Semantic Shape Editing with Parametric Implicit Templates

Uday Kusupati*
EPFL, Adobe Research
Switzerland
uday.kusupati@epfl.ch

Mathieu Gaillard
Adobe Research
United States of America
gaillard@adobe.com

Jean-Marc Thiery
Adobe Research
France
jthiery@adobe.com

Adrien Kaiser
Adobe Research
France
akaiser@adobe.com

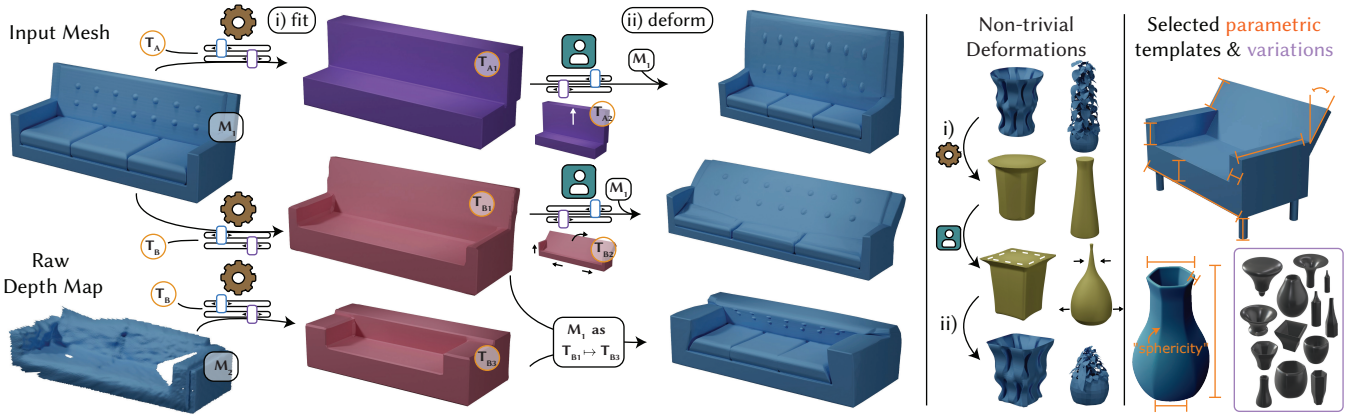

Figure 1: Left: We i) fit a parametric implicit surface template to an input mesh M_1 (GEAR+SLIDERS), and ii) deform M_1 by editing the template parameters (USER+SLIDERS). The top rows show deformations of M_1 using templates T_A and T_B. In the 3^{rd} row, T_B is fit to an input noisy scan M_2 to guide M_1's deformation: We retarget mesh M_1 onto mesh M_2 while following the *semantics* carried by T_B ($T_{B1} \mapsto T_{B3}$). Right: Our method allows using common implicit template models enabling non-trivial deformations while being controllable with a few intuitive parameters that can generate rich variations of a shape category.

ABSTRACT

We propose a semantic shape editing method to edit 3D triangle meshes using parametric implicit surface templates, benefiting from the many advantages offered by analytical implicit representations, such as infinite resolution and boolean or blending operations. We propose first a template fitting method to optimize its parameters to best capture the input mesh. For subsequent template edits, our novel mesh deformation method allows tracking the template's 0-set even when featuring anisotropic stretch and/or local volume change. We make few assumptions on the template implicit fields and only strictly require continuity. We demonstrate applications to interactive semantic shape editing and semantic mesh retargeting.

*Work partly done during an internship at Adobe Research

SIGGRAPH Conference Papers '24, July 27-August 1, 2024, Denver, CO, USA

ACM ISBN 979-8-4007-0525-0/24/07…$15.00
https://doi.org/10.1145/3641519.3657421

CCS CONCEPTS

• **Computing methodologies** → **Shape modeling**; *Parametric curve and surface models*; *Mesh models.*

KEYWORDS

implicit fields, parametric templates, mesh deformation

ACM Reference Format:
Uday Kusupati, Mathieu Gaillard, Jean-Marc Thiery, and Adrien Kaiser. 2024. Semantic Shape Editing with Parametric Implicit Templates. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers '24 (SIGGRAPH Conference Papers '24), July 27-August 1, 2024, Denver, CO, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3641519.3657421

1 INTRODUCTION

Implicit Modeling has been extensively researched and used by 3D artists for decades. In this context, an artist models a volumetric field $f_\Theta : \mathbb{R}^3 \to \mathbb{R}$ *parameterized* by Θ, and is exposed with a 3D surface $\mathcal{S}$ given by its 0-set: $\mathcal{S} := f_\Theta^{-1}(\{0\})$.

Famous examples include *metaballs* [Blinn 1982; Fujita et al. 1990], *convolution skeletons* [Suárez et al. 2019; Zanni et al. 2012, 2013, 2015, 2011], and enriched sketches [Sugihara et al. 2008;

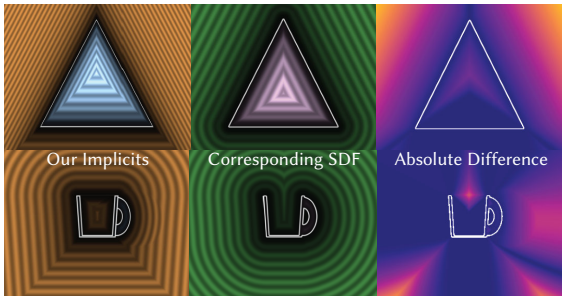

Figure 2: Our implicits, like most ones designed by CG artists, are not SDFs. Simple operations, such as intersection using a max **operator, indeed break the SDF assumption. Note the discontinuities in the spatial gradient of our implicit field.**

Wyvill et al. 2005]. They are part of most modeling software (e.g., `Blender` with *differentiable* graph-based solutions [BlenderImpl], `ZBrush`'s [ZSpheres], `Pixar`'s [Renderman]), and are the cornerstone of many recent modeling tools [Clavicula; MagicaCSG; Womp]. Implicit representations offer many benefits for 3D modeling:

- $\mathcal{S}$, the 0-set of f_Θ can be *infinitely smooth* for smooth f_Θ – which is almost unique in Computer Graphics;
- The artist can ignore polygonal surface quality during modeling, as $\mathcal{S}$ is extracted *as a post-process*;
- The *interior* and *exterior* of $\mathcal{S}$ are simply defined as $f_\Theta^{-1}(\mathbb{R}^-)$ and $f_\Theta^{-1}(\mathbb{R}^+)$ respectively, while those are typically difficult to compute for polygonal surfaces [Jacobson et al. 2013];
- Boolean operators (union, intersection, subtraction), highly challenging for meshed representations yet extremely useful for quick modeling and prototyping, are trivially implemented as min | max operators: $f^{A\cup B} := \min(f^A, f^B)$;
- As $\mathcal{S}$ is the sole object of interest, the definition of f_Θ *away from* $\mathcal{S}$ is in general of little to no concern to the artist, permitting flexible definitions that largely deviate from restrictive Signed Distance Fields (SDFs) – see Fig. 2.

Numerous works have tackled the design of *parametric blending operators* [Angles et al. 2017; Bernhardt et al. 2010; Gourmel et al. 2013], to allow for *smooth unions* for example (as opposed to *unions*, which lead to sharp creases at the exact junction between shapes being blended). Complementary to this, *warping* operators have allowed for the intuitive deformation and editing of implicit surfaces [Kleck 1989; Pasko et al. 2001; Zanni et al. 2011].

The extensive research and remarkable versatility of implicit surface representations have fostered creativity for years, and artists in the demoscene community (e.g., ShaderToy [Jeremias and Quilez 2013, 2014; Quilez and Jeremias 2024]) have used them intensively to design large scenes (sometimes infinite, defined as procedural fractals) that can be synthesized on-demand by extremely compact programs [Quilez 2008]. Due to their parametric nature, *dynamic, time-varying* implicit surfaces are also ubiquitous, and many works have focused on their efficient tracking/rendering [Jazar and Kry 2023] or their use for deforming 3D shapes [Vaillant et al. 2013].

Our work focuses on the latter problem: we study how parametric implicit surfaces can be used to deform 3D surfaces. In particular, we aim at enriching input raw detailed triangle meshes with *semantic deformation handles* described by parametric implicits (see Fig. 1, right), enabling mesh editing in a novel manner and allowing reusability of parametric implicits for the deformation of entire collections of objects. To this end, we introduce two complementary operators that constitute our two main technical contributions:

(i) **Fitting:** Our simple and robust template fitting operator allows finding adequate parameters for the 0-set of the input template to best match an input polygonal surface.
(ii) **Deformation:** Once the template is fit to the input mesh, our deformation operator allows *tracking* the template's 0-set as its semantic parameters are edited by a user. To cope with the severely ill-posed nature of this operation, we design a specific regularization energy adapted to the nature of common implicit templates, that typically allow for anisotropic stretch and/or local volume change.

To reach broad compatibility, we make little assumptions on our input implicit templates, and merely require those to be:

- *continuous* 3D fields: our templates representing 3D surfaces compatible with the objects we aim to deform, no Heaviside function nor discontinuous procedural noises are handled.
- *continuous* w.r.t. Θ: aiming at continuous deformations, we do not handle *discrete* parameters (e.g., grid instancing).
- reasonably *well-behaved* around their 0-set.

