Deep Feature Deformation Weights

Richard Liu
University of Chicago
guanzhi@uchicago.edu

Itai Lang
University of Chicago
itailang@uchicago.edu

Rana Hanocka
University of Chicago
ranahanocka@uchicago.edu

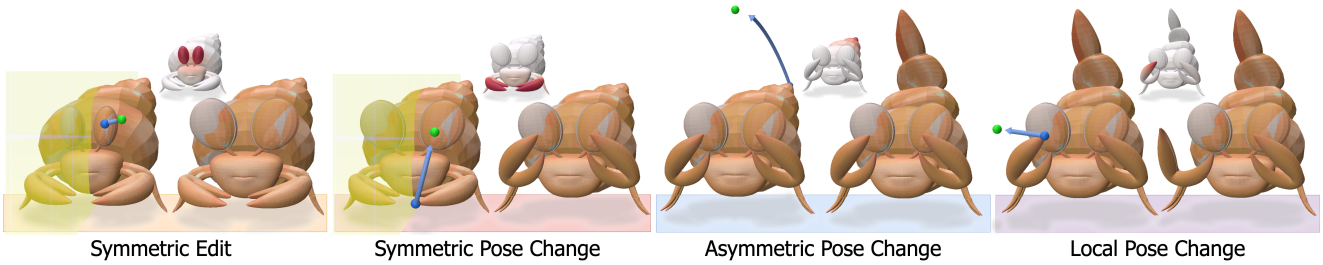

Figure 1. Our DFD framework enables flexible control over a wide range of deformations in real time. Symmetric deformations may be achieved through our automatically detected symmetry plane (yellow).

Abstract

Handle-based mesh deformation is a classic paradigm in computer graphics which enables intuitive edits from sparse controls. Classical techniques are fast and precise, but require users to know ideal handle placement apriori, which can be unintuitive and inconsistent. Handle sets cannot be adjusted easily, as weights are typically optimized through energies defined by the handles. Modern data-driven methods, on the other hand, provide semantic edits but sacrifice fine-grained control and speed. We propose a technique that achieves the best of both worlds: deep feature proximity yields smooth, visual-aware deformation weights with no additional regularization. Importantly, these weights are computed in real-time for any surface point, unlike prior methods which require expensive optimization. We introduce barycentric feature distillation, an improved feature distillation pipeline which leverages the full visual signal from shape renders to make distillation complexity robust to mesh resolution. This enables high resolution meshes to be processed in minutes versus potentially hours for prior methods. We preserve and extend classical properties through feature space constraints and locality weighting. Our field representation enables automatic visual symmetry detection, which we use to produce symmetry-preserving deformations. We show a proof-of-concept application which can produce deformations for meshes up to 1 million faces in real-time on a consumer-grade machine. Project page at https://threedle.github.io/dfd.

1. Introduction

Handle-based deformation frameworks enable intuitive editing with sparse inputs. Traditional methods solve an optimization problem to obtain either a weight matrix for linear blending of handles [16, 39] or the deformed mesh vertices directly [35]. Both types of methods require strategic placement of the handles to obtain desirable deformations [22]. Traditionally, local influence of the handles is enforced through the optimized energy (typically a Laplacian or rigidity energy). Local influence is assumed to be desirable, but in this work we argue that *global/semantic influence* can also be desirable (e.g. co-deformation of chair legs). In fact, prior work provides evidence for this [10, 25, 38, 44]. These data-driven methods offer semantic-aware edits (symmetry/structure preserving), but lack the fine-grained control and speed of traditional methods.

Our work aims to synthesize the strengths of the these competing approaches. Specifically, we desire the speed and fine-grained control of traditional handle-based deformation methods, while simultaneously capturing visual understanding from data priors. This brings us to *Deep Feature Deformation weights* (DFD weights), which use distilled deep features from pretrained 2D models to define the function mapping handle transformations to surface deformations. Importantly, this function is *not* conditioned on the choice of handle set and does not involve solving any optimization problem. We take a simple yet effective approach: we define linear blending weights based on feature similarities. We demonstrate that these weights are robust

across deformation types, handle choice, and shape.

We specifically propose to parameterize the space of weights through a neural field, so that *any* handle can be chosen from ambient space, without the need for expensive re-optimization. DFD weights correlate visually similar structures due to the data prior of 2D pre-trained models. Deformations using our weights are smooth without requiring any additional vertex constraints or regularization.

Each field is optimized per-shape, but distillation is fast and accelerated through a novel *barycentric feature distillation*. This procedure allows feature fields to be rapidly distilled on coarse shapes and transferred to high-resolution counterparts during inference. Existing distillation techniques distill to the mesh resolution. We instead make distillation a function of *render resolution*, and leverage the geometric prior of the mesh to efficiently sample the shape space from each render. We show that with barycentric distillation, feature fields for shapes ranging from 1000 to $>$10 million faces can all be optimized within a few minutes.

Though our weights by default give global deformations, we still account for locality and fixed point constraints. Specifically, we introduce locality through a geodesic-weighting adjustment. Under our framework, point constraints naturally extend to visual constraints, which we dub *feature space constraints.* Visually similar parts constrained by these fixed points are preserved under deformation.

We evaluate our DFD weights on shapes of varying quality, resolution, and type. Our weights are robust to topological deficiencies and our performance is on-par or outperforms all baselines on their specialized datasets. We develop a toy GUI for interactive mesh deformation.

2. Related Work

We divide related handle-based deformation work into traditional methods and data-driven ones. We then address a set of common desirable properties of such methods, propose some alterations, and contextualize our method against relevant baselines in Sec. 2.3.

2.1. Traditional Handle-Based Methods

Handle-based methods offer low-dimensional control structures for performing shape editing or animation tasks. These methods can be roughly categorized by their choice of control structure, the most common being handle points, cages, or skeletal rigs.

