DeepMetaHandles: Learning Deformation Meta-Handles of 3D Meshes with Biharmonic Coordinates

Problem Statement:

Learning to generate 3D meshes is much more challenging than 2D images due to the irregularity of mesh data structures and the difficulty in designing loss functions to measure geometrical and topological properties. For such reasons, to create new meshes, instead of generating a mesh from scratch, recent work assumes that the connectivity structure of geometries is known so that the creation space is restricted to changing the geometry without altering the structure.

Control Points

subset of mesh vertices -> control points, used as the deformation handles.

Restrict the transformations of the handles to pure translations.

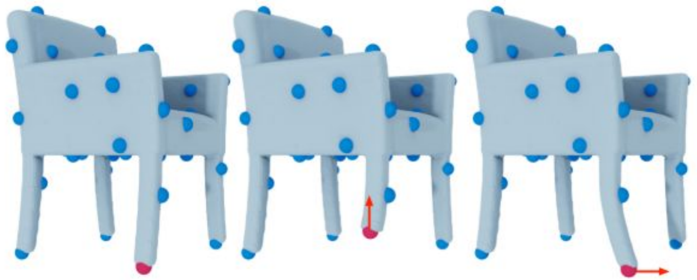


Figure 2: Two deformations resulted from moving the red control point along the arrow directions.

mesh vertices: $V \in R^{n \times 3}$ (n vertices)

control points: $C \in R^{c \times 3}$, sampled from V using farthest point sampling over geodesic distance

linear map $W \in R^{n \times c}$ between them ($V = WC$) is often called generalized barycentric coordinates’

Barycentric Coordinates

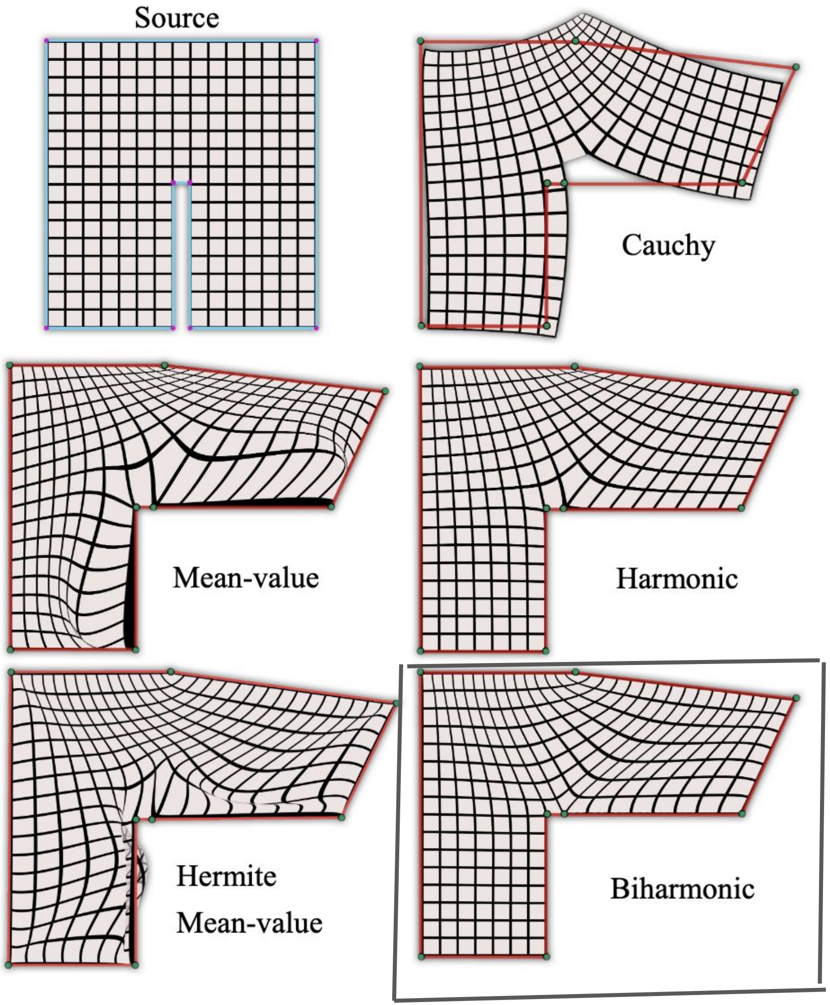
Very used for triangle subspace like mesh, point interpolation and adding analytical boundary constraints are easier wrt coordinate system.

