Supplementary Material for 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

1. Implementation Details

Our weighting function $\mathbf{w_p}$ is implemented as a neural network using a simple multi-layer perceptron (MLP) consisted of 2 hidden layers of width 1024 and ReLU activations. The output layer size is set to the amount of user-given control points. A softmax is applied to the output logits of the network in order to obtain probabilities, which we interpret as weights. We use the Adam optimizer [KB17] with learning rate of $3e-4$.

2. Proposition 1

Proposition 1 Let $\mathbf{x}$ be a point in space, $\mathcal{P}$ a set on control points, $\mathcal{Q}$ be their displacements and w the Euclidean-based weighting function, i.e. $w_i(\mathbf{x}) = \frac{1}{d(\mathbf{p_i},\mathbf{x})^{2\alpha}}$. Let $T_x = MLS(\mathbf{x},\mathcal{P},\mathcal{Q},w)$ be the deformation function obtained by plugging $\mathbf{x}$, $\mathcal{P}$, $\mathcal{Q}$ and w to the MLS framework with rigid-deformations constraint, namely, $T_x(\mathbf{y}) = M\mathbf{y}+\mathbf{r}$ is the optimal rigid solution for some rotation matrix M and translation vector $\mathbf{r}$. Also, let $\hat{w}$ be the normalized weights, i.e., $\hat{w}_i(\mathbf{x}) = \frac{w_i(\mathbf{x})}{\sum_i w_i(\mathbf{x})}$, and $\hat{T}_x = MLS(\mathbf{x},\mathcal{P},\mathcal{Q},\hat{w})$ be the optimal rigid solution of the MLS framework with the normalized weights. Then $\hat{T}_x(\mathbf{y}) = \hat{M}\mathbf{y}+\hat{\mathbf{r}} \equiv T_x(\mathbf{y})$.

Proof Our proof follows similar transitions and notations as in [ZG07] and [SMW06], but for completeness we attach the full proof here.
Let $\mathbf{p}_*$ and $\mathbf{q}_*$ be the weighted centroids of $\mathbf{p_i}$'s and $\mathbf{q_i}$'s, respectively:

$$\mathbf{p}_* = \frac{\sum_i w_i(\mathbf{x})\mathbf{p_i}}{\sum_i w_i(\mathbf{x})}, \quad \mathbf{q}_* = \frac{\sum_i w_i(\mathbf{x})\mathbf{q_i}}{\sum_i w_i(\mathbf{x})}$$

Similarly, define $\hat{\mathbf{p}}_*$ and $\hat{\mathbf{q}}_*$ as the weighted centroids using the normalized weights. Then we have:

$$\hat{\mathbf{p}}_* = \frac{\sum_i \hat{w}_i(\mathbf{x})\mathbf{p_i}}{\sum_i \hat{w}_i(\mathbf{x})} = \sum_i \hat{w}_i(\mathbf{x})\mathbf{p_i} = \sum_i \frac{w_i(\mathbf{x})\mathbf{p_i}}{\sum_j w_j(\mathbf{x})} = \mathbf{p}_*$$

And similarly, $\hat{\mathbf{q}}_* = \mathbf{q}_*$.
By plugging $T_x(\mathbf{y}) = \hat{T}_x(\mathbf{y}) = \hat{M}\mathbf{y}+\hat{\mathbf{r}}$ into equation 1 (main text) we get a quadratic dependency in $\hat{\mathbf{r}}$. Since the minimizer is where the derivatives with respect to each of the free variables in $\hat{T}_x$ are zero, we can solve directly for $\hat{\mathbf{r}}$ in the terms of the matrix $\hat{M}$. Taking the partial derivative w.r.t. the free variables in $\hat{\mathbf{r}}$ produces a linear system of equations. Solving for $\hat{\mathbf{r}}$ yields that

$$\hat{\mathbf{r}} = \hat{\mathbf{q}}_* - \hat{\mathbf{p}}_*\hat{M} = \mathbf{q}_* - \mathbf{p}_*\hat{M}$$

This leaves only $\hat{M}$ to be determined. Note:

$$\alpha_i = w_i(\mathbf{x})^{\frac{1}{2}}, \quad \tilde{\mathbf{p}}_\mathbf{i} = \mathbf{p_i} - \mathbf{p}_*, \quad \tilde{\mathbf{q}}_\mathbf{i} = \mathbf{q_i} - \mathbf{q}_*,$$
$$P = (\alpha_1\tilde{\mathbf{p}}_\mathbf{1}\cdots\alpha_N\tilde{\mathbf{p}}_\mathbf{N}), \quad Q = (\alpha_1\tilde{\mathbf{q}}_\mathbf{1}\cdots\alpha_N\tilde{\mathbf{q}}_\mathbf{N})$$

Similarly, we note for the normalized weights:

$$\hat{\alpha}_i = \hat{w}_i(\mathbf{x})^{\frac{1}{2}}, \quad \hat{\tilde{\mathbf{p}}}_\mathbf{i} = \mathbf{p_i} - \hat{\mathbf{p}}_* = \tilde{\mathbf{p}}_\mathbf{i}, \quad \hat{\tilde{\mathbf{q}}}_\mathbf{i} = \mathbf{q_i} - \hat{\mathbf{q}}_* = \tilde{\mathbf{q}}_\mathbf{i}$$
$$\hat{P} = (\hat{\alpha}_1\hat{\tilde{\mathbf{p}}}_\mathbf{1}\cdots\hat{\alpha}_N\hat{\tilde{\mathbf{p}}}_\mathbf{N}), \quad \hat{Q} = (\hat{\alpha}_1\hat{\tilde{\mathbf{q}}}_\mathbf{1}\cdots\hat{\alpha}_N\hat{\tilde{\mathbf{q}}}_\mathbf{N})$$

