Slippage-Preserving Reshaping of Human-Made 3D Content

CHRYSTIANO ARAÚJO, University of British Columbia, Canada
NICHOLAS VINING, University of British Columbia, Canada and NVIDIA, Canada
SILVER BURLA, University of British Columbia, Canada
MANUEL RUIVO DE OLIVEIRA, University of British Columbia, Canada
ENRIQUE ROSALES, University of British Columbia, Canada
ALLA SHEFFER, University of British Columbia, Canada

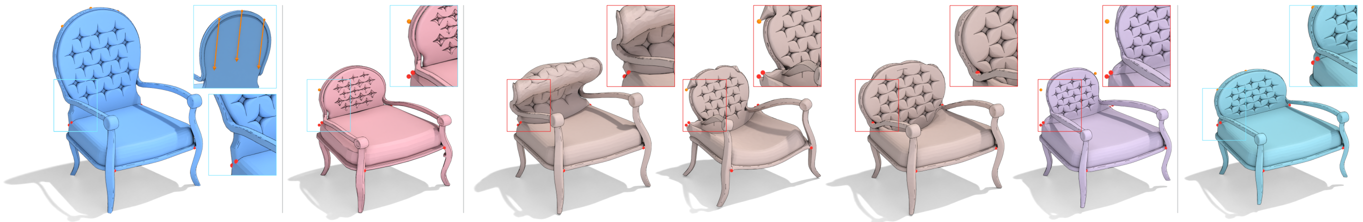


Fig. 1. Slippage-Preserving 3D Reshaping: (a) An input model and a reshaping gesture specified via new handle vertex positions (fixed handle locations in red, relocated in orange with arrows showing direction of change). (b) Reshaping output created by an artist based on the gesture. Given this input, state-of-the-art 3D deformation methods (c, As-Rigid-As-Possible [Chao et al. 2010]), conformal (d, [Vaxman et al. 2015]) and baseline Poisson deformation (e) produce unintuitive results. The outputs of our 3D extension of 2D ALUP reshaping [Araújo et al. 2022] (f) are closer to what viewers expect than all these alternatives, but exhibit undesirable scaling and orientation changes. Our slippage-preserving reshaping method (g) produces outputs well aligned with user expectations. Perceptual study participants unanimously preferred our result (g) over all algorithmic alternatives and judged it as on par with artist created one (b). The "chair" model is provided courtesy of Lun et al. [2015].

Artists often need to *reshape* 3D models of human-made objects by changing the relative proportions or scales of different model parts or elements while preserving the look and structure of the inputs. Manually reshaping inputs to satisfy these criteria is highly time-consuming; the edit in our teaser took an artist 5 hours to complete. However, existing methods for 3D shape editing are largely designed for other tasks and produce undesirable outputs when repurposed for reshaping. Prior work on 2D curve network reshaping suggests that in 2D settings the user-expected outcome is achieved when the reshaping edit keeps the *orientations* of the different model elements and when these elements *scale as-locally-uniformly-as-possible (ALUP)*. However, our observations suggest that in 3D viewers are tolerant of *non-uniform* tangential scaling if and when this scaling preserves *slippage* and reduces changes in element size, or *scale*, relative to the input. Slippage preservation requires surfaces which are locally *slippable* with respect to a given rigid motion to retain this property post-reshaping (a motion is slippable if when applied to the surface, it slides the surface along itself without gaps). We build on these observations by first extending the 2D ALUP framework to 3D and then modifying it to allow non-uniform scaling while promoting slippage and scale preservation. Our 3D ALUP extension produces reshaped outputs better aligned with viewer expectations than prior alternatives; our slippage-aware method further improves the outcome producing results on par with manual reshaping ones. Our method does not require any user input beyond specifying control handles and their target locations. We validate our method by applying it to over one hundred diverse inputs and by comparing our results to those generated by alternative approaches and manually. Comparative study participants preferred our outputs over the best performing traditional deformation method by a 65% margin and over our 3D ALUP extension by a 61% margin; they judged our outputs as at least on par with manually produced ones.

CCS Concepts: • **Computing methodologies → Shape modeling**.

Additional Key Words and Phrases: Slippage, Shape Editing, Reshaping

ACM Reference Format:
Chrystiano Araújo, Nicholas Vining, Silver Burla, Manuel Ruivo de Oliveira, Enrique Rosales, and Alla Sheffer. 2023. Slippage-Preserving Reshaping of Human-Made 3D Content. *ACM Trans. Graph.* 42, 6, Article 272 (December 2023), 18 pages. https://doi.org/10.1145/3618391

Authors' addresses: Chrystiano Araújo, University of British Columbia, Canada, araujoc@cs.ubc.ca; Nicholas Vining, University of British Columbia, Canada and NVIDIA, Canada, nvining@cs.ubc.ca; Silver Burla, University of British Columbia, Canada, silverbu@cs.ubc.ca; Manuel Ruivo de Oliveira, University of British Columbia, Canada, m.oliveira@math.ubc.ca; Enrique Rosales, University of British Columbia, Canada, albertr@cs.ubc.ca; Alla Sheffer, University of British Columbia, Canada, sheffa@cs.ubc.ca.

0730-0301/2023/12-ART272
https://doi.org/10.1145/3618391

1 INTRODUCTION

3D models of human-made objects (Figs. 1, 2) are ubiquitous in virtual environments and digital worlds. Creating such content from scratch is time consuming, and while large repositories of commercially licenseable content (e.g. [Ske 2023; Tur 2023]) exist, artists often need to modify this existing content for their needs [Araújo et al. 2022; Kraevoy et al. 2008]. In particular, our discussions with digital artists and designers suggest that they often want to *reshape*, or rescale, the inputs by changing the proportions, scale, or relative locations of different model parts, features, and geometric elements. See Figures 1-3 for typical examples, such as making the lighthouse stairs taller or the phone handle shorter. When manually reshaping content, they seek to preserve local and global surface structures [Araújo et al. 2022; Kraevoy et al. 2008]. Existing modeling packages do not provide tools specifically for reshaping, forcing artists to rely on repetitive basic modeling operations. This makes reshaping a complex task, requiring significant time and expertise - lowering the chair back (Fig. 1b) took an expert modeler over five hours. We propose an algorithmic reshaping approach that is capable of producing comparable outputs fully automatically and with minimal user guidance (Fig 1g, Fig 2g).

While shape deformation and editing has been extensively researched, existing methods are ill-suited for reshaping human-made content (Sec 2). Surface deformation methods typically target posing or animation of natural or organic objects, and are designed to be shape preserving. Repurposing these methods for reshaping human-made content produces outputs that are not aligned with user expectations (Figs. 1c-e, 2c-e). Kraevoy et al. [2008] support global axis-aligned resizing of human-made content; however, their framework does not generalize to local reshaping operations where users want to change the *relative* proportions of different object parts. Part-based editing methods are limited in the operations they support, and do not allow for global propagation of reshaping actions, such as the impact of raising the stairs on the overall structure of the lighthouse in Fig 2,top.

Araújo et al [Araújo et al. 2022] propose a targeted method for reshaping 2D curve networks. Their research suggests that viewers expect reshaping of such networks to preserve the *orientation* of input curves as much as possible, and to scale these curves *as-locally-uniformly-as-possible* (ALUP). As the authors indicate, extending their 2D reshaping method to 3D surfaces is a non-trivial task. We achieve this goal by first reformulating the ALUP criteria in the context of 3D surface reshaping, and then developing a method for computing 3D reshaping outputs that satisfy these criteria (Sec 3). We observe that, in the 3D setting, the expectation that surfaces scale as uniformly as possible is equivalent to optimizing for the mapping from the input to the reshaped surfaces to be as *conformal* as possible. We formulate 3D ALUP reshaping as a computation of 3D-to-3D surface mappings that satisfy the user specified reshaping gesture (specified via positional constraints), and that are as orientation preserving and as conformal as possible. We then discretize this formulation using triangle meshes as the input and output surface representation, and solve for the desired mapping and corresponding reshaping outputs using a custom solver (Fig. 1f). The results obtained using this approach are notably better aligned with viewer expectations than those produced by traditional ARAP [Chao et al. 2010], conformal [Vaxman et al. 2015], or Poisson deformation approaches (Fig. 1c-e). However, as confirmed by our user studies (Sec. 5), they frequently exhibit undesirable uniform scaling and other artifacts diverging from user expectations (see Fig. 1f versus Fig. 1b). Araújo et al. [2022] acknowledge such undesirable scaling as a possible, yet rare, drawback of their method; while rare in 2D, our experiments show such divergence from human expectations is significantly more frequent in 3D.

We avoid this undesirable scaling and related artifacts by re-examining the properties that 3D reshaping needs to satisfy in order to be well aligned with human expectations. Our analysis suggests that viewers neither expect, nor desire, 3D reshaping to be conformal everywhere. Instead, viewer expectations are impacted by *slippage* and *scale* related considerations. Following [Gelfand and Guibas 2004], we define a surface as locally *slippable* if it is locally invariant under translational or rotational motion. Formally, a rigid motion M is a slippable motion of a surface S if the velocity vector of each point $x \in S$ under M is tangential to S. Our observations suggest that viewers expect surfaces which are locally *slippable* with respect to a given rigid motion to retain this property after reshaping, and prefer reshaping outputs that keep the original element *scale*, or dimensions, as much as possible when doing so does not impact orientation or slippage preservation. As discussed in Sec. 4, slippage is strongly correlated with surface principal curvatures: on spherical surfaces slippage is preserved only under uniform scaling, but slippable anisotropic surfaces remain slippable when tangentially scaled by constant, but potentially different, factors along their principal curvature directions (for instance, reducing the radius of a cylinder while preserving its length). Critical to our setting, our observations suggest that in 3D viewers tolerate such a slippage preserving non-uniform tangential scaling. In particular, users prefer the outputs to locally keep their *scale* closer to the original one when doing so does not violate slippage or orientation; e.g. keeping the original thickness of the phone handle in Fig. 3 while making it shorter rather than shrinking it uniformly.

We use these observations to refine our reshaping formulation. First, we relax the conformality requirement in non-spherical surface areas; second, we complement the prior expectation of across-the-board orientation preservation with the need for across-the-board slippage preservation. Finally, we incorporate an incentive to preserve original scale when doing so does not negatively impact slippage or orientation. We formulate the computation of the outputs we seek as a variational optimization problem. We discretize this formulation by casting it as an optimization problem whose variables are the per-triangle reshaping transformations and the output mesh vertex positions (Sec 4), and find the transformations and positions that satisfy these properties using a custom alternating least-squares solver.

Our contribution is thus two-fold: we first extend the ALUP 2D curve reshaping framework to operate on 3D meshes, achieving better reshaping outputs than prior alternatives; we then further refine this extended framework to accurately capture human expectations of 3D reshaping.

We demonstrate our method's robustness and versatility by generating 128 different reshaping outputs, starting from diverse inputs and reshaping task specifications. We validate our approach by comparing our results to those created by artists and via algorithmic alternatives (Sec 5). In a perceptual study, participants preferred our slippage-aware results over the best performing algorithmic alternative by a margin of 65% (ours 75%, alternative 9%, equally good 11%, equally bad 5%) and judged them as on par with manual reshaping outputs. In comparisons against our 3D ALUP extension, our slippage-preserving method was preferred by a margin of 61% (our 67%, ALUP 6%, equally good 22%, equally bad 5%).

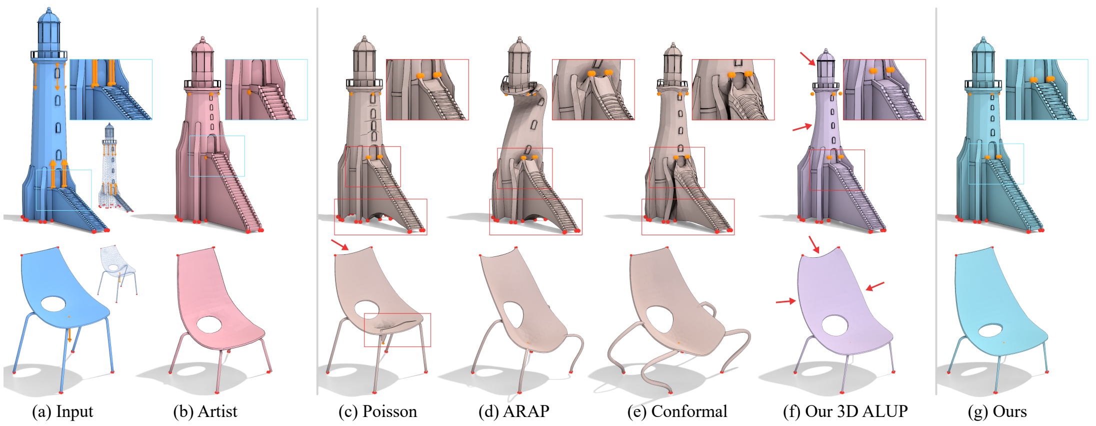


Fig. 2. Given the input models and gesture specifications (a) an expert modeler created the pink reshaped outputs (b). It took them three hours to edit the lighthouse and 70min to edit the chair. Given the editing gestures in (a), Poisson (c), state of the art ARAP [Chao et al. 2010; Sorkine and Alexa 2007] (d) and conformal [Vaxman et al. 2015] (e) deformation methods visibly shear the inputs and change surface orientation. Our 3D extension of 2D ALUP reshaping [Araújo et al. 2022] (f) has less pronounced artifacts, but still notably diverges from viewer expectations. Our new slippage-preserving reshaping method (g) preserves both orientation and slippage, and produces results of similar quality to those created by the expert modeler (b). Our method took 2.5 minutes to generate results for the lighthouse; 32 seconds for the chair. Lighthouse model © tivsol - sketchfab.com. The "chair" model is provided courtesy of Lun et al. [2015].

