As-Locally-Uniform-As-Possible Reshaping of Vector Clip-Art

CHRYSTIANO ARAÚJO, University of British Columbia, Canada
NICHOLAS VINING, University of British Columbia, Canada and NVIDIA, Canada
ENRIQUE ROSALES, University of British Columbia, Canada
GIORGIO GORI, Adobe Research, USA
ALLA SHEFFER, University of British Columbia, Canada

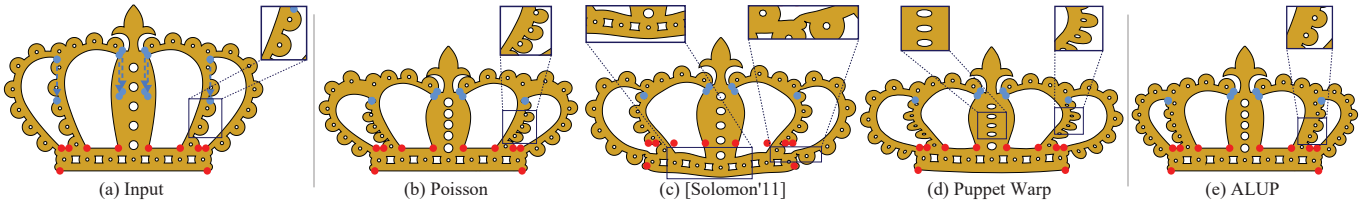


Fig. 1. *Reshaping* vector clip-art: (a) input image with proposed new locations of control handles (stationary handles in red, relocated in blue, arrows correspond original to new locations); (b-d) Using existing planar deformation methods to satisfy the new handle locations (baseline Poisson deformation (b); [Solomon et al. 2011] (c); Puppet Warp [Adobe Inc. 2019; Jacobson et al. 2012; Liu et al. 2014] (d)) destroys input structures and produces unintuitive results. Our *As-Locally-Uniform-As-Possible (ALUP)* reshaping method (e) maximally preserves curve orientations and scales geometric details as uniformly as possible, producing outputs that are consistent with user expectations. Please zoom in to see image details throughout the paper. Input image © dervish15 - stock.adobe.com.

Vector clip-art images consist of regions bounded by a network of vector curves. Users often wish to *reshape*, or rescale, existing clip-art images by changing the locations, proportions, or scales of different image elements. When reshaping images depicting synthetic content they seek to preserve global and local structures. These structures are best preserved when the gradient of the mapping between the original and the reshaped curve networks is locally as close as possible to a uniform scale; mappings that satisfy this property maximally preserve the input curve orientations and minimally change the shape of the input's geometric details, while allowing changes in the relative scales of the different features. The expectation of approximate scale uniformity is *local*; while reshaping operations are typically expected to change the *relative proportions* of a subset of network regions, users expect the change to be minimal away from the directly impacted regions and expect such changes to be gradual and distributed as evenly as possible. Unfortunately, existing methods for editing 2D curve networks do not satisfy these criteria. We propose a targeted *As-Locally-Uniform-as-Possible (ALUP)* vector clip-art reshaping method that satisfies the properties above. We formulate the computation of the desired output network as the solution of a constrained variational optimization problem. We effectively compute the desired solution by casting this continuous problem as a minimization of a non-linear discrete energy function, and obtain the desired minimizer by using a custom iterative solver. We validate our method via perceptual studies comparing our results to those created via algorithmic alternatives and manually generated ones. Participants preferred our results over the closest alternative by a ratio of 6 to 1.

Authors' addresses: Chrystiano Araújo, University of British Columbia, Canada, araujoc@cs.ubc.ca; Nicholas Vining, University of British Columbia, Canada , NVIDIA, Canada, nvining@cs.ubc.ca; Enrique Rosales, University of British Columbia, Canada, albertr@cs.ubc.ca; Giorgio Gori, Adobe Research, USA, gori@adobe.com; Alla Sheffer, University of British Columbia, Canada, sheffa@cs.ubc.ca.

CCS Concepts: • **Computing methodologies** → **Image manipulation**; **Shape modeling**.

Additional Key Words and Phrases: Clip-art, Shape Editing

ACM Reference Format:
Chrystiano Araújo, Nicholas Vining, Enrique Rosales, Giorgio Gori, and Alla Sheffer. 2022. As-Locally-Uniform-As-Possible Reshaping of Vector Clip-Art. *ACM Trans. Graph.* 41, 4, Article 160 (July 2022), 10 pages. https://doi.org/10.1145/3528223.3530098

1 INTRODUCTION

Vector format clip-art images, consisting of a collection of regions bounded by a network of curves and often depicting synthetic or human-made content, are ubiquitous in digital art (Fig 1a). To modify stock clip-art for specific applications, such as generating a new look or satisfying new visual constraints, users often need to *reshape* existing clip-art visuals by changing the relative scales, sizes, and proportions of different image elements or moving them with respect to one another — for instance making the body of the crown in Fig. 1a shorter while retaining the width and height of its rim. Performing these reshaping tasks manually is time consuming and requires significant expertise; it took an artist 45 minutes to perform the task above in Adobe Illustrator. Reshaping clip-art images algorithmically in accordance with a user-specified compact set of reshaping constraints (Fig 1a, red and blue dots) would lower expertise barriers and save both professional and amateur users significant time and effort. Unfortunately, existing shape deformation methods (Fig 1b-d) are unsuitable for this task, as the results that they generate are poorly aligned with user expectations. We propose a new targeted clip-art reshaping method that produces outputs consistent with user preferences (Fig. 1e).

