter colors) implies regions with similar geometric details.

Figure 8. **Sharpening and smoothing.** Our NCS naturally decomposes shapes into coarse shapes and fine details. Boosting or suppressing the fine details and reconstructing the shapes, naturally results in exaggeration or smoothing of surface features.

where we use the same kernels to a larger collection of surface, to learn more specific and robust features.

Implementation details. Patches are found by randomly sampling patch centers c_i on S and selecting all points within a geodesic radius ρ: $R_i = \{p \mid d^{\text{geo}}(p, c_i) le \rho\}$. We use an iterative approach: after creating a patch, all points inside the patch are marked as forbidden for the following patch centers with a probability η, which controls the



(a) Source **(b)** Target

(c) IDF [36] **(d)** Ours

Figure 9. **Detail transfer.** Our architecture intrinsically decomposes shapes into coarse base models and associated geometric details, which allows us to transfer learned details from one model to another base shape. In this case, from one pair of pants (a), to another pair of pants. The fitter coarse model for each of the two pairs is shown at the top. Here we compare our detail transfer results with those of a concurrent work [36]. To transfer details we replace the source coarse model with the target coarse one, and reconstruct the shape. Note, this is possible because the global geometry images, source and target, are aligned. In case of misalignment, an inter-surface map between the coarse models could be computed using, e.g., [26].

amount of overlap between patches. In our experiments, we set $\eta = 0.5$ and $\rho = 0.04$ times the maximum extent of S along any coordinate axis. Figure 3 shows the effect of the choice of number of patches.

In terms of training performance, we observed that we can achieve better results with a training schedule that starts by warming up the coarse model before slowly ramping up training of the fine model:

$$\mathcal{L} = (1-\lambda)\mathcal{L}_{\text{joint}} + \lambda\mathcal{L}_{\text{reg}} \tag{7}$$
$$\text{with } \mathcal{L}_{\text{reg}} = \int_{Q_S} \|g^c_\phi(q) - s(q)\|_2^2 \, dq.$$

We start with $\lambda = 1$ and progressively decrease $\lambda \to 0$ over the warm-up phase, which lasts for 100K iterations. At the same time, we increase the learning rate of the fine model and decrease the learning rate of the coarse model over the warm-up phase: the learning rate of the coarse model follows a cosine annealing schedule [19], down to a minimum learning rate of 0 at the end of the warm-up phase, while the learning rate of the fine model is set to $1e-4$ minus the coarse learning rate. The total number of iteration varies based on the complexity of the model, *e.g.*, between 800K to 1.4M iterations, using the RMSProp optimizer [13].

5. Conclusions

Neural convolutional surfaces enable faithfully representing a given surface via a neural network, with higher accuracy and a smaller network capacity (e.g., 10-80x) compared to multiple state-of-the-art alternatives. Key to our method is an inductive bias in the network architecture that results in a split representation, with an MLP producing a coarse abstraction of the shape, and a fine detail CNN-like layer that adds geometric displacements based on the local reference frame the local UV charts.

We demonstrate that this coarse-fine disentanglement emerges naturally, without any intermediate supervision, and leads to the fine module reusing its convolutional kernels, which in turn enable meaningful geometric operations like mesh smoothing and feature exaggeration.

Limitations and Future Work. While the CNN based architecture leads to significant compression by reusing the kernels across object-centric local coordinate frames, the kernels themselves are still regular, 2D Euclidean image kernels, and hence are not rotationally invariant, as they ideally should be to handle geometry. This hinders perfect reuse of kernels across the shape, e.g., in cases of asymmetric features that are reoriented on the shape (rotated on the local tangent space), for example, the scales on the dragon. Furthermore, the kernels cannot be reused to capture local *deformations* of the underlying geometric details. Lastly, we note that in some cases our pipeline, in absence of intermediate supervision, may associate coarse structures as fine, e.g., Lucy's (the angel) torch in Figure 8 is reconstructed mainly by the fine module of our networks, and as a result is reduced in size when the details are smoothed.

While we focused on faithfully representing individual shapes for the scope of this work, we intend to followup the next goal, of capturing *distributions* of shapes. We observe that geometric details is often reused across shapes and hence we can aspire to learn a universal dictionary of CNN detail kernels, that can then be applied across a diverse set of shapes, where the global structures are captured by shape-specific coarse abstractions. Such a universal dictionary of local geometric details will be a close analog of low level features learnt on images (e.g., VGG features learnt using ImageNet), which, in turn, will enable both manipulation or transfer of details, as well as compression of shapes with a fixed universal shape-dictionary.

Acknowledgements. LM was partially supported by the UCL Centre for AI and the UCL Adobe PhD program. This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 956585.