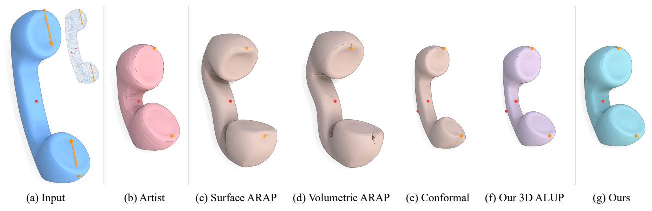


Fig. 3. Given the input models and gesture specifications (a) an expert modeler created the reshaped output in (b). It took him 75min to generate. Our algorithmic results (g) took 7 seconds to generate, are visually almost identical to the artist generated ones, and required the user to perform a handful of mouse click-and-drag operations. Our results are notably better aligned with artist ones than those created using surface (c) and volumetric (d) ARAP methods [Chao et al. 2010; Sorkine and Alexa 2007], conformal deformation [Vaxman et al. 2015] (e), and our 3D extension of 2D ALUP reshaping [Araújo et al. 2022] (f). The "phone" model is provided courtesy of Gori et al. [2017].

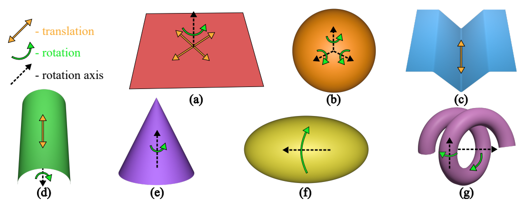


Fig. 4. Types of slippable surfaces and motions: (a) planar (translation and rotation around normal); (b) spherical (rotation around normal and around any sphere axis); (c) parabolic (translation) (d) elliptic (rotation around axis); (e) and (f) surfaces of revolution (rotation around axis); (g) helical - (approximate) rotation around two axes.

2 BACKGROUND AND RELATED WORK

Our work leverages slippage analysis and is related to methods for editing 2D and 3D content.

Slippage. The concept of slippage was introduced by Gelfand and Guibas [2004]. They define surfaces as locally slippable if they are locally invariant under translational or rotational motion (Fig 4). For example, spheres are invariant under rotation around any axis, planes are invariant under translation and rotation around the normal, and cylinders are translationally slippable along their axis and rotationally slippable around it. Gelfand and Guibas [2004] and follow-up works use slippage analysis to segment meshes into regions with different slippage properties for reverse engineering purposes. Kraevoy et al. [2008] suggest that viewer tolerance to surfaces being scaled non-uniformly along an axis during editing operations is a function of the degree to which the surface is translationally *slippable* along this axis. They utilize this property in the context of grid-based global resizing, as described below. Kurz et al. [2014] and Bokeloh et al. [2011] employ slippage analysis solely for the detection of translational symmetries; they do not attempt to preserve slippage during deformation. Nieser et al. [2012] locally classify surface regions as cylindrical, spherical, or hyperbolic, and employ this classification to guide directional constraints in a parametrization problem; they also do not address structure-preserving deformation. We speculate that, in the context of reshaping, viewers expect surfaces that are slippable with respect to a rigid motion to retain this property as much as possible post reshaping. We leverage this observation in our reshaping method (Sec. 4) and validate our results to be well-aligned with human expectation via perceptual studies (Sec. 5).

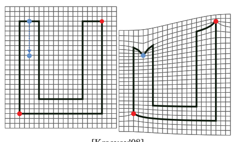
Shape-Preserving Deformation. Deforming shapes to conform to user specifications is a well researched computer graphics problem (see [Cohen-Or et al. 2015; Gain and Bechmann 2008; Yuan et al. 2021] for surveys.) The vast majority of deformation methods are designed for posing or animating natural shapes, and target local shape preservation. As-rigid-as-possible (ARAP) deformation methods seek to introduce a minimal amount of shear and scaling, thus preserving the local geometry, while allowing for rotation and translation [Alexa 2003; Chao et al. 2010; Igarashi et al. 2005; Jacobson et al. 2012; Joshi et al. 2007; Solomon et al. 2011; Sorkine and Alexa 2007; Yu et al. 2004]. Conformal deformation methods relax these requirements, allowing for uniform scaling in addition to rotation and translation [Crane et al. 2011; Vaxman et al. 2015; Weber and Gotsman 2010]. Reshaping, by definition, aims to change the absolute or relative scale and proportions of different elements in human-made objects while preserving their implicit and explicit structures. Unlike the shape-preserving setup, reshaping must therefore maintain surface orientation as much as possible. Rather than heavily penalizing non-uniform scaling as all above methods do, we allow such scaling if and when it preserves slippage and when doing so allows for better orientation and local scale preservation.

We compare our method against representative shape-preserving baselines: basic Poisson deformation, as described in [Cohen-Or et al. 2015]; ARAP deformation [Chao et al. 2010; Sorkine and Alexa 2007] as implemented in [Jacobson et al. 2018]; and conformal deformation [Vaxman et al. 2015] (e.g. Figs. 2, 3), Basic Poisson deformation penalizes any deviation from the original shape (including rotation), and was shown to outperform other alternatives for 2D reshaping [Araújo et al. 2022]. It thus provides an important baseline for our comparisons. Our outputs are significantly better aligned with user expectations than all alternatives (Sec. 5).

Cage-Based and Volumetric Deformation. Cage-based deformation methods [Li and Hu 2012; Lipman et al. 2008; Sumner et al. 2007; Thiery and Boubekeur 2022; Thiery et al. 2018; Zhang et al. 2014] use closed, low-polygon count polyhedral cages as a control mechanism for deforming 3D meshes; see [Nieto and Susín 2013] for a survey. Jacobson et al. [2011] propose the use of bounded biharmonic weights for deformation using a linear blending scheme. However, as acknowledged by the authors (see Fig. 17 in [Jacobson et al. 2011]), using their method for cage-based deformation can lead to orientation distortion in the underlying shape, violating a key requirement for reshaping tasks. Volumetric deformation methods [Chao et al. 2010; Shi et al. 2006; Zhou et al. 2005] operate on polyhedral meshes, which provide greater rigidity and help prevent self-intersections during deformation. Like the surface-based methods above, both cage and volumetric techniques are primarily designed for posing and animation and thus focus on locally shape-preserving deformations. Figs. 3 and 13 compare our outputs against a standard volumetric ARAP formulation applied to tetrahedral meshes of our input surfaces [Chao et al. 2010]; the outputs of this method exhibit similar characteristics to those of the surface based ARAP methods, and as such are ill-suited for reshaping needs.

Grid-Based Resizing and Retargeting. Global resizing or retargeting methods encase 2D or 3D shapes in bounding boxes, meshed using regular grids, resize or deform these boxes, and propagate the box's grid deformation to the enclosed shape [Artusi et al. 2016]. Most such methods [Gal et al. 2006; Wang et al. 2008; Wolf et al. 2007; Zhang et al. 2009] employ local shape preserving or ARAP formulations, similar to those discussed above and thus exhibit similar artifacts on our data. Xiao et al. [2014] utilize a tetrahedral mesh instead of a grid and allow enforcement of user-specified symmetries. Their method uses mean-value coordinates to deform the tet mesh and consequently does not preserve any of the properties we seek.

Kraevoy et al. [2008] address resizing of 3D human-made objects and identify orientation and slippage as key properties that need preservation when editing such content. They embed the inputs in hexahedral grids and estimate slippage relative to the grid directions; they then resize the inputs by deforming this grid using a variant of Poisson deformation that seeks to scale grid cells uniformly when they contain surface patches which are not translationally slippable along the resizing axis, while allowing cells that only contain patches slippable along the axis to scale non-uniformly.


[Kraevoy'08]

Grid-based resizing methods such as [Kraevoy et al. 2008] can produce grids with inverted elements, resulting in severe artifacts on the resized outputs [Panozzo et al. 2012]. As demonstrated by Araújo et al.[2022], using such methods for local reshaping instead of global resizing can result in inverted elements and corresponding catastrophic edit outputs even for simple gestures (see inset). Our method is inspired by the work of Kraevoy et al., but targets the much more challenging and general problem of reshaping, which requires support for fine grained localized edits. It operates directly on the input meshes and can robustly perform diverse local reshaping edits (Figs. 1, 2) as well as global resizing (Fig. 3).

Panozzo et al. [2012] prevent foldovers in grid-based 2D deformation by constraining all points that share a common x or y value to continue to do so; this approach dramatically constrains the user's degrees of freedom and as such cannot be used for the vast majority of edits shown in this paper.

Neural Deformation. Neural deformation methods use loss functions that encode the properties of the desired deformation operation, or alternatively learn these properties from deformation examples. Methods that fall into the first category include Hertz et al. [2022a], who deform source meshes using a progressive positional encoding that aims for as-rigid-as-possible deformation throughout the training process; and Yifan et el. [2020] who extend the cage-based approaches above to learned weights and share the same local shape preservation goals. Thus, the outputs of these methods exhibit similar characteristics to those of traditional shape preserving methods, making them unsuitable for our needs. We are not aware of any learning-based methods that use loss functions suitable for reshaping. In the second category, Jiang et al. [2020] learn deformation spaces on classes of 3D shapes from deformation examples, and demonstrate both traditional shape-preserving deformation and deformation of CAD models via explicit parametric controllers. Tang et al. [2022] deform meshes using neural shape deformation priors which they take from character animation sequences. Applying these approaches to reshaping would require manually creating enough diverse, and thus sufficiently representative, examples. Given the effort it takes to create a single example (Fig 1), this task would be prohibitively time consuming. Our learning-free method does not require such examples. Neuform [Lin et al. 2022] simultaneously learns both a latent shape space for a class of objects and an overfitted model of a specific input, then uses both representations simultaneously for limited reshaping operations. This approach requires both large shape databases and significant runtimes: to reshape one chair, Neuform requires a latent shape space trained on 7800 chairs, 33 hours of training to learn the general shape space, and 25 minutes to learn the specific chair being reshaped. We require no training data, and our median runtimes are under 30 seconds.

Part Based Editing. Part-based editing methods leverage a partition of models into separable parts, and support deformation of individual parts, either via simple axial scaling or shape preserving methods similar to those discussed above (e.g. [Chaudhuri et al. 2011; Funkhouser et al. 2004; Lun et al. 2015; Ma et al. 2014]). As discussed by [Kraevoy et al. 2008] axial scaling is ill-suited for editing of human-made content as it changes the orientation of surfaces not orthogonal to one of the axes, see Fig 14e for a typical outcome. Shape-preserving deformation is similarly ill-suited for reshaping, as discussed above. More importantly, many reshaping operations are inherently non-local - scaling the stairs on the lighthouse (Fig 2) requires adjusting feature locations and sizes across the rest of the building. Part-based methods are not designed to support such reshaping propagation. Part-based neural deformation methods like SPAGHETTI [Hertz et al. 2022b] operate on neural implicits, learning partwise decompositions directly on them. Their framework implicitly encourages changes in surface orientation, which we show is undesirable in a reshaping setting. DeformSyncNet [Sung et al. 2020] performs part-based deformation transfer across a latent shape space, targeting different models in a class of objects. The problem it solves is orthogonal to our direct editing method.

Structure-Preserving Deformation. Early methods for structure-preserving shape editing [Gleicher 1992; Hsu et al. 1993; Sutherland 1964] required users to explicitly manually specify all structural constraints, necessitating significant time and expertise to operate. Interfaces for semi-manual global resizing of 2D vector icons [Bernstein and Li 2015; Dragicevic et al. 2005] similarly require users to formulate the explicit structural constraints they want to enforce. Cabral et al. [2009] target structure-preserving reshaping of textured architectural models; they identify planar surfaces and preserve their orientations during editing. Gal et al. [2009] detect geometric relationships on 3D shapes and attempt to explicitly preserve them. They acknowledge that the method can easily become over-constrained, and has no built in mechanisms for automatically relaxing constraints users do not wish to enforce. Wu et al. [2014] preserve automatically detected or manually annotated symmetries during deformation, but do not target orientation or slippage preservation.

We target free-form shapes, for which preserving orientations alone without considering other properties can produce undesirable outputs (see e.g. Fig. 6), and support reshaping edits for which orientation preservation is not always achievable (e.g. the reshaping gesture on the bike in Fig 10 can only be satisfied by rotating the diagonal bar). Our framework preserves structures implicitly and relaxes orientation and slippage preservation when these cannot be strictly satisfied. This both prevents overconstraining and avoids the need for users to explicitly annotate features or relations they wish to preserve. By not relying on explicit detection of slippable areas (beyond approximately detecting spherical regions), we avoid the issues inherent in using fuzzy and fragile thresholds [Gelfand and Guibas 2004].

Reshaping Parametric Models. A large body of research addresses representation and editing of parametrically described CAD models [Cascaval et al. 2022; Kelly et al. 2015; Michel and Boubekeur 2021; Schulz et al. 2017]. Editing operations on such models preserve their structures by design. We operate on free-form meshes for which no structural information is available. Converting meshes into parametric models via reverse engineering is itself a long standing research problem [Buonamici et al. 2018].

2D Reshaping. As-Locally-Uniformly-as-Possible (ALUP) reshaping [Araújo et al. 2022] operates on 2D curve networks, and aims to preserve curve orientation and penalize non-uniform curve scaling. We extend their method to surfaces in 3D in Sec. 3. As with most geometry processing tasks, and specifically shape deformation [Cohen-Or et al. 2015], extending a 2D curve-based formulation to 3D surfaces requires both a different, more refined, formulation and a different corresponding solution mechanism. While this approach visibly outperforms traditional methods, its outputs significantly diverge from user expected ones (e.g. Fig 1f versus Fig 1b). Specifically, as noted above, unlike the 2D setting where users expect content to scale as-uniformly-as-possible in response to reshaping gestures, in 3D uniform scaling is often undesirable, see e.g. Fig 3. Consequently, as our comparisons show our novel slippage-preserving formulation (Sec. 4) significantly outperforms this 3D ALUP baseline.

3 3D ALUP

Before describing our slippage-preserving 3D reshaping method (Sec 4), we briefly review the 2D ALUP reshaping formulation of Araújo et al [2022] (Sec 3.1) and propose an extension of this formulation to 3D (Sec. 3.2).