Reshaping clip-art in a manner consistent with user expectations requires both identifying the geometric properties that users expect

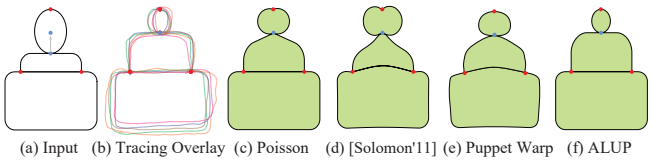


Fig. 2. Reshaping goals: (a) input shape and reshaping constraints; (b) overlay of desired reshaping outputs drawn by five study participants; (c-d) traditional 2D deformation approaches (Poisson deformation (c), [Solomon et al. 2011] (d), [Adobe Inc. 2019; Jacobson et al. 2012] (e)) produce results which significantly differ from those generated by the study participants; ALUP reshaping (f) outputs closely align with the participant drawn ones.

to be preserved under a reshaping operation, and developing a reshaping algorithm that satisfies these properties. Our analysis of prior research (Sec. 2), confirmed by an observational user study (Sec. 3.1, Fig 2), suggests that when users reshape clip-art imagery they seek to produce outputs which satisfy two core criteria. First, while any reshaping operation is expected to change the shape of the region boundaries in the clip-art image, users expect the *orientations* of these boundary curves to be preserved as much as possible. Second, while a reshaping operation is expected to change the *relative proportions* of a subset of elements in the image (e.g, in Fig 2, changing the size of the ellipse on the top and making the middle part taller), users expect the effect of the change to be minimal away from the directly impacted regions and expect any change in proportions to be gradual and distributed as evenly as possible. In other words, users seek the reshaping operation to be *As-Locally-Uniform-As-Possible (ALUP)*, namely that the local gradient of the mapping from the original to the reshaped boundary curves is as close as possible to a uniform scale. These expectations — specifically the requirement that the orientations of region boundaries are maximally preserved — drastically differ from the criteria used in typical deformation settings (Sec. 2), where the optimal solutions are expected to be as-rigid-as-possible, preserving local geometry while allowing and even encouraging rotation (Figs. 1b-d, 2b-d).

Our *ALUP reshaping* method takes as input a vector clip-art image, represented as a network of curves that are discretized into polylines; a set of control handle points on these curves; and the new target positions of these points. It then computes the corresponding reshaped output network such that the mapping between the input and output networks is as locally uniform as possible and maps the input control handles to their new target locations (Sec. 3). We formulate the computation of the desired mapping as a variational problem, and seek for maps that minimize the sum of two functionals which encode our twin goals of *normal preservation* and *local scale uniformity*. Starting from this continuous formulation, we cast the problem of finding a functional-minimizing curve in the discrete setting as an energy minimization problem on uniformly sampled curves. We efficiently solve the resulting problem using a custom iterative mechanism that alternates between solving quadratic optimization problems on disjoint sets of variables.

We evaluate our method on 115 inputs and validate it via extensive comparisons to prior art, potential alternatives, and manually reshaped networks. Participants in our comparative study preferred our results over the closest alternative by a ratio of 6 to 1.

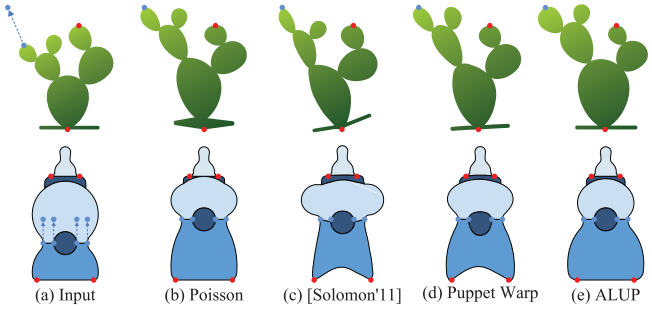


Fig. 3. Reshaping comparison: (a) input shape and reshaping constraints; (b-d) traditional 2D deformation approaches (baseline Poisson deformation (b), [Solomon et al. 2011] (c), [Adobe Inc. 2019; Jacobson et al. 2012] (d)) visibly shear the inputs ; ALUP reshaping (e) minimized changes in curve orientation and locally minimizes changes in scale. Feeder image adapted from dstarky - stock.adobe.com.

2 RELATED WORK

Vector Art Editing. Vector graphics images are made of curves and segments that use control points to define their shape [Hoschek and Lasser 1993; Yuksel 2020]. Commercial tools such as Adobe Illustrator [2019] and Inkscape [2003] enable users to create and manipulate vector imagery by creating and manipulating these geometric primitives. Reshaping vector images by directly manipulating primitive control points can be time consuming since most reshaping operations impact multiple primitives: for instance performing the suggested edit in Fig. 1 required an artist to manually edit over two hundred curve primitives. Chugh et al. [2015] cast vector art creation and editing as functional language programming. Their proposed method requires programming expertise and is not suitable for non-expert users. Dragicevic et al. [2005] and Bernstein et al. [2015] introduce interfaces for global resizing of vector icons that reduce the amount of manual labor; users of their systems still need to formulate the explicit structural constraints they wish to enforce. Our method addresses a complementary problem of supporting general vector art reshaping, and preserves structures that can be maintained implicitly.