We do not require our implicits to be differentiable everywhere nor be SDFs, which would otherwise highly limit the usability of our technique. For instance, common operations on SDFs like union (min) and substraction/intersection (max) do not generally produce SDFs (see Fig. 2) and fixing the resulting field is an open problem [Marschner et al. 2023]. Further, even if our implicits are mostly built as compositions and blending of simple parametric primitives – like most ones found in the demoscene community or typically used by modelers, we do not rely on per-primitive fields (e.g., like [Vaillant et al. 2013]), nor are we enriching our programs to track point deformations (e.g., like [Michel and Boubekeur 2021]).

2 RELATED WORK

We focus in this section on *shape editing methods*, and refer the interested reader to surveys on implicit surface modeling [O'Brien and Yoo 2005], rendering [Knoll 2008] and meshing [De Araújo et al. 2015]. In addition, Yuan et al. [2021] survey shape editing from traditional to the more recent neural editing approaches.

Template-based shape editing. Semantic shape templates [Gadelha et al. 2020; Yumer et al. 2015] restrict shape edits to a semantically meaningful space. Several approaches, e.g. Liu et al. [2021], use neural representations capturing semantic notions of everyday objects. Jakab et al. [2020] extract meaningful point handles for deformation while Shechter et al. [2022] learn inferred displacements by such handles. Primitive-fitting methods (e.g. bounding boxes [Mo et al. 2020; Wei et al. 2020] or 3D Gaussians [Genova et al. 2019]) offer light-weight abstractions and part-wise semantic control.

Implicit shape templates. Several approaches estimate the semantic parameters of a procedurally-modeled template or a latent semantic representation using neural networks [Deng et al. 2021; Hao et al. 2020; Hertz et al. 2022; Liu et al. 2023; Pearl et al. 2022;

Zheng et al. 2021]. Approaches using a neural implicit representation can either have parameters learned by training over a dataset or estimated on a per-shape basis [Yifan et al. 2021]. While interactive shape editing is possible through quick forward-passes, issues resulting from out of training-distribution inputs and generalization need to be handled. Hertz et al. [2022] perform a *shape inversion* optimization during inference for a better latent fit in addition to an interactive editing speed. Those last approaches re-generate new shapes, which has the drawback of losing the input's connectivity. Our approach is connectivity-preserving, and only uses the implicit template to track deformations to apply to the input.

Deformation transfer. Shape editing is closely related to detail-preserving deformation transfer. Classical methods include Porumbescu et al. [2005] for detail transfer, Dekkers and Kobbelt [2014] and Rohmer et al. [2015] for detail generation/deletion, and Botsch et al. [2006] for detail-preserving surface editing, while neural representations have been recently developed for the same tasks [Aigerman et al. 2022; Chen et al. 2021; Morreale et al. 2022; Yifan et al. 2020, 2021]. Yin et al. [2021] and Ma et al. [2014] propose shape analogies methods by learning the deformation from a *source* shape to a *target* shape, and allow replicating the learned deformations to new shapes; Sung et al. [2020] project deformation handle edits to a learned semantics-preserving shape space; Maesumi et al. [2023] build deformation fields from point clouds generated using a 2D exploration space. In our work, we focus on driving deformation transfer (mesh retargeting) with our parametric implicit templates, which capture the semantics of the intended transfer.

3 OVERVIEW

Our method takes as input **i)** a mesh $\mathcal{M}$ within a specific object category **c**, and **ii)** an *implicit template* $\mathbf{T^c}$ represented by a parametric implicit function $f_\Theta : \mathbb{R}^3 \to \mathbb{R}$, Θ defining its parameters.

We manipulate $\mathcal{M}$ using $\mathbf{T^c}$ in two stages:

(i) **Fitting** (Sec. 4): Our fitting process computes parameters Θ_S for the template to best conform to $\mathcal{M}$ (in the sense that $\mathcal{M}$ and $\mathcal{S} := f^{-1}_{\Theta_S}(\{0\})$ are colocated). $\mathcal{M}$ is then *encoded* using local *signatures* in f_{Θ_S} found at its vertices.

(ii) **Deformation** (Sec. 5): Given target Θ_T (e.g., from a user-edit), we deform $\mathcal{M}$ by *tracking* its *implicit signatures* as Θ_S is advected towards Θ_T while minimizing a global deformation energy, ensuring quality preservation.

Implicit Template. $\mathbf{T^c}$ has minimal requirements as defined in Section 1. We designed our templates using an in-house tool, similar to [BlenderImpl]. Our data is identical in nature to the ones produced by popular software ([Clavicula; MagicaCSG; Womp]). Figure 1 (right) shows two examples of our templates, their interpretable structural parameters and the various implicit function instances of the same template with different parameters. Section 1 of the supplementary document details our implicit templates, their design logic and the way parameters interact with them.

4 IMPLICIT TEMPLATE FITTING

To fit a parametric implicit template to an input mesh $\mathcal{M}$, we design a simple iterative method inspired from Iterative Closest Point (ICP) methods. Consider *surface samples* $\mathcal{P}_S$ evenly distributed on $\mathcal{M}$ and *volume samples* $\mathcal{P}_V$ randomly distributed away from $\mathcal{M}$. We optimize the parameters Θ to minimize the following energy:

$$\mathcal{E}^\sigma_{\text{FIT}} := \sum_{p\in\mathcal{P}_S} |f_\Theta(p)|^2 + \alpha \sum_{p\in\mathcal{P}_V} w_\sigma(p)|f_\Theta(p) - d(p)|^2, \quad (1)$$

$d(p)$ denoting the approximate signed distance from p to $\mathcal{M}$ (we use fast pseudo-normal test [Bærentzen and Aanæs 2005]), and $w_\sigma(p)$ being a *confidence* weight quickly decreasing with increasing $d(p)$.

We minimize Eq. (1) with L-BFGS-B using *auto-differentiation* to compute the gradient $\nabla_\Theta(f_\Theta)$.

Optimizing f_Θ to best fit $\mathcal{M}$ only (i.e., setting $\alpha = 0$ in Eq. (1)) is not sufficient, because templates initialized far off the target $\mathcal{M}$ are not attracted to it with such a simple local approach, as $\nabla_\Theta(f_\Theta)$ might indicate wrong directions (or be null) in those situations (see Fig. 2 illustrating our implicits far from their 0-set). We therefore use the sparse volume samples $\mathcal{P}_V$ to mainly *repulse* the 0-set away from them and ensure non-zero gradients in such cases.

Note that, as we cannot ensure that f_Θ reproduces an SDF behavior (see Fig. 2), we mildly penalize deviation from those targets $\mathcal{P}_V$ in $\mathcal{E}_{\text{FIT}}$, which is sufficient to guide the optimization during the early iterations without compromising the final result. In our implementation, we define those confidence weights as

$$w_\sigma(p) := \exp\left(-d(p)^2/\sigma^2\right). \quad (2)$$

where σ is the standard deviation hyperparameter. We adopt a coarse to fine approach by running the optimization four times with decreasing σ values starting from the bounding box diagonal (almost uniform repulsion) to 3% of it (almost no repulsion).

Figure 3 shows examples of parametric shapes fitting (input) target meshes. The template shapes are shown both in their initial configuration, and after being fitted to an input mesh.

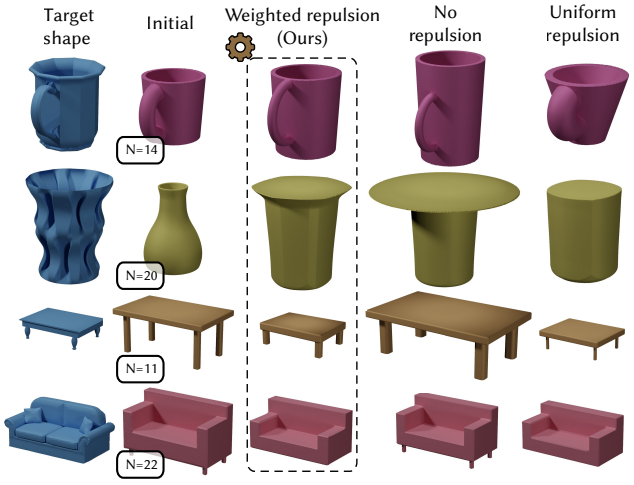

Figure 3: Fitting parametric implicit templates to meshes. N is the number of (scalar) template parameters. When using only surface samples (4^{th} col.), gradients far off the isosurface fail to bring it close to the target shape (blue). With uniform weighting (5^{th} col.), the fit is close to the target but slightly off as volume samples do not yield true sdf values.

To illustrate the robustness of our method, we conduct an ablation study, and compare in Fig. 3 our weighting strategy with:

- no repulsion: setting $\alpha = 0$;
- uniform repulsion: setting $\alpha = 1$ and $w(p) = 1\ \forall p \in \mathcal{P}_V$.

In the first case, the template might not converge towards the target surface, due to null gradients far off the global minimizer of $\mathcal{E}_{\text{FIT}}$. In the second case, the final template is slightly off the target surface, since the assumption that it follows an SDF everywhere is not true.

