NeuralMLS: Geometry-Aware Control Point Deformation

M. Shechter[1], R. Hanocka[2], G. Metzer[1], R. Giryes[1], D. Cohen-Or[1]

[1]Tel-Aviv University, Israel, [2]University of Chicago, USA

Abstract

*We introduce NeuralMLS, a space-based deformation technique, guided by a set of displaced control points. We leverage the power of neural networks to inject the underlying shape geometry into the deformation parameters. The goal of our technique is to enable a realistic and intuitive shape deformation. Our method is built upon moving least-squares (MLS), since it minimizes a weighted sum of the given control point displacements. Traditionally, the influence of each control point on every point in space (*i.e.*, the weighting function) is defined using inverse distance heuristics. In this work, we opt to learn the weighting function, by training a neural network on the control points from a single input shape, and exploit the innate smoothness of neural networks. Our geometry-aware control point deformation is agnostic to the surface representation and quality; it can be applied to point clouds or meshes, including non-manifold and disconnected surface soups. We show that our technique facilitates intuitive piecewise smooth deformations, which are well suited for manufactured objects. We show the advantages of our approach compared to existing surface and space-based deformation techniques, both quantitatively and qualitatively.*

CCS Concepts

• ***Computing methodologies*** → *Computer graphics; Machine learning;*

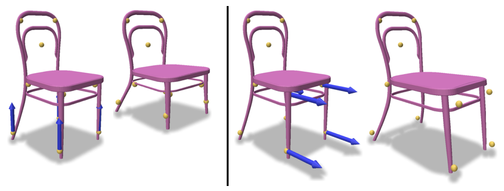

Figure 1: Interactive shape deformation using NeuralMLS. *We train a neural network on a given set of control points (yellow), in a displacement-agnostic manner. Then, the user applies various displacements and receives different deformations.*

1. Introduction

Interactive deformation of 3D shapes is a long-standing problem in computer graphics [YLW*21]. The goal is to provide the user a simple and intuitive interface for shape manipulation.

We present a space-deformation technique, where the user manipulates a set of control points (CP) which are interactively displaced to define the deformation. The technique is built upon moving least-squares (MLS), as it minimizes a weighted sum of the given CP displacements. However, in traditional MLS the weighting function is crafted based on pre-defined heuristics, which are agnostic to the given input shape. Instead, we opt to *learn* a weighting function, which is *tailored* to the given input shape using a neural network, as illustrated in Figure 2, leading to a NeuralMLS.

Our technique benefits from the innate inductive biases of neural networks (e.g., intrinsically favor smooth and desirable solutions [BGG*20] and a tendency to converge to a piecewise smooth solution [OWSS20]). Those biases enable us to train the network to learn a smooth piecewise weighting function from only the given CP, and avoids having to train on a large dataset. Piecewise smooth deformations are desirable when editing manufactured objects. It is particularly useful for keypoint-based deformations, since we want to deform or displace each part separately while smoothly interpolating within each part independently. Our interface and a set of shape manipulations are shown in Figure 3. Our method: (i) provides an intuitive and simple framework for shape manipulation; (ii) requires no training dataset; (iii) produces piecewise smooth deformations; and (iv) provides control over the relaxation degree of the interpolation property. We demonstrate quantitative and qualitative advantages of our approach compared to existing surface and space-based deformations. Our code is available at `https://github.com/MeitarShechter/NeuralMLS`.

2. Related Work

Our paper focuses on shape deformation which has been studied extensively in the literature for the past several decades [YLW*21]. One of the common schemes are referred to as space-based deformation, where the deformation is applied over the entire ambient space containing the shape. Among space-based methods are free-form deformation (FFD) [SP86]), cage-based deformation (CBD) [JSW05; LLC08] and moving least squares

DOI: 10.2312/egs.20221034

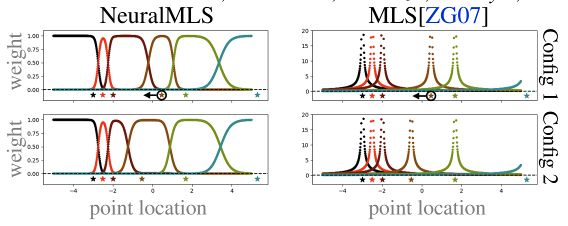

Figure 2: NeuralMLS learns a geometry-aware weighting function. *We show two different (1D) control point configurations (denote by the star locations). We visualize the learned/calculated weighting function of each control point for both NeuralMLS (left) and MLS (right) by color coding. Note the difference in the weight distribution of the points which are adjacent to the moved point in NeuralMLS, compared to the lack of change in MLS.*

(MLS) [ZG07].As space-based deformations seek to interpolate a given set of point displacements over the entire space, the main challenge that arises is: how to define an interpolation that follows desired deformation properties, such as piecewise smoothness which facilitate detail preservation of the input shape.