Shape Deformation. Shape deformation has been a problem of interest for many years in computer graphics; see [Yuan et al. 2021] for a recent survey. Most shape deformation frameworks in both two and three dimensions aim to produce deformations that are shape preserving, namely that are as close to rigid movement (rotation and translation) as possible, with minimal scaling and shearing [Alexa et al. 2000; Floater 2003; Igarashi et al. 2005; Joshi et al. 2007; Ju et al. 2005; Lipman et al. 2008; Solomon et al. 2011; Sorkine and Alexa 2007; Sorkine et al. 2004]. Our goal is nearly opposite: when applying a reshaping operation users anticipate significant changes in the relative scales of different parts of the input but seek to preserve curve orientations. Conformal shape deformation methods (e.g. [Lipman et al. 2008; Weber and Gotsman 2010]) address the related problem of finding a shape deformation consisting solely of rotation and locally uniform scaling, and penalize shear. While these methods do allow locally uniform scaling, they also explicitly allow for rotation which in our context is highly undesirable.

We compare our reshaping results against three representative methods: a baseline Poisson deformation approach described by [Cohen-Or et al. 2015; Panozzo et al. 2013], the As-Killing-as-Possible vector field method [Solomon et al. 2011], and Adobe Illustrator's Puppet Warp tool (which implements the Bézier spline skinning method of Liu et al. [2014] using the bounded biharmonic weights of Jacobson et al. [2011]) in Figs 1, 2, 7 and Sec. 4; the two latter methods are specifically designed for 2D inputs. The baseline Poisson deformation we compare against limits the deformation degrees of freedom, minimizing the amount of rotation along the input curves and providing a highly relevant baseline for our method; see App. B for details. Our comparisons show that in the context of clip-art reshaping this baseline outperforms the more recent approaches; our outputs are significantly better aligned with user expectations than those produced by all three methods.

Structure-Preserving Deformation. Attempts to preserve global structures during 2D or 3D edits date back to Sketchpad [Sutherland 1964]. Earlier systems [Gleicher 1992; Hsu et al. 1993; Sutherland 1964] required users to assign structural constraints manually. Such systems require significant user expertise and do not allow quick, high-level operations. iWires [Gal et al. 2009] detects and attempts to explicitly preserve geometric constraints and relationships on an input mesh under deformation. As they note, their method can easily become over-constrained as it is not clear which of the observed geometric relationships users may wish to preserve. Cabral et al. [2009] employ a similar constraint-based strategy for structure-preserving reshape of architectural scenes, clustering edges based on length and directional similarity as a proxy for semantic information. Their method is designed to be used in an interactive mode where users can relax or introduce constraints when necessary. By promoting orientation and local scale uniformity, our ALUP reshaping formulation automatically and implicitly distinguishes between structures that can be preserved and those that need to be relaxed (e.g. in Fig. 6 we keep the shape of the circular arc surrounding the light blue region in the windmill, while minimally rotating the normals on the bottom part to satisfy the constraints).

Resizing and Retargeting. Methods for resizing and retargeting of raster images focus on changing the aspect ratio or boundary shape of the input images [Artusi et al. 2016]. Reshaping encompasses a much larger set of editing tasks, necessitating a finer level of control; thus, as discussed below, modifying existing retargeting methods to enable reshaping of vector data is not practical.

Both traditional [Avidan and Shamir 2007; Lefebvre et al. 2010; Setlur et al. 2007; Simakov et al. 2008] and learning based image resizing [Cho et al. 2017; Nam et al. 2019] methods leverage the regular structure of raster imagery. It is not clear how to extend either of these approaches to resizing vector curve networks, which have arbitrary topology and geometry, and where topology must be preserved under the reshaping operations. Li et al. [Li et al. 2020] introduce a differentiable rasterizer for vector graphics; this work has many potential applications, including interactive editing by back-propagating image space edits of a rendered raster image to the original vector art domain. However, currently, no raster space reshaping methods exist.

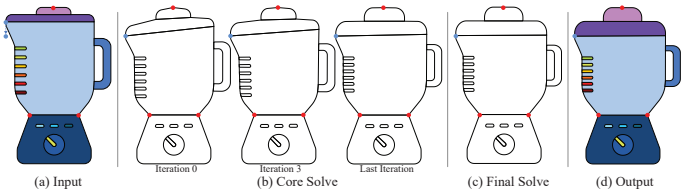
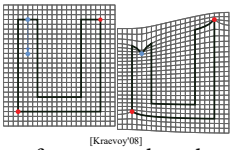


Fig. 4. ALUP reshaping overview: (a) input curve network with control handles highlighted (stationary in red, relocated in blue, dashed lines show correspondence between before and after locations); (b) core solution iterations and output; (c) final solve output, (d) output clip art image. Input image adapted from bsd studio - stock.adobe.com.

Grid-based image retargeting methods [Gal et al. 2006; Wang et al. 2008; Wolf et al. 2007; Zhang et al. 2009] overlay a grid over the input image, and compute new locations for all grid vertices subject to constraints along the grid boundaries. They employ shape preserving deformation formulations, which as discussed above are ill-suited for our needs.



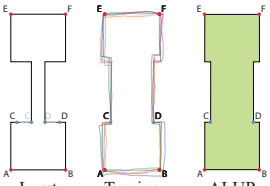
[Kraevoy et al. 2008] resize 3D models by embedding them in a voxel grid, using a method that can be applied in 2D space (see inset). They observe that when resizing human-made shapes users expect surface normals to be preserved as much as possible. They use a variant of Poisson deformation and penalize non-uniform scaling of grid cells containing surfaces whose normals are not aligned with the major axes. [Xiao et al. 2014] resize 3D content using a tetrahedral overlay grid and enforce user specified symmetries. As pointed out by [Panozzo et al. 2012], and illustrated in the inset, all these methods can easily produce grid foldovers; more importantly, as the inset illustrates, the grid structure links the positions of input points which are far apart along the curve network but close in Euclidean space, introducing noticeable artifacts for even simple edits. [Panozzo et al. 2012] prevent foldovers and grid rotation by forcing all grid points that share the same x or y coordinate to continue to do so; this constraint prevents many basic edits where users want to move such points apart (e.g. raising one branch of the cactus above the other (Fig 7). We avoid all these limitations by operating directly on curve networks instead of an overlay grid.