To further demonstrate the robustness of our approach on *severely damaged implicits*, we show results of our strategy for modulated targets in Fig. 4 of the supp. and show that the optimized implicit surfaces still overall fit the target surfaces, indicating that the use of volume samples is indeed harmless for the final fitting.

5 IMPLICIT-DRIVEN MESH DEFORMATION

We propose a novel algorithm to compute the deformation map ϕ (the target mesh $\mathcal{M}' = \phi(\mathcal{M})$) given the source (input) mesh $\mathcal{M}$, associated source implicit representation f_Θ (fit to $\mathcal{M}$ using the method presented in Sec. 4), and target parameters Θ' (typically corresponding to a user edit). Note that target and source meshes have a one-to-one correspondence, and we are merely looking for a deformation map $v' = \phi(v)$ for every mesh vertex v.

We cast our ill-posed problem as an energy minimization, and introduce a novel mesh deformation energy:

$$\mathcal{E}_{\text{DEF}}(\mathcal{M}') \coloneqq \mathcal{E}_{\text{TRACKING}}(\mathcal{M}') + \mathcal{E}_{\text{REG}}(\mathcal{M}'), \tag{3}$$

both tracking $\mathcal{E}_{\text{TRACKING}}(\mathcal{M}')$ and regularization $\mathcal{E}_{\text{REG}}(\mathcal{M}')$ energies specifically designed for our problem. Since both the terms are highly non-linear, we adopt an iterative optimization scheme.

For the rest of this section, v denotes an input mesh vertex, and v' its unknown transformation under the corresponding template edit $\Theta \mapsto \Theta'$. $(\cdot)'$ denotes *target* quantities. We further denote v^k its 3D position at step/iteration k, and d_v its displacement update at step $k+1$ ($v^{k+1} = v^k + d_v$), converging to $v' = \lim_{k\to\infty}(v^k) = \operatorname{argmin} \mathcal{E}_{\text{DEF}}$. We detail our energy terms and their associated one-step descent constraint in Sec. 5.1 and 5.2, under the assumption of continuous smooth edit steps $\Theta \mapsto \Theta'$ in the parameter space, with the complete iterative algorithm in Sec. 5.3.

We take inspiration from well-established ARAP deformations [Sorkine and Alexa 2007], that advocate the use of auxiliary local transformation maps R_v associated with each vertex v. The main (geometric) principle of those works is to minimize a non-linear deformation energy of the form:

$$\mathcal{E}_{\text{ARAP}} = \sum_{\bar{v}\in\text{HANDLES}} \|v' - \bar{v}\|^2 + \sum_{v\in V} \sum_{(u,w)\in E(v)} h_{uw} \|R_v(w-u) - (w'-u')\|^2,$$

that enforces the rigid transformation of v's edge-set $E(v)$ by a rotation $R_v \in \mathbb{R}^{3\times3}$ while enforcing *handle* constraints $v \mapsto \bar{v}$, $\{h_{uw}\}$ being surface discretization weights. $E(v)$ can simply be the v's adjacent edges, but larger/different neighborhoods (e.g., a one-ring edge-set) can be considered. R_v is typically initialized to I and constrained to be 3D rotations, and $\{v', R_v\}$ are optimized iteratively using a standard *local/global* solver minimizing efficiently $\mathcal{E}_{\text{ARAP}}$, which is the main take-away we retain from those works.

5.1 Tracking energy

5.1.1 Implicit signatures. Many mesh deformation techniques involve a user interacting with a few handle vertices to specify 3D positions for them and the rest of the mesh is optimized for a shape regularization energy ensuring well-behaved deformations. In our context, we rather bind the source mesh $\mathcal{M}$ to the source implicit f_Θ. We do this by preserving the following *implicit signatures*:

order 0: The value of $f_{\Theta'}$ at vertex $v' = \phi(v)$ should equal the value of f_Θ at its source location $v \in \mathcal{M}$, i.e.

$$f_{\Theta'}(v') = f_\Theta(v) =: f_v \tag{4}$$

order 1: The gradient of $f_{\Theta'}$ at v' should equal the re-aligned gradient of f_Θ at $v \in \mathcal{M}$ (with v's local map $J_v \in \mathbb{R}^{3\times3}$, Sec. 5.2), i.e.

$$\nabla f_{\Theta'}(v') = J_v \cdot \nabla f_\Theta(v) =: J_v \cdot g_v \tag{5}$$

When J_v is not a rotation, we re-normalize the right hand side in Eq. 5 to match the norm of $\nabla f_{\Theta'}(v')$. These $(0^{\text{th}}, 1^{\text{st}})$-order coupling conditions enforced over $\mathcal{M}$ "bind" the mesh geometry to the underlying implicit representation. Our tracking energy is:

$$\mathcal{E}_{\text{TRACKING}}(\mathcal{M}') = \sum_v |f_{\Theta'}(v') - f_v|^2 + \|\nabla f_{\Theta'}(v') - J_v \cdot g_v\|^2 \tag{6}$$

5.1.2 Mesh updates. We translate those conditions to our update scheme $v^{k+1} = v^k + d_v$, by writing the explicit Taylor expansion of our target field $f_{\Theta'}$ and its gradient $\nabla f_{\Theta'}$ at current location v^k as

$$\begin{cases} f_{\Theta'}(v^{k+1}) & = f_{\Theta'}(v^k) + \nabla f_{\Theta'}(v^k)^T d_v + \frac{d_v^T H f_{\Theta'}(v^k) d_v}{2} + O(\|d_v\|^3) \\ \nabla f_{\Theta'}(v^{k+1}) & = \nabla f_{\Theta'}(v^k) + H f_{\Theta'}(v^k) \cdot d_v + O(\|d_v\|^2) \end{cases}$$

Enforcing v's target implicit signature at v^{k+1} (i.e., $f_{\Theta'}(v^{k+1}) = f_v$, $\nabla f_{\Theta'}(v^{k+1}) = J_v \cdot g_v$) and discarding $O(\|d_v\|^3)$ terms gives

$$\begin{cases} f_v & = f_{\Theta'}(v^k) + \nabla f_{\Theta'}(v^k)^T \cdot d_v + \frac{d_v^T H f_{\Theta'}(v^k) \cdot d_v}{2} \\ J_v \cdot g_v & = \nabla f_{\Theta'}(v^k) + H f_{\Theta'}(v^k) \cdot d_v \end{cases} \tag{7}$$

Eliminating $H f_{\Theta'}$ gives the following tracking linear constraint on our vertex update d_v:

$$\boxed{\left(\frac{\nabla f_{\Theta'}(v^k) + J_v \cdot g_v}{2}\right)^T \cdot d_v = f_v - f_{\Theta'}(v^k)} \ \forall v. \tag{8}$$

5.1.3 Analysis and implementation notes. Although the updates use only up-to first order derivatives in its final formulation, the resulting error terms are of order 3, as we implicitly use the second order information of the deformation $H f_{\Theta'}$ in our construction (Eq. (7)). This construction leads to a modified gradient descent update (Eq. (8)) with the standard gradient $\nabla f_{\Theta'}$ blended with the re-aligned source gradient signature $J_v \cdot g_v$.

In practice we approximate a reliable spatial gradient ∇f using a finite differences scheme rather than evaluating it explicitly – which would only work for implicits that are differentiable everywhere. We notice no significant difference in results using automatic differentiation instead. Our update step constraint (Eq. (8)) does not strictly correspond to the precise least-squares minimization of Eq. 6, for the reason that the 0^{th} and 1^{st} order constraints are not evenly balanced in Eq. (7). Our simpler update step avoids however the finite-difference approximation of the Hessian $H f_{\Theta'}$, which we

do both for simplicity and robustness reasons, and it corresponds to the least-squares optimization under infinitesimal updates d_v.

5.2 Regularization energy

Tracking our implicit signatures leads to under-constrained deformations as the updates are insensitive to the vertices sliding on the implicit's iso-surfaces (especially in flat regions, see Fig. 4), warranting the need for regularization. We specialize our regularization energy to deformations containing large anisotropic stretch. Inspired from ARAP deformations, we weakly enforce each vertex v' neighborhood to be rigidly transformed by a single map $J_v \in \mathbb{R}^{3\times3}$. Contrary to ARAP, we do not constrain those to be rotations, as we target stretched deformations. In our case, we enforce local normal isometries while controlling the stretch in the tangent plane.

We further study the use of an optional anisotropic smoothing term, weakly enforcing continuous consistent regions (i.e., not separated by sharp features) to remain uniformly deformed if needed. Our regularization energy is defined as:

$$\mathcal{E}_{\text{REG}}(\mathcal{M}') = \sum_{v\in V} \sum_{(u,w)\in E(v)} c_{vuw} \|J_v \cdot (w-u) - (w'-u')\|^2 + \alpha_s \mathcal{E}_{\text{SMOOTH}}(\{J_v\}) \tag{9}$$

where c_{vuw} are the cotangent weights and J_v belongs to a class of local transformations enforcing isometry in the normal direction.