3.1 2D ALUP

Araújo et al. [2022] formulate 2D reshaping as a computation of a mapping from input to output curve networks that satisfies a set of user-defined positional and optional straightness constraints. They seek a mapping that is locally as similar as possible to a uniform scale, and express this property via two conditions: first, they expect normals at the corresponding points on the input and output curves to be maximally similar; second, they expect the gradient of the tangent length at the corresponding points on the curves to be similar as well. They express these two properties variationally. Given an *arc-length parameterized* smooth input curve $C_i(u) : [0, s] \to \mathbb{R}^2$, with pointwise tangents $\tau_i(u)$ and unit normals $n_i(u)$, they find a function $C_o : [0, s] \to \mathbb{R}^2$ representing the output curve, with pointwise tangents $\tau_o(u)$ and unit normals $n_o(u)$, that minimizes the sum of the following two functionals:

$$E_{normal} = \int_{u=0}^{s} \omega_n(u)(n_i(u) \cdot \frac{\tau_o(u)}{\|\tau_o u\|})^2 du \tag{1}$$

$$E_{tangent} = \int_{u=0}^{s} \omega_t(u)(\frac{d\|\tau_o(u)\|}{du} - \frac{d\|\tau_i(u)\|}{du})^2 du. \tag{2}$$

subject to the aforementioned constraints. The first functional, E_{normal}, encourages normal preservation. The second functional,

$E_{tangent}$, encourages the gradient of the lengths of the output tangents $\tau_o(u)$ to be maximally similar to that of the input ones $\tau_i(u)$. Here $\omega_n(u)$ and $\omega_t(u)$ are weight functions encoding the degree to which normal preservation and scale uniformity need to be maintained at different points along the curve.

For efficiency, and to allow processing of inputs where their constraints may not be strictly satisfiable, they enforce them softly by incorporating corresponding energy terms into the minimized functional, assigning these terms a very high weight $w_{constr} = 10^5$. They express positional constraints that stipulate that points $C_i(u_0), \ldots, C_i(u_k)$ on the curve should map to positions $p_0, \ldots, p_k$ as

$$E_{pos} = \sum_{i=0}^{k} \|C_o(u_i) - p_i\|^2 \tag{3}$$

and express straightness preservation along *a priori* straight sections $s \in S$ of the input curve $[u_s, u_e]$ as:

$$E_{straight} = \sum_{s \in S} \int_{u=u_s}^{u_e} \left\| \frac{d^2 C_o(u)}{du^2} \right\| du. \tag{4}$$

They then formulate reshaping goals as minimizing:

$$E^{2D}_{ALUP} = E_{normal} + E_{tangent} + w_{constr}(E_{pos} + E_{straight}). \tag{5}$$

They proceed to discretize this formulation by approximating each input curve by a densely sampled polyline $< V, E >$ where V are polyline vertices and E polyline edges. They reformulate the different terms (where E^d denotes discretized energy terms) as functions of the input shape vertex positions v_i^0 and their corresponding output positions v_i, aiming for a formulation that allows for an efficient minimization strategy:

$$E^d_{normal} = \sum_{\langle i,j \rangle \in E} \omega(v_j, v_i) \left(n^i_{ij} \cdot \frac{v_j - v_i}{\|v_j - v_i\|} \right)^2$$

$$E^d_{tangent} = \sum_{i \in V} \sum_{j \neq k \in N_i} \frac{\omega^d_t(ijk)}{L_{avg}} \left(\|v_j - v_i\| - \frac{\|v^0_j - v^0_i\|}{\|v^0_k - v^0_i\|} \|v_k - v_i\| \right)^2$$

$$E^d_{straight} = \sum_{i \in S} \sum_{j,k \in N_i; j \neq k} \left\| \frac{(v_j - v_i)}{\|v_j - v_i\|} - \frac{(v_i - v_k)}{\|v_k - v_i\|} \right\|^2 .$$

Here, $\omega(v_j, v_i)$ is a discretization of $\omega_n(u)$ and is defined as

$$\omega(v_j, v_i) = \max\left(1, \left(\left\|v_j - v_i\right\| / L_{avg}\right)^2\right) \tag{6}$$

where L_{avg} denotes the average length of the input polyline edges. These weights are designed penalize deviation more along longer edges. The weight $\frac{\omega^d_t(ijk)}{L_{avg}}$ is the discretizations of $\omega_t(u)$, where $\omega^d_t(ijk)$ reflect the visual smoothness at the center vertices j in the *input* curve network (see [Araújo et al. 2022] for details). The set S includes all vertices that are interior to the straight lines in the original curve network.

They minimize the discretized nonlinear energy $E^d_{ALUP}(v)$:

$$E^d_{ALUP} = E^d_{normal} + E^d_{tangent} + w_{constr}(E_{pos} + E^d_{straight}) \tag{7}$$

using an iterative least-squares solver which uses target output edge lengths l_{ij} as auxiliary variables. Key to their method is the observation that using these auxiliary variables lets normal preservation be expressed as preserving the directions of the polyline edges:

$$E'_{edge} = \sum_{\langle i,j \rangle \in E} \omega^l(v_i, v_j) \left\| \frac{v_i - v_j}{l_{ij}} - \frac{v^0_i - v^0_j}{l^0_{ij}} \right\|^2 \tag{8}$$

where $l^0_{ij} = \|v^0_i - v^0_j\|$, and $\omega^l(v_j, v_i) = \max\left(1, \left(l_{ij}/L_{avg}\right)^2\right)$.

This term is minimized when the output edges have similar directions to the original, and satisfy $\|v_i - v_j\| = l_{ij}$. They then rewrite the main energy terms as

$$E'_{normal} = \sum_{(i,j) \in E} \omega^l(v_i, v_j) \left(n^i_{ij} \cdot \frac{v_j - v_i}{l_{ij}} \right)^2 \tag{9}$$

$$E'_{tangents} = \sum_{i \in V} \sum_{j,k \in N_i; j \neq k} \omega^d_t(ijk) \frac{L_{avg}}{\|v_j - v_i\|} \left(l_{ij} - r_{ijk} l_{ik} \right)^2 \tag{10}$$

$$E'_{straight} = \sum_{i \in S} \sum_{j,k \in N_i; j \neq k} \left\| \frac{(v_j - v_i)}{l_{ij}} - \frac{(v_i - v_k)}{l_{ik}} \right\|^2 . \tag{11}$$

and approximate minimizing E^d_{ALUP} as minimizing

$$E'_{ALUP} = E'_{normal} + E'_{edge} + E'_{tangent} + w_{constr}(E_{pos} + E'_{straight}) \tag{12}$$

This formulation allows for a solution using least squares iterations which alternate between solving for positions while keeping the lengths fixed, and solving for lengths with fixed positions.

3.2 Extending ALUP to 3D

Our 3D equivalent of ALUP looks for a mapping τ from an input surface S_i to an output surface S_o that similarly satisfies a set of positional and optional straightness constraints, and is locally as-close-as-possible to a uniform scaling. For simplicity, we assume S_i and S_o are smooth and oriented and that τ is a diffeomorphism.

Our positional constraints, similar to 2D, stipulate that points C_k on the surface should map to positions $p_0, \ldots, p_k \in \mathbb{R}^3$; thus E_{pos} in 3D is a straightforward extension of its 2D counterpart. Assuming that each straight line on the surface can be parameterized as a function of a single parameter $u \in [u_s, u_e]$, the straightness energy formulation $E_{straight}$ can be extended basically as-is from 2D to 3D. Straightness preservation remains a desired property in 3D as planar or non-planar surface patches can meet along straight sections.

To formulate our ALUP goals in 3D, we therefore only need to define 3D equivalents to E_{normal} and $E_{tangent}$. Given a point p on S_i, let $n_i(p)$ be the unit normal vector at point $p \in S_i$, and let $\varphi = (u, v)$ be a local coordinate chart from a neighbourhood of p to $\Omega \subset \mathbb{R}^2$. The tangent plane at $\tau(u, v)$ is then spanned by $\frac{\partial}{\partial u}\tau(u, v)$ and $\frac{\partial}{\partial v}\tau(u, v)$. We can therefore express our desire for normal preservation as:

$$E^{3D}_{normal} = \int_{\Omega} \omega_n(u, v) \left((n_i(u, v) \cdot \frac{\frac{\partial}{\partial u}\tau(u, v)}{\|\frac{\partial}{\partial u}\tau(u, v)\|})^2 + (n_i(u, v) \cdot \frac{\frac{\partial}{\partial v}\tau(u, v)}{\|\frac{\partial}{\partial v}\tau(u, v)\|})^2 \right) dudv \tag{13}$$

In other words, E^{3D}_{normal} is minimized when the input normals remain orthogonal to the tangent plane at the output surface. Here

$\omega_n(u,v) = \max(1, \frac{\|\nabla\tau(u,v)\|}{\nabla_{avg}})$, where ∇_{avg} is the average length of the gradient of τ over all S_i.

We observe that, in the 3D context, requiring local uniformity is equivalent to requiring that τ is a conformal transformation. The conformality property can be expressed in terms of the *conformal energy* of the map τ, or the difference between the Dirichlet energy of τ and the area of the output surface S_o, given by integrating the Jacobian of τ over S_i:

$$E_{conformal} = \int_{S_i} (\frac{1}{2}|\nabla\tau|^2 - \mathcal{J}ac(\tau))dA \tag{14}$$

where dA is the area form on S_i. This energy is always nonnegative, and is known to be zero exactly when τ is conformal [Pinkall and Polthier 1993]. With these in place we formulate 3D ALUP reshaping as the minimization of

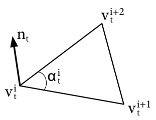
$$E^{3D}_{ALUP} = E^{3D}_{normal} + E_{conformal} + w_{constr}(E_{pos} + E_{straight}). \tag{15}$$

3.2.1 3D ALUP Discretization. Our next task is to apply this framework to typical 3D models we want to reshape, which are most commonly represented as triangular meshes. We denote the triangles of the input mesh as $t \in T$, and the input normals on these triangles as $\bar{n}_t$. We refer to the output mesh vertex positions as $v \in V$, and their corresponding input positions as $\bar{v} \in \bar{V}$. We denote the input positions of the vertices of a triangle t as $\bar{v}^0_t, \bar{v}^1_t, \bar{v}^2_t$, and the corresponding output positions as v^0_t, v^1_t, v^2_t. We formulate our goal of obtaining a desired reshaping as computing a piecewise linear mapping τ from the input mesh $\langle\bar{V}, T\rangle$ to the unknown output mesh $\langle V, T\rangle$. For simplicity, we assume without loss of generality that the handle points C_k form a subset of the mesh vertices V.

Normal Preservation. We encode E^{3D}_{normal} in discrete form as

$$E^{D}_{normal} = \sum_{t\in T}\sum_{i=0}^{2} \omega(v^{i+1}_t, v^i_t)\left(\bar{n}_t \cdot \frac{v^{i+1}_t - v^i_t}{\|v^{i+1}_t - v^i_t\|}\right)^2. \tag{16}$$

Conformality. The conformal energy $E^{3D}_{conformal}$ requires non-trivial and typically indirect machinery to optimize [Crane et al. 2011; Mullen et al. 2008; Vaxman et al. 2015]. We formulate our expectation of conformality in a much simpler quadratic form by implicitly reinforcing our goal of normal preservation and recasting conformality as the expectation that the input and output triangles have the same normals and are similar, or have the same corner angles.



Let R^i_t be a rotation matrix encoding a counterclockwise rotation around the input triangle normal $\bar{n}^t$ by the input triangle's angle α^i_t at the corresponding corner vertex $\bar{v}^i_t$. The triangle t is scaled uniformly while preserving its normal if, and only if, for any of its corners $i \in 0, 1, 2$ (where $i_3 = i_0$, etc.; see inset):

$$\frac{(v^{i+2}_t - v^i_t)}{\|\bar{v}^{i+2}_t - \bar{v}^i_t\|} - R^i_t\frac{(v^{i+1}_t - v^i_t)}{\|\bar{v}^{i+1}_t - \bar{v}^i_t\|} = 0. \tag{17}$$

With this relation in place, we define the discrete conformal energy as:

$$E^{D}_{conformal} = \sum_{t\in T}\sum_{i=0}^{2}\left\|\frac{(v^{i+2}_t - v^i_t)}{\|\bar{v}^{i+2}_t - \bar{v}^i_t\|} - R^i_t\frac{(v^{i+1}_t - v^i_t)}{\|\bar{v}^{i+1}_t - \bar{v}^i_t\|}\right\|^2 \tag{18}$$

This formulation builds on the LSCM parametrization (see Eq. 3.3 in [Sheffer et al. 2006]), extending it to the case where conformality needs to be preserved while keeping 3D triangle normals fixed.

Straightness. For each detected straight line, expressed as a vertex sequence $S = \{v^s_0, \dots v^s_{n(S)}\} \in \tilde{S}$, we encode the expectation of straightness preservation as

$$E^{D}_{straight} = \sum_{S\in\tilde{S}}\sum_{i=1}^{n(S)-1}\left\|\frac{v^s_{i+1} - v^s_i}{\|v^s_{i+1} - v^s_i\|} - \frac{v^s_i - v^s_{i-1}}{\|v^s_i - v^s_{i-1}\|}\right\|^2 \tag{19}$$

Replacing the continuous terms in E^{3D}_{ALUP} with their discrete counterparts to form a new energy E^{D}_{ALUP} results in a nonlinear optimization problem. Unfortunately, this problem cannot be solved with the strategy used for 2D ALUP, as the 3D framework requires supporting additional degrees of freedom. In particular, and in contrast to the 2D case, preserving normals in 3D space does not require preserving input edge directions. Constraining the solution to preserve edge directions unnecessarily constrains the outputs. This means that the relationship implied by Eq. 8 no longer holds for the solutions we seek. Since this relationship is central to the solution method of [Araújo et al. 2022], we require a different approach that does not penalize changes in output edge directions as long as these edges remain orthogonal to the input normal, and as long as conformality is maximally preserved.

3.3 3D ALUP Solver.

We recast the optimization of E^{3D}_{ALUP} in a solver-amenable form by explicitly considering the transformations from the input to output mesh triangles as additional variables in our formulation. We note that the mapping from the input to the output mesh is, by definition, linear on each triangle. Consequently, it can be thought of as a combination of per-triangle linear transformations τ_t (3×3 matrices), defined on the triangle's edges, $v^{i+1}_t - v^i_t = \tau_t(\bar{v}^{i+1}_t - \bar{v}^i_t)$ and translations. Since we seek to preserve surface orientation, an inherently global property, our linear transformations are defined using axis directions aligned with the global coordinate system. This choice differs from traditional deformation transfer [Sumner and Popović 2004] and similar setups, which use local per-triangle coordinate systems whose axes are dictated by triangle normal and edge directions.