3 METHOD

Our core reshaping method takes as input a connected vector curve network, with a set of user specified control handle locations (points or sections) on these curves, and the desired new locations of these handles (Fig. 4a). It outputs a new reshaped network that satisfies these handle locations (Fig 4d). We first discuss how our reshaping formulation is motivated by prior research and our informal user study, then propose a continuous problem formulation, and finally discretize our continuous formulation and show how it may be solved using an efficient optimization strategy.

3.1 Perception of 2D Reshaping

Our reshaping formulation is motivated by prior research [Kraevoy et al. 2008; Mehra et al. 2009; Panozzo et al. 2012] that highlights the importance that humans place on avoiding rotation and shearing and preserving straight lines when editing synthetic shapes.



These observations do not address reshaping-specific user preferences. To understand how the general preferences above apply to reshaping, we performed a study (App. A) which tasked participants with performing different reshaping operations on 2D icons using a tablet and stylus (inset, middle and Figs. 2, 9). In line with prior research, study participants consistently preferred solutions that avoided or minimized curve rotation, and that locally scaled the icon's curves as uniformly as possible. Participant-drawn outputs strongly suggest that, when presented with a task where some amount of shear or rotation is inevitable, humans prioritize preserving curve orientations, when possible, at the expense of introducing some non-uniform scale. We similarly observed that participants strictly preserved straight lines, even when doing so resulted in increased rotation across the board (App. A). Our observations and participant debriefing also suggest that when presented with reshaping constraints that do not allow for perfect uniform scaling, humans typically expect *some* parts of the network to noticeably stretch or shrink, and for there to be smooth transitions in scale along curves connecting these parts with other portions of the network which are expected to maintain their original scale (Fig. 2, middle block). Lastly and notably, consistent with observations made by Kraevoy et al. [2008], humans are less sensitive to changes in scale across sharp discontinuities: for instance, when asked to reshape the input in the inset, participants preserved the orientations and length of all vertical edges and simply shrank the horizontal ones. We design our reshaping strategy to reflect these preferences.

3.2 Continuous Problem Formulation

Motivated by the observations above, we formulate our problem as one of reshaping input curve networks *as locally uniformly as possible*: given an input vector curve network and a set of constraints that the reshaped network needs to conform with, we seek an output network that (1) satisfies these constraints; (2) supports a pointwise bijective continuous mapping from the input to the output that is locally as similar as possible to a uniform scale; and (3) preserves straightness as much as possible. We express our second requirement via two conditions. First, we expect normals at the corresponding points on the two networks to be maximally similar; second, we expect the gradient of the tangent length at the corresponding points on the two networks to be maximally similar. We can express these two properties variationally: for a single, *arc-length parameterized* smooth input curve $C_i(u) : [0, s] \rightarrow \mathbb{R}^2$, with pointwise tangents $\tau_i(u)$ and unit normals $n_i(u)$, and a set of constraints that map points $C_i(u_0), \ldots, C_i(u_k)$ on the curve to positions $p_0, \ldots, p_k \in R^2$ respectively, we must find a function $C_o : [0, s] \rightarrow \mathbb{R}^2$ representing the output curve, with pointwise tangents $\tau_o(u)$ and unit normals $n_o(u)$, that satisfies the positional constraints

$$C_o(u_i) = p_i, \; i \in 0 \ldots k, \tag{1}$$

and minimizes the weighted sum of the following two functionals:

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

$$E_{tangent} = \int_{u=0}^{s} \omega_t(u) (\frac{d\|\tau_o(u)\|}{du} - \frac{d\|\tau_i(u)\|}{du})^2 du. \tag{3}$$

The first functional, E_{normal}, encourages normal similarity. 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 $\tau_i(u)$, or equivalently encourages the mapping to have *constant speed* (note that since $C_i(u)$ is arc-length parameterized, its tangents have unit length and thus their gradient is zero). $\omega_n(u)$ 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 (Sec. 3.2.1). The two terms jointly express our desire for the mapping from the input curve to the output curve to be as locally close as possible to uniform scale.

Preserving straightness along a straight section of the input curve $[u_s, u_e]$ can be formulated as an additional per-section functional:

$$E_{straight} = \int_{u=u_s}^{u_e} \frac{d^2 C_o(u)}{du^2} du. \tag{4}$$

With these definitions in place, and treating the positional constraints at handle points $C_o(u_i)$ (Eqn. 1) as soft, we can formulate our goal as finding a new function that minimizes the sum of functionals:

$$E_{ALUP} = E_{normal} + E_{tangent} + w_s \sum_{S} E_{straight}(S) + w_c \sum_{i=0}^{k} \|C_o(u_i) - p_i\|^2 \tag{5}$$

The first two terms are computed over *all* network curves; S is the set of straight line segments along all of these curves (identified as discussed in Sec. 3.4). To strictly enforce straightness whenever feasible we set $w_s = 10,000$. We set $w_c = 100,000$ so that it outweighs all other terms.