Then

$$E = \sum_i \hat{w}_i(\mathbf{x})|\hat{M}\hat{\tilde{\mathbf{p}}}_\mathbf{i} - \hat{\tilde{\mathbf{q}}}_\mathbf{i}|^2 = \sum_i |\hat{M}\hat{\alpha}_i\hat{\tilde{\mathbf{p}}}_\mathbf{i} - \hat{\alpha}_i\hat{\tilde{\mathbf{q}}}_\mathbf{i}|^2 =$$
$$||\hat{M}\hat{P} - \hat{Q}||_F^2 = tr((\hat{M}\hat{P} - \hat{Q})^t(\hat{M}\hat{P} - \hat{Q})) =$$
$$tr(\hat{P}^t\hat{P}) + tr(\hat{Q}^t\hat{Q}) - 2tr(\hat{Q}^t\hat{M}\hat{P})$$

where $||\cdot||_F$ is the Frobenius norm. Since $\hat{P}$ and $\hat{Q}$ are constant, minimizing E corresponds to maximizing $\psi = tr(\hat{Q}^t\hat{M}\hat{P}) = tr(\hat{M}\hat{P}\hat{Q}^t)$.
Now, observe that the following holds:

$$\hat{P}\hat{Q}^t = \sum_i \hat{\alpha}_i\hat{\alpha}_i\hat{\tilde{\mathbf{p}}}_\mathbf{i}(\hat{\tilde{\mathbf{q}}}_\mathbf{i})^t = \sum_i \hat{\alpha}_i\hat{\alpha}_i\tilde{\mathbf{p}}_\mathbf{i}(\tilde{\mathbf{q}}_\mathbf{i})^t = \sum_i \hat{w}_i(\mathbf{x})\tilde{\mathbf{p}}_\mathbf{i}(\tilde{\mathbf{q}}_\mathbf{i})^t =$$
$$\sum_i (\frac{w_i}{\sum_k w_k}\tilde{\mathbf{p}}_\mathbf{i}(\tilde{\mathbf{q}}_\mathbf{i})^t) = \frac{1}{\sum_k w_k}\sum_i w_i\tilde{\mathbf{p}}_\mathbf{i}(\tilde{\mathbf{q}}_\mathbf{i})^t =$$
$$\frac{1}{\sum_k w_k}\sum_i \alpha_i\alpha_i\tilde{\mathbf{p}}_\mathbf{i}(\tilde{\mathbf{q}}_\mathbf{i})^t = [c \equiv \frac{1}{\sum_k w_k}] = c\sum_i \alpha_i\alpha_i\tilde{\mathbf{p}}_\mathbf{i}(\tilde{\mathbf{q}}_\mathbf{i})^t = cPQ^t$$

Therefore, the singular value decomposition of $\hat{P}\hat{Q}^t = \hat{U}\hat{\Lambda}\hat{V}^t$ satisfies $\hat{U} = U, \hat{V} = V$ and $\hat{\Lambda} = c\Lambda$, where $U\Lambda V^t$ is the singular value decomposition of PQ^t. Thus

$$\psi = tr(\hat{M}cPQ^t) = tr(\hat{M}cU\Lambda V^t) = tr(U^t\hat{M}^tVc\Lambda)$$

Write $N = U^t\hat{M}^tV$, then N is orthogonal since U, $\hat{M}$ and V are orthogonal. It follows that $|N_{i,j}| \leq 1$, and

$$\psi = tr(Nc\Lambda) = \sum_{i=1}^{3} N_{i,i}c\lambda_i \leq \sum_{i=1}^{3} c\lambda_i$$

Hence, ψ is maximized when $N = I \iff \hat{M} = VU^t = M$ and therefore $\hat{\mathbf{r}} = \mathbf{q}_* - \mathbf{p}_*M = \mathbf{r}$. □

3. Additional Experiments and Results

3.1. Experimental Setup

Data. We demonstrate our performance abilities on data from different categories. In order to compare to KeypointDeformer [JTM*20] we trained their method on the relevant ShapeNet [CFG*15] category, as it can not be applied on a single input and requires pre-training.

Control points annotation. All methods we compare to, except KeypointDeformer [JTM*20], require and allow control points that are manually annotated by the user. In the evaluations against KeypointDeformer, we use the control points given by their keypoint predictor, as KeypointDeformer can not be conditioned on arbitrarily positioned control points. Yet, the displacement of each control-point is still annotated by the user. Also, in the case of ARAP [SA07], a mesh vertex must be chosen as a control point. Therefore, we simply use the nearest vertex to each of the user annotated control points, when comparing to ARAP.

3.2. Approximation vs. Interpolation

As we use a softmax normalization layer in our construction of the weighting function, we can add a temperature scaling parameter to the network's output, i.e.

$$[w_p(\mathbf{x})]_i = \frac{e^{\frac{w_i(\mathbf{x})}{T}}}{\sum_i e^{\frac{w_i(\mathbf{x})}{T}}}, \tag{1}$$

where $w_i(\mathbf{x})$ are the network's outputs.

The temperature enables the user to control the degree of approximation versus interpolation. As setting $T \to 0$ result in a *sharper* weight distribution i.e. making the weight of the most dominant control point approach 1 and the other weights approach 0.

We showcase our technique's ability to enable users to control the degree of approximation v.s. interpolation, using a single temperature parameter. A low temperature value results in sharper weight assignments, mainly considering the closest control point, similar to nearest neighbour interpolation of the input displacements. On the other hand, a high temperature value results in a smoother combination of the control point specified displacements, i.e. approximation, as the resulting displacement at each control point does not necessarily equal the specified user input exactly. Changing the temperature value does not require additional training, as it only amounts to scaling the trained network outputs. Figure 1 presents deformation results under different temperatures. The control points and their displacements are displayed to facilitate the visualization of the approximation quality.