Consider a triangle T defined by its three vertices, $\mathbf{r}_1$, $\mathbf{r}_2$ and $\mathbf{r}_3$. Each point $\mathbf{r}$ located inside this triangle can be written as a unique convex combination of the three vertices. In other words, for each $\mathbf{r}$ there is a unique sequence of three numbers, $\lambda_1, \lambda_2, \lambda_3 \geq 0$ such that $\lambda_1 + \lambda_2 + \lambda_3 = 1$ and

$$\mathbf{r} = \lambda_1 \mathbf{r}_1 + \lambda_2 \mathbf{r}_2 + \lambda_3 \mathbf{r}_3 ,$$

Barycentric Coordinates

https://www.cs.technion.ac.il/~gotsman/AmendedPubl/Roi/biharmonic.pdf



Biharmonic Coordinates

All deformations on the right depends linearly on the derivatives of the coordinates, characterizes the local behavior of the deformation.

Adding a constraint for coordinates to have large derivatives leads to overlapping maps in that local regions.

Figure on right is obtained by keeping the boundary fixed while manually changing the normal derivative vector direction and magnitude.

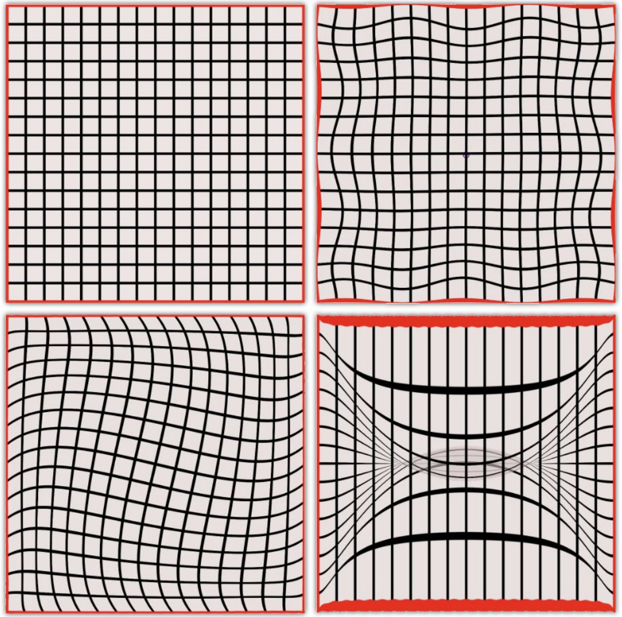


Figure 4: *A square (top-left) is mapped to itself in various ways using biharmonic coordinates, varying only the normal derivatives. Forcing the derivative to be too large (bottom-right) leads to self-overlap.*

Closed Form Solutions

Closed-form expressions with respect to the handles can be easily calculated after a pre-computation. Closed form solution gives exact result as opposed to numerical methods which give approximations.

Therefore it is stable solution to deformation problem.

deformation function $f : R^{c\times3} \rightarrow R^{n\times3}$ defined over the given control points C,

$f(C) = WC$, has 3c degrees of freedom. **f(C) are new deformed vertices of mesh.**

Why 3c?

$C\_0 = 2\hat{i} + 3\hat{j} + 4\hat{k}$, Each of these axis counts for 1 degree of freedom.

This individual freedom for each control points might lead to many implausible deformations, so there is need to further limit the freedom.

Meta Handles

meta-handle $M_i \in R^{c\times3}$ is represented as offsets over the c control points:

$M_i = [t_{i1}, \cdot \cdot \cdot ,t_{ic}]^T$, $t_{ij} \in R^3$ indicates the offset of the j-th control point for the i-th meta-handle.

These meta-handles are trained under a constraint that they don't only affect a local region rather they should represent some semantic meaning like deformation for chair legs, etc.