3.2.1 Balancing Normal Preservation versus Tangent Length Uniformity. Given the formulation above our remaining challenge is to properly balance normal preservation versus tangent length uniformity, by appropriately defining $\omega_n(u)$ and $\omega_t(u)$. While some constraints and curves may allow for a zero distortion mapping which satisfies $E_{ALUP} = 0$ (for instance, when the constraints enforced allow for strictly uniform scaling), other curves and constraints may require change in normals or $\nabla \|\tau_o\|$ relative to the input. This raises the question: which of the two terms should be prioritized and where? The observations in Sec. 3.1 suggest that while users aim to minimize changes in normals and tangent length gradients overall, they often expect reshaping to introduce significant changes in the scale of different input elements; therefore, users are much more tolerant of changes in $\nabla \|\tau_o\|$ in some parts of the output than others. Unfortunately, our algorithm does not know the user's expected distribution of these changes in advance, we thus cannot directly specify $\omega_t(u)$ at each point along the curve network so as to match user expectations of change in $\nabla \|\tau_o\|$.

We address this challenge by leveraging the following observations. We first note that human observers are less tolerant of large increases in tangent length gradients if these can be avoided or significantly reduced at the expense of a small decrease in normal preservation. However they prioritize normal preservation if and when normals can be strictly preserved at the expense of a small increase in $\nabla \|\tau_o\|$ distortion.

Following the first observations we define ω_n so as to penalize normal deviation more when it coincides with larger than average stretch,

$$\omega_n(u) = \max(1, \frac{\|\tau_o(u)\|}{\tau_{avg}})^2 \tag{6}$$

where τ_{avg} is the average tangent length along the input curve. This weight prevents solutions where normal orientation is satisfied at the expense of much higher $\nabla \|\tau_o\|$.

We leverage the second observation by introducing a two step solution process (Fig. 4). We first obtain a solution that distributes both normal and $\nabla \|\tau_o\|$ errors as evenly as possible. The local tangent length gradients in this solution reflect unavoidable $\nabla \|\tau_o\|$ distortion. We then use these distorted local tangent length gradients as the target gradients in a final solution round that balances normal preservation against minimal increase in $\nabla \|\tau_o\|$ beyond this unavoidable distortion. In both rounds we set $\omega_t(u) \in [\epsilon_r, 1]$ to reflect the degree of visual discontinuity at $C(u)$ (measured as described in Sec 3.3, Eq. 11); $\omega_t(u) = 1$ when the curve is viewed as continuous at u and drops to $\epsilon_r = 1e^{-3}$ when it is discontinuous.

Core Solution Round. More formally, our core solution step (Fig 4b) finds a network that minimizes E_{ALUP} (Eq. 13). At this stage of the method we do not enforce straightness at network corners (vertices with valence higher than two), nor at control handles. Relaxing the constraints at handles and corners leads to a solution that better preserves both normals and $\nabla \|\tau_o\|$, providing a better overall approximation of the desired output. We refer to the network that minimizes Eq. 13 as $C^{updated}$, and the tangents of the curves inside this network as $\tau^{updated}(u)$. When computing the minimizer of E_{ALUP} we use identity as the initial mapping function guess, promoting solutions that retain $\nabla \|\tau_o(u)\|$ as much as possible for all u.

Final Solution Round. We use the output of the distortion estimation stage to obtain accurate targets for the user-anticipated $\nabla \|\tau_o\|$. To this end we replace $\tau_i(u)$ in our tangent energy term $E_{tangent}$ (Eq. 3) with $\tau^{updated}(u)$,

$$E_{tangent} = \int_{u=0}^{s} \omega_t(u)(\frac{d\|\tau_o(u)\|}{du} - \frac{d\|\tau^{updated}(u)\|}{du})^2 du. \tag{7}$$

We then compute the output network curves that minimize the updated overall energy function E_{ALUP} using the positions of the intermediate network curves $C^{updated}$ as the initial guess (Fig 4bc). This choice promotes solutions that retain the initial tangent lengths $\|\tau(u)\|$ as much as possible. We include corner and handle straight line points in the straightness energy term, if they remain approximately straight in the core solution output. As Fig. 4 illustrates this step allows the method to strictly satisfy normal preservation on inputs where the core solution almost satisfies them.

3.3 Discrete Formulation

Data Discretization. We first discretize all input vector curves as densely and evenly sampled polylines. When discretizing, we mark vertices that lie in the *interior* of straight lines in the original vector input as *straight-line*, and use this information during subsequent processing. After reshaping the polyline network, we convert the output back into vector form (Sec. 3.4).

Notations. We define E as the set of input polyline network edges, V as the complete set of vertices, and $H \subset V$ as the set of control handle vertices. We denote the output vertex positions we solve for in each round as v_j for all $j \in V$ and denote the vertex positions at the start of this round as v_j^0. We denote the *initial* normal of each edge $\langle i, j \rangle$ as n_{ij}^i (normals remain fixed across both rounds), and the set of straight-line vertices detected during discretization as S.

3.3.1 Discrete Formulation. We replace our continuous energy terms with corresponding discrete ones as follows. In the discrete setting we encode E_{normal} as:

$$E_{normal}^d = \sum_{\langle i,j \rangle \in E} \max\left(1, \left(\frac{\|v_j - v_i\|}{L_{avg}}\right)^2\right)\left(n_{ij}^i \cdot \frac{v_j - v_i}{\|v_j - v_i\|}\right)^2 \tag{8}$$

where the first term is the discretization of $\omega_n(u)$ and L_{avg} denotes the average length of the curve network edges at the start of the current solution round.