3.3. MLS Ablation

MLS hyper-parameters. MLS [ZG07] allows the user a "fall-out" hyperparameter, i.e., α, to control the *typical affecting distance* of each control point. Figure 2 shows that even though α has a smoothing effect on the deformation, the results still suffers from significant artifacts. This is evident in the collateral damage caused by high influence of irrelevant control points, seen on the legs of the bottom row left most chair, and on the back rest of the top row chair. This phenomenon can also be foreseen mathematically from Equation 2 (main text), as changing the α parameter does not resolve the large weights and gradient norms near control points.

Another approach for trying to achieve a better weighting functions based on an Euclidean distance, is to add a small value ε, to the denominator of the weighting function, i.e., $w_i(\mathbf{x}) = \frac{1}{d(\mathbf{p_i},\mathbf{x})^{2\alpha}+\varepsilon}$. Figure 3 reveals that adding ε does not sufficiently resolve local artifacts and undesirable deformation properties, such as the bending of the legs in the top row chair.

3.4. Additional Results

Qualitative Results. Further comparisons to other methods can be seen in Figure 4 that also demonstrate the limitations of previous approaches. For the armadillo shape (bottom row), KPD [JTM*20] result can not be obtained, as the armadillo shape does not fall into any of the categories KPD was trained on. Also, note that ARAP [SA07] results for the top 3 rows are meaningless as ShapeNet [CFG*15] meshes are often a mesh soup and therefore are not a manifold.

Additional comparison to plain MLS can be seen in Figure 5, where it shows that MLS suffers from local artifacts. We claim that the artifacts in Figure 5 are due to the very high weights, experienced by surface points that are close to the input control points, as further analysis shows in Section 3.5 in the supplementary.

Quantitative Results. We measured the distortion in discretized Laplacian magnitude and mean curvature of the deformed shapes. We used the difference in the Laplacian magnitude rather than the actual Laplacian as we want to be invariant to rotations, that are a desirable property of the deformation, and only want to penalize loss of details that is expressed through the Laplacian magnitude. The full qualitative results discussed in the paper are attached in Table 1 and Figure 6.

Table 1: *Quantitative evaluation of our method in comparison to other approaches, using the average mean curvature difference between the source shape and the deformed shape. ARAP results for the top 3 shapes are omitted as they are meaningless due to the deformations failing, as can be seen in Figure 4.*

	Average Mean Curvature (↓)			
Shape	Ours	MLS	KPD	ARAP
Chair	**16**	19	42	-
Airplane	**1764**	1880	2142	-
Car	**5.2**	8.9	17.8	-
Armadillo	**0.412**	0.578	-	0.574

3.5. Piecewise Smooth Deformation

Figure 6 (main text) shows the MLS and NeuralMLS weights and resulting deformation functions, and demonstrate the difficulty of MLS to produce piecewise solutions. It can be seen that Euclidean-based weighting function starts off at a very high value, close to the control points (bottom-left), which causes the local artifacts that are observed across the MLS results. The weighting function then

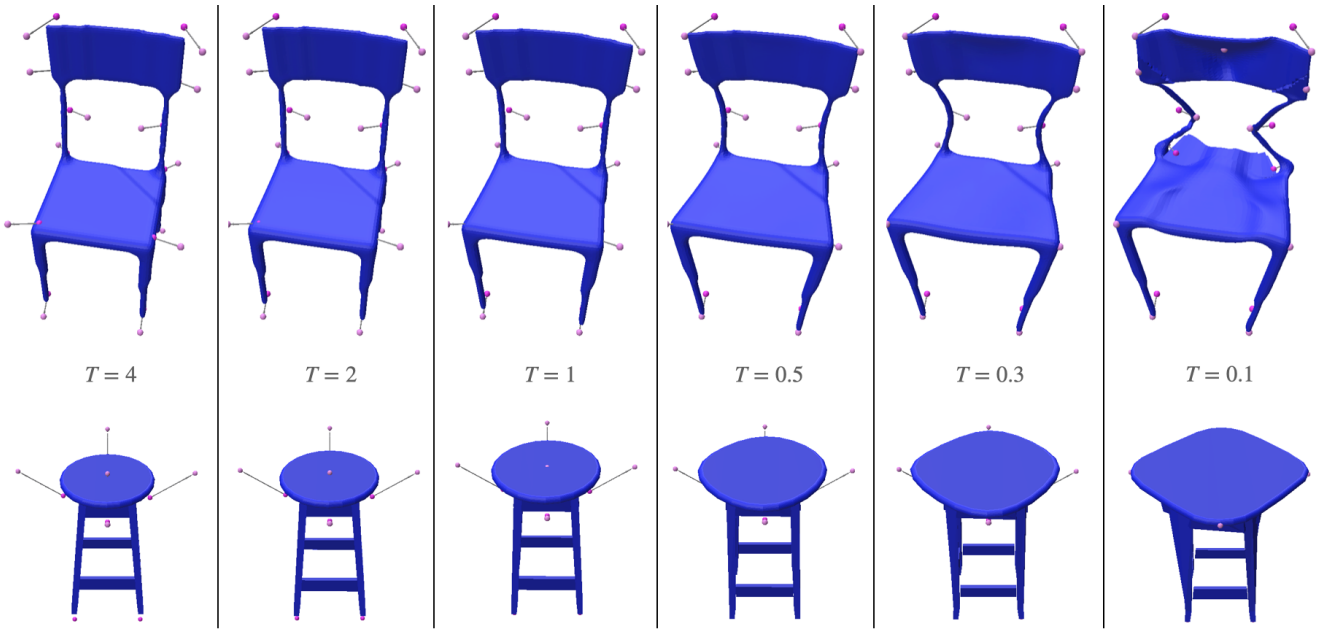

Figure 1: The temperature hyper-parameter effect demonstrated on various chairs. *The temperature value gives the user control over the approximation level of the deformation, as a trade off between approximation and interpolation of the input control point guidance.*

decrease rapidly as we move further away from the control point until reaching a non-zero plateau. This result in deformation function (top-left) that is both less smooth (sharp gradient near the control points) and less piecewise (the deformation interpolation between two points is not linear). Our method, on the other hand, produces weights that are much smaller near control points (bottom-right) ,i.e. achieving an approximation instead of interpolation, then slowly and smoothly decrease until reaching a negligible value at some point, creating a more piecewise solution (top-right).