MetaHandleNet

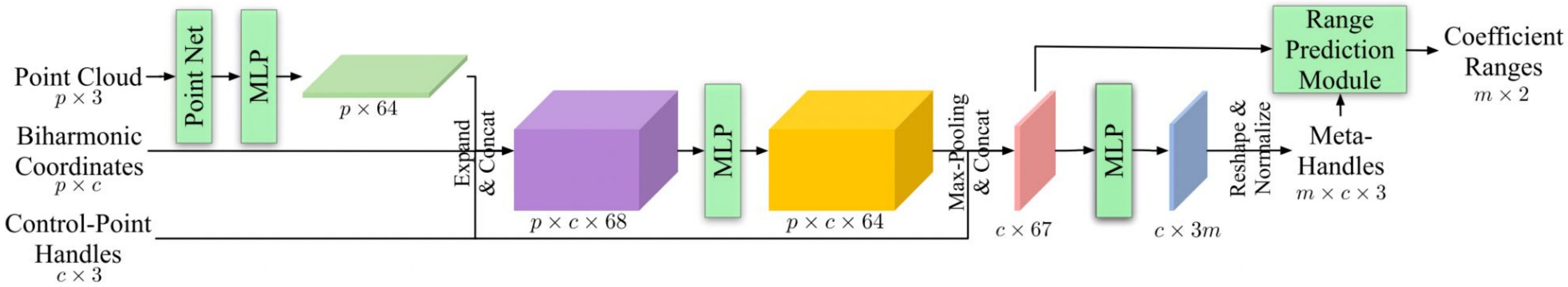


Figure 3: Architecture of MetaHandleNet: it incorporates the information from the shape (point cloud), control-point handles, and biharmonic coordinates by building a 3D tensor, and predicts a set of meta-handles with the corresponding coefficient ranges for the shape.

MetaHandleNet Inputs

Point cloud uniformly sampled from mesh, P $\in$ $R^{p\times 3}$

Control points sampled from vertices of mesh, C $\in$ $R^{c\times 3}$

Biharmonic coordinates, W $\in$ $R^{p\times c}$ -> interpolated from the mesh vertices to the point cloud (i.e., W $\in$ Rp×c) according to the barycentric coordinates.

Explicitly, the formulae for the barycentric coordinates of point $\mathbf{r}$ in terms of its Cartesian coordinates (*x, y*) and in terms of the Cartesian coordinates of the triangle's vertices are:

$$\lambda_1 = \frac{(y_2 - y_3)(x - x_3) + (x_3 - x_2)(y - y_3)}{\det(T)} = \frac{(y_2 - y_3)(x - x_3) + (x_3 - x_2)(y - y_3)}{(y_2 - y_3)(x_1 - x_3) + (x_3 - x_2)(y_1 - y_3)},$$

$$\lambda_2 = \frac{(y_3 - y_1)(x - x_3) + (x_1 - x_3)(y - y_3)}{\det(T)} = \frac{(y_3 - y_1)(x - x_3) + (x_1 - x_3)(y - y_3)}{(y_2 - y_3)(x_1 - x_3) + (x_3 - x_2)(y_1 - y_3)},$$

$$\lambda_3 = 1 - \lambda_1 - \lambda_2 .$$

Deformation Function

$g : R^m \rightarrow R^{n\times3}$, wrt m meta handles

the meta-handles $\{\mathbf{M}_i\}_{i=1\cdots m}$ and their linear combination coefficients $\mathbf{a} = [a_1, \cdots, a_m]$:

$$g\left(\mathbf{a}; \{\mathbf{M}_i\}_{i=1\cdots m}\right) = \mathbf{W}(\mathbf{C}_0 + \sum_{i=1}^{m} a_i \mathbf{M}_i), \qquad (2)$$

$C_0 \in R^{c\times3}$ denotes the rest positions of the given control points.

The degrees of freedom of the deformation function g is typically much smaller than that of the deformation function f, i.e., $m \ll 3c$.

This results in metahandlenet model forming correlation between control points and also learn structural properties of object like all legs of chair should be deformed together.