Recently, there has been a rising interest in using deep learning to edit shapes through deformations. In ALIGNet [HFW*18], the authors learn a FFD grid which can warp source shapes to incomplete target shapes. Neural cages [YAK*20] uses a neural network to learn a CBD. Other works harness neural networks to learn a per-point displacement [WCMN19]. A limitation with the above methods is their inability to provide a simple and intuitive interface for shape manipulation. Constructing a cage is a rather complex task, where as annotating a few control points is straight-forward.

Shape deformation based on a sparse set of control points (CP) provides an intuitive interface for shape manipulation. CP based methods enable the user to guide the deformation using various constraints. In As-Rigid-As-Possible (ARAP) [SA07], the CP impose hard constraints on a distortion minimization objective. In KeypointDeformer (KPD) [JTM*20], the CP are used to influence a cage enclosing the shape. MLS [ZG07] uses the CP to construct a least squares minimization problem. Although CP based methods often yield desirable results, they still suffer from various drawbacks. ARAP is impractical for large meshes and requires high quality, manifold, single-connected component mesh, which is a considerable demand especially for scanned objects. KPD is required to be trained on the same class used at inference time, and only provides a fixed amount and initial placement of CP for the user. MLS might suffer from local artifacts around the CP and over smoothing of rigid parts, as seen in Figures 4, 5 and 6.

The spectral bias and smoothness properties of neural networks suggest that they learn low frequencies first and provide a type of smooth interpolation of the training data [BGG*20]. Furthermore, [OWSS20] showed that the types of functions learned by neural networks tend to be piecewise smooth, which is particularly interesting for shape manipulation.

3. Method

NeuralMLS learns a weighting function, which is then used within the MLS framework. To learn such function, we soft-divide the space into regions of control point (CP) influence by treating each CP as a class, and train a network to output the corresponding class, based on the control point's xyz position in space. In the following, we elaborate more on the technique. We first give a brief overview on plain MLS, and then explain how we train and use the network.

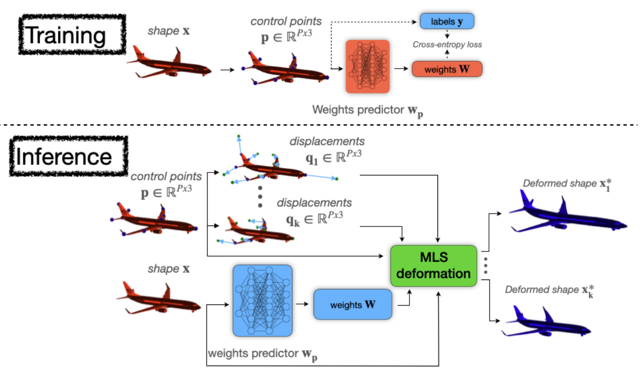
Training
shape $\mathbf{x}$
control points $\mathbf{p} \in \mathbb{R}^{Px3}$
labels $\mathbf{y}$
Cross-entropy loss
weights $\mathbf{W}$
Weights predictor $\mathbf{w_p}$
Inference
control points $\mathbf{p} \in \mathbb{R}^{Px3}$
displacements $\mathbf{q_1} \in \mathbb{R}^{Px3}$
displacements $\mathbf{q_k} \in \mathbb{R}^{Px3}$
shape $\mathbf{x}$
weights $\mathbf{W}$
MLS deformation
weights predictor $\mathbf{w_p}$
Deformed shape $\mathbf{x}_1^*$
Deformed shape $\mathbf{x}_k^*$

Figure 3: Method overview. ***Training (top)****: given a shape* $\mathbf{x}$ *and control points* $\mathbf{p}$*, we train a weighting function network* $\mathbf{w_p}$ *to learn to map each control point position to an assigned 1-hot encoding.* ***Inference (bottom)****: during inference, we freeze the network weights and each point in the shape* $\mathbf{x}$ *passes through* $\mathbf{w_p}$ *to generates the per-point weights* $\mathbf{W}$*. The user can then interactively displace the control points to new locations* $\mathbf{q}$*, which will generates a new shape* $\mathbf{x}^*$ *using MLS.*