Since our initial discretization samples curve points uniformly, the lengths of the output network edges approximate the tangent lengths of the mapped curves. We therefore recast the tangent length gradient as the difference between the lengths of adjacent output edges, and can consequently reformulate $E_{tangent}$ as preserving the ratios between the lengths of consecutive edges along network curves (edges are defined as consecutive if they share common end vertices):

$$E_{tangent}^d = \sum_{i \in V} \sum_{j,k \in N_i; j \neq k} \omega_t^d(ijk) \frac{1}{L_{avg}} \left(\|v_j - v_i\| - r_{ijk}^0 \|v_k - v_i\|\right)^2 \tag{9}$$

We set r_{ijk}^0 to the ratios between the lengths of the edges $\langle i, j \rangle$ and $\langle i, k \rangle$ at the start of the current solution round.

$$r_{ijk}^0 = \|v_j^0 - v_i^0\| / \|v_k^0 - v_i^0\| \tag{10}$$

We define the weights $\omega_t^d(ijk)$ to reflect the visual smoothness at the center vertices j in the *input* curve network. For the purposes of our computation, we approximate the detection of visible discontinuities by using the angle at each vertex as a proxy for the likelihood of the vertex being a visible discontinuity:

$$\omega_t^d(ijk) = \begin{cases} e^{\left(\frac{-(\theta_{ijk} - \pi)^2}{2\sigma^2}\right)} & \theta_{ijk} > \pi \frac{95}{180} \\ \epsilon_r & \theta_{ijk} < \pi \frac{95}{180} \end{cases} \tag{11}$$

where $\sigma = \frac{\pi}{6}$. We set this weight to ϵ_r at high-valence network vertices, since the network is visually discontinuous at corners.

We discretize the straightness energy $E_{straight}$ as

$$E^d_{straight} = \sum_{i \in S} \sum_{j,k \in N_i; j \neq k} \left\| \frac{(v_j - v_i)}{\|v_j - v_i\|} - \frac{(v_i - v_k)}{\|v_k - v_i\|} \right\|^2. \quad (12)$$

The set S includes all vertices that are interior to the straight lines in the original curve network.

We now can minimize the discretized nonlinear energy $E^d_{ALUP}(v)$,

$$E^d_{ALUP} = E^d_{normal} + E^d_{tangent} + w_s \sum_S E^d_{straight}(S) + w_c \sum_{i=0}^{k} \|C_o(u_i) - p_i\|^2 \quad (13)$$

using the tailored iterative solution method described below.

3.3.2 Discrete Energy Minimization. In both stages of our method we seek to compute the vertex positions $v_j \in V$ that minimize the energy function E^d_{ALUP}, starting from the current stage input vertex positions $v^0_j \in V$ (where V are either the initial or updated positions depending on the current stage). Our nonlinear energy function includes fractional terms whose denominators contain square roots of intra-vertex distances. Optimizing such functions using brute force approaches is highly time consuming, with off the shelf optimizers failing to reach a local minimum within a reasonable timeframe. We compute the desired minimum using a tailored efficient optimization strategy that combines three core technical elements. First, we introduce auxiliary variables l_{ij} designed to capture the desired lengths of the network edges $\langle i, j \rangle$. We then rewrite the energy function so that once either the values of these auxiliary variables or the vertex positions are fixed, we may compute the energy-minimizing values of the other set of variables by solving a standard least squares problem. We then use an alternating least-squares minimization process to obtain the desired minimum.

Optimization With Auxiliary Variables. Rewriting our main energy terms using the auxiliary variables l_{ij}, we have:

$$E'_{normal} = \sum_{(i,j) \in E} \max\left(1, \left(\frac{l_{ij}}{L_{avg}}\right)^2\right) \left(n^i_{ij} \cdot \frac{v_j - v_i}{l_{ij}}\right)^2 \quad (14)$$

$$E'_{tangents} = \sum_{i \in V} \sum_{j,k \in N_i; j \neq k} \omega^d_t(ijk) \frac{L_{avg}}{\|v_j - v_i\|} \left(l_{ij} - r_{ijk} l_{ik}\right)^2 \quad (15)$$

$$E'_{straight} = \sum_{i \in S} \sum_{j,k \in N_i; j \neq k} \left\| \frac{(v_j - v_i)}{l_{ij}} - \frac{(v_i - v_k)}{l_{ik}} \right\|^2. \quad (16)$$

We seek for the values of the auxiliary variables l_{ij} to equal the lengths of the edges $\langle i, j \rangle$. Instead of enforcing this relationship via hard constraints, we introduce an additional energy term that articulates this relationship:

$$E'_{edge} = \sum_{\langle i,j \rangle \in E} \max\left(1, \left(\frac{l_{ij}}{L_{avg}}\right)^2\right) \left\| \frac{v_i - v_j}{l_{ij}} - \frac{v^0_i - v^0_j}{l^0_{ij}} \right\|^2 \quad (17)$$

where $l^0_{ij} = \|v^0_i - v^0_j\|$. This term is minimized when the output edges have similar directions to the original, and satisfy $\|v_i - v_j\| = l_{ij}$. Notably the first of these properties is a reinforcement of our desire to avoid rotations.

With this term added we approximate minimizing E^d_{ALUP} as minimizing

$$E'_{ALUP} = E'_{normal} + E'_{edge} + E'_{tangent} + w_s E'_{straight} + w_c \sum_{i=0}^{k} \|C_o(u_i) - p_i\|^2 \quad (18)$$

While this function remains nonlinear with respect to the augmented set of variables, it can now be robustly optimized using an alternating least squares strategy. We initialize both sets of variables as $v_i = v^0_i$ and $l_{ij} = l^0_{ij}$. At each iteration we first keep the target lengths l_{ij} fixed and solve for positions that minimize the energy function given these fixed lengths. We then keep the positions fixed and update the target lengths l_{ij}. Each iteration reduces the overall energy E^d_{ALUP}. The process terminates once we reach a minimum (Sec. 3.4).