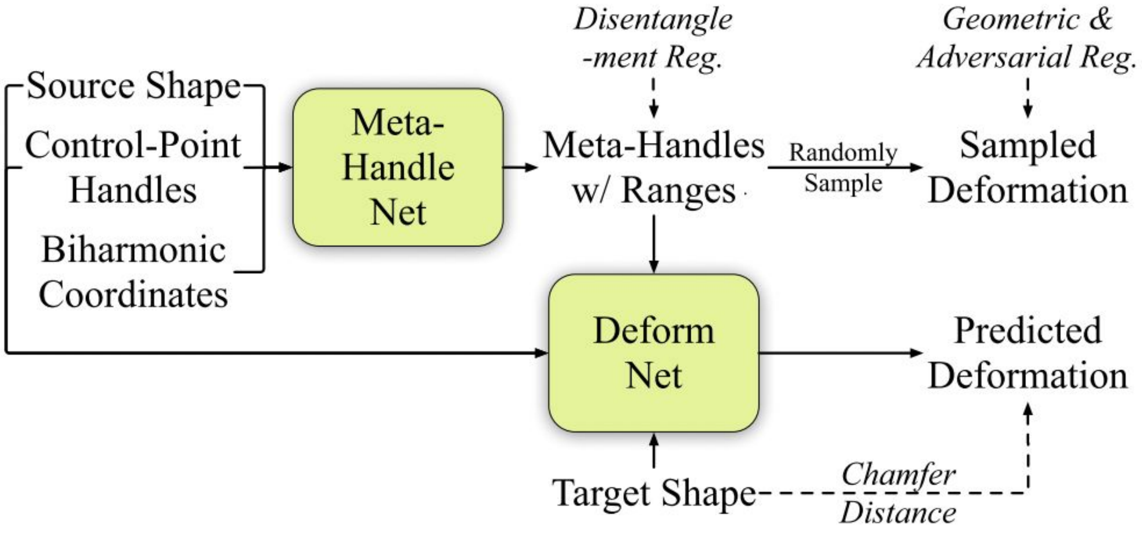


Figure 4: Overview of our method. We learn the meta-handles in an unsupervised fashion.

DeformNet inputs

Inputs to the model are 2 shapes of same categories, so the DeformNet learns to deform source shape to target shape of same category but of different structure. So essentially this design methodology constraints the MetaHandle to learn structural significance as well.

Source shape is broken into 3 parts point clouds, biharmonic coordinates, control points

Training Objectives

1. Match deformed input shape to target shape
2. Any deformation sampled from the learned ranges is plausible
3. Learned meta-handles properly disentangle the deformation space

Loss Function

$$\mathcal{L} = \mathcal{L}_{fit} + \mathcal{L}_{geo} + \mathcal{L}_{adv} + \mathcal{L}_{disen}.$$

L_{fit} minimizes the Chamfer distance between the deformed source point cloud and the target point cloud. (1st objective)

L_{geo} and L_{adv} are geometry loss and adversarial loss, added for the second objective.

$$d_{CD}(S_1, S_2) = \sum_{x \in S_1} \min_{y \in S_2} \|x - y\|_2^2 + \sum_{y \in S_2} \min_{x \in S_1} \|x - y\|_2^2$$

Geometric loss

In each iteration, deformation is sampled within predicted ranges and any implausible deformation is penalized.

$L_{geo} = L_{symm} + L_{nor} + L_{Lap}$

L_{symm}-> Minimize chamfer distance between point clouds reflected over x axis. (Because apparently all shapes in their dataset are symmetric on x axis)

L_{nor}-> Minimizes the angle difference between the face normals of the source mesh and the deformed mesh.

L_{Lap} minimizes l1-norm of the difference of Cotangent Laplacian.
https://igl.ethz.ch/projects/Laplacian-mesh-processing/Laplacian-mesh-optimization/lmo.pdf