3.1. Moving Least Squares (MLS)

MLS is a space-deformation technique, where the set of the given CP and their displacements, denoted by $\mathcal{P}$ and $\mathcal{Q}$, are projected into the entire space. Specifically, it finds for any given point in space $\mathbf{x} \in \mathbb{R}^3$, the deformation $T_x(\cdot)$ that minimizes the following:

$$\sum_i w_i(\mathbf{x})|T_x(\mathbf{p_i}) - \mathbf{q_i}|^2, \qquad (1)$$

where $\mathbf{p_i}$ and $\mathbf{q_i}$ are the CP and their displacements, respectively, and w_i are the weighting schemes of each CP. Euclidean-based weighting function is typically being used:

$$w_i(\mathbf{x}) = \frac{1}{d(\mathbf{p_i}, \mathbf{x})^{2\alpha}}, \qquad (2)$$

where d is the distance metric (Euclidean in MLS) and α is a fall-off parameter that weigh the distance function. A closed-form solution for T_x can be obtained when the deformation is constrained to be an affine, similarity, or rigid transformation [ZG07].

3.2. Learning a Geometry Aware Weighting Function

The MLS framework can be seen as a deformation function $f : \mathbb{R}^3 \times \mathbb{R}^{Px3} \times \mathbb{R}^{Px3} \times w_p \rightarrow \mathbb{R}^3$, where the inputs are a point in space $\mathbf{x}$, the CP $\mathcal{P}$, the CP after displacement $\mathcal{Q}$ and a weighing function $w_p : \mathbb{R}^3 \rightarrow \mathbb{R}^P_{\geq 0}$, and the output is the deformed point location. For performing the deformation, we apply f on all the points of the given shape. Our work focuses on learning a new weighting function, which we plug into the MLS framework and gain the following desired deformation properties, for the deformation function f: 1) *Approximation*: the control points $\mathcal{P}$ should map *approximately* to their displaced locations $\mathcal{Q}$, i.e., $f(\mathbf{p_i}) \approx \mathbf{q_i}$; 2) *piecewise Smoothness*: f should produce piecewise smooth deformations; and 3) *Identity*: If $\forall i : \mathbf{p_i} = \mathbf{q_i}$ then T_x should be the identity function, i.e., $f(\mathbf{x}) = \mathbf{x}$. As the third property is guaranteed by the principles of MLS, our effort concentrate on the first two.

Our weighting function $w_p : \mathbb{R}^3 \rightarrow \mathbb{R}^P_{\geq 0}$ is computed using a multi-layer perceptron (see supplementary for implementation details) and we use a proxy classification task, where each given control point $\mathbf{p_i} \in \mathcal{P}$ is considered as a class, to train the network. The training is done by minimizing a categorical cross-entropy classification loss such that the input is the CP location and its label is its corresponding class. As a result, given a point in space, the network provides us with a probability for each class, that we interpret as a weight assignment for the corresponding CP, where the weight function has a piecewise smooth structure due to the network inherent bias. In order to initialize the training we first construct the network to contain an output size defined by the number of CP. We then train our network on the given set of CP (note the training is agnostic to the CP displacements), which takes only a few seconds. Given the trained weighting function, the user can displace the CP to interactively deform the shape.

By casting the problem as a classification task, we implicitly inject awareness to the underlying CP geometric configuration, as each CP also acts as a negative classification example for all the others. This makes the learned weight function depend on an entire CP configuration, rather than being affected by each CP independently, as illustrated in Figure 2. This property, for rigid deformations, is not manifested in the Euclidean-based weighting function (or any hand-crafted weighting scheme that is independent of the rest of the CP), as can be concluded from the following proposition:

Proposition 1 (Informal) The optimal rigid transformation obtained by the MLS framework is the same whether the weighting functions are normalized or not.

For formal wording and proof, see the supplementary material. From Proposition 1, we conclude that for rigid solutions, observing the weights w_i with no normalization is enough as the solutions are the same, and therefore adding a new CP or manipulating others has so effect on the *weight* of a certain CP. Thus, the weight of each CP is *not* adaptive to the specific CP configuration.

3.3. Approximating vs. Interpolating

We apply a softmax to the weighting function, which enables controlling the temperature through a scaling parameter. This enables the user to control the degree of approximation versus interpolation. See supplementary material for further details.