Methods which use handle sets either use variational methods to optimize each deformation according to target handle positions ([3, 4, 35], or optimize a weight matrix to perform linear blending of the prescribed handle positions to produce the deformed shape ([16, 39]). Implicit-ARAP [1] is an interesting recent work which extends as-rigid-as-possible optimization to implicit representations.

Cages [13, 19, 20, 24, 41] are similar in that they define control handles which form the nodes of a coarse polytope enclosing the mesh. Each vertex is defined as a linear combination of node positions, the weights of which are typically generalized barycentric coordinates [9].

Skeleton rigs define a hierarchy of bones related through joint rotations, and deformations applied to the bones are transferred to the mesh surface through skinning weights [2, 21, 23, 26, 40]. The hierarchical structure allows for movements made to the extremities to propagate through the kinematic chain, causing larger scale pose changes and shape motions. This property is what makes skeleton rigs preferred in animation and shape-preserving applications.

All traditional methods aim to produce some form of "natural deformation" [4], where in lieu of a well-defined mathematical notion of "natural", distortion minimization becomes the stand-in objective. Distortion minimization is an effective approach to achieving "pose change" types of deformations, where articulated parts are bent or twisted while preserving total shape volume, but these are not the only types of deformations a user may desire. For instance, a user may wish to elongate all the legs of a chair mesh in a symmetric way to preserve the chair structure and function. Data-driven approaches have since emerged to champion these types of surface-distorting, yet desirable, edits.

2.2. Data-Driven Handle-Based Methods

Early data-driven shape deformation works explored the space of semantic design attributes through curated datasets [45]. Recent works use data to predict the parameters of the traditional control structures cited in the previous section. Neural Cages [43] introduces a neural network which predicts both the cage node positions and the corresponding node translations to deform a shape towards a target. Similar works have utilized shape targets for optimizing control structure quantities, such as AlignNet [12], OptCtrlPoints [22], KeypointDeformer [18], DeepMetaHandles [25], and DeformSyncNet [37]. NeuralMLS [34] leverages the smooth prior of neural networks to obtain deformation weights based on input handles.

APAP [44] is a recent method which combines the classical approach of ARAP with the modern approach of supervision from pretrained text-to-image models to generate "plausibility-aware" shape deformations from handle positions and anchor points.

Importantly by learning through data, these works demonstrate user-desirable deformations which are not necessarily near-isometric pose changes. In particular, IWires [10], DeepMetaHandles [25], and APAP [44] show surface distorting transformations (e.g. part scaling and stretching) that preserve global symmetries and shape structure. Our method aligns with this line of work, though we emphasize that our method is ultimately capable of both types of edits,

as demonstrated in Fig. 1.

2.3. Properties of Handle-Based Methods

Most methods have abided by a set of desirable properties for handle-based deformation. As outlined by OptCtrlPoints [22], these properties are:

1. Identity: The original shape must be reconstructed under zero movement of shape handles.
2. Locality: The deformation produced by each individual handle must be local and smooth.
3. Closed-form: The deformed shape is a closed-form expression of the target point transformations
4. Flexibility: The deformation handles and function is defined without any additional information about the shape (e.g. skeleton hierarchy).

We agree with properties 1,3,4 but argue that locality is not essential. Rather, deformations may be global as long as they are smooth. Locality may be preferred depending on the user's intent, but allowing global deformations opens up the possibility of modeling with *semantics* – editing while preserving shape symmetries (visual/intrinsic/extrinsic) and global structure/function. As explained in the previous section, we are not the first work to demonstrate that such deformations could be desirable.

We propose two more desiderata: efficient compute under changing handles and mesh resolution. All cited works require solving an optimization problem to obtain either the weights or the final deformation. These works also scale poorly – at best quadratically with vertex count. This is a fundamental bottleneck towards real-time deformation, as users will frequently iterate over handles. OptCtrlPoints [22] takes an important step towards reducing the bottleneck, but does not resolve the underlying issues of optimization and scaling. Our method makes handle-based recomputation fast by simply assigning weights in terms of feature distances, bypassing the need to solve large linear systems. We downsample shapes using a fast decimation algorithm, and use the coarse renders to optimize feature fields, making us highly robust to resolution (see Fig. 10).

Based on these properties, we place our work in context with the relevant baselines in Sec. 2.3. We emphasize that all existing works must re-solve an optimization problem for new handles ("New Handle w/o Optim."), and ours is the only method which affords both local and global control.

3. Method

DFD weights map control handle deformations to the rest of the shape through linear blending, enabling real-time and interactive mesh deformations (3.1). We accelerate neural field optimization (pre-processing) through barycentric feature distillation, which maximizes the feature signal extracted from each render and allows efficient distillation of shapes with millions of elements (3.2). Additionally, we demonstrate locality control, feature anchoring, and automatic symmetry evaluation capabilities (3.3).

	Global Semantics	Local Control	Size Robust	New Handle w/o Optim.
Classical				
ARAP [35]	✗	✓	✗	✗
Biharmonic [39]	✗	✓	✗	✗
Neural				
APAP [44]	✓	✗	✗	✗
DMH [25]	✓	✗	✗	✗
NeuralMLS [34]	✗	✓	✓	✗
DFD (Ours)	✓	✓	✓	✓

Table 1. **Properties of Control-Based Deformation Methods.** "Global Semantics" is whether the method can make visual-driven global deformations. "Local Control" is the ability to smoothly deform the surface local to the transformed handle. "Size Robust" is whether the weight computation scales robustly with mesh resolution. "New Handle w/o Optim." is whether the method solves an optimization problem with every update to the control handle set. Our method (DFD) is the only work which incorporates these desiderata.