Global step. Given estimated $\{J_v^{k-1}\}$ from step $k-1$, the following linear constraint on the vertex update at step k is enforced:

$$\boxed{d_w - d_u = J_v^{k-1} \cdot (w-u) - (w^k - u^k)} \; \forall v, \forall (u,w) \in E(v). \tag{10}$$

Local step. Given source vertex positions $\{v\}$ and current vertex positions $\{v^k\}$ at step k, we estimate the local maps $\{J_v^k\}$ by enforcing those to map the source normal n_v precisely to the deformed normal n_v^k, while best aligning edge-set $E(v)$ in the corresponding tangent spaces. With $P_n := I - n \cdot n^T$, we confine the maps $\{J_v^k\}$ to:

$$J_v^k = n_v^k \cdot {n_v}^T + P_{n_v^k} \cdot J_v \cdot P_{n_v} \tag{11}$$

In the absence of $\mathcal{E}_{\text{SMOOTH}}$, substituting Eq. (11) in Eq. (9) gives

$$J_v^k = \underset{J}{\text{argmin}} \sum_{(u,w)\in E(v)} c_{vuw} \|J P_{n_v} \cdot (w-u) - P_{n_v^k} \cdot (w^k - u^k)\|^2 \tag{12}$$

J_v^k can be computed in closed-form as

$$\boxed{J_v^k = \underbrace{n_v^k \cdot {n_v}^T}_{\text{rank 1}} + \underbrace{P_{n_v^k} \cdot E_v^k \cdot \left(P_{n_v} \cdot E_v\right)^\dagger}_{\text{complementary, rank 2}}} \tag{13}$$

with E_v (resp. E_v^k) the $3 \times m$ matrix whose columns are given by $\sqrt{c_{uvw}}(w-u)$ (resp. $\sqrt{c_{uvw}}(w^k-u^k)$), and $M^\dagger$ denoting the pseudo-inverse of M. Note that Eq. (13) is symmetric, and that the inverse map J_v' is obtained by switching (n_v^k, E_v^k) with (n_v, E_v).

Anisotropic smoothing. We optionally perform a smoothing step of our local maps $\{J_v\}$. Regularizing those maps was first proven (to the best of our knowledge) to help preserving large structures in the context of ARAP optimizations in [Levi and Gotsman 2014]. Inspired by this work, we design our smoothing operator by

- enforcing smoothing on $\{J_v^k\}$ after aligning adjacent maps through a trivial connection, and
- using *geometry-aware* blending weights w_{ij} that preserve the structure defined by sharp features.

The optional smoothing energy is defined over all edges E as:

$$\mathcal{E}_{\text{SMOOTH}} = \sum_{(i,j)\in E} w_{ij} \|J_i^k - T(J_j^k, j \mapsto i)\|_F^2 \tag{14}$$

with the trivial connection operator [Crane et al. 2010] transporting maps from vertex j to vertex i given by

$$T(J_j, j \mapsto i) = R(n_j^k \mapsto n_i^k)^T J_j R(n_j \mapsto n_i) \tag{15}$$

where $R(n_j \mapsto n_i)$ is the shortest rotation mapping n_j to n_i. Parallel to Eq. (12), we can show that in presence of $\mathcal{E}_{\text{SMOOTH}}$,

$$J_i^k = \underset{J}{\text{argmin}} \sum_{(u,w)\in E(i)} c_{iuw} \|J \cdot P_{n_i} \cdot (w-u) - P_{n_i^k} \cdot (w^k - u^k)\|^2 + \sum_{j\in\mathcal{N}(i)} w_{ij} \|J \cdot P_{n_i} - P_{n_i^k} \cdot T(J_j^k, j \mapsto i) \cdot P_{n_i}\|^2 \tag{16}$$

While we implemented a vectorized least-squares solve to compute $J_i^k, \forall k$, we observed in practice that using neighboring maps J_j^k from the previous step lets us compute J_i^k in parallel $\forall i$ without compromising the solution quality (given we already use an iterative approach). With $T_j^k = T(J_j^k, j \mapsto i)$, we obtain $\forall i$:

$$J_i^k = n_i^k \cdot {n_i}^T + P_{n_i^k} \cdot \Big[E_i^k \dots \underbrace{\sqrt{w_{ij}} \cdot T_j^k \cdot P_{n_i}}_{\forall j\in\mathcal{N}(i)} \dots\Big] \Big(P_{n_i} \cdot \Big[E_i \dots \underbrace{\sqrt{w_{ij}} \cdot I}_{\forall j\in\mathcal{N}(i)} \dots\Big]\Big)^\dagger \tag{17}$$

with E_i, E_i^k defined as in Eq. (13), $\mathcal{N}(i)$ the neighboring vertices of vertex i, and I the 3×3 identity matrix. We define our *geometry-aware* weights using distance functions common in mesh segmentation [Yan et al. 2006], for *flat* and *spherical* regions:

$$w_{ij} = \exp\left(-d(j, \text{proxy } i)^2 / \sigma_s^2\right), \tag{18}$$

$\sigma_s \geq 0$ controlling the process.

5.3 Progressive tracking

With both energy terms in account, we optimize Eq. (3) iteratively, alternating *global* optimization of $d = \{d_v\}$ updates with *local* estimation of the auxiliary transformation matrices $\{J_v\}$.

For a specific edit from the source (input) parameters $\Theta_S \mapsto \Theta_T$, we perform the associated mesh deformation by (optionally) splitting the edit into smaller sub-edits $\Theta_S = \Theta_0 \mapsto \Theta_1 \mapsto \cdots \mapsto \Theta_T$. This is motivated in Section 3 of the supp. by the need for infinitesimal updates as explained in Section 5.1.3. Algorithm 1 summarizes our progressive tracking. Note that we typically use a small number of iterations k_{max} for the (intermediate) subedit steps, while ending with a larger number of k_{max} iterations (especially if anisotropic smoothing is enforced) for interactive speeds.

5.4 Ablation and experimental validation

We validate our choices for the deformation energy incrementally. Fig. 4 shows a couch mesh fitted with a couch template subjected to a large edit in its length . The edited mesh with no $\mathcal{E}_{\text{REG}}$ shows

ALGORITHM 1: Our Implicit-driven mesh deformation

```
// Input: source mesh M, source/target implicit parameters Θ/Θ'
// Output: deformed mesh M'
for subedit step Θ ↦ Θ' do
    for k = 0 ↦ k_max do
        Gather Eqs. (8),(10) and build system A · d = b, for
          d = {d_v}
        Solve for d_v = argmin_d ||A · d − b||^2
        Update mesh: v ← v + d_v for all v
        Update local maps {J_v} (Eq. (13)/ (17))
```

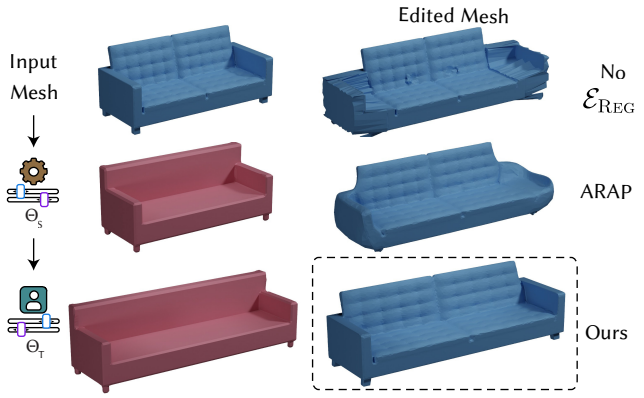

Figure 4: In contrast to ARAP, we allow for anisotropic stretch to the desirable extent.

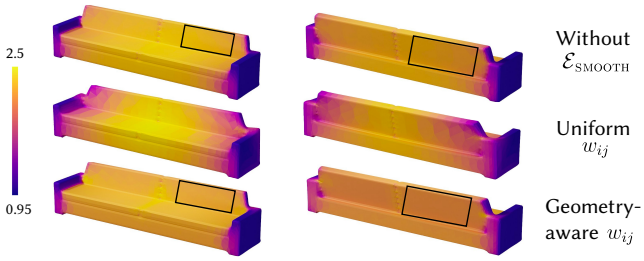

Figure 5: 1^{st} principal stretch factor of the deformation for an edit doubling the couch length. Our $\mathcal{E}_{\text{SMOOTH}}$ gives control to the user to preserve consistent regions, while smoothing the local transformations within them (see insets). Region boundaries (see the back and side) are also better demarcated.

the need for regularization since the fitting energy alone is not sufficient. While ARAP [Sorkine and Alexa 2007] provides some regularization, it does not permit deformation involving a large stretch. Our approach provides the desired stretch while preserving features from the input mesh. We show the effect of geometry-aware weights in the smoothing energy in Fig. 5. The smoothing energy can be turned on when the user needs smooth stretch in continuous consistent regions (controlled using σ_s).

6 APPLICATIONS

We demonstrate two main applications of our shape editing framework: *interactive semantic shape editing* of input 3D shapes (Sec. 6.1).