4. Experiments

In this section we demonstrate the key benefits of our method, compared to plain MLS and other techniques. In our experiments, we show that our method is able to achieve more intuitive and realistic shape deformations both qualitatively and quantitatively (Section 4.2) and explore our more piece-wise smooth weighting function (Section 4.3). Additional experiments that demonstrate the control over the approximation degree and the limitations of the Euclidean-based weighting function are in the supplementary.

4.1. Experimental Setup

We evaluate our technique on data from different categories. Our method requires no training data besides the user specified control points (CP) and input shape to deform. All results are obtained using a softmax temperature of 1 and constraining the MLS solution to rigid deformations. See supplementary for more details.

4.2. Shape Deformation

We demonstrate the advantages of our method compared to both classic and learning-based methods.

Qualitative Evaluation. Figure 4 contains visual comparisons to MLS [ZG07], ARAP [SA07] and KPD [JTM*20]. Our method is able to better preserve local features of the source shape, while still adhering the CP guidance. The CP displacements are manually selected to reflect reasonable shape deformations. Additional results and comparison to MLS can be seen in Figure 5, where it shows that MLS suffers from local artifacts. More results and further discussion can be found in the supplementary.

Quantitative Evaluation. We measured the distortion in discretized Laplacian magnitude and mean curvature of the deformed shapes, compared with the source shapes appear in Figure 4 and in the supplementary. These measures infer the amount of details preserved which indicates a desirable deformation. As one trivial way to achieve zero distortion in the above measures, is to not deform the shape at all, we also compare the approximation level of the deformed CP to their given displacements. Our approach is able to achieve the lowest distortion across all categories by a large margin, while still adhering the CP guidance. The full results can be found in the supplementary.

User Study. We conducted a user study in order to evaluate how realistic our deformations look compared to MLS and KPD, ARAP was excluded from the study, as the deformations failed due to bad input mesh. The study concludes that our method is more likely to produce realistic looking results, compared to MLS and KPD. Complete details and results are in the supplementary.

4.3. Piecewise Smooth Deformation

Piecewise smoothness is a desired property for shape deformations. It enables applying different deformations to different parts of the shape, while preserving a natural and smooth interpolation within and between parts. The visualizations depicted in Figure 6, show the MLS and NeuralMLS weights and resulting deformation functions, and demonstrate the difficulty of MLS to produce piecewise solutions (see supplementary for further analysis).

5. Conclusions

We presented a geometry-aware space-deformation technique based on the MLS framework. Our key idea is to leverage the power

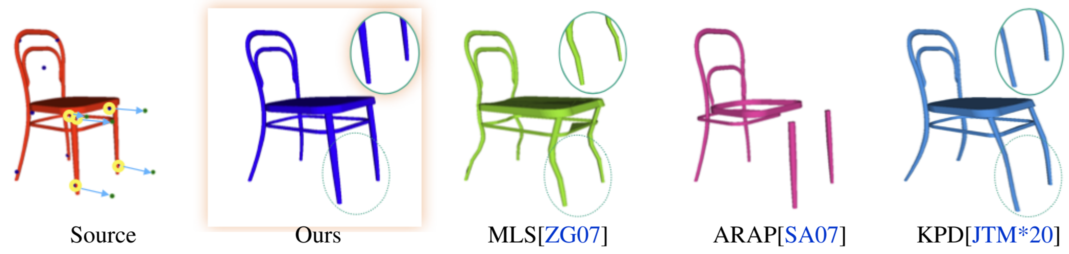

Figure 4: Comparison to existing control point based deformation methods. *Our method achieves desirable results compared to traditional and learning-based deformation techniques. Note that the chair is a non-manifold, which is not supported by ARAP.*

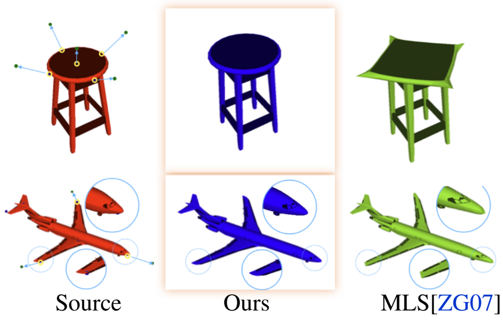

Figure 5: Deforming source shapes (left) with user annotated control points (blue arrows). *Observe how MLS suffers from local artifact near the control points, while our method avoids these issues.*