3.1. Preliminaries

Given a mesh $\mathcal{M} = (V, F)$, with vertices $V \subseteq \mathbb{R}^3$ and faces F, our method predicts a DFD weight matrix $\mathcal{W} \in \mathbb{R}^{n \times n}$ ($n = |V|$) which defines the basis for a deformation subspace of the mesh. For a given set of deformations $\{D_1, \ldots, D_K\}$ (affine transformations assigned to vertices $v_{j_1}, \ldots, v_{j_k}$), we can compute the final position of vertex i through an extended form of standard linear blending.

$$V_i' = (\max(1 - \sum_{k=1}^{K} \mathcal{W}_{ij_k}, 0) D_0 + \sum_{k=1}^{K} \mathcal{W}_{ij_k} D_k) V_i \quad (1)$$

The term $\max(1 - \sum_{j=1}^{K} \hat{\mathcal{W}}_{ij}, 0) D_0$ represents a simulated control point which has some default transformation D_0. We assume in all our results that D_0 is the identity transformation, but can be set to some default affine transformation to be applied globally if desired. The max function is applied to avoid negative weights on the simulated control point, which can result in unintuitive behavior [17].

By including a simulated control point, we satisfy partition of unity for sparse handles (where $\sum_{k=1}^{K} \mathcal{W}_{ij_k} \leq 1$). This property is primarily enforced to guarantee no movement under identity transformation, and otherwise does not serve much practical purpose. Past work has demonstrated that dropping this constraint yields negligible changes to deformation quality (fig. 6 in [16]). We also show in supplemental Fig. 21 that our method also produces reasonable results for dense handle configurations, where partition of

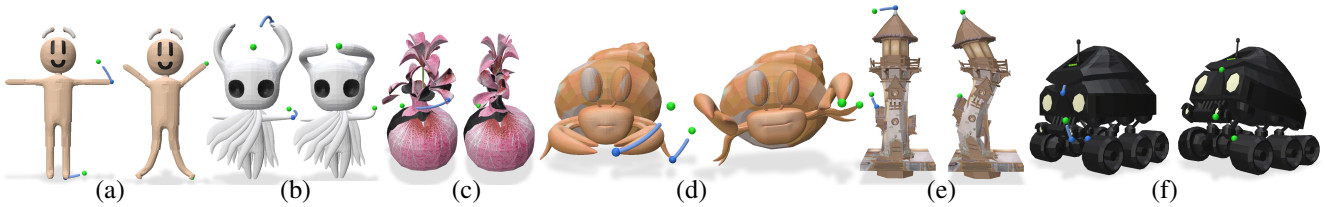

Figure 2. **General Affine Transformations.** DFD weights effectively interpolate affine transformations to generate plausible pose changes. We can generate a variety of deformations by leveraging detected symmetries (a,b) (3.3.3) and localization control (b, d) (3.3.2).

Figure 3. **Translation Edits.** Edit results from different handle (blue) translations to target locations (green) using translations prescribed by APAP-Bench 3D. [44]. Weights are visualized as heatmap insets. A larger set of results from the dataset are shown in the supplemental.

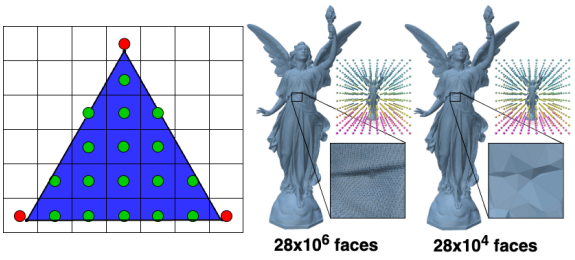

Figure 4. **Barycentric Feature Distillation.** (Left) Existing feature distillation methods use only pixels intersected by raster vertices. Barycentric distillation takes advantage of the known geometry to supervise with features at all pixels intersected by a triangle. (Right) High resolution meshes look visually unchanged even with extreme reduction using QEM (99%). Consequently their feature fields are virtually identical (PCA insets). We opt to distill features using renders of low-resolution meshes, and use them to deform meshes at their original resolution.

unity is not guaranteed. Linear blending of affine transformations follows the same approach as past work [16, 39], which themselves follow a long tradition of linear averaging deformations in skeletal animation [17].

3.2. Barycentric Feature Distillation

Barycentric feature distillation is motivated by the observation that existing work on feature distillation for surfaces waste much of the visual signal from renders [7, 42]. Fig. 4 (left) demonstrates this with a simple example of a rasterized triangle on the image plane. Current works distill features onto mesh vertices, which means only the pixels containing a vertex (red dots) contribute to features on the surface. Using a neural field and the triangle geometry, we can instead extract the 3D coordinates for all the pixels covered by the triangle (green), and optimize our neural field on the features from these pixels. Formally, we can use the rasterization process to define a function which assigns a 3D surface coordinate to each render pixel (i, j).

$$P_{ij} = B(i,j)T(i,j) \tag{2}$$

$T(i,j)$ is a matrix where the columns are the vertex positions of the triangle covering the center of pixel (i,j) and $B(i,j)$ is a row vector containing the barycentric coordinates of the surface point at the pixel center. Note that P_{ij} is only defined for pixels which have triangles intersecting the center. Let Z_{ij} be encoded feature for pixel (i,j). The training loss for our neural field Φ is

$$\mathcal{L} = \sum_{(i,j)\in\Omega} \left\| \Phi(P_{ij}) - \frac{Z_{ij}}{||Z_{ij}||} \right\|^2 \tag{3}$$

where Ω contains the set of all pixels which have centers covered by a raster triangle. This method of supervising on features-per-pixel rather than features-per-vertex results in *complete disentanglement* of the neural field sampling and the mesh resolution. Two meshes occupying the same volume in the field will induce the *same sampling resolution*.