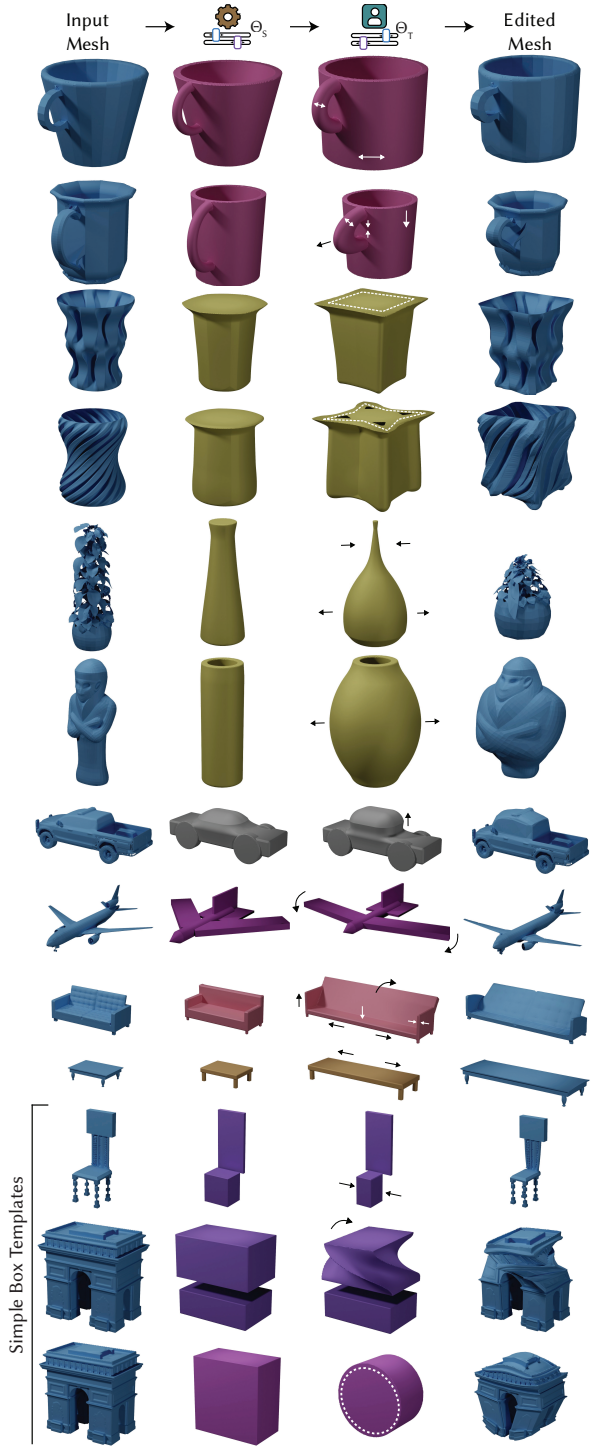

Figure 6: Applying semantic edits from implicit templates (second column) onto 3D meshes (left). Same colors of templates denote same template used. Our implicit templates allow complex deformation like smooth morphing or inflating. The last three rows show edits performed using simple box templates. Fig. 7 shows more results of semantic edits.

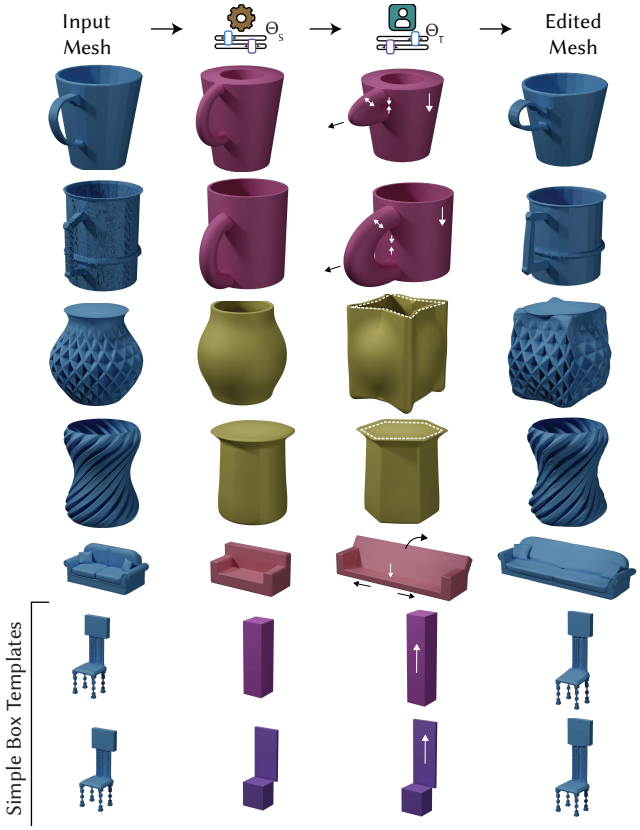

Figure 7: Applying semantic edits from implicit templates (second column) onto 3D meshes (left). Same colors of templates denote same template used. The last two rows show edits performed using simple box templates.

and deformation of a source mesh to *match the semantic structure* of a target one (Sec. 6.2). We show the robustness of our approach to the template used, while illustrating how different templates incur different types of deformations.

Sec. 5 of the supp. gives details on our implementation and performance insights. In addition to all the results shown in figures in the main and supplementary documents, we provide an HTML interface (in supp.) to browse all results at high resolution.

6.1 Interactive shape editing

Our shape parameters capture the structural semantics of objects and give a sparse set of shape handles to perform intuitive edits. Our implementation demonstrates shape editing with interactive feedback on a consumer computer, as shown in the supp. video. We illustrate shape editing by varying the parameters for shape categories like couch, table, vase, mug, car and airplane. We show that simpler box templates also allow performing useful edits, while being very easy to create and agnostic to the shape category.

6.1.1 Semantic templates. Figs. 6 and 7 show semantic edits of different natures and scales applied to shapes of multiple categories (specified by the user). The same template is used for different inputs

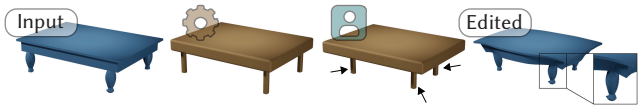

Figure 8: Failure case: our approach may struggle to make large edits localized to a tiny region.

showing the versatility of our proposed approach and its generality to inputs of different structures and topologies, as also seen in Fig. 12. By design, our framework supports multiple edits at once as it only relies on the edited template. Implicit function templates enable complex editing operations, such as morphing between primitive shapes or squeeze and inflation, visible on the vase examples. Fig. 8 of the supp. shows additional results of semantic edits.

Fig. 8 shows a failure case of our deformation method. Though we focus on connectivity-preservation deformations, further improvement to our method could be achieved through promoting sparse stretch relaxation singularities and remeshing, as well as detail synthesis/removal when necessary [Dekkers and Kobbelt 2014; Rohmer et al. 2015].

6.1.2 Simple templates. The last three rows of Fig. 6 show easy access to quick complex deformations (like twisting) using very simple templates. Fitting simple box templates differently to the same inputs shows ways to deform parts of the inputs separately in a desired way. Fig. 9 of the supp. shows additional results.

6.2 Semantic mesh retargeting

We leverage our template-based deformation to warp a source shape into the structure of a target. We fit templates of a common category to the source and target meshes. This enables moving from the source to the target templates through a continuous parametric space. Since our approach enables tracking edits in this space, we can deform the source mesh to be edited to conform to the target template resulting in mesh retargeting. In addition, interpolating parameter values between templates fit to different shapes allows interpolating the structure of the two meshes.

As shown in Fig. 9, this process allows preserving the style of a given shape while warping it into a different structure. For instance, we show the deformation of an input couch mesh to match the structure of a noisy scan. This provides an easy way to replace dirty meshes from a single view depth map with complete ones. Fig. 10 of the supp. shows additional results.

6.3 Other parametric templates

Fig. 10 shows deformation of human meshes fit with *SMPL* templates [Loper et al. 2015] that provide a parametric model of body pose and shape. While our approach focuses on tracking implicit templates, we can easily deploy other existing specialized templates (like *SMPL*) by simply computing the SDF from them. Given the template's ability to parametrically vary both body pose and shape (Fig. 10 rows 1 & 2), our approach enables easy deformation transfer to textured human inputs even from noisy scans.

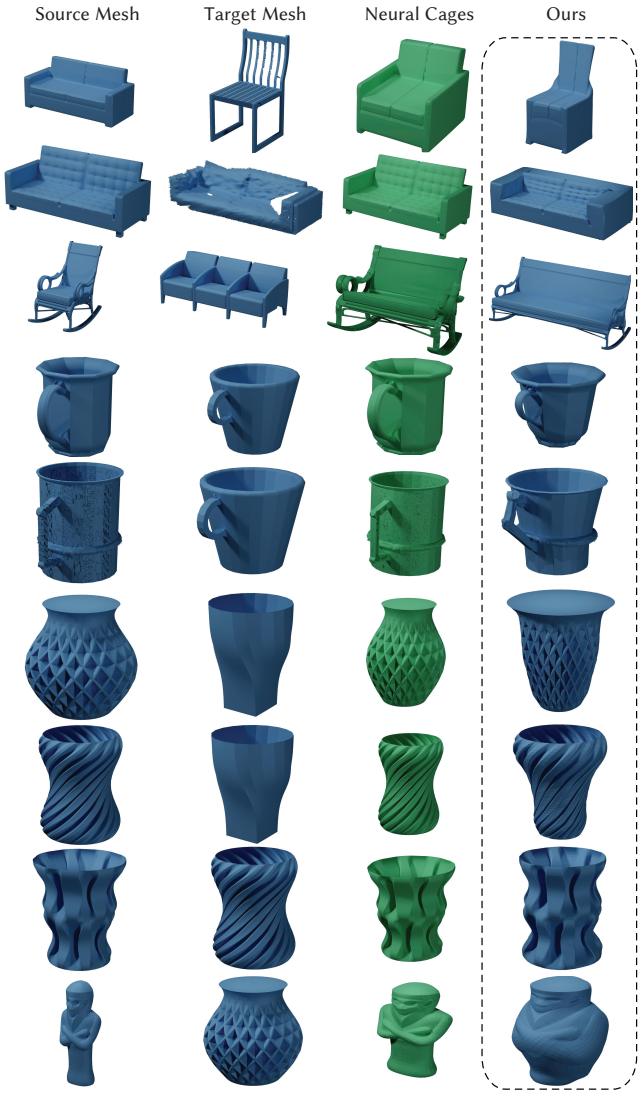

Figure 9: Detail-preserving mesh retargeting. The target mesh in the second row is a single-view scanned couch. The third column (green) shows the result of *Neural Cages* [Yifan et al. 2020] on the same source and target meshes.