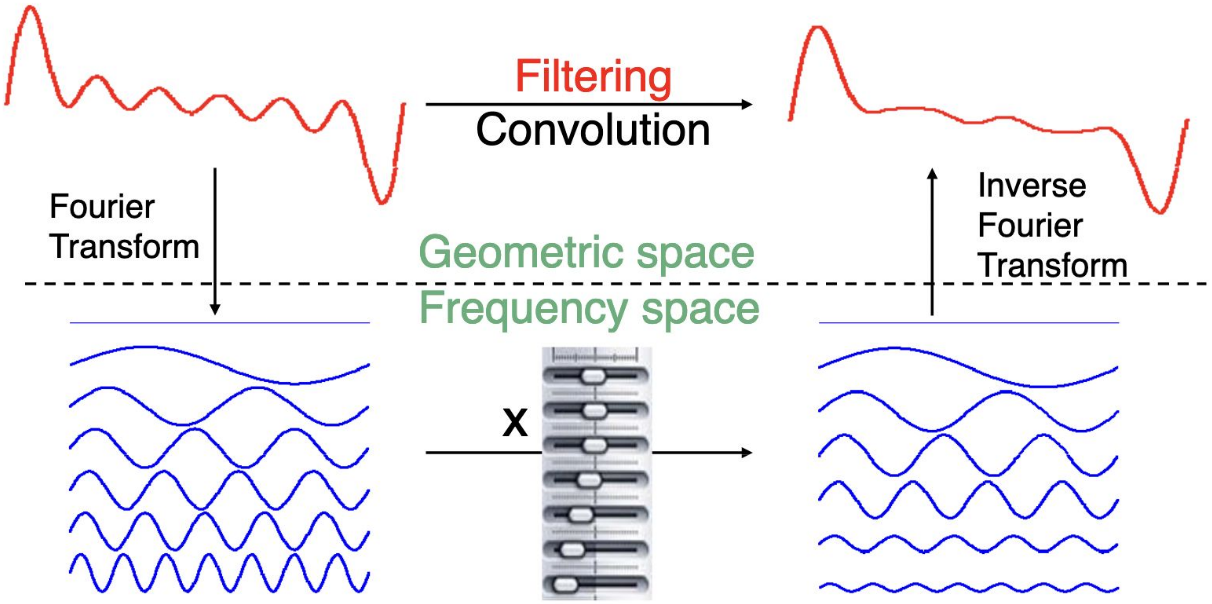
Mesh Smoothing

Input:Noisy mesh (scanned or other)

Output:Smooth mesh

How:Filter out high frequency noise

Smoothing by Filtering



Laplacian for high curvature surface

These either uniformly smooth the mesh, shown in Fig. 7(b) and (e),

or attempt to retain features by placing more (positional) weight on high curvature vertices, as seen in Fig. 7(c) and (f).

To further increase feature preservation, practically any function which reduces the weight on Laplacian smoothness constraints of feature vertices can be applied. Fig. 7(d) and (g).