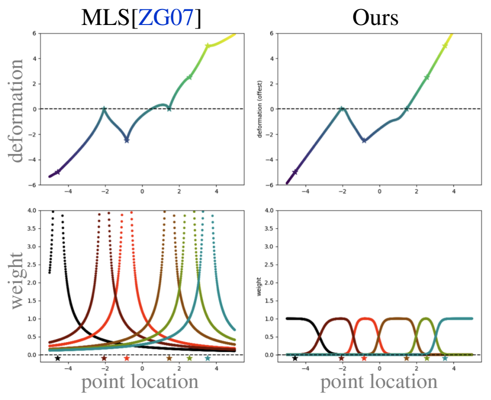

Figure 6: The resulting deformations (top) and weights (bottom) of our method (right) and MLS (left). *This illustrates our method's ability to produce more piecewise smooth deformations compared to MLS, as our weights are more evenly spread over each control point adjacent region, compared to the sharp peaks of MLS.*

of neural networks to learn a weighting function associated with the given control points (CP). The learned weights adhere to the geometric configuration of the CP which implicitly respects the underlying shape geometry and results in piecewise smoothness. Traditional MLS treats the CP displacements as *hard-constraints*. On the other hand, our NeuralMLS treats the displacements as *soft-constraints* in order to obtain piecewise-smooth deformations. Indeed, there is an inherent trade-off between adherence to the constraints and a piecewise-smooth weighting function. Our framework provides a relaxation parameter which can trade-off smoothness for constraint-adherence, which enables intuitive and interactive manipulation of the shape.

Our framework is built for an interactive scenario with user-specified CP and constraints. In the future, we would like to train a network to learn the location and number of the CP based on analyzing the input shape. We would like also to consider grouping and structuring the CP to reflect the symmetries and relationships of the input shape, such that the editing will be faithful to the input shape semantics.

References

[BGG*20] Basri, Ronen, Galun, Meirav, Geifman, Amnon, et al. *Frequency Bias in Neural Networks for Input of Non-Uniform Density*. 2020. arXiv: 2003.04560 [cs.LG] 1, 2.

[HFW*18] Hanocka, Rana, Fish, Noa, Wang, Zhenhua, et al. "ALIGNet: partial-shape agnostic alignment via unsupervised learning". *ACM Transactions on Graphics (TOG)* 38.1 (2018), 1 2.

[JSW05] Ju, Tao, Schaefer, Scott, and Warren, Joe. "Mean value coordinates for closed triangular meshes". *ACM Trans. Graph.* 24 (July 2005), 561–566. DOI: 10.1145/1073204.1073229 1.

[JTM*20] Jakab, Tomas, Tucker, Richard, Makadia, Ameesh, et al. "KeypointDeformer: Unsupervised 3D Keypoint Discovery for Shape Control". *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020 2–4.

[LLC08] Lipman, Yaron, Levin, David, and Cohen-Or, Daniel. "Green coordinates". *ACM Transactions on Graphics* 27.3 (Aug. 2008). ISSN: 0730-0301. DOI: 10.1145/1360612.1360677 1.

[OWSS20] Ongie, Greg, Willett, Rebecca, Soudry, Daniel, and Srebro, Nathan. "A Function Space View of Bounded Norm Infinite Width ReLU Nets: The Multivariate Case". *International Conference on Learning Representations*. 2020 1, 2.

[SA07] Sorkine, Olga and Alexa, Marc. "As-Rigid-As-Possible Surface Modeling". *Proceedings of EUROGRAPHICS/ACM SIGGRAPH Symposium on Geometry Processing*. 2007, 109–116 2–4.

[SP86] Sederberg, Thomas and Parry, Scott. "Free-form deformation of solid geometric models". Vol. 20. Aug. 1986, 151–160. ISBN: 0897911962. DOI: 10.1145/15886.15903 1.

[WCMN19] Wang, Weiyue, Ceylan, Duygu, Mech, Radomir, and Neumann, Ulrich. "3DN: 3D Deformation Network". *CVPR*. 2019 2.

[YAK*20] Yifan, Wang, Aigerman, Noam, Kim, Vladimir G., et al. "Neural Cages for Detail-Preserving 3D Deformations". *CVPR*. 2020 2.

[YLW*21] Yuan, Yu-Jie, Lai, Yu-Kun, Wu, Tong, et al. *A Revisit of Shape Editing Techniques: from the Geometric to the Neural Viewpoint*. 2021. arXiv: 2103.01694 [cs.GR] 1.

[ZG07] Zhu, Yuanchen and Gortler, S. "3D Deformation Using Moving Least Squares". 2007 2–4.