7 COMPARISONS

We evaluate our deformation scheme against Wei et al. [2020] (edits meshes using parametric templates), and Yifan et al. [2020](mesh retargeting). We also compare to an implicit shape-editing approach *Spaghetti* [Hertz et al. 2022]. In addition to the ones presented here, Figs. 12, 13 and 14 of the suppl. demonstrate more results. Sec. 7 of the supp. displays results of a perceptual study showing user preference of our results compared to these three related works.

7.1 Mesh Deformation based editing

7.1.1 Comparison with Wei et al. [2020]. This work deforms a given shape using non-implicit templates made of different types of primitives. It uses a simple deformation scheme based on weighted average of displaced control points. In contrast, our regularization energy was specifically designed to preserve local orientation of details. It leverages a neural network to estimate the parameters of the template for a given input, which can limit its representativeness to the training data. Code for this method is not available but the authors kindly provided result meshes and corresponding edits, that we reproduce in our setup in Fig. 11. Our approach preserves details better and displays greater fidelity to the template.

7.1.2 Comparison with Neural Cages [Yifan et al. 2020]. We compare our mesh retargeting application to *Neural Cages* which uses a *neural* cage as a low-level representation to track and deform the input vertices using *Mean Value Coordinates*. In Fig. 9, we display greater fidelity to the target shape structure and preserve the source details at the appropriate scale of the target. In addition, we do not suffer the problem of under-performing on out-of-distribution inputs not generalized by a learnt network. While the *Neural Cages* are limited to low frequency edits, other subspaces [Wang et al. 2015] allow more localized and sharp edits.

7.2 Semantic editing using Neural Implicits

We compare our method to the implicit-based *Spaghetti* method [Hertz et al. 2022], which captures semantics of objects using a neural representation and *generates* the shape again after editing. While this approach suffers from a generalization problem too, the fixed resolution and expressiveness of the neural latent representation can affect the details of the output.

Fig. 12(a) shows a comparison on chairs from the *Spaghetti* samples, where the method can not accurately re-generate the shape after structural edits. In Fig. 12(b), we run the shape inversion from *Spaghetti* to apply edits on new shapes. In addition to the lack of geometric details output by *Spaghetti*, large scale edits such as back tilting on couches lead to global deformation of the input and do not preserve the locality of the edits, due to the entanglement of features in the neural representation.

8 CONCLUSION

We presented a novel approach to fit parametric implicit templates to input shapes, and deform them by tracking the evolving implicit template. With the ubiquity and versatility of implicit modeling, our minimal requirements on the template ensure ease of use. In addition, our deformation algorithm contributes tracking implicit signatures enabling a semantic link between the template and shape spaces. We show compatibility with existing parametric templates and provide a complementary approach to learning based methods that can suffer from generalization problems.

ACKNOWLEDGMENTS

We thank Tamy Boubekeur, Pierre Gueth, Thibault Groueix, Matheus Gadelha and Vojtech Krs for fruitful discussions on template fitting.

REFERENCES

Noam Aigerman, Kunal Gupta, Vladimir G. Kim, Siddhartha Chaudhuri, Jun Saito, and Thibault Groueix. 2022. Neural jacobian fields: learning intrinsic mappings of arbitrary meshes. *ACM Trans. Graph.* 41, 4, Article 109 (jul 2022), 17 pages. https://doi.org/10.1145/3528223.3530141

Baptiste Angles, Marco Tarini, Brian Wyvill, Loïc Barthe, and Andrea Tagliasacchi. 2017. Sketch-based implicit blending. *ACM Trans. Graph.* 36, 6, Article 181 (nov 2017), 13 pages. https://doi.org/10.1145/3130800.3130825

Jakob Andreas Bærentzen and Henrik Aanæs. 2005. Signed distance computation using the angle weighted pseudonormal. *IEEE Transactions on Visualization and Computer Graphics* 11, 3 (2005), 243–253. https://doi.org/10.1109/TVCG.2005.49

Adrien Bernhardt, Loic Barthe, Marie-Paule Cani, and Brian Wyvill. 2010. Implicit Blending Revisited. *Computer Graphics Forum* 29, 2 (2010), 367–375. https://doi.org/10.1111/j.1467-8659.2009.01606.x

BlenderImpl. [n. d.]. ConjureSDF - smooth, non destructive booleans add-on for Blender. https://blenderartists.org/t/conjuresdf-smooth-non-destructive-booleans

James F. Blinn. 1982. A Generalization of Algebraic Surface Drawing. *ACM Trans. Graph.* 1, 3 (jul 1982), 235–256. https://doi.org/10.1145/357306.357310

Mario Botsch, Robert Sumner, Mark Pauly, and Markus Gross. 2006. Deformation transfer for detail-preserving surface editing. In *Vision, Modeling & Visualization*. Citeseer, 357–364.

Yunlu Chen, Basura Fernando, Hakan Bilen, Thomas Mensink, and Efstratios Gavves. 2021. Neural feature matching in implicit 3D representations. 139 (18–24 Jul 2021), 1582–1593. https://doi.org/11245.1/67c7cd5a-ed8f-4199-ae10-087a9574ed52

Clavicula. 2022. *Clavicula*. Retrieved May 23, 2023 from https://clavicula.link/

Keenan Crane, Mathieu Desbrun, and Peter Schröder. 2010. Trivial Connections on Discrete Surfaces. *Computer Graphics Forum (SGP)* 29, 5 (2010), 1525–1533. https://doi.org/10.1111/j.1467-8659.2010.01761.x

Bruno Rodrigues De Araújo, Daniel S Lopes, Pauline Jepp, Joaquim A Jorge, and Brian Wyvill. 2015. A Survey on Implicit Surface Polygonization. *ACM Computing Surveys (CSUR)* 47, 4, Article 60 (may 2015), 39 pages. https://doi.org/10.1145/2732197

Ellen Dekkers and Leif Kobbelt. 2014. Geometry seam carving. *Computer-Aided Design* 46 (2014), 120–128. https://doi.org/10.1016/j.cad.2013.08.024 2013 SIAM Conference on Geometric and Physical Modeling.

Yu Deng, Jiaolong Yang, and Xin Tong. 2021. Deformed Implicit Field: Modeling 3D Shapes with Learned Dense Correspondence. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, Los Alamitos, CA, USA, 10286–10296. https://doi.org/10.1109/CVPR46437.2021.01015

Takushi Fujita, Katsuhiko Hirota, and Kouichi Murakami. 1990. Representation of splashing water using metaball model. *Fujitsu* 41, 2 (1990), 159–165.

Matheus Gadelha, Giorgio Gori, Duygu Ceylan, Radomir Mech, Nathan Carr, Tamy Boubekeur, Rui Wang, and Subhransu Maji. 2020. Learning Generative Models of Shape Handles. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 399–408. https://doi.org/10.1109/CVPR42600.2020.00048

Kyle Genova, Forrester Cole, Daniel Vlasic, Aaron Sarna, William T Freeman, and Thomas Funkhouser. 2019. Learning Shape Templates With Structured Implicit Functions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 7154–7164. https://doi.org/10.1109/ICCV.2019.00725

Olivier Gourmel, Loic Barthe, Marie-Paule Cani, Brian Wyvill, Adrien Bernhardt, Mathias Paulin, and Herbert Grasberger. 2013. A gradient-based implicit blend. *ACM Transactions on Graphics (TOG)* 32, 2 (2013), 1–12. https://doi.org/10.1145/2451236.2451238

Zekun Hao, Hadar Averbuch-Elor, Noah Snavely, and Serge Belongie. 2020. Dualsdf: Semantic shape manipulation using a two-level representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 7631–7641. https://doi.org/10.1109/CVPR42600.2020.00765

Amir Hertz, Or Perel, Raja Giryes, Olga Sorkine-Hornung, and Daniel Cohen-Or. 2022. Spaghetti: Editing implicit shapes through part aware generation. *ACM Transactions on Graphics (TOG)* 41, 4 (2022), 1–20. https://doi.org/10.1145/3528223.3530084

Alec Jacobson, Ladislav Kavan, and Olga Sorkine-Hornung. 2013. Robust inside-outside segmentation using generalized winding numbers. *ACM Transactions on Graphics (TOG)* 32, 4 (2013), 1–12. https://doi.org/10.1145/2461912.2461916

Tomas Jakab, Richard Tucker, Ameesh Makadia, Jiajun Wu, Noah Snavely, and Angjoo Kanazawa. 2020. KeypointDeformer: Unsupervised 3D Keypoint Discovery for Shape Control. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. https://doi.org/10.1109/CVPR46437.2021.01259

Kavosh Jazar and Paul G Kry. 2023. Temporal set inversion for animated implicits. *ACM Transactions on Graphics (TOG)* 42, 4 (2023), 1–18. https://doi.org/10.1145/3592448

Pol Jeremias and Iñigo Quilez. 2013. Shadertoy: live coding for reactive shaders. In *ACM SIGGRAPH 2013 Computer Animation Festival*. 1–1.