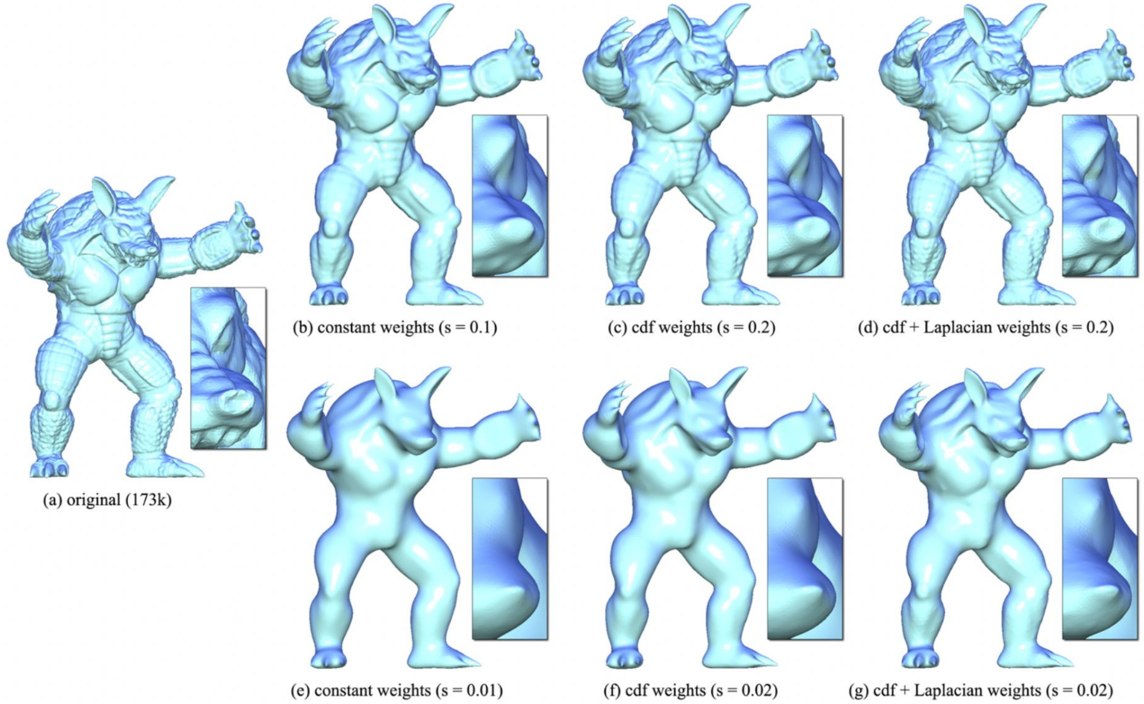
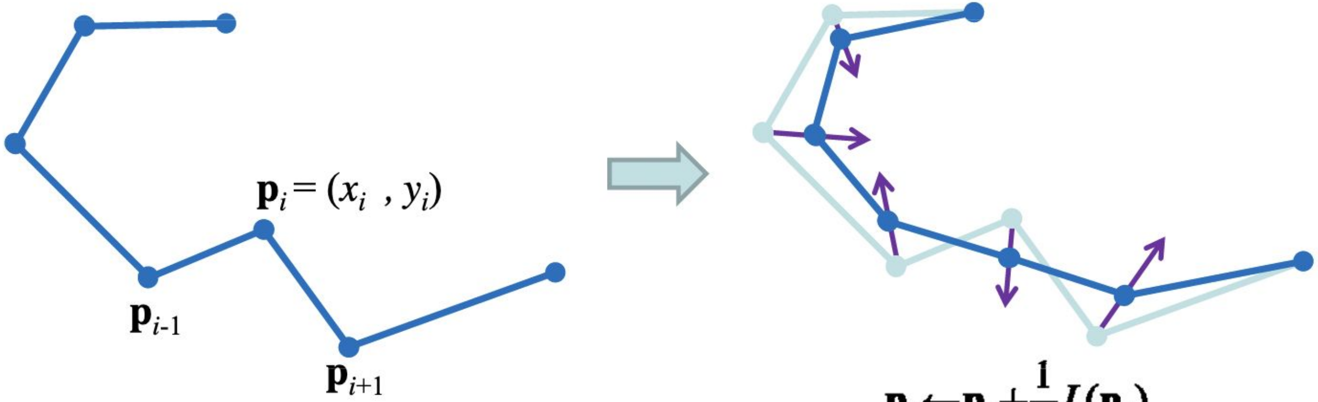


Figure 7: Comparison of smoothing weights. Columns 2 and 3 are generated using the weight functions introduced in Section 5, with different scaling factors s. The right column shows the effect of reducing the weights on the (Laplacian) smoothness constraints of high curvature vertices. All results in this figure were generated with $\mathbf{L} = \mathbf{L}_u$.

Laplacian Smoothing



$$\mathbf{p}_i \leftarrow \mathbf{p}_i + \frac{1}{2} L(\mathbf{p}_i)$$

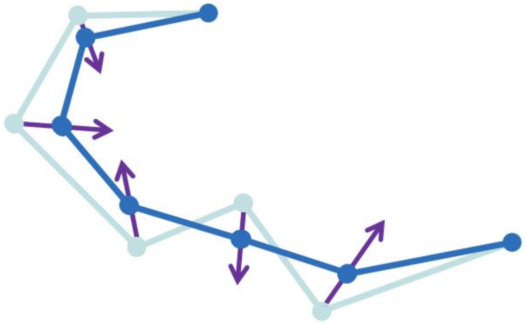
Finite difference discretization of second derivative = Laplace operator in one dimension

$$L(\mathbf{p}_i) = \frac{1}{2}(\mathbf{p}_{i+1} - \mathbf{p}_i) + \frac{1}{2}(\mathbf{p}_{i-1} - \mathbf{p}_i)$$

Algorithm:

Repeat for *m* iterations (for non boundary points):

$$\mathbf{p}_i \leftarrow \mathbf{p}_i + \lambda L(\mathbf{p}_i)$$



For which λ?