We relate vertex positions and triangle transformations by stating the relationship between them along the mesh edges:

$$E_{transform} = \sum_{t\in T}\sum_{i=0}^{2}\omega(v^{i+1}_t, v^i_t)\left\|\tau^{-1}_t\frac{v^{i+1}_t - v^i_t}{\|v^{i+1}_t - v^i_t\|} - \frac{\bar{v}^{i+1}_t - \bar{v}^i_t}{\|\bar{v}^{i+1}_t - \bar{v}^i_t\|}\right\|^2$$

Applying the inverse of our mapping τ^{-1}_t to the output edges, rather than applying the mapping τ_t to the input ones, prevents error reduction through the decrease of $\|\tau_t\|$, and hence discourages undesirable edge shrinkage. We use the same weights ω_{ti} as the normal preservation energy to penalize deviation more along longer edges, while still penalizing it along shorter ones. We normalize the edge vectors to make this term independent of the input mesh resolution.

With these variables in place, our solver alternates between minimizing two quadratic energies approximating E^{3D}_{ALUP}, where one uses positions as unknowns and the other uses transformations as unknowns. At each iteration, we solve for vertex positions while keeping the transformation matrices fixed, then solve for transformations while keeping the positions fixed. Formulating our alternating steps requires addressing two challenges: first, we desire that the problems solved at each step are quadratic, allowing for straightforward minimization; second, we want the two alternating steps to share a common minimum to avoid oscillation.

3.3.1 Vertex Solve. In our first solve, we approximate E^{D}_{ALUP} by an energy which is quadratic with respect to vertex positions by keeping the transformations τ_t fixed, and by using fixed values l^i_t for edge lengths in lieu of the mesh edge length terms $\|v^{i+1}_t - v^i_t\|$ within the energy function. We write:

$$l^i_t = \|\tau_t(\bar{v}^{i+1}_t - \bar{v}^i_t)\|$$

and define $\omega_{ti} = \max\left(1, \left(l^i_t/L_{avg}\right)^2\right)$. We substitute l^i_t for the norms $\|v^{i+1}_t - v^i_t\|$ throughout E^D_{ALUP}. With this substitution, our approximating quadratic energy terms (denoted by E^L) become:

$$E^L_{transform} = \sum_{t\in T}\sum_{i=0}^{2}\omega_{ti}\left\|\tau_t^{-1}\frac{v^{i+1}_t - v^i_t}{l^i_t} - \frac{\bar{v}^{i+1}_t - \bar{v}^i_t}{\|\bar{v}^{i+1}_t - \bar{v}^i_t\|}\right\|^2, \quad (20)$$

$$E^L_{normal} = \sum_{t\in T}\sum_{i=0}^{2}\omega_{ti}(\bar{n}_t \cdot \frac{v^{i+1}_t - v^i_t}{l^i_t})^2, \quad (21)$$

$$E^L_{straight} = \sum_{S\in\bar{S}}\sum_{i=1}^{n(S)-1}\left\|\frac{v^s_{i+1} - v^s_i}{l^{i,i+1}_s} - \frac{v^s_i - v^s_{i-1}}{l^{i,i-1}_s}\right\|^2 \quad (22)$$

Since $E^D_{conformal}$ is already quadratic in the vertex positions, we use it as is. Our the overall reshaping energy thus becomes,

$$E^L_{ALUP} = E^L_{transform} + E^D_{conformal} + E^L_{normal} + w_{constr}(E_{pos} + E^L_{straight}) \quad (23)$$

We minimize this energy, and obtain updated output vertex positions using a standard least squares solver in the Eigen linear algebra package [Guennebaud et al. 2010].

3.3.2 Transformation Solve. In our second solve, we keep positions fixed and solve for the transformations that best satisfy our overall energy. We linearize $E_{transform}$ with respect to transformations by multiplying each term in the sum by τ_t,

$$E^{TL}_{transform} = \sum_{t\in T}\sum_{i=0}^{2}\omega_{ti}\left\|\frac{v^{i+1}_t - v^i_t}{\|v^{i+1}_t - v^i_t\|} - \tau_t\frac{\bar{v}^{i+1}_t - \bar{v}^i_t}{\|\bar{v}^{i+1}_t - \bar{v}^i_t\|}\right\|^2 \quad (24)$$

While the mesh connectivity is explicitly accounted for in position space, this is not the case for transformations. We explicitly require transformations for adjacent faces to agree on common edges by adding a connectivity preservation term for pairs of adjacent triangles:

$$E^{TL}_{connect} = \sum_{t\in T, s\in N(t)}\left\|\tau_t\frac{\bar{v}^1_t - \bar{v}^0_t}{\|\bar{v}^1_t - \bar{v}^0_t\|} - \tau_s\frac{\bar{v}^1_t - \bar{v}^0_t}{\|\bar{v}^1_t - \bar{v}^0_t\|}\right\|^2 \quad (25)$$

where, without loss of generality, we assume that $\bar{v}^0_t = \bar{v}^1_s, \bar{v}^1_t = \bar{v}^0_s$. We formulate our normal and conformality requirements in terms of transformations, ensuring that both steps in our solve have maximally overlapping minima:

$$E^{TL}_{normal} = \sum_{t\in T}\sum_{i=0}^{2}\omega_{ti}\left(\left(\tau_t\frac{\bar{v}^{i+1}_t - \bar{v}^i_t}{\|\bar{v}^{i+1}_t - \bar{v}^i_t\|}\right)\cdot\bar{n}_t\right)^2 \quad (26)$$

$$E^{TL}_{conformal} = \sum_{t\in T}\sum_{i=0}^{2}\left\|\frac{\tau_t(\bar{v}^{i+2}_t - \bar{v}^i_t)}{\|\bar{v}^{i+2}_t - \bar{v}^i_t\|} - R^i_t\frac{\tau_t(\bar{v}^{i+1}_t - \bar{v}^i_t)}{\|\bar{v}^{i+1}_t - \bar{v}^i_t\|}\right\|^2 \quad (27)$$

Three of our terms $E^{TL}_{connect}$, E^{TL}_{normal} and $E^{TL}_{conformal}$ are minimized when $\tau_t = 0$, leading the linear system we solve to be poorly conditioned. To stabilize the solution, we add a regularizer which promotes identity transformations

$$E^{TL}_{regularizer} = \sum_t \|\tau_t - I\|^2_F \quad (28)$$

where I is a 3×3 identity matrix and F is the Frobenius norm, assigning it a tiny weight $w_{reg} = 10^{-4}$.

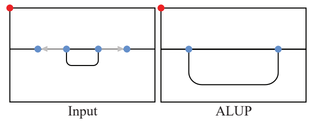
With these elements in place, our quadratic approximate reshaping energy, expressed in terms of transformations, is:

$$E^{TL}_{ALUP} = E^{TL}_{transform} + E^{TL}_{connect} + E^{TL}_{normal} + E^{TL}_{conformal} + w_{reg}E^{TL}_{regularizer}$$

Note that the straightness term is omitted from E^{TL}_{ALUP} as the edge length terms in the denominator would be non-quadratic with respect to τ_t. We minimize this energy using the same solver as above. See App. A for initialization and termination criteria.

Finalizing Outputs. Once the optimization converges, we apply an optional finalization step whose goals are to improve ALUP error distribution and degree of constraint satisfaction, App. A. This step addresses a common problem to surface deformation methods [Sorkine et al. 2004]: when the constraints cannot be satisfied without significant increase in the optimized energy, solutions tend to concentrate the error immediately next to the handles. Our finalization redistributes the error away from the handles.

Fig. 5 shows representative results generated using this framework. For some inputs (such as the sphere Fig. 5b), the results are consistent with human expectations; in many cases, however, this approach results in unexpected uniform scaling (Fig. 5e) and undesirable orientation changes (Fig. 5h). In particular, since many inputs do not allow simultaneous normal and local scale preservation, the ALUP output balances the two properties, sacrificing normal preservation to better preserve scale uniformity.



As acknowledged by Araújo et al. [2022], their goal of promoting uniformity everywhere potentially results in undesirable uniform scaling (see inset). While rare in 2D, undesirable uniform scaling becomes a larger issue in 3D. In [Araújo et al. 2022], uniform scaling propagation is avoided across the entire curve network by effectively segmenting the network at handles, and at sharp/high-valence corners. In 3D, handles and sharp features do not provide a natural segmentation. Consequently, to produce

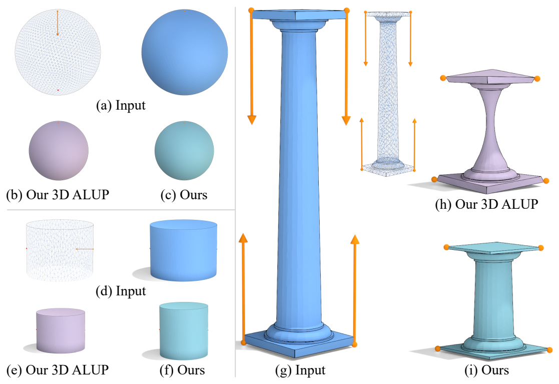


Fig. 5. Our 3D ALUP baseline produces acceptable results on simple inputs such as the sphere (a-b), but does not preserve input scale on anisotropic inputs (e), and sacrifices orientation to better preserve scale uniformity (h). Our slippage aware method does not exhibit such artifacts and generates results consistent with user expectation on all inputs. The "column" model is provided courtesy of Lun et al. [2015].

outputs consistent with user expectations we relax scale uniformity and refine this formulation as discussed in Sec. 4.

4 SLIPPAGE-PRESERVING RESHAPING

We seek to replace the enforcement of conformality used in the ALUP formulation with a weaker requirement of preserving slippage, and to use the extra degree of freedom this relaxation provides to better preserve the original input scale (when compatible with orientation and slippage preservation). To this end, we need to address two challenges: defining and enforcing the geometric requirements on our reshaping mapping τ that will promote slippage preservation, and detecting the slippable surfaces where these requirements need to be enforced. We note that slippable motion can be grouped into three distinct groups (Fig 4): translational motion along a straight path (parabolic and planar surfaces, e.g. Fig 4ac), rotational motion around an axis (cylindrical, conical, elliptic or helical surfaces, e.g. Fig 4d-f), and rotation around the surface normal (planar and spherical surfaces, e.g. Fig 4ab). Notably, while all slippable motions are preserved under conformal mappings. our key observation is that the first two motions are maintained under more relaxed conditions. Specifically, we observe that planar surface regions, by definition, remain planar under orientation preserving transformations. When analyzing nonplanar surface regions we distinguish between isotropic and anisotropic ones: a surface region is isotropic if, for all points p in the region, $\kappa_1(p) = \kappa_2(p)$; such a region is known to either be part of a sphere ($\kappa_i(p) \neq 0$) or a plane ($\kappa_i(p) = 0$) [do Carmo 1976]; a surface region is anisotropic if $\kappa_1(p) \neq \kappa_2(p)$. Surfaces slippable around the normal are by definition isotropic. Notably, the translational motion direction on anisotropic surfaces by definition coincides with the zero principal curvature direction (where one of $\kappa_i(p)$ is zero); similarly, for surfaces slippable around an axis, the rotation direction coincides with one of the principal curvature directions. These observations jointly suggest that slippable anisotropic surface regions remain slippable under a mapping that (1) preserves the surface Frenet frame (normal, and principal curvature directions); and (2) scales the surface uniformly along each principal curvature direction (e.g. scaling a cylinder's diameter and length by different scale factors). In other words, these surface regions remain slippable under tangential non-uniform scale, as long as the same amount of tangential scaling is applied at all points and as long as the transformation does not exhibit shear (while sufficient, this condition may not be always necessary). We refer to this property as *similarity*.

In contrast to planar or anisotropic areas, spherical surface regions remain slippable around the normal *only* under conformal mapping. Thus, to retain slippage across spherical surface regions, we look for a mapping that is conformal in these areas. We refer to this property as *sphericity*.

With these definitions in place, our second challenge is to identify slippable surface regions. Detecting approximately planar or spherical regions is relatively straightforward, as methods for robustly computing principal curvatures on meshes (e.g. [Panozzo et al. 2010]) are readily available. However, identifying which anisotropic surface regions are locally slippable is highly non-trivial [Gelfand and Guibas 2004; Kraevoy et al. 2008] and the distinction between anisotropic and isotropic surface regions on real data can often be fuzzy. Therefore, for practical purposes, we simply promote similarity across the entire output surface, and enforce conformality in approximately spherical regions; in our experiments, this approach produced outputs well-aligned with viewer expectations. Fig 6 shows the impact of slippage enforcement, demonstrating the impact of relaxing similarity or sphericity.

4.1 Slippage-Preserving Formulation

We support slippage-preserving reshaping by replacing the conformality functional $E_{conformal}$ with two terms, one that strongly promotes slippage preservation and one that encourages preservation of the input scale.