Pol Jeremias and Inigo Quilez. 2014. Shadertoy: Learn to create everything in a fragment shader. In *SIGGRAPH Asia 2014 Courses*. 1–15.

James Edgar Kleck. 1989. *Modeling using implicit surfaces*. University of California, Santa Cruz, Computer Research Laboratory.

Alois Knoll. 2008. A survey of implicit surface rendering methods, and a proposal for a common sampling framework. (2008), 164–177.

Zohar Levi and Craig Gotsman. 2014. Smooth rotation enhanced as-rigid-as-possible mesh animation. *IEEE transactions on visualization and computer graphics* 21, 2 (2014), 264–277. https://doi.org/10.1109/TVCG.2014.2359463

Mengya Liu, Ajad Chhatkuli, Janis Postels, Luc Van Gool, and Federico Tombari. 2023. Unsupervised Template Warp Consistency for Implicit Surface Correspondences. In *Computer Graphics Forum*, Vol. 42. Wiley Online Library, 77–87. https://doi.org/10.1111/cgf.14745

Minghua Liu, Minhyuk Sung, Radomir Mech, and Hao Su. 2021. Deepmetahandles: Learning deformation meta-handles of 3d meshes with biharmonic coordinates. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 12–21. https://doi.org/10.1109/CVPR46437.2021.00008

Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. 2015. SMPL: A Skinned Multi-Person Linear Model. *ACM Trans. Graphics (Proc. SIGGRAPH Asia)* 34, 6 (Oct. 2015), 248:1–248:16. https://doi.org/10.1145/2816795.2818013

Chongyang Ma, Haibin Huang, Alla Sheffer, Evangelos Kalogerakis, and Rui Wang. 2014. Analogy-driven 3D style transfer. In *Computer Graphics Forum*, Vol. 33. Wiley Online Library, 175–184. https://doi.org/10.1111/cgf.12307

Arman Maesumi, Paul Guerrero, Noam Aigerman, Vladimir Kim, Matthew Fisher, Siddhartha Chaudhuri, and Daniel Ritchie. 2023. Explorable Mesh Deformation Subspaces from Unstructured 3D Generative Models. In *SIGGRAPH Asia 2023 Conference Papers*. 1–11. https://doi.org/10.1145/3610548.3618192

MagicaCSG. 2021. *MagicaCSG ephtracy, A lightweight signed distance field editor and path tracing renderer.* Retrieved May 23, 2023 from https://ephtracy.github.io/index.html?page=magicacsg

Zoë Marschner, Silvia Sellán, Hsueh-Ti Derek Liu, and Alec Jacobson. 2023. Constructive Solid Geometry on Neural Signed Distance Fields. In *SIGGRAPH Asia 2023 Conference Papers*. 1–12. https://doi.org/10.1145/3610548.3618170

Elie Michel and Tamy Boubekeur. 2021. DAG amendment for inverse control of parametric shapes. *ACM Transactions on Graphics (TOG)* 40, 4 (2021), 1–14. https://doi.org/10.1145/3450626.3459823

Kaichun Mo, Paul Guerrero, Li Yi, Hao Su, Peter Wonka, Niloy J Mitra, and Leonidas J Guibas. 2020. StructEdit: Learning structural shape variations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 8859–8868. https://doi.org/10.1109/CVPR42600.2020.00888

Luca Morreale, Noam Aigerman, Paul Guerrero, Vladimir G Kim, and Niloy J Mitra. 2022. Neural convolutional surfaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 19333–19342. https://doi.org/10.1109/CVPR52688.2022.01873

James F O'Brien and Terry S Yoo. 2005. Modern techniques for implicit modeling. In *ACM SIGGRAPH 2005 Courses*. 1–es. https://doi.org/10.1145/1198555.1198638

Alexander Pasko, Valery Adzhiev, Benjamin Schmitt, and Christophe Schlick. 2001. Constructive hypervolume modeling. *Graphical models* 63, 6 (2001), 413–442. https://doi.org/10.1006/gmod.2001.0560

Ofek Pearl, Itai Lang, Yuhua Hu, Raymond A. Yeh, and Rana Hanocka. 2022. GeoCode: Interpretable Shape Programs. (2022). https://doi.org/10.48550/arXiv.2212.11715

Serban D Porumbescu, Brian Budge, Louis Feng, and Kenneth I Joy. 2005. Shell maps. *ACM Transactions on Graphics (TOG)* 24, 3 (2005), 626–633. https://doi.org/10.1145/1073204.1073239

Ínigo Quilez. 2008. Rendering Worlds with Two Triangles with raytracing on the GPU in 4096 bytes.

Inigo Quilez and Pol Jeremias. 2024. Shadertoy. *Retrieved April* 26 (2024), 2024.

Pixar Renderman. [n. d.]. Renderman implicit surfaces. https://renderman.pixar.com/resources/RenderMan_20/risBlobbies.html

Damien Rohmer, Stefanie Hahmann, and Marie-Paule Cani. 2015. Real-time continuous self-replicating details for shape deformation. *Computers & Graphics* 51 (2015), 67–73. https://doi.org/10.1016/j.cag.2015.05.011

Meitar Shechter, Rana Hanocka, Gal Metzer, Raja Giryes, and Daniel Cohen-Or. 2022. NeuralMLS: Geometry-Aware Control Point Deformation. (2022). https://doi.org/10.2312/egs.20221034

Olga Sorkine and Marc Alexa. 2007. As-Rigid-As-Possible Surface Modeling. In *Symposium on Geometry processing*, Vol. 4. 109–116. https://doi.org/10.2312/SGP/SGP07/109-116

Alvaro Javier Fuentes Suárez, Evelyne Hubert, and Cédric Zanni. 2019. Anisotropic convolution surfaces. *Computers & Graphics* 82 (2019), 106–116. https://doi.org/10.1016/j.cag.2019.05.018

Masamichi Sugihara, Erwin De Groot, Brian Wyvill, and Ryan M Schmidt. 2008. A Sketch-Based Method to Control Deformation in a Skeletal Implicit Surface Modeler. In *SBIM*. The Eurographics Association, 65–72. https://doi.org/10.2312/SBM/SBM08/065-072

Minhyuk Sung, Zhenyu Jiang, Panos Achlioptas, Niloy J. Mitra, and Leonidas J. Guibas. 2020. DeformSyncNet: Deformation transfer via synchronized shape deformation spaces. *ACM Trans. Graph.* 39, 6, Article 261 (nov 2020), 16 pages. https://doi.org/10.1145/3414685.3417783

Rodolphe Vaillant, Loïc Barthe, Gaël Guennebaud, Marie-Paule Cani, Damien Rohmer, Brian Wyvill, Olivier Gourmel, and Mathias Paulin. 2013. Implicit skinning: Real-time skin deformation with contact modeling. *ACM Transactions on Graphics (TOG)* 32, 4 (2013), 1–12. https://doi.org/10.1145/2461912.2461960

Daniel Vlasic, Ilya Baran, Wojciech Matusik, and Jovan Popović. 2008. Articulated mesh animation from multi-view silhouettes. In *Acm Siggraph 2008 papers*. 1–9. https://doi.org/10.1145/1360612.1360696

Yu Wang, Alec Jacobson, Jernej Barbič, and Ladislav Kavan. 2015. Linear subspace design for real-time shape deformation. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 1–11. https://doi.org/10.1145/2766952

Fangyin Wei, Elena Sizikova, Avneesh Sud, Szymon Rusinkiewicz, and Thomas Funkhouser. 2020. Learning to infer semantic parameters for 3D shape editing. In *2020 International Conference on 3D Vision (3DV)*. IEEE, 434–442. https://doi.org/10.1109/3DV50981.2020.00053

Womp. 2022. *Womp*. Retrieved May 23, 2023 from https://womp.com

Brian Wyvill, Kevin Foster, Pauline Jepp, Ryan M Schmidt, Mario Costa Sousa, and Joaquim A Jorge. 2005. Sketch based construction and rendering of implicit models. In *CAe*. 67–74. https://doi.org/10.2312/COMPAESTH/COMPAESTH05/067-074

Dong-Ming Yan, Yang Liu, and Wenping Wang. 2006. Quadric surface extraction by variational shape approximation. In *Geometric Modeling and Processing-GMP 2006: 4th International Conference, Pittsburgh, PA, USA, July 26-28, 2006. Proceedings 4*. Springer, 73–86. https://doi.org/10.1007/11802914_6