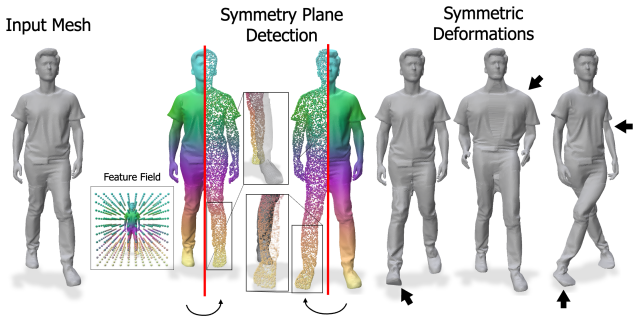

Figure 5. **Visual Symmetry Detection.** Our neural field representation returns visual features for arbitrary 3D points. This allows us to evaluate candidate symmetry planes on points *away from* the shape surface, which we use to identify symmetry planes where visual features are reflected. This identification is not constrained by extrinsic geometry or isometry constraints. Our symmetric deformations are generated by manipulating only *one side of the shape.*

Though rendering is typically fast, it becomes a non-trivial bottleneck for high resolution shapes. We observe that high-resolution meshes do not visually change much even with aggressive decimation, which motivates our decision to first downsample the shape with quadric error simplification (QEM) prior to rendering. We find empirically QEM to be substantially faster than rendering for shapes at most resolutions. In Fig. 4 (right), a single render of the Lucy mesh (28 million faces) with Pytorch3D [31] takes 5.7 minutes, whereas quadric mesh simplification with 99% reduction and then rendering takes 3.7 seconds. Despite the reduction, the two meshes look visually identical. Distilling with barycentric features, however, is critical to ensuring the neural field distilled on the coarse mesh is effective for the mesh at its original resolution. We show in supplemental Fig. 13 that standard vertex-based distillation on the coarse shape results in much worse DFD weights, even for the same number of training FLOPs.

3.3. Feature Proximity Weighting

Given a trained feature field Φ and mesh with vertices V, we have unit-norm distilled features $Z = \Phi(V)$. For some similarity function $F(x, y)$ which maps points x, y to a similarity value between -1 and 1, our weights are defined as

$$\mathcal{W}_{ij} = \max(F(Z_i, Z_j), 0) \tag{4}$$

F can be any similarity function which falls within [-1, 1], and we find a simple $L2$-based function works well.

$$F(Z_i, Z_j) = 1 - ||Z_i - Z_j||_2 \tag{5}$$

For unit norm features, F is 1 when Z_i and Z_j are the same and -1 when the features are the furthest apart. Negative weights are undesirable [16], so we interpret all negative weights to represent unrelated features and clamp them to 0. For vertices i, j, the weight W_{ij} is given by Eq. 4. Note that Z is precomputed from a single forward pass of Φ, so weights for new handles are obtained from computing F. Our choice of linear F (5) makes this weight calculation linear with respect to both vertex and handle count.

Figure 6. **Feature Space Constraints.** Fixed points in our framework constrain points with similar deep features. For example, we place a fixed point on the robot treads, which prevents it from twisting with the torso.

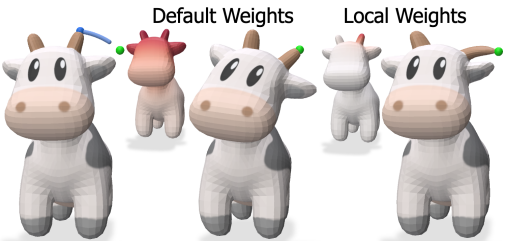

Figure 7. **Locality Weighting.** Locality weighting (7) allows precise control over the spatial extent of the target deformation. With default DFD weights, the deformation is a reasonable rotation of the cow head, and with locality weighting the same deformation instead bends just the horn.

3.3.1. Feature Space Constraints

We propose a simple extension to our framework to account for point constraints. For fixed vertex indices $\{p_1, \ldots, p_k\}$, we can update $\mathcal{W}$ such that

$$\mathcal{W}_{ij} = \max(\mathcal{W}_{ij} - \max_{p_k}(W_{ip_k}), 0) \tag{6}$$

Fixed points update the weights for vertex i by subtracting the maximum weight between i and the fixed points. This ensures $W_{ik} \leq 0$ for all fixed points k. The outer max ensures none of these adjusted weights become negative, so $W_{ik} = 0$. Practically, all points with similar visual features to the fixed points will be constrained. We demonstrate the effectiveness of these constraints in Fig. 6.

3.3.2. Locality Weighting

A user may not always want deformations to result in global shape change. We introduce a user-defined parameter λ which determines localization extent. If $\lambda > 0$, we update the weight matrix to localized weights $\mathcal{W}'$

$$\mathcal{W}'_{ij} = \mathcal{W}_{ij}(1 - G_{ij})^{\lambda} \tag{7}$$

where G_{ij} is the geodesic distance between vertex i and vertex j [33] normalized such that the maximum geodesic distance is 1. As geodesic distance increases, the weights between points to the handle drops off to 0 with speed determined by λ. Fig. 7 shows that by allowing user control over the dropoff, specific local features can be deformed.

3.3.3. Visual Symmetry Detection

Our neural field representation enables automatic evaluation of candidate symmetry planes. For a candidate plane P, let V^+ be the set of all mesh vertices on the positive side of the plane normal, and let V^- be the vertices on the negative side. Let R_P be the function which reflects points across P. We say there exists a visual symmetry along P if

$$\frac{1}{|V|}\Big(\sum_i ||\Phi(V_i^+) - \Phi(R_P(V_i^+))||_2 + \sum_j ||\Phi(V_j^-) - \Phi(R_P(V_j^-))||_2\Big) < \epsilon$$

For symmetry-preserving deformations along plane P, we update the right-side term in Eq. (1)

$$\sum_{k=1}^{K} \mathcal{W}_{ij_k} \Rightarrow \sum_{k\in\Omega_i^+} \mathcal{W}_{j_k i} D_k + \sum_{k\in\Omega_i^-} \mathcal{W}_{j_k i} R_P(D_k) \quad (8)$$

Ω_i^+ is the set of handles on the same side of P as vertex i, and vice-versa for Ω_i^-. Handles in Ω_i^- have their deformations reflected across P before being applied to V_i.