Slippage. We encode slippage as a combination of similarity and sphericity. We express our preference for similarity as:

$$E_{similarity} = \int_{\Omega} \left(\left\| \nabla_{S_o} \left|\left| \frac{\partial \tau}{\partial u} \right|\right| - \nabla_{S_i} \left|\left| \frac{\partial}{\partial u} \right|\right| \right\|^2 + \left\| \nabla_{S_o} \left|\left| \frac{\partial \tau}{\partial v} \right|\right| - \nabla_{S_i} \left|\left| \frac{\partial}{\partial v} \right|\right| \right\|^2 \right) dudv \tag{29}$$

To retain slippage across spherical surface regions, we look for a mapping that is conformal in these areas. We formulate this property as:

$$E_{sphericity} = \int_{S_i} \Gamma(\kappa_1(p), \kappa_2(p)) \left(\frac{1}{2} |\nabla \tau|^2 - \mathcal{J}ac(\tau) \right) dA \tag{30}$$

$\Gamma(\kappa_1, \kappa_2)$ serves as an indicator function which is 1 on spherical surfaces, and 0 elsewhere. On real life data, the distinction between spherical and non-spherical surfaces is never precise. Thus in practice, we define $\Gamma(\kappa_1, \kappa_2)$ to smoothly vary between 0 and 1 based on the magnitude of the two curvature values and degree of similarity between them:

$$f_{iso}(\kappa_1, \kappa_2) = (\kappa_1 - \kappa_2)/(\max(|\kappa_1|, |\kappa_2|) + \epsilon)$$
$$f_{sph}(\kappa_1, \kappa_2) = 1 - \min(|\kappa_1|, |\kappa_2|, k_{max})/k_{max}$$

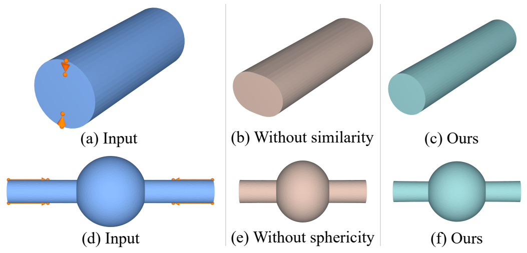


Fig. 6. Slippage preservation: (top) Preserving surface orientation only, without optimizing for similarity (b), results in a surface that is correctly oriented but is no longer rotationally slippable around the input cylinder's axis; our similarity preserving reshaping (c) maintains this property. (bottom) Absent sphericity preservation (e), the middle part of the output surface does not preserve the slippable motion around the surface normal. Our result (f) maintains this property.

$$\Gamma(\kappa_1, \kappa_2) = e^{-\left(\frac{f_{iso}(\kappa_1,\kappa_2)^2}{2\sigma_1^2}\right)} e^{-\left(\frac{f_{sph}(\kappa_1,\kappa_2)^2}{2\sigma_2^2}\right)}$$

Normalizing the input surface (and the corresponding curvature values) to the unit bounding box, we set $k_{max} = 2$, $\sigma_1 = 0.2$, $\sigma_2 = 0.3$ and $\epsilon = 10^{-3}$.

Our slippage functional then becomes

$$E_{slippage} = E_{similarity} + E_{sphericity} \tag{31}$$

Scale. We express our expectation of scale preservation while reinforcing orientation preservation by asking that the derivative of the mapping τ at each point $p \in S_i$ be as close as possible to the identity map I:

$$E_{scale} = \int_{S_i} \|\nabla\tau - I\|_F^2 \, dA \tag{32}$$

Our combined slippage preserving reshaping energy then becomes:

$$\begin{aligned} E^{3D}_{reshape} =& w_{normal}E^{3D}_{normal} + E_{slippage} + w_{scale}E_{scale} \\ &+ w_{constr}(E_{pos} + E_{straight}). \end{aligned} \tag{33}$$

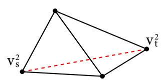
Our slippage formulation implicitly relies on normal preservation, leading us to prioritize it as a prerequisite by setting $w_{normal} = 10$. We set $w_{scale} = 10^{-4}$ to deprioritize scale compared to the other two properties. These weights are internal to the method and fixed across all inputs shown.

Discretization. We discretize our new terms, expressing them in terms of the input and output mesh vertex positions, as follows.

Sphericity. We replace the point-wise principal curvature values $\kappa_1(u,v)$ and $\kappa_2(u,v)$ in the continuous formulation with per-triangle ones, $\kappa_1(t)$ and $\kappa_2(t)$, computed using an off-the-shelf framework [Jacobson et al. 2018], and define the sphericity energy as:

$$E^D_{sphericity} = \sum_{t\in T} \Gamma(\kappa_1(t), \kappa_2(t)) \sum_{i=0}^{2} \left\| \frac{(v_t^{i+2} - v_t^i)}{\|\bar{v}_t^{i+2} - \bar{v}_t^i\|} - R_t^i \frac{(v_t^{i+1} - v_t^i)}{\|\bar{v}_t^{i+1} - \bar{v}_t^i\|} \right\|^2$$

Similarity. To encode similarity, we consider a pair of adjacent triangles t and s which, without loss of generality, share the vertices $\bar{v}_t^0 = \bar{v}_s^1$ and $\bar{v}_t^1 = \bar{v}_s^0$. Similarity is best preserved when the two triangles undergo similar non-uniform scaling in their respective tangent planes. By definition the scale is the same along their common edges, but absent any constraints triangles can scale differently orthogonally to the common edge. Directly operating on vectors orthogonal to the edges leads to a complex, hard to optimize formulation. Instead, we approximate the impact of the per-triangle transformations τ_t and τ_s on the orthogonal stretch of the two triangles by considering their impact on the vector connecting the non-shared vertices v_t^2 and v_s^2 of the two triangles (dashed red line in the inset). If the transformed vectors $\tau_t(\bar{v}_t^2 - \bar{v}_s^2)$ and $\tau_s(\bar{v}_t^2 - \bar{v}_s^2)$ do not coincide, that suggests that one triangle is transformed differently than the other along the orthogonal direction, contrary to our goal of having both triangles exhibit similar tangent plane transformations along the orthogonal direction. Based on this observation, we define our similarity term as:



$$E^D_{similarity} = \sum_{t\in T, s\in N(t)} \omega_{feature}(t,s) \left\| \tau_t \frac{\bar{v}_t^2 - \bar{v}_s^2}{\|\bar{v}_t^2 - \bar{v}_s^2\|} - \tau_s \frac{\bar{v}_t^2 - \bar{v}_s^2}{\|\bar{v}_t^2 - \bar{v}_s^2\|} \right\|^2$$

where $N(t)$ are the triangles sharing edges with t. We define $\omega_{feature}$ to be 0 if the edge $< \bar{v}_t^0, \bar{v}_t^1 >$ shared by the two triangles is a feature edge, and define it as 1 otherwise. This allows surfaces connected via sharp features to scale independently. Feature edges can be manually annotated or automatically identified using a simple normal difference threshold.

Scale. We encode scale preservation as a desire to maintain the original lengths and directions of all mesh edges:

$$E^D_{scale} = \frac{1}{L_{avg}} \sum_{t\in T} \sum_{i=0}^{2} \left\| (v_t^{i+1} - v_t^i) - (\bar{v}_t^{i+1} - \bar{v}_t^i) \right\|^2 . \tag{34}$$

4.2 Solving for Slippage-Preserving Reshaping

We modify the solver in Sec 3.3 to support the new formulation as follows.

Vertex Solve. Our scale E^D_{scale} and sphericity $E^D_{sphericity}$ terms are both quadratic in the vertex positions and can be therefore used as-is during the vertex solve. Plugging these terms into the overall reshaping energy, expressed in terms of output vertex positions, yields:

$$\begin{aligned} E^L_{reshape} =& E^L_{transform} + w_{constr}(E_{pos} + E^L_{straight}) + w_{normal}E^L_{normal} \\ &+ E^D_{sphericity} + w_{scale}E^D_{scale} \end{aligned}$$

We do not include the $E_{similarity}$ term as it does not depend on output vertex positions. While it may be advantageous to formulate similarity in terms of positions and include it in the optimized energy to promote better alignment between the vertex and transformation solves, we expect such a formulation to be highly non-linear and thus harder to optimize, and so we avoid it. We minimize this energy, and obtain updated output vertex positions using the same least squares solver [Guennebaud et al. 2010] as for 3D ALUP.

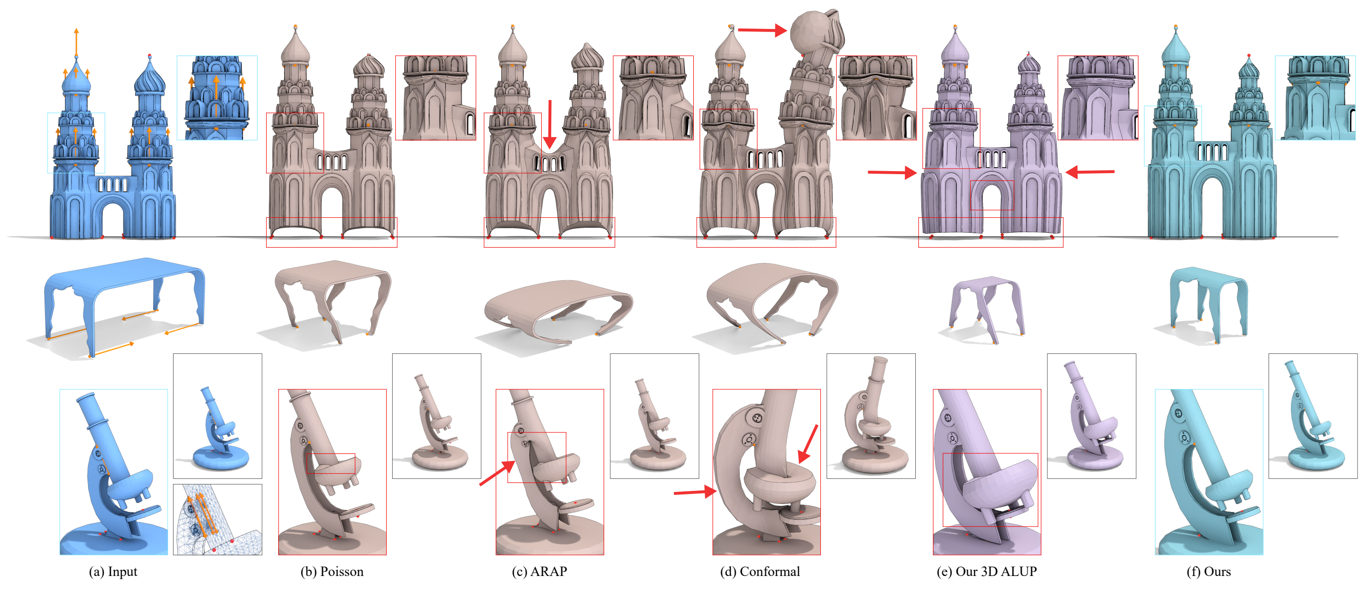


Fig. 7. Additional comparison of slippage-preserving reshaping (f) against (b) Poisson deformation, (c) ARAP deformation [Chao et al. 2010], (d) conformal deformation [Vaxman et al. 2015], and our 3D ALUP (e). While all alternative outputs exhibit visible artifacts, ours (f) are well aligned with viewer expectations. The "table" model is provided courtesy of Lun et al. [2015].

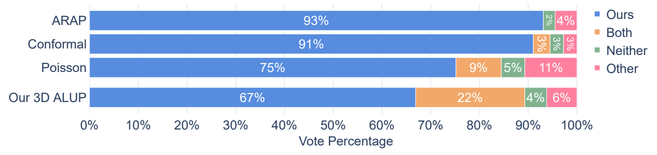


Fig. 8. Study summary. Participants preferred our slippage-preserving method's results over both pre-existing alternatives and our 3D ALUP extension by a large margin.

Transformation Solve. We formulate sphericity in terms of transformations as:

$$E^{TL}_{sphericity} = \sum_{t \in T} \Gamma(\kappa_1(t), \kappa_2(t)) \sum_{i=0}^{2} \left\| \frac{\tau_t(\bar{v}_t^{i+2} - \bar{v}_t^i)}{\|\bar{v}_t^{i+2} - \bar{v}_t^i\|} - R_t^i \frac{\tau_t(\bar{v}_t^{i+1} - \bar{v}_t^i)}{\|\bar{v}_t^{i+1} - \bar{v}_t^i\|} \right\|^2$$

We note that our regularizer term $E^{TL}_{regularizer}$ (Eq. 28) already promotes scale preservation, and thus no other term needs to be added. Similarity is already defined in terms of transformations.

With these elements in place, our quadratic approximate reshaping energy expressed in terms of transformations is:

$$\begin{aligned} E^{TL}_{reshape} = & E^{TL}_{transform} + E^{TL}_{connect} + w_{normal} E^{TL}_{normal} \\ & + E^{TL}_{sphericity} + E^{D}_{similarity} + w_{scale} E^{TL}_{regularizer} \end{aligned} \tag{35}$$

We minimize this energy using the same solver as above.

5 RESULTS AND VALIDATION

We evaluate our slippage-aware 3D reshaping method on 128 gestures across 59 input objects. Our input choices were motivated by interviews with artists and designers, and span the type of edits they expect to need. 37 reshaping edits are demonstrated in the paper, and the rest are shown in the supplementary material. Our examples include diverse household items, architectural structures, abstract shapes, furniture, and other human-made content. The edits demonstrated include both gestures that can be supported with subtle or minimal orientation changes (e.g. Fig. 1, 3), as well as ones where some rotation is unavoidable and thus expected (e.g. bicycle Fig. 10). Our framework supports reshaping of both closed and open (see sphere in Fig 10,top-right) surfaces, as well as non-manifold (Fig 18) and multi-component (Fig 19) geometries. In all cases our results are consistent with viewer expectations. Please see the supplementary materials for additional examples and details. Our code and data is available at (to be included in the non-anonymized version).

Comparison to Prior Art. We compare our slippage-aware reshaping outputs against those of representative state-of-the-art 3D surface deformation methods. We compare to representative surface and volumetric As-Rigid-As-Possible (ARAP) methods [Chao et al. 2010] implemented in [Jacobson et al. 2018]. Notably, both methods produce nearly identical results on our data (Fig 3, 13). To apply the volumetric method we tetrahedralize the inputs using [Si 2015]. We compare against the conformal deformation method of [Vaxman et al. 2015]. Both their and our methods share the goal of preserving conformality, albeit in our case only on spherical surfaces. However, in contrast to their approach we seek to also minimize changes in surface orientation and to preserve slippage rather than conformality on anisotropic surfaces. These differences lead to drastically different reshaping outputs (e.g. Fig 11, 1). We compare our approach to baseline Poisson deformation [Cohen-Or et al. 2015] which aims to keep the output mesh edges as close as possible to the input ones (see Appendix for exact formulation). On simpler anisotropic inputs, this approach often preserves normals and slippage, providing an important baseline for our needs. Last, but not least we compare our

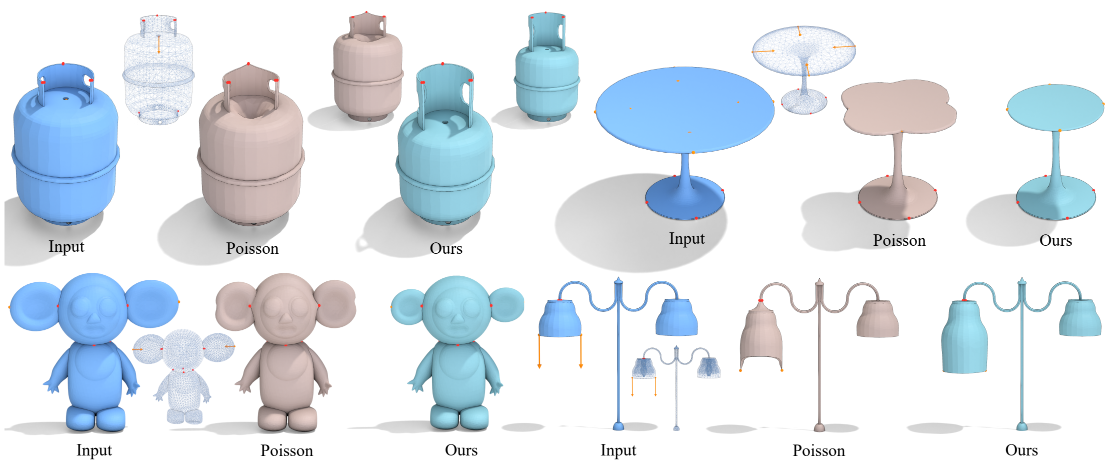


Fig. 9. Additional comparisons against Poisson deformation. The "table" and "lamp" models are provided courtesy of Lun et al. [2015].