4. Geometry Awareness Weighting Function

We attach an additional example for the geometry awareness nature of our method. Figure 8 shows a 2D example of this property.

5. User Study

The user study was done over three different categories (Car, Chair, Airplane), each category contained three different shape, and three different deformation were applied on each shape (a total of 27 different deformations). We asked users to choose the most realistic-looking deformation, among the different deformation methods applied on each set of shape and control point configuration. 23 different users, composed mostly of computer vision/computer graphics students and researchers, replied to our study and the results are displayed in Table 2.

We attach the full user study questions for the chair shapes in Figure 9, for the airplane shapes in Figure 10 and for the car shapes in Figure 11. Shapes label with 1 are produced by MLS[ZG07], shapes label with 2 are produced by NeuralMLS and shapes label with 3 are produced by KPD[JTM*20].

The full user study answers are attaches in Figures 12, 13 and 14. MLS[ZG07] answers are colored with blue, NeuralMLS with green and KPD[JTM*20] with yellow.

Table 2: *User study results. Users were asked to choose the deformation that looks the most realistic, across results produce by NeuralMLS, MLS and KPD. The numbers in the table represent the % of users that chose the result corresponding to each method. The study concludes that our method is more likely to produce realistic looking results, compared to MLS and KPD, as indicated by more users choosing NeuralMLS for every category in the study.*

	% of users (↑)		
Category	Ours	MLS	KPD
Chair	**77**%	14%	9%
Airplane	**56**%	12%	32%
Car	**71**%	16%	13%

References

[CFG*15] CHANG, ANGEL X., FUNKHOUSER, THOMAS, GUIBAS, LEONIDAS, et al. *ShapeNet: An Information-Rich 3D Model Repository*. Tech. rep. arXiv:1512.03012 [cs.GR]. Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015 2.

[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, 3, 6.

[KB17] KINGMA, DIEDERIK P. and BA, JIMMY. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG] 1.

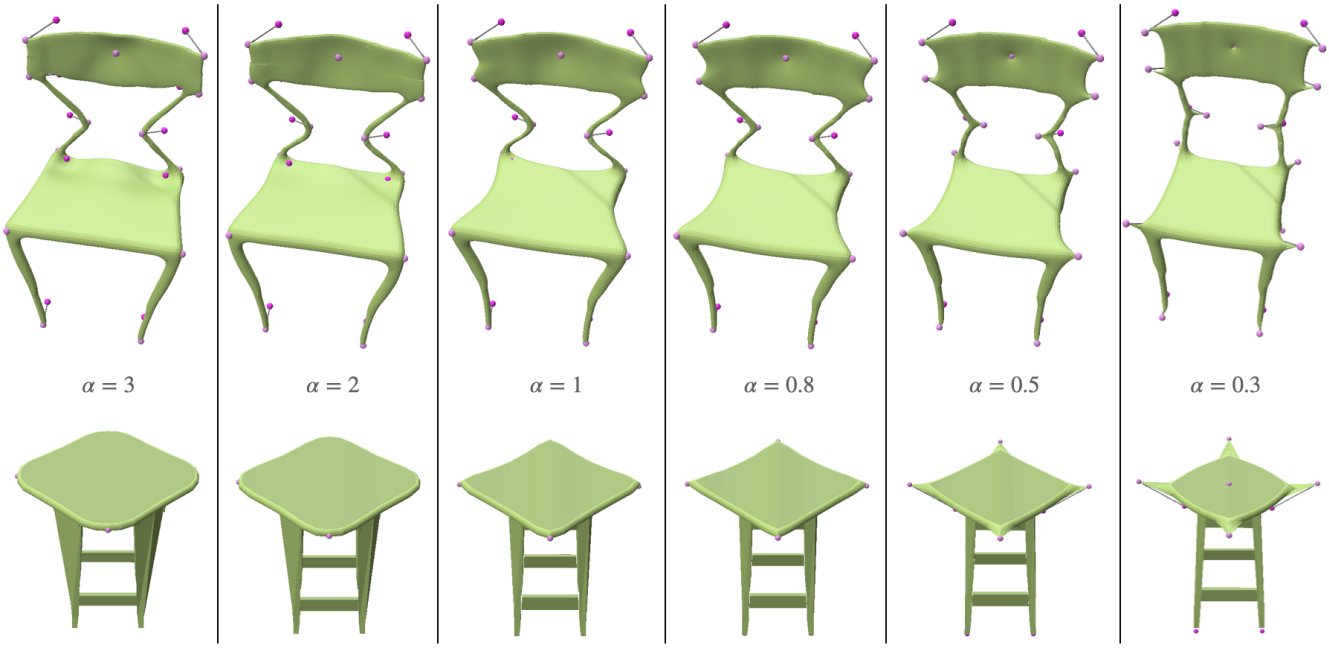

Figure 2: The α hyper-parameter effect in the plain MLS method. *This parameter provides the user control over the region of influence of each control point. Observe how altering this parameter, even though mitigating certain artifacts, such as sharp edges, causing new ones, e.g. bending rigid legs in the top chair or incompatible control point influence, manifested in the legs of the bottom leftmost chair.*

[SA07] SORKINE, OLGA and ALEXA, MARC. "As-Rigid-As-Possible Surface Modeling". *Geometry Processing*. Ed. by BELYAEV, ALEXANDER and GARLAND, MICHAEL. The Eurographics Association, 2007. ISBN: 978-3-905673-46-3. DOI: 10.2312/SGP/SGP07/109-116 2, 6.