Since similarity is computed based on visual features, the detected symmetries are not *geometric*, but rather *visual*. The closest related concept is intrinsic symmetry [28], in which identical geometric parts in different poses are identified as symmetric. Our notion of visual symmetry takes this one step further, where the parts need not be intrinsically identical, but simply visually.

Fig. 5 shows an example of a shape which has visual but not extrinsic symmetry. Our neural field representation allows for evaluation of features that are not on the shape surface. In this example, we identify a vertical symmetry plane that enables symmetric deformations, such as switching the leg poses, broadening the shoulders, and even crossing the legs, by manipulating just one side of the shape.

For all results shown, we evaluate symmetry planes spanning the primary axes and specify when the results apply detected symmetries. We set $\epsilon = 0.1$ for all results.

4. Experiments

We evaluate against the baselines in Sec. 2.3, and use the datasets from APAP (APAP-Bench 3D) [44] and DeepMetaHandles (DMH) [25] (1,363 shapes from the cars, tables, and chairs categories in ShapeNet [5]). Some validation shapes are non-manifold (causing DMH, ARAP, and biharmonic coordinates to fail), so we generate manifold versions using [15]. Our method is robust to non-manifoldness, as shown in supplemental Fig. 17. All DFD weights are distilled from DINOv2 [27]. We QEM simplify shapes with over 50k faces to $\leq$50k for barycentric feature distillation and do not simplify lower resolution shapes.

APAP-Bench 3D comes with prescribed handles and target positions. The DMH dataset comes with basis vectors and handles predicted by the trained DMH model. For our method, we use a single handle and target position by taking the largest-norm offset from the predicted basis.

Biharmonic coordinates require a tet mesh, so we tetrahedralize all shapes using FTetWild [14]. We correspond the original surface and the tet mesh with nearest neighbors, and we transfer the handles accordingly. All biharmonic results are visualized with the deformed tet mesh. Texture information is not transferred well, so we exclude textures.

4.1. Qualitative Results

Affine Transformations. DFD weights smoothly interpolate affine transformations while respecting shape semantics. We show affine deformations on shapes from Objaverse [6] in Fig. 2, which demonstrate the many axes of control within our framework. We show symmetric deformations enabled by our symmetry detection (a,b) (3.3.3), local deformations using locality weighting (b,d) (3.3.2), and pose changes using our base weights (a-c,d,f).

Translation-Based Editing. Fig. 3 shows translation-based shape edits using DFD weights. These weights propagate edits to visually relevant features, such as the nose of the Moai sculpture, the arms of the robot, the chair cushion, and the horns on the demon mask.

Qualitative Comparisons. Fig. 8 compares the same shapes and handle transformations shown in APAP [44] across the baselines. Our weights perfectly correspond key features on the shape, allowing for uniform stretching of the fox ears (row 2). Furthermore, symmetry detection allows us to generate uniform and symmetric scaling of the owl head and axe blades (rows 1,3). APAP does not consistently preserve symmetry due to its reliance on a noisy score distillation signal. Both ARAP and biharmonic coordinates are Laplacian-based, so the deformations are unsurprisingly non-visual and often result in global rotations/offsets due to poor placement of fixed points. Though handle sets in rows 1 and 2 are similarly placed (fixed points under the feet), the behavior of ARAP/biharmonic is inconsistent, demonstrating the difficulty in choosing performant handle sets. We show additional comparisons, with and without 0.01-sampling, in supplemental figures 14 and 15.

Fig. 9 shows deformation comparisons on the DMH dataset. DMH uses biharmonic coordinates as its deformation framework, so they are synonymous in this comparison. We use handle sets and deformations predicted by