$0 < \lambda < 1$

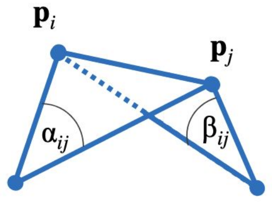
Closed curve converges to?

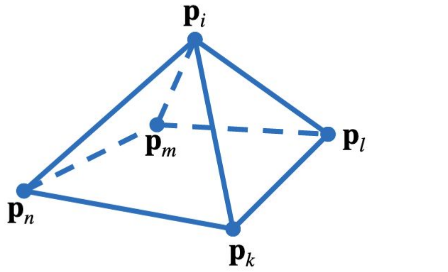
Single point

Cotangent Laplacian

Same weight to both neighboring points P_{i+1}, P_{i-1}. This isn't optimal when smoothing, weighted smoothing works better.

$$L(\mathbf{p}_i)=\frac{1}{2}(\mathbf{p}_{i+1}-\mathbf{p}_i)+\frac{1}{2}(\mathbf{p}_{i-1}-\mathbf{p}_i)$$





$$w_{ij}=\frac{h_{ij}^1+h_{ij}^2}{l_{ij}}=\frac{1}{2}\left(\cot\alpha_{ij}+\cot\beta_{ij}\right)$$

$$L(\mathbf{p}_i)=\frac{1}{\sum_{j\in N_i} w_{ij}}\left(\sum_{j\in N_i} w_{ij}\mathbf{p}_j\right)-\mathbf{p}_i$$

Planar meshes will be invariant to smoothing

Cotanget Laplacian

L_{Lap} maintains the curvature flow, this moves each vertex along its normal, leaving the tangential component unchanged. It also prevents distortion this

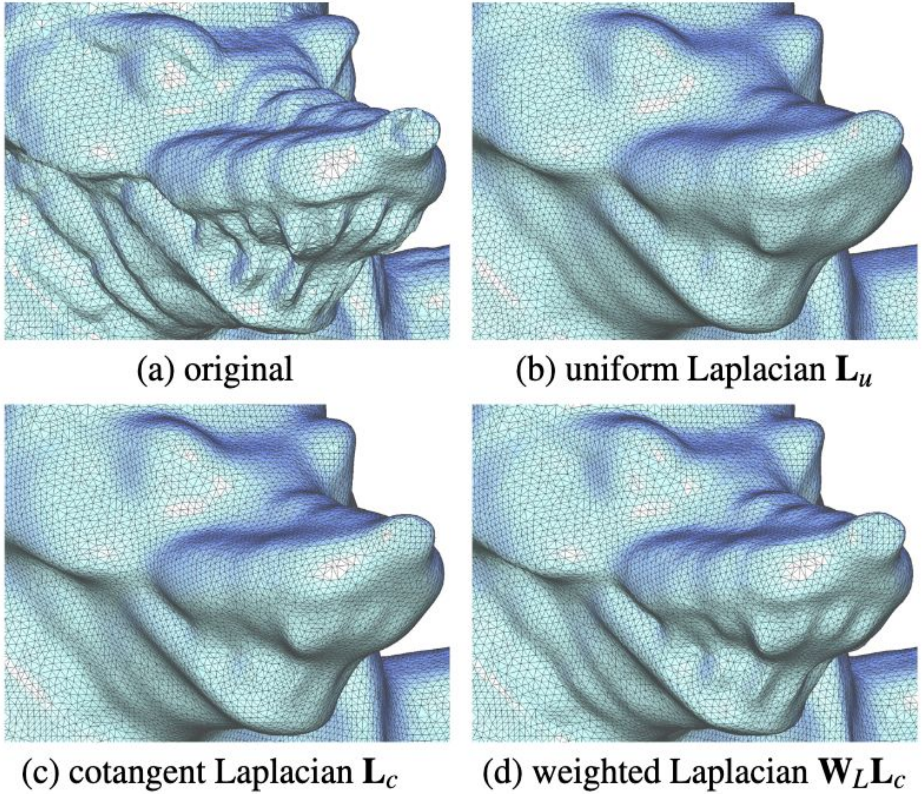


Figure 9: Comparison of umbrella (b) and cotangent (c) discretization of the $\mathbf{L}$ matrix. In (d), Laplacian constraints $\mathbf{L}_c\mathbf{V}'_d = 0$ are relaxed on feature vertices.