[SMW06] SCHAEFER, SCOTT, MCPHAIL, TRAVIS, and WARREN, JOE. "Image Deformation Using Moving Least Squares". *ACM Trans. Graph.* 25.3 (July 2006), 533–540. ISSN: 0730-0301. DOI: 10.1145/1141911.1141920. URL: https://doi.org/10.1145/1141911.1141920 1.

[ZG07] ZHU, YUANCHEN and GORTLER, S. "3D Deformation Using Moving Least Squares". 2007 1–3, 6, 7.

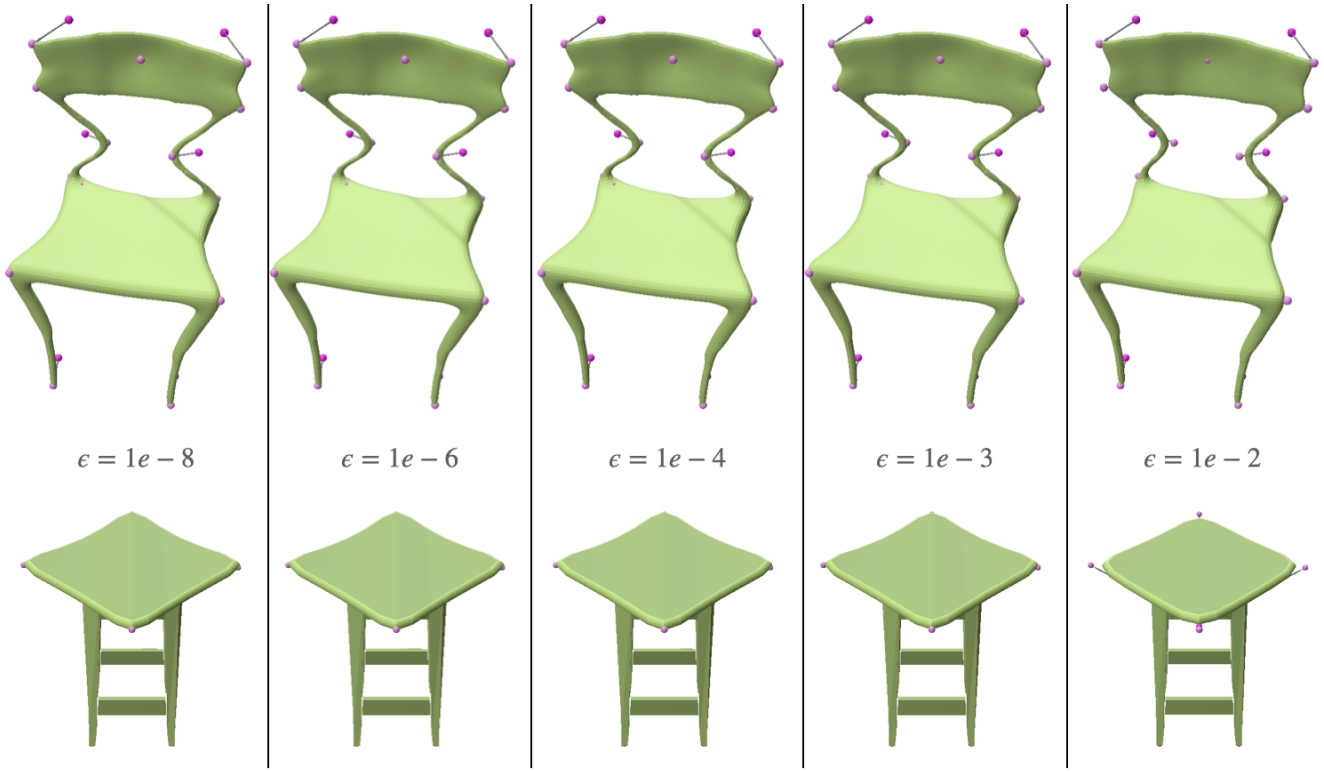

Figure 3: Demonstrating the effect of adding an ε parameter to the denominator of the weighting function of plain MLS on the chair shape. *This value is used for numerical stability but can also, as a by-product, enable the user to smooth out the deformation of the shape close to control points. Observe that although indeed mitigating the sharp artifacts the deformation still suffers from them, as well as from other problems, such as bending of rigid parts (top chair legs).*

Figure 4: Comparison of our method to other control point based deformation methods. *Our method achieves desirable results compared to traditional and learning-based deformation techniques. Note that the first three rows are non-manifold meshes, which ARAP does not support. Since KPD requires a dataset to be trained on, results for the "armadillo" shape (bottom row) cannot be obtained.*