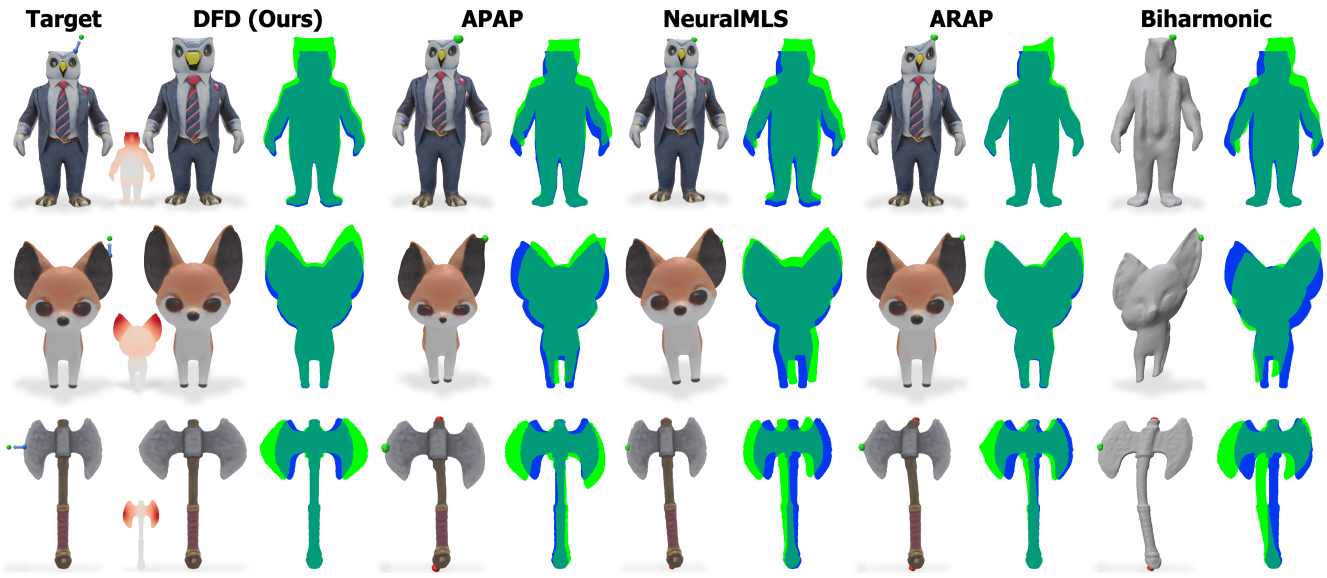

Figure 8. **APAP-Bench 3D Comparison.** Qualitative results using the shapes and handles shown in the APAP paper. DFD uses the single prescribed handle translation *with no fixed points*. All other methods require many constraints to achieve reasonable deformations. These are generated with 0.01-ball sampling of handles/fixed points, following APAP. Control handle initial positions are in blue and target positions in green. Fixed points are in red. Transparent silhouettes show deformation change, where the initial mesh is blue and final mesh is green. DFD weights are shown in small heatmap insets. Biharmonic coordinates requires a tetrahedral mesh, whereby the texture is lost in conversion. Additional comparisons shown in supplemental figures 14 and 15, with and without 0.01-ball sampling, respectively.

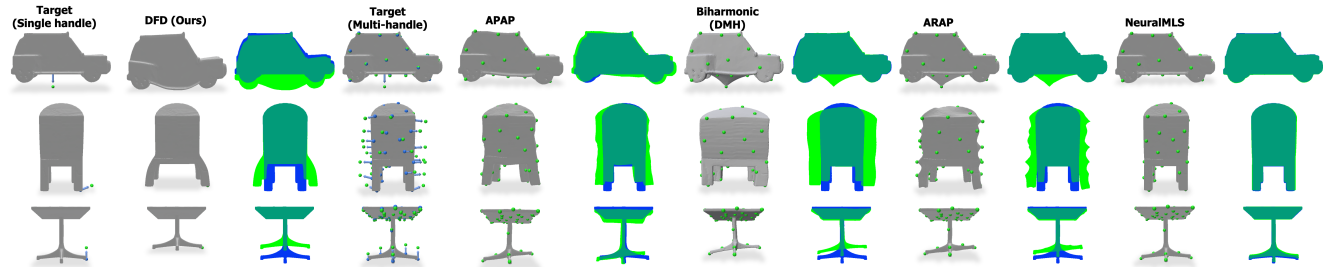

Figure 9. **DMH Comparison.** We compare against baselines using the DMH dataset [25]. The baselines use 50 control handles with offsets predicted by DMH, while our method takes the single handle with highest-norm offset. DMH uses biharmonic coordinates as its deformation model, so they are equivalent. Our method generates deformations that are just as smooth as DMH and more visual/structure-aware.

DMH, making this a very strong baseline. Nevertheless, our method demonstrates smoothness on par with DMH with greater visual awareness. DMH will generate global rescaling of the chair, whereas our method can restrict the deformation to the legs (row 2). DMH will sometimes over-constrain (sharp artifacts in row 1) and lacks in symmetry awareness (uneven legs in row 3). Additional comparisons are in supplemental Fig. 16.

4.2. Quantitative Results

Timing. We conduct a timing analysis of DFD against the baselines to quantify our efficiency claims. To cover all methods, we measure timing of 3 phases (which may not all apply to each method): preprocess, bind, and pose.

- **Preprocess** time involves all the steps involved prior to computing the handle weights (bind). Biharmonic coordinates converts the surface mesh to a tetrahedral mesh, and prefactorizes the bilaplacian system for the linear solve. APAP renders the shape and finetunes a LORA model, and DFD (our method) distills a feature field.
- **Bind** time is the time taken to compute handle weights. Biharmonic solves a linear system over the tet elements, DFD (our method) does a feedforward pass and feature distance calculation, and NeuralMLS trains a neural field.
- **Pose** time involves computing the final deformed mesh. Both DFD and biharmonic leverage the speed of linear blending, whereas ARAP solves a linear system, APAP conducts score distillation sampling [29], and Neu-

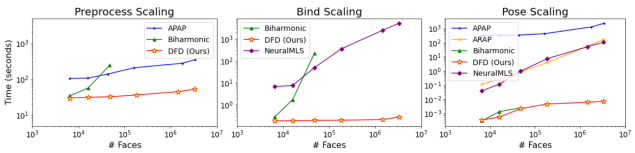

Figure 10. **Method Timing.** We compare timings across three stages (preprocess, bind, pose) for our datasets remeshed to different resolutions. Biharmonic coordinates fails in both the tetrahedralization preprocess and bind steps at resolutions higher than 10^5 faces. Our method (DFD) is just as fast as biharmonic coordinates at the lowest resolution, and scales better than all methods across all phases. Our preprocess time is robust to mesh resolution due to barycentric feature distillation (3.2). DFD also demonstrates sublinear scaling in both bind and pose time.

ralMLS solves a moving least squares problem.

We remesh each shape in our dataset to resolutions between 10^3 to 10^7 faces (∼6,000 shapes) and compute method timings over them (Fig. 10, log scale). Biharmonic coordinates scale very poorly in the tetrahedralization and bind phases, and fails at resolutions past 10^5 faces. Our method, on the other hand, demonstrates robust scaling for all phases across all resolutions. APAP also demonstrates good scaling but has a base runtime several orders of magnitude larger than DFD. ARAP and NeuralMLS exhibit both higher base runtimes and worse scaling than our method.