Adversarial Loss

The 2D discriminator network is jointly trained with MetaHandleNet and Deform-Net with a classification loss function.

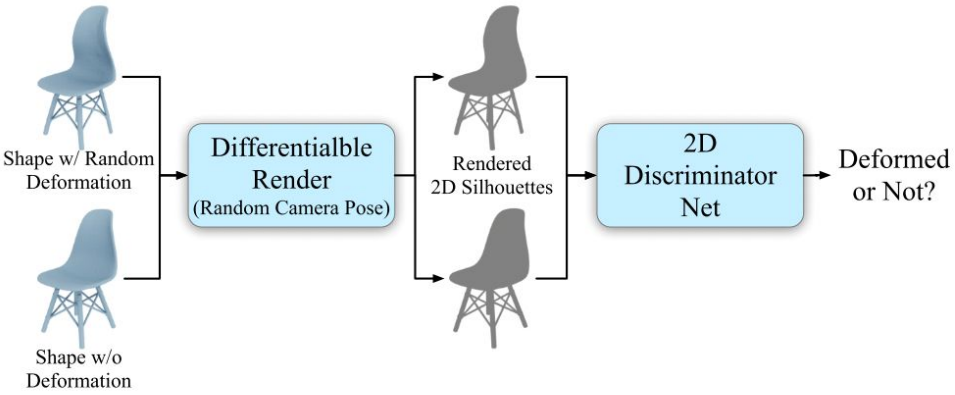


Figure 5: We utilize a soft rasterizer [25] and a 2D discriminator network to penalize unrealistic deformations.

Disentanglement Loss

$$\mathcal{L}_{disen} = \mathcal{L}_{sp} + \mathcal{L}_{cov} + \mathcal{L}_{ortho} + \mathcal{L}_{SVD}.$$

Constraint to properly disentangle the deformation space.

1. L_{sp} encourages the meta-handles Mi and the coefficient vector a to be sparse by penalizing their l1- norm.
2. L_{cov} penalizes the covariance matrix (calculated for each batch) of the coefficients a.
3. L_{ortho} encourages meta-handles to cover different parts of the control-point offsets by penalizing dot products between the meta-handles.
4. L_{SVD} encourages the control points to translate in a single direction within each meta-handle.

Experiments

Eval on 15,522 shapenet models of three categories: chair, table, and car.

Control points, c = 50

Metahandles, m = 15

Table 2: Chamfer distance (×100) and Cotangent Laplacian (×10) between different ablated versions (on chair category). For both metrics, lower is better. DoF indicates degrees of freedom.

Meta-handle / Handle	DoF	$\mathcal{L}_{adv}$	CD↓	CotLap↓
Handle	50×3	w/o	4.78	5.60
Meta-handle	15	w/o	5.76	8.61
Handle	50×3	w/	7.98	7.69
Meta-handle	15	w/	6.28	5.75

Results

For a set of generated shapes A and a set of ground truth shapes B, coverage measures the fraction of the shapes in B that can be roughly represented within A, while MMD (minimum matching distance) measures how well shapes in B can be represented by shapes in A.

A & B both had 500 shapes

Table 1: Coverage (higher is better) and MMD (×100, lower is better) comparison between different methods.

	Chair		Car		Table	
	COV ↑	MMD ↓	COV ↑	MMD ↓	COV ↑	MMD ↓
3DN [39]	32.0%	4.56	46.6%	2.91	30.6%	4.26
CC [10]	51.0%	4.26	50.3%	2.79	50.2%	3.88
NC [46]	54.4%	4.23	66.6%	2.65	44.7%	3.85
Ours	64.6%	4.28	76.5%	2.97	54.9%	3.70

Table 3: Coverage (higher is better) and MMD (×100, lower is better) for different ablated versions (on Chair category).

	COV↑	MMD↓
w/o meta-handle	48.4%	4.69
w/o $\mathcal{L}_{adv}$	56.3%	4.64
w/o $\mathcal{L}_{disen}$	64.1%	4.14
Ours	64.6%	4.28