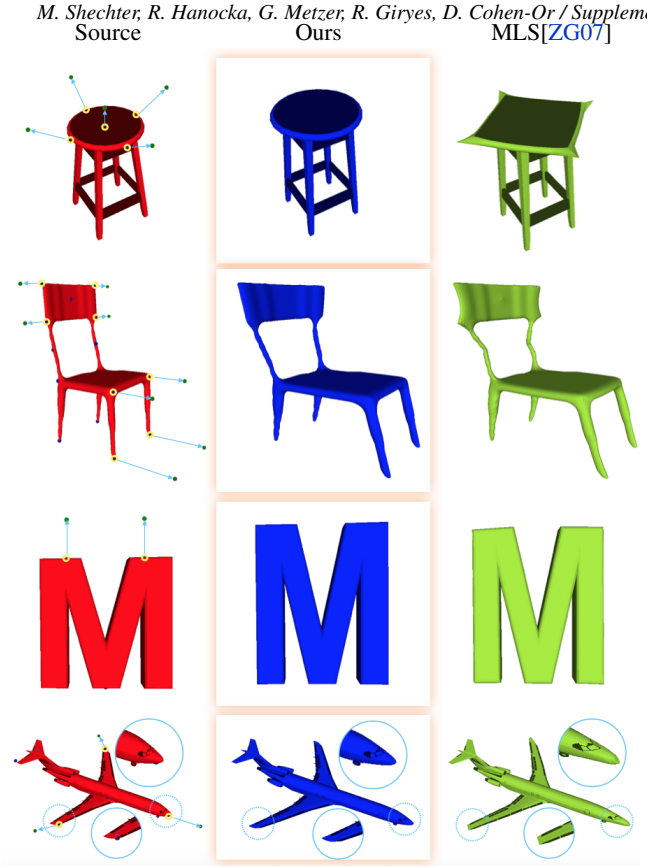

Figure 5: Deforming source shapes from the left column with user annotated control points, visualized by the blue arrows. *Observe how MLS suffers from local artifact near the input control points, while our method is able to avoid these issues.*

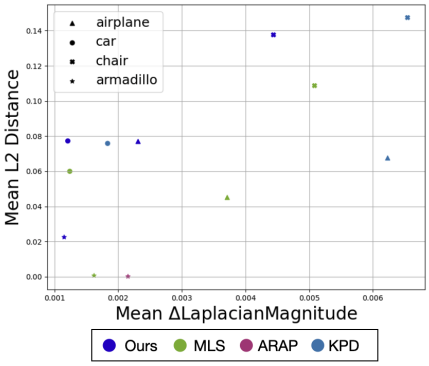

Figure 6: Quantitative comparison of our method against other approaches from Figure 4. *Each point is embedded according to its distortion (mean ΔLaplacianMagnitude) and approximation degree of the control points (Mean L2 Distance). Point near the bottom left corner are considered as "better". ARAP results are omitted for the Chair, Airplane and Car shapes as the deformations failed.*

Figure 7: *In this Figure, the weight function and its gradient norms are visualized as heat maps for the control points circled in blue at the leftmost column, with the MLS weights and gradient norms being in log-scale. This illustrates our method's ability to produce more piecewise smooth deformations compared to plain MLS, as our weight function is more evenly distributed over each control point adjacent region, where adjacent is with respect to the entire control point configuration, compared to the sharp peaks produced by plain MLS. This Figure also emphasises NeuralMLS's approximation characteristics compared to MLS that is an interpolation technique, as weights near control points are not very steep compared to their surroundings.*

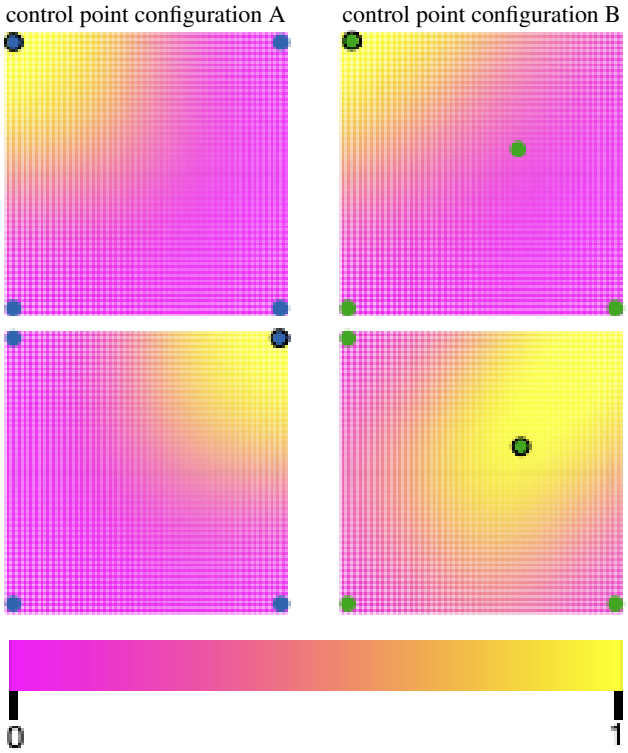

Figure 8: NeuralMLS learns a geometry-aware weighting function. *We show two different control point configurations in the left (blue) and right (green) columns. We visualize the learned weighting function for the highlighted control point (circled in black). Note the difference between the weight of the top left control point in the top row.*