Wang Yifan, Noam Aigerman, Vladimir G Kim, Siddhartha Chaudhuri, and Olga Sorkine-Hornung. 2020. Neural cages for detail-preserving 3d deformations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 75–83. https://doi.org/10.1109/CVPR42600.2020.00015

Wang Yifan, Lukas Rahmann, and Olga Sorkine-Hornung. 2021. Geometry-consistent neural shape representation with implicit displacement fields. *arXiv preprint arXiv:2106.05187* (2021). https://openreview.net/forum?id=yhCp5RcZD7

Kangxue Yin, Jun Gao, Maria Shugrina, Sameh Khamis, and Sanja Fidler. 2021. 3DStyleNet: Creating 3D Shapes with Geometric and Texture Style Variations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 12456–12465. https://doi.org/10.1109/ICCV48922.2021.01223

Yu-Jie Yuan, Yu-Kun Lai, Tong Wu, Lin Gao, and Ligang Liu. 2021. A revisit of shape editing techniques: From the geometric to the neural viewpoint. *Journal of Computer Science and Technology* 36, 3 (2021), 520–554. https://doi.org/10.1007/s11390-021-1414-9

Mehmet Ersin Yumer, Siddhartha Chaudhuri, Jessica K Hodgins, and Levent Burak Kara. 2015. Semantic shape editing using deformation handles. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 1–12. https://doi.org/10.1145/2766908

Cédric Zanni, Paul Bares, Ares Lagae, Maxime Quiblier, and Marie-Paule Cani. 2012. Geometric details on Skeleton-based implicit surfaces. In *Eurographics 2012-33rd Annual Conference of the European Association for Computer Graphics*. Eurographics, 49–52. https://doi.org/10.2312/conf/EG2012/short/049-052

Cédric Zanni, Adrien Bernhardt, Maxime Quiblier, and M-P Cani. 2013. SCALe-invariant Integral Surfaces. In *Computer Graphics Forum*, Vol. 32. Wiley Online Library, 219–232. https://doi.org/10.1111/cgf.12199

Cédric Zanni, Michael Gleicher, and M-P Cani. 2015. N-ary implicit blends with topology control. *Computers & Graphics* 46 (2015), 1–13. https://doi.org/10.1016/j.cag.2014.09.012

Cédric Zanni, Evelyne Hubert, and M-P Cani. 2011. Warp-based helical implicit primitives. *Computers & Graphics* 35, 3 (2011), 517–523. https://doi.org/10.1016/j.cag.2011.03.027

Zerong Zheng, Tao Yu, Qionghai Dai, and Yebin Liu. 2021. Deep implicit templates for 3d shape representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 1429–1439. https://doi.org/10.1109/CVPR46437.2021.00148

Maxon ZSpheres. [n. d.]. ZSphere documentation. https://docs.pixologic.com/user-guide/3d-modeling/modeling-basics/creating-meshes/zspheres/

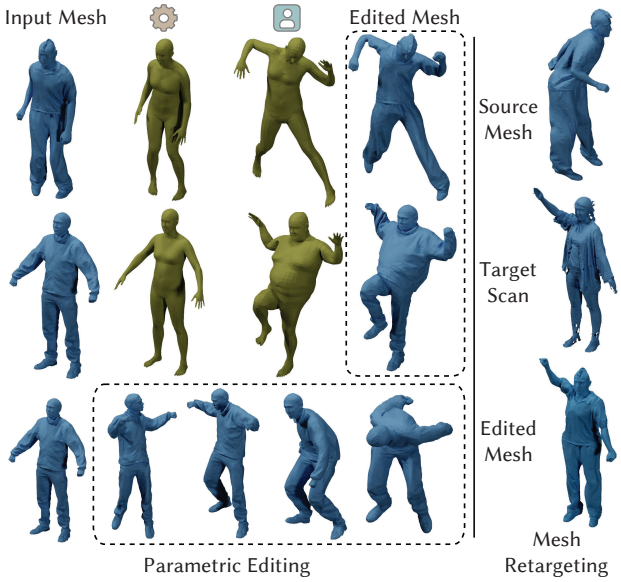

Figure 10: Left: Parametric editing of clothed human meshes (from [Vlasic et al. 2008]) using *SMPL* templates. First row shows an edit to the body pose parameters while the second row shows edits to both pose and shape parameters. The third row shows variations of a source human mesh. Right: Mesh retargeting to fit the pose and shape of an input target scan. This demonstrates transferability of our approach to existing specialized templates.

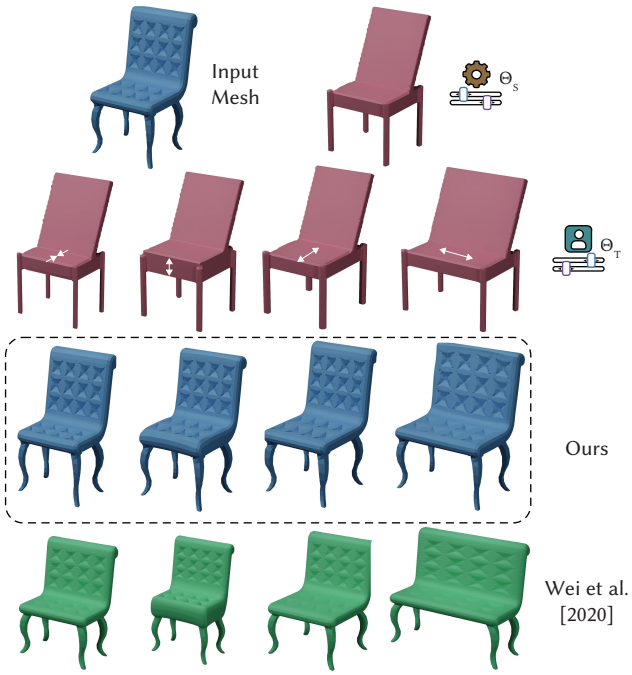

Figure 11: Our deformation formulation better preserves geometric details for large edits while portraying greater fidelity to the target template. Note that here we are using the couch template which is able to represent chairs as well.

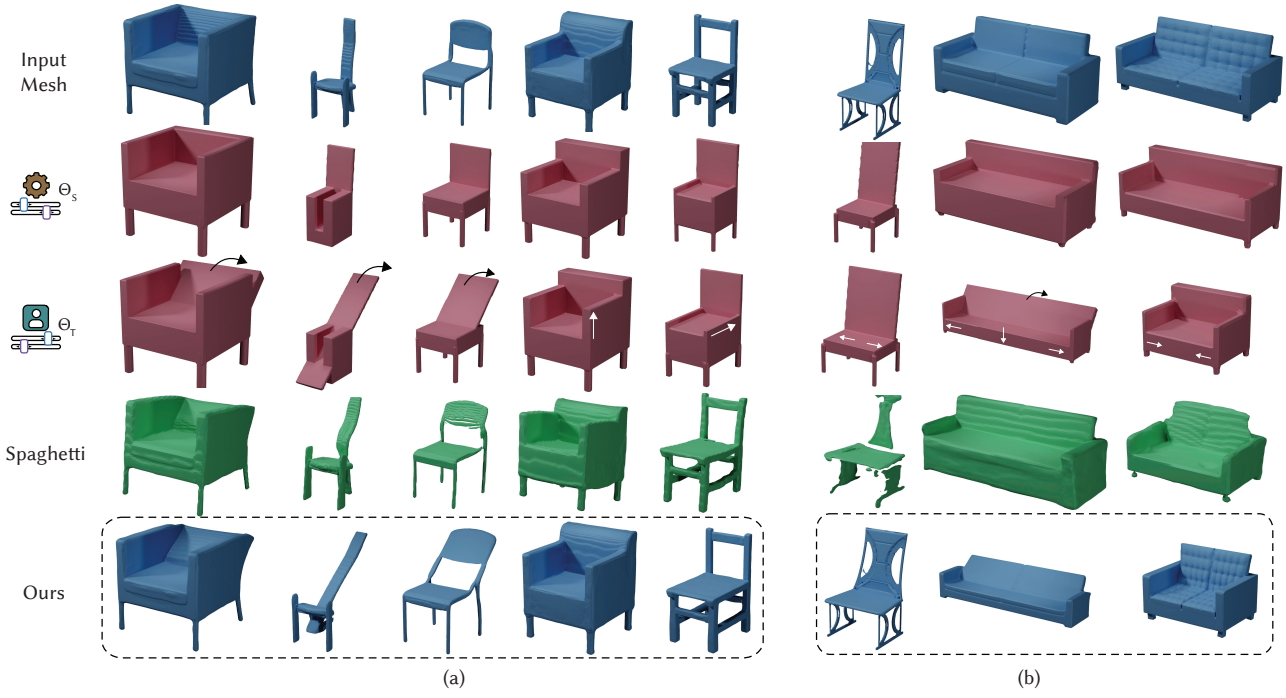

Figure 12: Comparison to the results of *Spaghetti* [Hertz et al. 2022] on (a) samples from their test set: We observe our approach works robustly even for large edits and preserves the structure better. (b) new samples: *Spaghetti* suffers from generalization problem on out of distribution samples.