Solving for Positions. When the lengths of the desired edges l_{ij} are fixed, minimizing E'_{ALUP} is equivalent to minimizing

$$E'_{vertex} = E'_{normal} + E'_{edge} + w_s E'_{straight} + w_c \sum_{i=0}^{k} \|C_o(u_i) - p_i\|^2 \quad (19)$$

as the other terms do not depend on vertex positions. This energy is quadratic with respect to the vertex positions, and finding the positions that minimize it amounts to a simple linear solve.

Solving for Desired Edge Lengths. We now fix the newly computed positions and update the desired edge lengths. Several of the terms in E'_{ALUP} use edge lengths in their denominator, making direct optimization with respect to lengths challenging. We observe that our auxiliary term E'_{edge} encodes both edge directions and lengths and as such serves as a suitable proxy for the non-linear normal term E'_{normal}. We linearize the optimized function by dropping the normal term E'_{normal} and rewriting E'_{edge} as a quadratic function of edge lengths. Specifically, as the quadratic function $f(x) = x^2$ is strictly monotonically increasing over the nonnegative reals, we may rewrite E'_{edge} to square the edge lengths without changing the minimizer:

$$E^{linearized}_{edge} = \frac{1}{L_{avg}} \sum_{\langle i,j \rangle \in E} (l_{ij} - \|v_i - v_j\|)^2) \quad (20)$$

The function we minimize then becomes:

$$\min_{l_i} E_l(l_i) = E^{linearized}_{edge} + E^{'}_{tangents} \quad (21)$$

We compute the minimizer of this quadratic function using the same solver as for positions.

3.4 Implementation Details

Discretization. We uniformly sample each input curve in the curve network, with a spacing between samples of 0.5% of the length of the largest side of the network's bounding box. To enforce positional constraints our polygon vertices include all user specified point handles and includes at least two vertices on each user specified curve handle. We mark polyline segments as straight if their originating vector segments were either straight lines or curves

with colinear control points. Converting from polylines to curve networks is a well-researched problem; we use the built-in tools in Adobe Illustrator.

Termination. Our alternating least-squares iterations terminate when one of the following conditions is satisfied: (1) The decrease in the value of E^d_{ALUP} from one iteration to the next drops below $\epsilon = 10^{-4}$; (2) The largest change in a vertex position from one iteration to the next drops below $\epsilon \cdot L_{avg}$); (3) The value of E^d_{ALUP} increases from one iteration to the next. In this case we return the set of positions, lengths and ratios that resulted in the smallest value of E^d_{ALUP}. (4) The maximum allowed number of iterations ($iter_{max} = 100$) is reached.

Solver Implementation and Runtimes. We use the simplicial LDL^T solver from the Eigen linear algebra library to solve both least squares problems. As the curve network topology does not change between iterations, we precompute the LDL^T decomposition once at the start of each solution round; each iteration then consists only of forwards and backwards substitution. Our runtimes range from 16 msec to 2.2 sec, with the median runtime being 0.3 seconds. Performance was measured on an Intel I7-7700 2.80GHz, with 32GB RAM, running Windows 10.

3.5 Extensions.

We augment our core method with support for multiple components and intersection avoidance.

Multiple Connected Components. Vector clip-art images frequently contain multiple connected components (e.g. the crown in Figure 1 in our paper). While users can apply our core method independently to each component, it is often advantageous to adjust components in tandem — for instance moving all the details in the crown together with the outline. We support this tandem motion by computing a constrained Delaunay triangulation scaffold connecting all user-specified components of interest. Our approach is similar to [Kraevoy et al. 2003] amongst other methods: we construct a constrained Delaunay triangulation of the points of the input curve network and the vertices of the clip-art's axis-aligned bounding box, and with the edges of the curve network as required edge constraints, using the *Triangle* software library [Shewchuk 1996]. We then augment our energy function E'_{ALUP} with a weak term that preserves edges in this triangulation belonging to different components, mirroring E'_{edge}:

$$E'_{comp} = \sum_{(i,j)\in T} \left\| \frac{v_i - v_j}{l^0_{ij}} - \frac{v^0_i - v^0_j}{l^0_{ij}} \right\|^2 . \tag{22}$$

Here T are the triangulation edges connecting *different* components of interest, v^0_i, v^0_j are the original locations of their vertices, and l^0_{ij} are the original edge lengths. We add this term to our overall energy E'_{ALUP}, and incorporate it into the position solve (Sec. 3.3.2) by adding this term to E'_{vertex} (Eq. 19). To ensure that this term only minimally impacts the shape of the network curves, we assign it a weight of $w_{comp} = 0.01/|T|$.

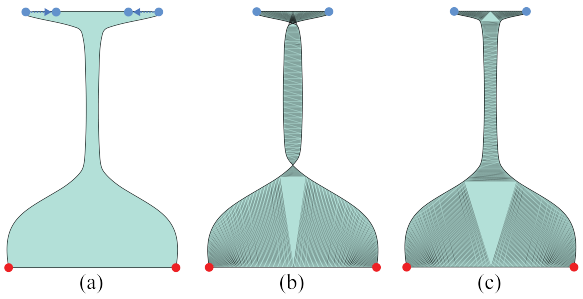


Fig. 5. Given the input in (a), our unconstrained solution (b) introduces some self-intersections along the stem of the wine glass; (c) we resolve these intersections using a second solution iteration.