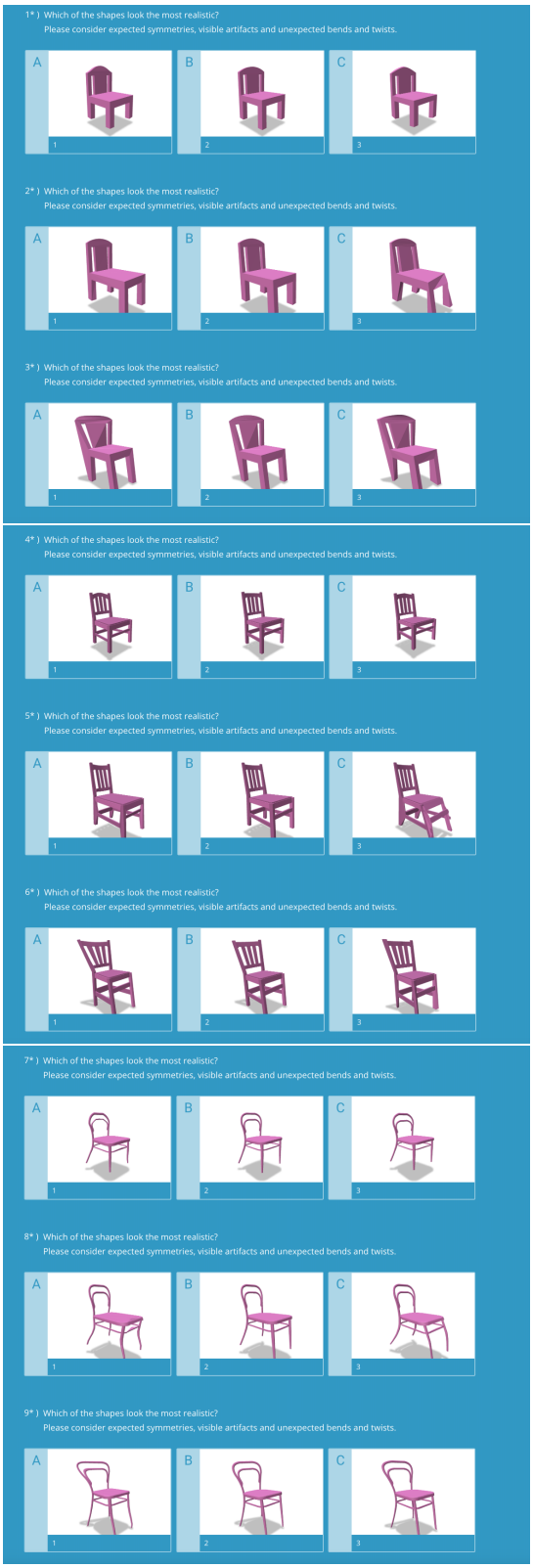

Figure 9: *User study chair questions.*

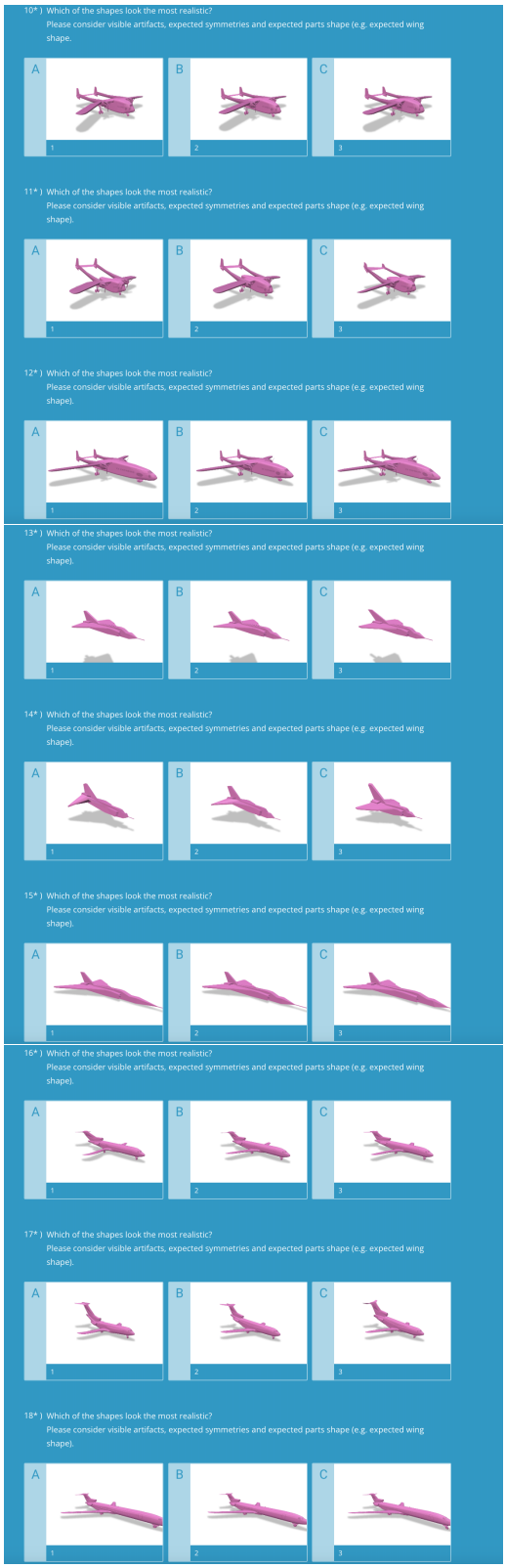

Figure 10: *User study airplane questions.*

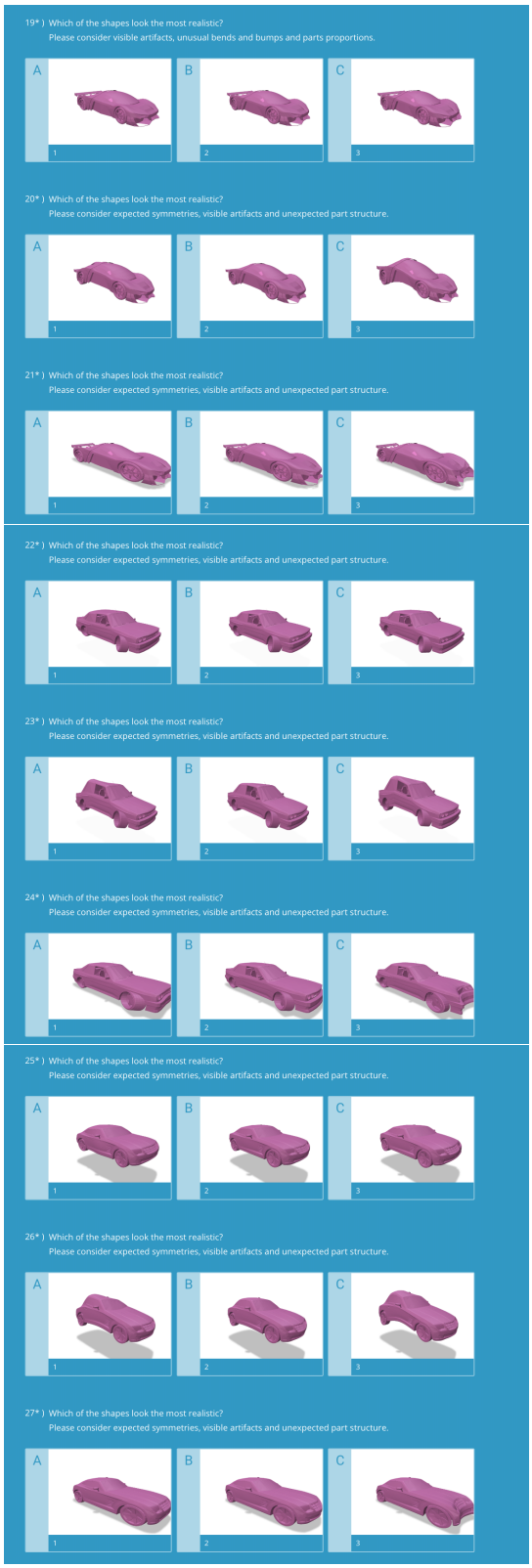

Figure 11: *User study car questions.*