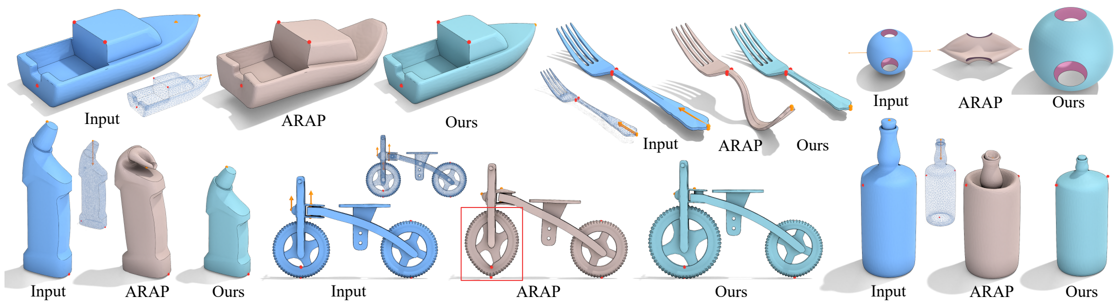


Fig. 10. Additional comparisons against ARAP [Chao et al. 2010] deformation. The "row boat" and "cleaning bottle" models are provided courtesy of Gori et al. [2017], and "fork" of Lun et al. [2015]. Bottle model © 3Diamante - turbosquid.com.

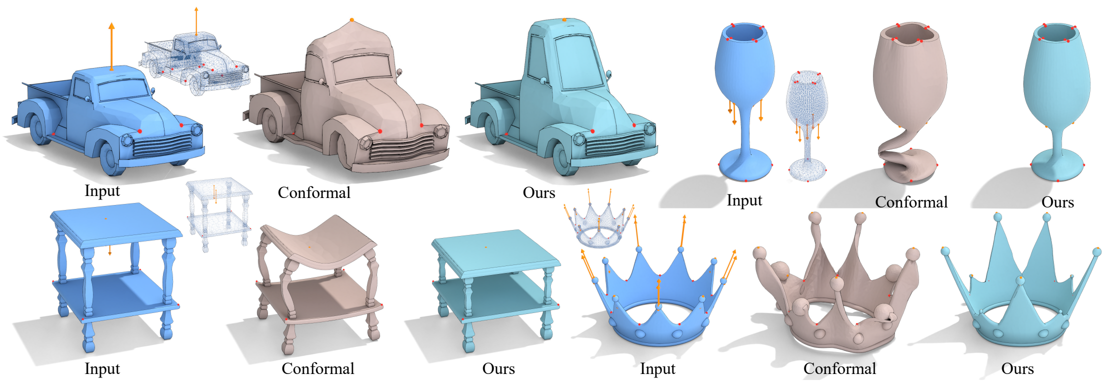


Fig. 11. Additional comparisons against conformal deformation [Vaxman et al. 2015]. The "wineglass" model is provided courtesy of Gori et al. [2017], and "table" of Lun et al. [2015]. Truck model © Polygon.by - sketchfab.com. Crown model © Javidan - turbosquid.com.

slippage-preserving method to our 3D extension of the 2D ALUP formulation of [Araújo et al. 2022] (e.g. Figs 1, 2, 12). Additional comparisons are in the supplementary.

We validate our improvement over these key alternatives via a perceptual comparative study. Study participants were shown input models together with reshaping gestures, our corresponding reshaping outputs, and reshaping outputs produced by alternative methods. The input was shown on top and marked as ‘A’, and the two reshaping results were placed at the bottom in random order and marked as ‘B’ and ‘C’. To better visualize the gesture, the input was rendered twice: once as a solid and once as a wireframe (see Fig 20). Participants were then asked: "The images on the top

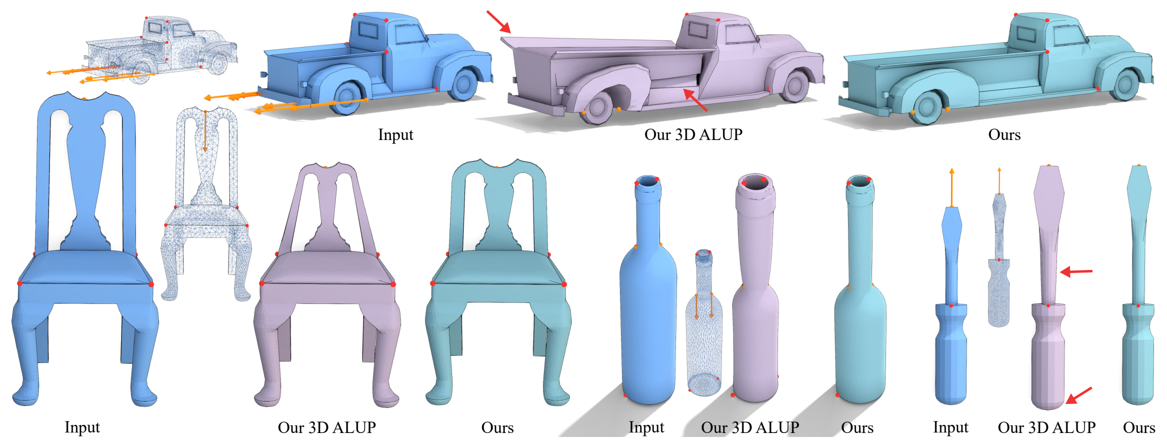


Fig. 12. Additional comparisons against our 3D ALUP extension. Truck model © Polygon.by - sketchfab.com. Chair model © PrettySmallThings - Thingi10K. Wineglass model © EnragedOstrich - turbosquid.com. Screwdriver model © Moox - turbosquid.com.

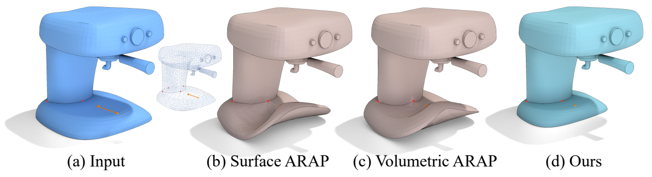


Fig. 13. Given the input in (a), both surface-based (b) and volumetric (c) ARAP deformation methods [Chao et al. 2010] produce outputs with undesirable changes in surface orientation. Our slippage-aware method (d) produces results consistent with user expectations. The "espresso" model is provided courtesy of Gori et al. [2017].

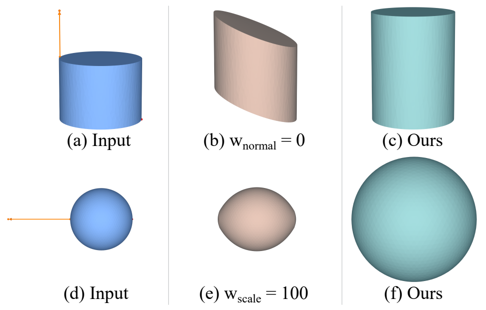


Fig. 14. Reshaping ablations: (a-c) Normal preservation. When normals are not preserved ($w_{normal} = 0$) the output exhibits undesirable rotation (b). (d-e) Allowing the scale term to dominate by increasing w_{scale} by a factor of 1000 (e) produces outputs more similar to those of Poisson deformation, and does not preserve normals or slippage. Our method produces the viewer expected results on these inputs (c,f).

(A) include an object and a suggested resizing/rescaling/proportion change edit indicated via handle displacement gestures. The orange points are moved as suggested by the arrows, while the red points are held in place. The wireframe rendering is provided to better visualize partially occluded displacements when present. Mentally perform the suggested resizing/rescaling/proportion change. Which of the objects on the bottom (B) or (C) comes closer to the edit you envisioned?" The answer options were "B", "C", "Both", and "Neither". To avoid expertise bias, our study was conducted with a diverse set of participants from various backgrounds, ages, and genders. Participants had no prior knowledge of our research or tool, and were only provided with the information on the study questionnaires. Additional protocol details are listed in the Appendix. All study data is provided in the supplemental material.

We included 29 questions comparing our results against each of basic Poisson deformation [Cohen-Or et al. 2015], surface ARAP deformation [Chao et al. 2010], conformal deformation [Vaxman et al. 2015], and our 3D ALUP baseline. We also include 4 questions comparing our results to manually created ones as discussed below (120 questions total). Questions were randomly distributed into 4 questionnaires, so that each includes 30 comparisons. Study inputs were selected at random, with the constraint that a questionnaire does not contain the same reshaping gesture twice. The study had a total of 40 valid participants (26 male, 14 female). In total we collected 10 answers to each question.

Fig. 8 summarizes the study results. Participants preferred our slippage-aware reshapings over all alternatives. In comparisons to the closest pre-existing alternative, basic Poisson deformation, participants preferred our results 75% of the time and preferred the alternative 11% of the time, judging them on par the rest of the time. In comparison against our 3D ALUP extension they preferred our outputs 67% of the time, and judged the results as on-par 27% of the time. We conducted t-tests on the study results and found that the results were highly statistically significant (two-tailed t-tests; $p < 0.005$ for comparisons vs. all methods). These results indicate that our reshaping algorithm produces outputs that are more consistent with participant expectations than those produced by existing alternatives. Fig 15 shows the only two gestures out of 116 included in our study where a plurality of participants preferred a result produced by an alternative algorithm over ours. In both cases, the alternative chosen was computed using basic Poisson deformation, and in both the results are visually very close. This suggests that our method does not just outperform the alternatives

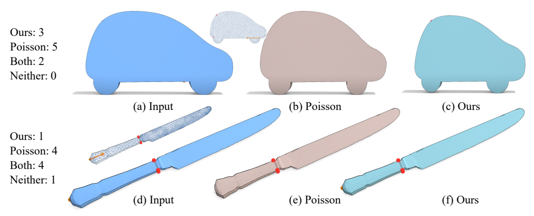


Fig. 15. The only two examples in our study where participants preferred the alternative result over ours. The "knife" model is provided courtesy of Lun et al. [2015].

on average but does so across a wide range of inputs. Following [Yin et al. 2022], we use the frequency of the "neither" option to serve as a proxy for measuring absolute alignment. Participants choose "neither" when neither result aligns with their expectations. We note that the number of neither responses in the study was extremely low (under 5%), suggesting that our outputs are not just better than the alternatives but are indeed well aligned with viewer expectations.

Comparison to Manual Reshaping. We asked a professional computer artist, with over ten years of experience, to reshape four 3D models (armchair in Fig 1, chair and lighthouse in Fig 2, and phone in Fig. 3) based on gestures provided. It took the artist 70 minutes to reshape the modern chair, 75min to reshape the phone, 180 minutes to reshape the lighthouse, and 330min (five and a half hours) to reshape the antique chair. Our results are visually very similar to the artist generated ones. They were generated using just a few mouse clicks and took on average under a minute to compute, with the antique chair taking the longest (3 minutes).

The artist had no beforehand knowledge of our tool. After he completed the manual reshapings, we showed him our results and tool. He judged our results on the armchair, and phone to be better than his, and judged the others as of equal quality. For the phone he commented that "the result of the algorithm is way better than mine." For the armchair, he stated "With all the pain in my heart, I have to admit that yours [algorithm] is better!" For the chair in Fig 2, bottom, he said that "In regards to the hole, I believe my result is better. But, in regards to both the legs and the curvature of the chair, yours [algorithm] is better". For the lighthouse he commented "It is [artist and algorithm outputs] very similar!"

We asked the artist about our methods' usefulness. We quote their responses verbatim: "I think this will be very very useful, especially for attacking new ideas and conceptualizing."; "With this kind of tools, we would spend more time on the creative side and even making more use of the assets"; and "With this tool, we know that we can bring this type of assets back to life because we can do lots of changes without working everything from the ground up." This feedback confirms that our results are well aligned with artist expectations, and that our tool has immediate real-life applications.

To further evaluate our results against artist generated ones, we incorporated queries comparing our and artist results into the comparative perceptual study above. We included a query for each artist generated output and our corresponding result, and collected 10 answers for each query. Participants preferred our reshaping outputs over the artist efforts 55% of the time, judged them as on par 27% of the time, and preferred the artist result only 18% of the time. We confirm the statistical significance of our results by running a paired two-tailed t-test, which found that our results were statistically significant ($p < 0.005$). We therefore conclude that our method produces outputs that are at least on par with those created by a professional artist, and in this case even visibly better. All study data is provided in the additional supplemental material.

Input Meshes. Our core method is designed to operate on sufficiently fine connected triangle meshes, with well-shaped triangles and roughly uniform mesh density. We remesh inputs that do not conform to this assumption using an off-the-shelf remeshing tool [Hu et al. 2018]. Our input mesh sizes range from 1K to 82K triangles with a median of 8.5K triangles. We extend the method to general meshes as discussed in Sec. 5.1.

Compute and User Time. Our median runtime is 27 seconds, and depends on both the mesh resolution and the number of iterations required to converge to the desired reshaping output. Our most time consuming input is the gate (Fig 7) which has 84K triangles and takes 25 minutes to reshape; 90% of our inputs converge in 3 minutes or less. Performance was measured on an Intel I7-7700 2.80GHz, with 32GB RAM, running Windows 10.