Though the analysis above already demonstrates our method's efficiency, it doesn't take into account the fact all other methods must re-optimize for new handles. OptCtrl-Points [22] is a recent method that attempts to make the re-solve for biharmonic coordinates more efficient. We evaluate this accelerated re-solve against our method for new handle bind times in supplemental Sec. A and show our method is still several orders of magnitude faster.

User Study. We conduct a user study using deformations from our evaluation datasets and 6 additional large handle deformations. To show our weights robustly interpolate both translations and rotations, we evaluate both translation-only (DFD-T) and affine (DFD-A) variants of our method. Users (N=37) selected the 2 "most desirable" deformations for each example, and we report the frequency each method is chosen in Sec. 4.2. Both versions of our method are significantly preferred over the baselines (82% for DFD-T and 79% for DFD-A). Screenshots are in the supplemental Fig. 23. To measure perceptual realism, we conduct a second user study (N=23) which asks users to select the "deformation which is most realistic and best preserves shape detail". Our method is chosen by users 64% of the time, ARAP 17.7%, NeuralMLS 15.2%, biharmonic 2.2%, and APAP 0.93%. Additional detail in Sec. K.

DFD-T	**DFD-A**	ARAP	Biharmonic	APAP	NMLS
82%	**79%**	19%	3%	4%	11%

Table 2. **User Study.** We evaluate the translation (DFD-T) and affine (DFD-A) variants of DFD against baselines. Users (N=37) select the top 2 deformations for each example, and we report the frequency each method is chosen. NMLS stands for NeuralMLS.

4.3. Ablations

Barycentric feature distillation. We ablate on barycentric distillation in supplemental Fig. 13. Specifically, we take the same approach as prior work and supervise the neural field solely on pixels which contain a vertex. We distill using the same decimated mesh and train for additional iterations to match the total # FLOPs trained with under barycentric distillation. Despite this, the resulting DFD field produces weights which are neither smooth nor visual-aware on the original resolution shapes.

Different image encoders. We explore DFD weights extracted from different modern image models and find that they give surprisingly similar deformation results. Specifically we find that different image features tend to correlate the same structures, which indicates a convergence in semantic understanding of these different models. We show these results in supplemental Fig. 11 and Fig. 12.

5. Conclusion

Deep Feature Deformation generates deformation weights using feature distances. These weights, without regularization, yield smooth and shape-preserving deformations. Barycentric feature distillation ensures our distillation is fast and resolution-agnostic linear blending enables interactive deformation, and the field representation allows visual symmetry detection. We expose classical axes of control through locality weighting and feature space constraints. Unlike prior methods we incorporate new handles without re-optimization, taking an important step towards true user-interactivity, which we demonstrate through a proof-of-concept GUI (supplemental videos).

Limitations. DFD weights are distilled on high resolution shapes in around a minute but still require per-shape optimization. Linear blending of extreme deformations has known issues (e.g. volume collapse) [17] we do not re-solve.

6. Acknowledgements

This project was funded by NSF 2402894, 2304481, the United States - Israel Binational Science Foundation (BSF) 2022363, gifts from Adobe, Snap, Google, and The Bennett Family AI + Science Collaborative Research Program.

References

[1] Daniele Baieri, Filippo Maggioli, Emanuele Rodolà, Simone Melzi, and Zorah Lähner. Implicit-arap: Efficient handle-guided neural field deformation via local patch meshing. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. 2, 3

[2] Ilya Baran and Jovan Popović. Automatic rigging and animation of 3d characters. *ACM Transactions on graphics (TOG)*, 26(3):72–es, 2007. 2

[3] Mirela Ben-Chen, Ofir Weber, and Craig Gotsman. Variational harmonic maps for space deformation. *ACM Trans. Graph.*, 28(3), 2009. 2

[4] Mario Botsch and Olga Sorkine. On linear variational surface deformation methods. *IEEE Transactions on Visualization and Computer Graphics*, 14(1):213–230, 2008. 2

[5] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. 6

[6] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects, 2022. 6

[7] Niladri Shekhar Dutt, Sanjeev Muralikrishnan, and Niloy J. Mitra. Diffusion 3d features (diff3f): Decorating untextured shapes with distilled semantic features. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4494–4504, 2024. 4

[8] Niladri Shekhar Dutt, Sanjeev Muralikrishnan, and Niloy J Mitra. Diffusion 3d features (diff3f): Decorating untextured shapes with distilled semantic features. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4494–4504, 2024. 1

[9] Michael S. Floater. Generalized barycentric coordinates and applications. *Acta Numerica*, 24:161–214, 2015. 2

[10] Ran Gal, Olga Sorkine, Niloy J. Mitra, and Daniel Cohen-Or. iwires: an analyze-and-edit approach to shape manipulation. *ACM Trans. Graph.*, 28(3), 2009. 1, 2

[11] Michael Garland and Paul S Heckbert. Surface simplification using quadric error metrics. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 209–216, 1997. 5

[12] Rana Hanocka, Noa Fish, Zhenhua Wang, Raja Giryes, Shachar Fleishman, and Daniel Cohen-Or. Alignet: partial-shape agnostic alignment via unsupervised learning. *ACM Transactions on Graphics (TOG)*, 38(1):1, 2018. 2

[13] Kai Hormann and Natarajan Sukumar. Maximum entropy coordinates for arbitrary polytopes. In *Computer Graphics Forum*, pages 1513–1520. Wiley Online Library, 2008. 2

[14] Yixin Hu, Teseo Schneider, Bolun Wang, Denis Zorin, and Daniele Panozzo. Fast tetrahedral meshing in the wild. *ACM Trans. Graph.*, 39(4), 2020. 6

[15] Jingwei Huang, Hao Su, and Leonidas Guibas. Robust watertight manifold surface generation method for shapenet models. *arXiv preprint arXiv:1802.01698*, 2018. 6

[16] Alec Jacobson, Ilya Baran, Jovan Popović, and Olga Sorkine. Bounded biharmonic weights for real-time deformation. *ACM Trans. Graph.*, 30(4), 2011. 1, 2, 3, 4, 5