References

[1] P. Achlioptas, O. Diamanti, I. Mitliagkas, and L. J. Guibas. Learning representations and generative models for 3d point clouds. *ICML*, 2018. 2

[2] M. Atzmon and Y. 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. 2

[3] M. Atzmon and Y. Lipman. SAL++: Sign agnostic learning with derivatives. *arXiv preprint arXiv:2006.05400*, 2020. 2

[4] J. Bednarik, S. Parashar, E. Gundogdu, M. Salzmann, and P. Fua. Shape reconstruction by learning differentiable surface representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4716–4725, 2020. 2

[5] A. Brock, T. Lim, J. M. Ritchie, and N. Weston. Generative and discriminative voxel modeling with convolutional neural networks. *CoRR*, 2016. 2

[6] A. Dai and M. Nießner. Scan2mesh: From unstructured range scans to 3d meshes. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2019. 2

[7] A. Dai, C. R. Qi, and M. Nießner. Shape completion using 3d-encoder-predictor cnns and shape synthesis. *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017. 2

[8] T. Davies, D. Nowrouzezahrai, and A. Jacobson. Overfit neural networks as a compact shape representation, 2020. 2

[9] T. Deprelle, T. Groueix, M. Fisher, V. G. Kim, B. C. Russell, and M. Aubry. Learning elementary structures for 3d shape generation and matching. *arXiv preprint arXiv:1908.04725*, 2019. 2

[10] M. Garland and P. Heckbert. Surface simplification using quadric error metrics. *Proceedings of the ACM SIGGRAPH Conference on Computer Graphics*, 1997, 07 1997. 3

[11] K. Genova, F. Cole, D. Vlasic, A. Sarna, W. T. Freeman, and T. Funkhouser. Learning shape templates with structured implicit functions. In *ICCV*, 2019. 2

[12] R. Girdhar, D. F. Fouhey, M. Rodriguez, and A. Gupta. Learning a predictable and generative vector representation for objects. *CoRR*, abs/1603.08637, 2016. 2

[13] A. Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013. 8

[14] A. Gropp, L. Yariv, N. Haim, M. Atzmon, and Y. Lipman. Implicit geometric regularization for learning shapes. *arXiv preprint arXiv:2002.10099*, 2020. 2

[15] T. Groueix, M. Fisher, V. G. Kim, B. C. Russell, and M. 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. 2

[16] X. Gu, S. J. Gortler, and H. Hoppe. Geometry images. In *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '02, page 355–361, New York, NY, USA, 2002. Association for Computing Machinery. 2

[17] G. Littwin and L. Wolf. Deep meta functionals for shape representation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1824–1833, 2019. 2

[18] J. Liu, F. Yu, and T. Funkhouser. Interactive 3d modeling with a generative adversarial network. *International Conference on 3D Vision (3DV)*, 2017. 2

[19] I. Loshchilov and F. Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *ICLR*, 2017. 8

[20] K.-L. Low and T.-S. Tan. Model simplification using vertex-clustering. pages 75–82, 188, 01 1997. 3

[21] D. Luebke and C. Erikson. View-dependent simplification of arbitrary polygonal environments. *SIGGRAPH Comput. Graph.*, 1997. 3

[22] D. Luebke, B. Watson, J. D. Cohen, M. Reddy, and A. Varshney. *Level of Detail for 3D Graphics*. Elsevier Science Inc., USA, 2002. 3

[23] J. N. Martel, D. B. Lindell, C. Z. Lin, E. R. Chan, M. Monteiro, and G. Wetzstein. Acorn: Adaptive coordinate networks for neural scene representation. *arXiv preprint arXiv:2105.02788*, 2021. 1, 2, 5, 6

[24] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[25] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In *European Conference on Computer Vision*, pages 405–421. Springer, 2020. 2

[26] L. Morreale, N. Aigerman, V. G. Kim, and N. J. Mitra. Neural surface maps. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4639–4648, 2021. 1, 2, 3, 5, 8

[27] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 165–174, 2019. 2

[28] O. Poursaeed, M. Fisher, N. Aigerman, and V. G. Kim. Coupling explicit and implicit surface representations for generative 3d modeling. *ECCV*, 2020. 2

[29] M. Rabinovich, R. Poranne, D. Panozzo, and O. Sorkine-Hornung. Scalable locally injective mappings. *ACM Transactions on Graphics (TOG)*, 36(4):1, 2017. 3

[30] W. Schroeder, J. Zarge, and W. Lorensen. Decimation of triangle meshes. *SIGGRAPH Comput. Graph.*, 26:65–70, 06 1997. 3

[31] A. Sinha, J. Bai, and K. Ramani. Deep learning 3d shape surfaces using geometry images. In *ECCV*, 2016. 2

[32] V. Sitzmann, J. N. Martel, A. W. Bergman, D. B. Lindell, and G. Wetzstein. Implicit neural representations with periodic activation functions. *arXiv preprint arXiv:2006.09661*, 2020. 2

[33] H. Su, H. Fan, and L. Guibas. A point set generation network for 3d object reconstruction from a single image. *CVPR*, 2017. 2

[34] T. Takikawa, J. Litalien, K. Yin, K. Kreis, C. Loop, D. Nowrouzezahrai, A. Jacobson, M. McGuire, and S. Fidler. Neural geometric level of detail: Real-time rendering with implicit 3d shapes. In *Proc. CVPR*, pages 11358–11367, 2021. 1, 2, 5, 6

[35] Y. Yang, C. Feng, Y. Shen, and D. Tian. FoldingNet: Point cloud auto-encoder via deep grid deformation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 206–215, 2018. 2

[36] W. Yifan, L. Rahmann, and O. Sorkine-Hornung. Geometry-consistent neural shape representation with implicit displacement fields, 2021. 3, 5, 8