Using our UI it typically takes a user 5 to 10 minutes to specify the control handles and their desired displacements. Our combined user and compute times are negligable compared to typical manual editing times. To confirm that our runtimes are acceptable, we interviewed the artist who performed the manual reshaping and asked their opinion. When told about the user and compute times for the armchair (Fig. 1, 3 min runtime) his response was "You are kidding me! It is super fast; 3 minutes against what it took me [5.5 hours]?!"

Ablations. The weights in our reshaping energy differ by orders of magnitude and are chosen to prioritize constraint satisfaction first, then normal preservation, then slippage, and then finally to deprioritize scale relative to all other properties. We use the same weights for all results generated in the paper; control over the weights of the different terms is not exposed to the user. We evaluate the impact of disabling the different energy terms we use in Figs 6 and 14. Notably, disabling scale preservation leads the the energy $E^{TL}_{reshape}$ minimized during the transformation solve to become ill-posed, a clearly undesirable outcome. In our experiments, changing the weights for normal, slippage, and scale preservation made negligible differences to the results as long as the relative weight order was maintained. Decreasing the weight for constraints predictably reduces the degree to which they are satisfied. Extreme weight changes, such as weighing scale preservation an order of magnitude above normal and slippage preservation (Fig 14e), predictably degrades the visual quality of the outputs.

Limitations. Humans often implicitly expect reshaping outputs to satisfy semantic constraints based on functional or aesthetic properties of the input. Our framework is purely geometric, and is not designed to account for such constraints. Despite this, we demonstrate our outputs to be well aligned with human expectations. Our

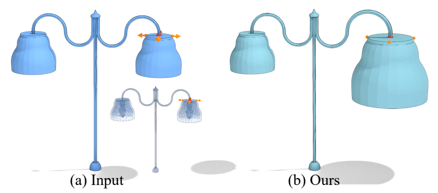


Fig. 16. Our method is not designed to explicitly preserve symmetries in the input shape: (a) input mesh containing structural symmetry and edit operation modifying a single part, (b) reshaped output where the modified part corresponds to the input gesture - the input symmetry is not preserved. The "lamp" model is provided courtesy of Lun et al. [2015].

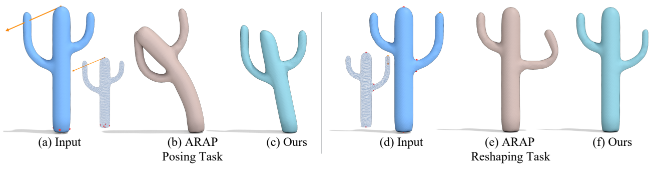


Fig. 17. Reshaping vs. Posing. Existing deformation methods are suitable for edit operations where surface orientation is required to change (a-c). Our method specifically targets reshaping (d), an operation unsuitable for existing deformation methods (e-f). The "cactus" model is provided courtesy of Sorkine and Alexa [2007].

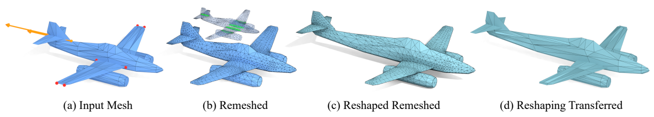


Fig. 18. Connectivity preserving reshaping: (a) input mesh and gesture, (b) remeshed input (non-manifold edges highlighted in green in the inset), (c) reshaped remeshed model, (d) reshaping transferred to original mesh. The "airplane" model is provided by ShapeNet [Chang et al. 2015].

method does not detect or explicitly enforce high order regularities such as symmetries (see e.g. Fig. 16) or planarity. Our outputs typically preserve these properties when consistent with user gestures. Our method can potentially be be extended to explicitly account for these and other regularities.

Our method is specifically designed to address reshaping tasks (e.g. Fig. 17d), where users desire to edit an input shape by changing the proportions, scale, or relative location of different model parts while preserving surface structures. There exist other editing tasks where existing deformation methods (e.g. ARAP [Chao et al. 2010]) are the appropriate choice. For instance, posing-like tasks (e.g. Fig. 17a) require surface orientation to be changed while preserving input details. This violates a key requirement in reshaping, making our method unsuitable for this task (Fig. 17bc). Our slippage-preserving algorithm complements the space of shape editing tools by allowing users to perform reshaping, an operation unsuitable for existing deformation methods (Fig. 17d-f).

5.1 Extensions

Connectivity Preserving Reshaping. There may be cases where even when the input meshes are not well shaped where users wish to retain the input mesh connectivity as-is. We support this preference facilitating connectivity preserving reshaping as illustrated in Fig 18.

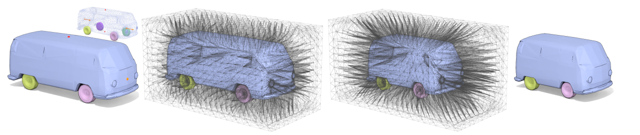


Fig. 19. Reshaping multi-component models: (a) input mesh and gesture, (b) volumetric mesh, (c) reshaped volumetric mesh, (d) reshaping output. The "minivan" model is provided by ShapeNet [Chang et al. 2015].

We first remesh the input using TetWild [Hu et al. 2018], then, for each vertex of the original mesh, we find its location on the remeshed model by casting rays in a uniform sphere centered around each vertex and finding the nearest ray-triangle intersection. This assigns a triangle and barycentric coordinates to each vertex on the input mesh. We then perform reshaping on the remeshed model, and find new positions for each vertex of the original input mesh by barycentrically interpolating the vertices of its corresponding reshaped triangle on the remeshed model. Fig. 18 demonstrates the application of our method in this scenario.

Multi-Component Models. We extend our method to support models with multiple connected components by applying a 3D variant of the scaffolding method presented in [Araújo et al. 2022]. We create a constrained Delaunay tetrahedralization scaffold connecting all mesh components, in which the input surface and vertices of an axis-aligned bounding box are included as additional constraints, and where the creation of Steiner points on internal surfaces is disallowed. We then deform this scaffold by augmenting our vertex-solve energy with an extra edge energy term $E_{ij}^{edge} = \| \frac{v_i - v_j}{l_{ij}^0} - \frac{v_i^0 - v_j^0}{l_{ij}^0} \|^2$ for edges connecting different components, with a weak weight of $w_{edge} = 10^{-2}$. See Fig. 19 for an example.

6 CONCLUSIONS

We presented a novel method for reshaping human-made 3D content, which produces outputs significantly better aligned with human expectation than state of the art alternatives. We achieved this goal by identifying the properties of surfaces that viewers expect to be preserved under a typical reshaping edit, formulating those in mathematical terms, and developing an effective and efficient method that produces outputs that satisfy these properties.

Our work offers several avenues for future work. Our outputs can be used as ground truth data for potential learning based reshaping approach that could potentially facilitate real-time reshaping as well as reshaping of implicit or neural models. Another interesting line of inquiry is to explore alternative control and interaction mechanisms for reshaping.

ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for their insightful comments. We are also deeply grateful to Jinfan Yang for her help with input gesture creation. We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC) grant RGPIN-2018-03944 (“Broad-Based Computational Shape Design”).

REFERENCES

2023. SketchFab. https://www.sketchfab.com. Accessed 2023-01-12.

2023. TurboSquid by ShutterStock. https://www.turbosquid.com. Accessed 2023-01-12.

Marc Alexa. 2003. Differential coordinates for local mesh morphing and deformation. *The Visual Computer* 19, 2 (2003), 105–114.

Chrystiano Araújo, Nicholas Vining, Enrique Rosales, Giorgio Gori, and Alla Sheffer. 2022. As-Locally-Uniform-As-Possible Reshaping of Vector Clip-Art. *ACM Transaction on Graphics* 41, 4 (2022).

A. Artusi, F. Banterle, T.O. Aydın, D. Panozzo, and O. Sorkine-Hornung. 2016. *Image Content Retargeting: Maintaining Color, Tone, and Spatial Consistency.* CRC Press.

Gilbert Louis Bernstein and Wilmot Li. 2015. Lillicon: Using Transient Widgets to Create Scale Variations of Icons. *ACM Trans. Graph.* 34, 4 (2015).

Martin Bokeloh, Michael Wand, Vladlen Koltun, and Hans-Peter Seidel. 2011. Pattern-Aware Shape Deformation Using Sliding Dockers. *ACM Trans. Graph.* 30, 6 (2011), 1–10.

Francesco Buonamici, Monica Carfagni, Rocco Furferi, Lapo Governi, Alessandro Lapini, and Yary Volpe. 2018. Reverse engineering modeling methods and tools: a survey. *Computer-Aided Design and Applications* 15, 3 (2018), 443–464.

Marcio Cabral, Sylvain Lefebvre, Carsten Dachsbacher, and George Drettakis. 2009. Structure Preserving Reshape for Textured Architectural Scenes. *Computer Graphics Forum (Proceedings of the Eurographics conference)* (2009).

Dan Cascaval, Mira Shalah, Phillip Quinn, Rastislav Bodik, Maneesh Agrawala, and Adriana Schulz. 2022. Differentiable 3D CAD Programs for Bidirectional Editing. *Computer Graphics Forum* 41, 2 (2022), 309–323.

Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. 2015. *ShapeNet: An Information-Rich 3D Model Repository.* Technical Report arXiv:1512.03012 [cs.GR].

Isaac Chao, Ulrich Pinkall, Patrick Sanan, and Peter Schröder. 2010. A Simple Geometric Model for Elastic Deformations. *ACM Trans. Graph.* 29, 4, Article 38 (2010).

Siddhartha Chaudhuri, Evangelos Kalogerakis, Leonidas Guibas, and Vladlen Koltun. 2011. Probabilistic Reasoning for Assembly-Based 3D Modeling. In *ACM SIGGRAPH 2011 Papers.* Article 35, 10 pages.

Daniel Cohen-Or, Chen Greif, Tao Ju, Niloy J. Mitra, Ariel Shamir, Olga Sorkine-Hornung, and Hao (Richard) Zhang. 2015. *A Sampler of Useful Computational Tools for Applied Geometry, Computer Graphics, and Image Processing* (1st ed.). A. K. Peters, Ltd., USA.

Keenan Crane, Ulrich Pinkall, and Peter Schröder. 2011. Spin Transformations of Discrete Surfaces. *ACM Trans. Graph.* 30 (2011). Issue 4.

Manfredo P. do Carmo. 1976. *Differential geometry of curves and surfaces.* Prentice Hall.

Pierre Dragicevic, Stéphane Chatty, David Thevenin, and Jean-Luc Vinot. 2005. Artistic resizing: A technique for rich scale-sensitive vector graphics. *ACM SIGGRAPH 2006*, 201–210.

Thomas Funkhouser, Michael Kazhdan, Philip Shilane, Patrick Min, William Kiefer, Ayellet Tal, Szymon Rusinkiewicz, and David Dobkin. 2004. Modeling by Example. *ACM Trans. Graph.* 23, 3 (aug 2004), 652–663.

James Gain and Dominique Bechmann. 2008. A Survey of Spatial Deformation from a User-Centered Perspective. *ACM Trans. Graph.* 27, 4 (2008).

Ran Gal, Olga Sorkine, and Daniel Cohen-Or. 2006. Feature-Aware Texturing. *Rendering Techniques* 11, 297–303.

Ran Gal, Olga Sorkine, Niloy J. Mitra, and Daniel Cohen-Or. 2009. IWIRES: An Analyze-and-Edit Approach to Shape Manipulation. In *Proc. SIGGRAPH 2009.* ACM.

Natasha Gelfand and Leonidas J Guibas. 2004. Shape segmentation using local slippage analysis. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*. 214–223.

Michael Gleicher. 1992. Briar: A Constraint-Based Drawing Program. In *Proc. SIGCHI 1992.*

Giorgio Gori, Alla Sheffer, Nicholas Vining, Enrique Rosales, Nathan Carr, and Tao Ju. 2017. FlowRep: Descriptive Curve Networks for Free-Form Design Shapes. *ACM Transaction on Graphics* 36, 4 (2017). https://doi.org/10.1145/3072959.3073639

Gaël Guennebaud, Benoît Jacob, et al. 2010. Eigen v3. http://eigen.tuxfamily.org.

Amir Hertz, Or Perel, Raja Giryes, Olga Sorkine-Hornung, and Daniel Cohen-Or. 2022a. Mesh Draping: Parametrization-Free Neural Mesh Transfer. *Computer Graphics Forum* (2022).

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

S. Hsu, Irene H. H. Lee, and N. Wiseman. 1993. Skeletal strokes. In *UIST '93.*

Yixin Hu, Qingnan Zhou, Xifeng Gao, Alec Jacobson, Denis Zorin, and Daniele Panozzo. 2018. Tetrahedral Meshing in the Wild. *ACM Trans. Graph.* 37, 4, Article 60 (2018).

Takeo Igarashi, Tomer Moscovich, and John F. Hughes. 2005. As-Rigid-as-Possible Shape Manipulation. *ACM Trans. Graph.* 24, 3 (2005).

Alec Jacobson, Ilya Baran, Ladislav Kavan, Jovan Popović, and Olga Sorkine. 2012. Fast Automatic Skinning Transformations. *ACM Trans. Graph.* 31, 4 (2012).

Alec Jacobson, Ilya Baran, Jovan Popović, and Olga Sorkine. 2011. Bounded Biharmonic Weights for Real-Time Deformation. In *Proc. SIGGRAPH 2011.* Association for Computing Machinery.

Alec Jacobson, Daniele Panozzo, et al. 2018. libigl: A simple C++ geometry processing library. https://libigl.github.io/.

Chiyu Jiang, Jingwei Huang, Andrea Tagliasacchi, and Leonidas Guibas. 2020. ShapeFlow: Learnable Deformations Among 3D Shapes. In *Neural Information Processing Systems (NeurIPS).*

Pushkar Joshi, Mark Meyer, Tony DeRose, Brian Green, and Tom Sanocki. 2007. Harmonic coordinates for character articulation. *ACM Transactions on Graphics (TOG)* 26, 3 (2007), 71–es.

Tom Kelly, Peter Wonka, and Pascal Müller. 2015. Interactive dimensioning of parametric models. In *Computer Graphics Forum*, Vol. 34. 117–129.

Vladislav Kraevoy, Alla Sheffer, Ariel Shamir, and Daniel Cohen-Or. 2008. Non-Homogeneous Resizing of Complex Models. *ACM Transactions on Graphics (TOG)* 27, 5 (2008), 1–9.