[17] Alec Jacobson, Zhigang Deng, Ladislav Kavan, and JP Lewis. Skinning: Real-time shape deformation. In *ACM SIGGRAPH 2014 Courses*, 2014. 3, 4, 8

[18] Tomas Jakab, Richard Tucker, Ameesh Makadia, Jiajun Wu, Noah Snavely, and Angjoo Kanazawa. Keypointdeformer: Unsupervised 3d keypoint discovery for shape control. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020. 2

[19] Pushkar Joshi, Mark Meyer, Tony DeRose, Brian Green, and Tom Sanocki. Harmonic coordinates for character articulation. *ACM transactions on graphics (TOG)*, 26(3):71–es, 2007. 2

[20] Tao Ju, Scott Schaefer, and Joe Warren. Mean value coordinates for closed triangular meshes. In *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*, pages 223–228. 2023. 2

[21] Ladislav Kavan, Steven Collins, Jiří Žára, and Carol O'Sullivan. Skinning with dual quaternions. In *Proceedings of the 2007 symposium on Interactive 3D graphics and games*, pages 39–46, 2007. 2

[22] Kunho Kim, Mikaela Angelina Uy, Despoina Paschalidou, Alec Jacobson, Leonidas J Guibas, and Minhyuk Sung. Optctrlpoints: Finding the optimal control points for biharmonic 3d shape deformation. In *Computer Graphics Forum*, page e14963. Wiley Online Library, 2023. 1, 2, 3, 8

[23] Binh Huy Le and Jessica K Hodgins. Real-time skeletal skinning with optimized centers of rotation. *ACM Trans. Graph.*, 35(4):37–1, 2016. 2

[24] Yaron Lipman, David Levin, and Daniel Cohen-Or. Green coordinates. *ACM transactions on graphics (TOG)*, 27(3): 1–10, 2008. 2

[25] Minghua Liu, Minhyuk Sung, Radomir Mech, and Hao Su. Deepmetahandles: Learning deformation meta-handles of 3d meshes with biharmonic coordinates. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12–21, 2021. 1, 2, 3, 6, 7

[26] Nadia Magnenat-Thalmann, Richard Laperrière, and Daniel Thalmann. Joint-dependent local deformations for hand animation and object grasping. In *Proceedings on Graphics interface'88*, pages 26–33, 1989. 2

[27] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision, 2024. 6, 1

[28] Maks Ovsjanikov, Jian Sun, and Leonidas Guibas. Global Intrinsic Symmetries of Shapes. *Computer Graphics Forum*, 27(5):1341–1348, 2008. 6

[29] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv*, 2022. 7

[30] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021. 1

[31] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3d deep learning with pytorch3d. *arXiv:2007.08501*, 2020. 5

[32] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollár, and Christoph Feichtenhofer. Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024. 1

[33] Nicholas Sharp, Yousuf Soliman, and Keenan Crane. The vector heat method. *ACM Trans. Graph.*, 38(3), 2019. 6

[34] M Shechter, R Hanocka, G Metzer, R Giryes, and D Cohen-Or. Neuralmls: Geometry-aware control point deformation. 2022. 2, 3

[35] Olga Sorkine and Marc Alexa. As-rigid-as-possible surface modeling. In *Proceedings of EUROGRAPHICS/ACM SIGGRAPH Symposium on Geometry Processing*, pages 109–116, 2007. 1, 2, 3

[36] Bane Sullivan and Alexander Kaszynski. PyVista: 3D plotting and mesh analysis through a streamlined interface for the Visualization Toolkit (VTK). *Journal of Open Source Software*, 4(37):1450, 2019. 5

[37] Minhyuk Sung, Zhenyu Jiang, Panos Achlioptas, Niloy J. Mitra, and Leonidas J. Guibas. Deformsyncnet: Deformation transfer via synchronized shape deformation spaces. *ACM Trans. Graph.*, 39(6), 2020. 2

[38] Weiyue Wang, Duygu Ceylan, Radomir Mech, and Ulrich Neumann. 3DN: 3D Deformation Network. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1038–1046, Long Beach, CA, USA, 2019. IEEE. 1

[39] Yu Wang, Alec Jacobson, Jernej Barbič, and Ladislav Kavan. Linear subspace design for real-time shape deformation. *ACM Transactions on Graphics*, 34(4):1–11, 2015. 1, 2, 3, 4

[40] Ofir Weber, Olga Sorkine, Yaron Lipman, and Craig Gotsman. Context-aware skeletal shape deformation. In *Computer Graphics Forum*, pages 265–274. Wiley Online Library, 2007. 2

[41] Ofir Weber, Mirela Ben-Chen, Craig Gotsman, and Kai Hormann. A complex view of barycentric mappings. In *Computer Graphics Forum*, pages 1533–1542. Wiley Online Library, 2011. 2

[42] Thomas Wimmer, Peter Wonka, and Maks Ovsjanikov. Back to 3d: Few-shot 3d keypoint detection with back-projected 2d features. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024. 4

[43] Wang Yifan, Noam Aigerman, Vladimir G Kim, Siddhartha Chaudhuri, and Olga Sorkine-Hornung. Neural cages for detail-preserving 3d deformations. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 75–83, 2020. 2

[44] Seungwoo Yoo, Kunho Kim, Vladimir G Kim, and Minhyuk Sung. As-plausible-as-possible: Plausibility-aware mesh deformation using 2d diffusion priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4315–4324, 2024. 1, 2, 3, 4, 6

[45] Mehmet Ersin Yumer, Siddhartha Chaudhuri, Jessica K. Hodgins, and Levent Burak Kara. Semantic shape editing using deformation handles. *ACM Trans. Graph.*, 34(4), 2015. 2