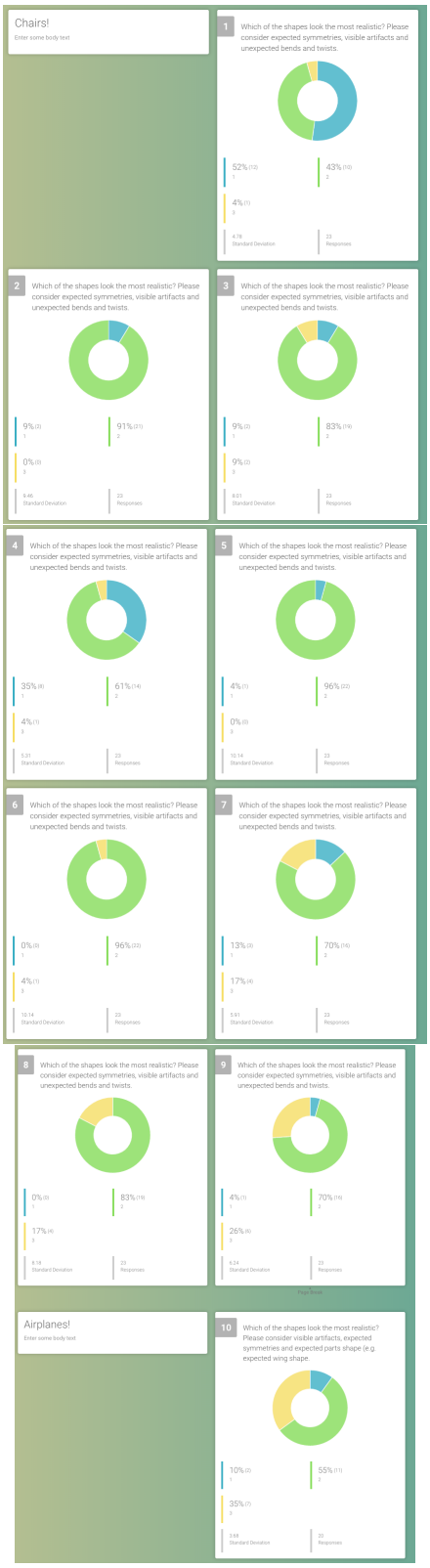

Figure 12: *User study answers for questions 1-10.*

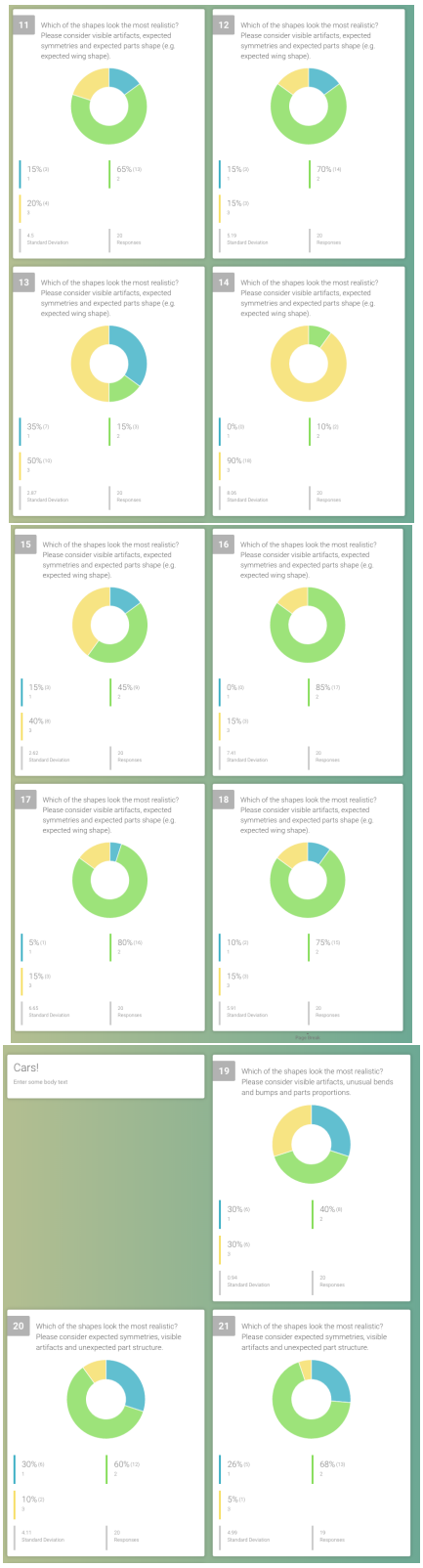

Figure 13: *User study answers for questions 11-21.*

Figure 14: *User study answers for questions 22-27.*