C. Kurz, X. Wu, M. Wand, T. Thormählen, P. Kohli, and H.-P. Seidel. 2014. Symmetry-Aware Template Deformation and Fitting. *Computer Graphics Forum* 33, 6 (2014), 205–219.

Xian-Ying Li and Shi-Min Hu. 2012. Poisson coordinates. *IEEE Transactions on visualization and computer graphics* 19, 2 (2012), 344–352.

Connor Lin, Niloy Mitra, Gordon Wetzstein, Leonidas J Guibas, and Paul Guerrero. 2022. NeuForm: Adaptive Overfitting for Neural Shape Editing. *Advances in Neural Information Processing Systems* 35 (2022), 15217–15229.

Yaron Lipman, David Levin, and Daniel Cohen-Or. 2008. Green coordinates. *ACM Trans. Graph.* 27, 3 (2008), 1–10.

Zhaoliang Lun, Evangelos Kalogerakis, and Alla Sheffer. 2015. Elements of Style: Learning Perceptual Shape Style Similarity. *ACM Trans. Graph.* 34, 4, Article 84 (jul 2015), 14 pages.

Chongyang Ma, Haibin Huang, Alla Sheffer, Evangelos Kalogerakis, and Rui Wang. 2014. Analogy-driven 3D style transfer. In *Computer Graphics Forum*, Vol. 33. 175–184.

Elie Michel and Tamy Boubekeur. 2021. DAG Amendment for Inverse Control of Parametric Shapes. *ACM Transactions on Graphics* 40, 4 (2021), 173:1–173:14.

Patrick Mullen, Yiying Tong, Pierre Alliez, and Mathieu Desbrun. 2008. Spectral conformal parameterization. In *Computer Graphics Forum*, Vol. 27. Wiley Online Library, 1487–1494.

Matthias Nieser, Jonathan Palacios, Konrad Polthier, and Eugene Zhang. 2012. Hexagonal Global Parameterization of Arbitrary Surfaces. *IEEE Transactions on Visualization and Computer Graphics* 18, 6 (2012), 865–878. https://doi.org/10.1109/TVCG.2011.118

Jesús R Nieto and Antonio Susín. 2013. Cage based deformations: a survey. In *Deformation models.* Springer, 75–99.

Daniele Panozzo, Enrico Puppo, and Luigi Rocca. 2010. Efficient multi-scale curvature and crease estimation. *Proceedings of Computer Graphics, Computer Vision and Mathematics (Brno, Czech Rapubic* 1, 6 (2010).

Daniele Panozzo, Ofir Weber, and Olga Sorkine. 2012. Robust image retargeting via axis-aligned deformation. In *Computer Graphics Forum*, Vol. 31. 229–236.

Ulrich Pinkall and Konrad Polthier. 1993. Computing discrete minimal surfaces and their conjugates. *Experimental mathematics* 2, 1 (1993), 15–36.

Adriana Schulz, Jie Xu, Bo Zhu, Changxi Zheng, Eitan Grinspun, and Wojciech Matusik. 2017. Interactive Design Space Exploration and Optimization for CAD Models. *ACM Transactions on Graphics* 36, 4 (2017).

Alla Sheffer, Emil Praun, and Kenneth Rose. 2006. Mesh Parameterization Methods and Their Applications. *Foundations and Trends in Computer Graphics and Vision* 2, 2 (2006), 105–171.

Lin Shi, Yizhou Yu, Nathan Bell, and Wei-Wen Feng. 2006. A Fast Multigrid Algorithm for Mesh Deformation. *ACM Trans. Graph.* 25, 3 (2006), 1108–1117.

H. Si. 2015. TetGen, a Delaunay-based quality tetrahedral mesh generator. *ACM Trans. Math. Software* 41, 2 (2015), 11.

Justin Solomon, Mirela Ben-Chen, Adrian Butscher, and Leonidas Guibas. 2011. As-Killing-As-Possible Vector Fields for Planar Deformation. *Computer Graph. Forum* 30 (2011), 1543–1552.

Olga Sorkine and Marc Alexa. 2007. As-Rigid-As-Possible Surface Modeling. In *Proc. EUROGRAPHICS/ACM SIGGRAPH Symposium on Geometry Processing*. 109–116.

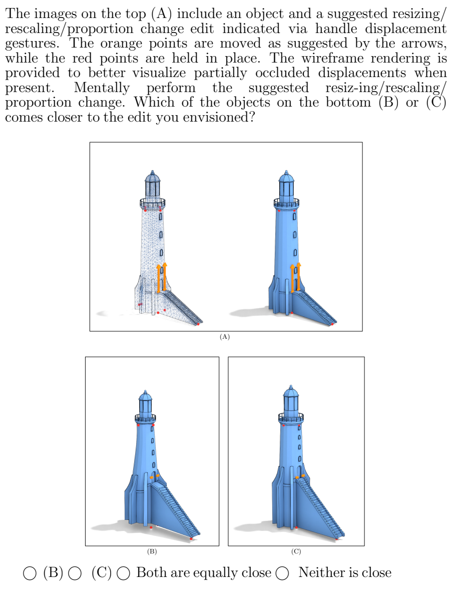
O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössl, and H.-P. Seidel. 2004. Laplacian Surface Editing. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing (SGP '04)*. 175–184.

Robert W. Sumner and Jovan Popović. 2004. Deformation Transfer for Triangle Meshes. *ACM Trans. Graph.* 23, 3 (2004), 399–405.

Robert W. Sumner, Johannes Schmid, and Mark Pauly. 2007. Embedded Deformation for Shape Manipulation. In *ACM SIGGRAPH 2007 Papers.* 80–es.

Minhyuk Sung, Zhenyu Jiang, Panos Achlioptas, Niloy J Mitra, and Leonidas J Guibas. 2020. DeformSyncNet: Deformation transfer via synchronized shape deformation spaces. *arXiv preprint arXiv:2009.01456* (2020).

Ivan E. Sutherland. 1964. Sketchpad: a Man-Machine Graphical Communication System. *Simulation* 2, 5, R–3.



The images on the top (A) include an object and a suggested resizing/rescaling/proportion change edit indicated via handle displacement gestures. The orange points are moved as suggested by the arrows, while the red points are held in place. The wireframe rendering is provided to better visualize partially occluded displacements when present. Mentally perform the suggested resiz-ing/rescaling/proportion change. Which of the objects on the bottom (B) or (C) comes closer to the edit you envisioned?

◯ (B) ◯ (C) ◯ Both are equally close ◯ Neither is close

Fig. 20. Example study question.

Jiapeng Tang, Markhasin Lev, Wang Bi, Thies Justus, and Matthias Nießner. 2022. Neural Shape Deformation Priors. In *Advances in Neural Information Processing Systems*.

Jean-Marc Thiery and Tamy Boubekeur. 2022. Green Coordinates for Triquad Cages in 3D. In *SIGGRAPH Asia 2022 Conference Papers*. Article 38, 8 pages.

Jean-Marc Thiery, Pooran Memari, and Tamy Boubekeur. 2018. Mean value coordinates for quad cages in 3D. *ACM Transactions on Graphics (Proc. SIGGRAPH 2018)* (2018).

Amir Vaxman, Christian Müller, and Ofir Weber. 2015. Conformal Mesh Deformations with Möbius Transformations. *ACM Trans. Graph.* 34, 4 (2015), 55:1–55:11.

Yu-Shuen Wang, Chiew-Lan Tai, Olga Sorkine, and Tong-Yee Lee. 2008. Optimized Scale-and-Stretch for Image Resizing. *ACM Trans. Graph.* (2008).

Ofir Weber and Craig Gotsman. 2010. Controllable Conformal Maps for Shape Deformation and Interpolation. *ACM Trans. Graph.* 29, 4, Article 78 (2010).

Lior Wolf, Moshe Guttmann, and Daniel Cohen-Or. 2007. Non-homogeneous content-driven video-retargeting. In *Proc. IEEE 11th International Conference on Computer Vision*. IEEE, 1–6.

Xiaokun Wu, Michael Wand, Klaus Hildebrandt, Pushmeet Kohli, and Hans-Peter Seidel. 2014. Real-Time Symmetry-Preserving Deformation. *Computer Graphics Forum* 33, 7 (2014), 229–238.

Chunxia Xiao, Liqiang Jin, Yongwei Nie, Renfang Wang, Hanqiu Sun, and Kwan-Liu Ma. 2014. Content-aware model resizing with symmetry-preservation. *The Visual Computer* 31 (2014), 155–167.

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.

Jerry Yin, Chenxi Liu, Rebecca Lin, Nicholas Vining, Helge Rhodin, and Alla Sheffer. 2022. Detecting Viewer-Perceived Intended Vector Sketch Connectivity. *ACM Transactions on Graphics* 41 (2022). Issue 4.

Yizhou Yu, Kun Zhou, Dong Xu, Xiaohan Shi, Hujun Bao, Baining Guo, and Heung-Yeung Shum. 2004. Mesh Editing with Poisson-Based Gradient Field Manipulation. *ACM Trans. Graph.* 23, 3 (2004), 644–651.

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. *CoRR* (2021). https://arxiv.org/abs/2103.01694

Guo-Xin Zhang, Ming-Ming Cheng, Shi-Min Hu, and Ralph R. Martin. 2009. A Shape-Preserving Approach to Image Resizing. *Computer Graphics Forum* (2009).

Juyong Zhang, Bailin Deng, Zishun Liu, Giuseppe Patanè, Sofien Bouaziz, Kai Hormann, and Ligang Liu. 2014. Local Barycentric Coordinates. *ACM Trans. Graph.* 33, 6 (2014).

Kun Zhou, Jin Huang, John Snyder, Xinguo Liu, Hujun Bao, Baining Guo, and Heung-Yeung Shum. 2005. Large Mesh Deformation Using the Volumetric Graph Laplacian. *ACM Trans. Graph.* 24, 3 (2005), 496–503.

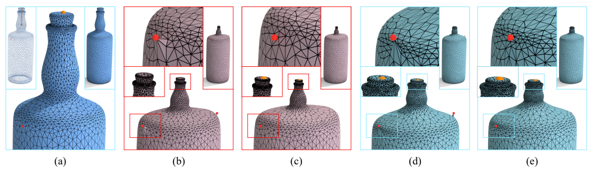


Fig. 21. Finalizing the output: (a) input (b) ALUP solver output (c) finalized output with normal error redistributed away from control handles and constraints well satisfied (note the improved satisfaction at the top orange handle). (d-e) finalization applied to our slippage-aware method (notably constraints are better satisfied from the start).

A ADDITIONAL IMPLEMENTATION DETAILS

Initialization and Termination. We initialize the vertex positions $\bar{v} \in \bar{V}$ using the locations of the input vertices, and initialize the transformations as $\tau_t = I$. Our first iteration solves for vertex positions using the initial transformations. Our method terminates on one of four conditions: if the maximal change in vertex positions between consecutive iterations drops below $L_{avg}/100$; if the improvement in the energy $E_{reshape}$ between consecutive iterations drops below $1/(100|E|)$; if the energy $E_{reshape}$ increases instead of decreases across consecutive iterations (in this case we return the solution with the lower energy); and finally, if the number of alternating least squares iterations exceeds our fixed maximum of 100 iterations.

Finalizing Outputs. Once the optimization converges, we apply an optional step whose goals are to improve normal error distribution and degree of constraint satisfaction (Fig. 21). We note that when the constraints cannot be satisfied without significant increase in other components of the minimized energy, our solution method tends to concentrate error immediately next to the handles (Fig. 21b). Similar behavior had been observed by authors of other surface deformation methods with handle-based positional constraints [Sorkine et al. 2004]. We automatically detect instances where this behavior happens and re-distribute the error using a process inspired by [Sorkine et al. 2004].

We consider the region around a handle as exhibiting error concentration if the normal error in one of the triangles next to the handle is above $10°$ and is at least twice higher than two triangle rings away. To resolve a discontinuity, for all edges outside a $K = 4$ ring circle, we use the solution edges and normals as the target edges and normals. For edges within the K-rings, we compute the target normals as the weighted average of the original and the current ones (assigning a weight of one to original normals for triangles immediately next to handles, and smoothly averaging for further away rings, so that the weight is zero K rings away). We define all vertices K rings or more away as additional handles and set $w_{constr} = 1000$. We set $\tau_t = I$ for all triangles and compute a new minimizer of $E^L_{reshape}$. We use this minimizer as the final output. Discontinuities only occur when there is no solution that satisfies orientation preservation well, which is rare. Overall handle discontinuities were present in less than 10% of gesture outputs in our data. Since we use soft rather than hard positional constraints our initial solution may not precisely satisfy them when doing so requires significantly increasing other components of the reshaping energy. To achieve better constraint satisfaction, if the deviation is above a threshold (0.5% of the bounding box diagonal in all our examples) we repeat the optimization step using the output of the initial solution as the input and using the same handle positions.

Gestures. In the examples in the paper we use handle vertex displacements to communicate the desired edits - users first mark the handle vertices, and then specify the displacements for vertices they want to move. Our interface supports specifying the same displacement for multiple vertices, allowing gesture replication for symmetric features or other related vertex groups. Similar to other editing setups users can select entire groups of vertices as handles, e.g. by selecting all vertices on a shared feature curve section, or by using a lasso tool.

Solver Implementation. We use the simplicial LDL^T solver from the Eigen linear algebra library to solve the least squares problems in our alternating solver. As the mesh topology does not change between iterations, we perform symbolic factorization once, and then re-use it within each solution iteration.

B COMPARATIVE STUDIES

Protocol Details. Participants were provided a task description and shown two simple reshaping examples; no other explanation was provided. Similar to other perception studies (e.g. [Araújo et al. 2022]) we used a screening question to discard all answers from participants who did not read the task description. The screening question used the second example in the task description. Three participants failed the screening question; additional participants were invited till the target number of 40 valid participants was reached. The task description is in the supplementary material. Figure 20 shows the layout of our study questions.

Poisson Deformation. Our implementation of the baseline Poisson deformation approach follows [Cohen-Or et al. 2015]: we deform the input mesh by finding new vertex positions that minimize $1/L_{avg} \sum_{t \in T} \sum_{i=0}^{2} \left\| v_t^{i+1} - v_t^i \right\|^2$ with appropriate handle constraints held in place with weight w_{constr}.