Self-Intersections. Our core formulation does not explicitly prevent self-intersections. While rare, these can and do happen when narrow features are compressed (Fig 5). We resolve intersections using a mechanism similar to the one for handling multiple components. We first compute a solution using our standard method and check for self-intersections in the output. If and when a self-intersection is detected, we compute a Delaunay triangulation of the input network, identify the boundaries of the overlapped region and identify the Delaunay triangulation edges S connecting different sections along these overlapped boundaries. We then add these edges into the set T above, including them in the component energy term E'_{comp}, and recompute the reshaping solution. We use the same weight w_{comp} as before. Out of all 115 inputs we tested, only two inputs (the wine glass shown in Fig. 5, and the N shape in the supplementary) required intersection resolution. In our experiments, this process was sufficient to resolve all intersections encountered.

4 RESULTS AND VALIDATION

Throughout the paper we demonstrate our ALUP reshaping method on 18 examples, including 3 individual closed contours and 15 curve networks. An additional 97 examples are included in the supplementary material. These examples include both inputs where the changes in the control handle locations allows for solutions where the orientation of the input network curves can be largely preserved (e.g windmill and lotion bottle Fig 6) as well as changes that cannot be satisfied without significantly altering curve orientation (e.g. car in Fig. 6) In all cases our method produces results consistent with human expectations. Please see the Appendix for ablations and details.

Comparison to Prior Art. As discussed in Sec 2, our method's goals are conceptually different from those of most prior shape deformation approaches. Figs. 1, 2, and 7 highlight the differences in the outputs produced by ALUP reshaping, a baseline Poisson deformation approach [Cohen-Or et al. 2015], the state of the art 2D deformation method of [Solomon et al. 2011], and Adobe Illustrator's Puppet Warp tool, which implements [Liu et al. 2014] with the biharmonic weights of [Jacobson et al. 2012].

We validate our improvement over prior art via comparative evaluation and a qualitative study. In our comparative study, participants were shown input clip-art visuals with specified control

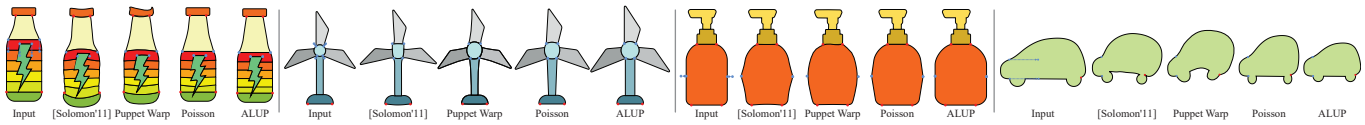


Fig. 6. Additional comparisons with prior methods. Input images adapted from: Bottle © dstarky, Shampoo © Svitlana - stock.adobe.com. Wind-turbine © DinosoftLabs - www.flaticon.com.

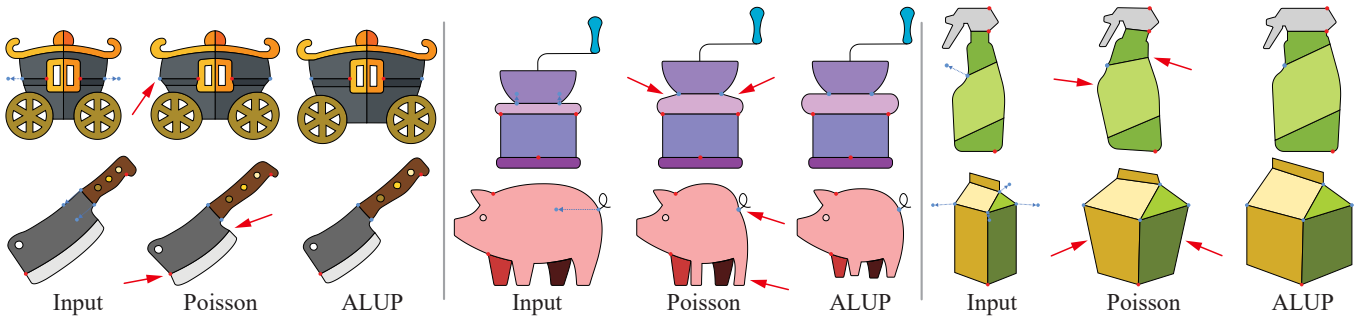


Fig. 7. Additional comparisons with Poisson deformation. Input images adapted from: Carriage © Freepik - www.flaticon.com. Coffee grinder © bsd studio, Cleaning bottle © Svitlana, Santoku © bsd studio, Pig © Nataliia, Milk box © dstarky - stock.adobe.com.

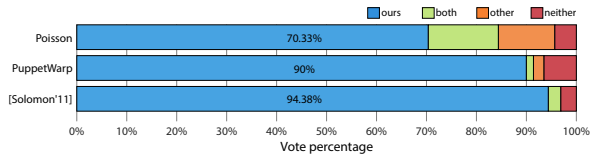


Fig. 8. Comparative study summary. Participants consistently preferred our results over all alternatives.

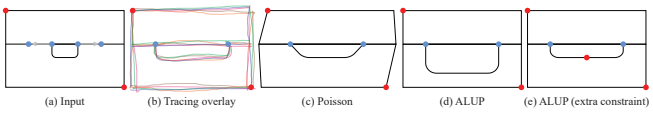


Fig. 9. Limitation: given the input image and constraints in (a), the ALUP method scales the buckle uniformly and in contrast to Poisson deformation (c) strictly preserves curve orientations. (b) Notably, similar to us the study participants preserved curve orientations, but scaled the buckle non-uniformly preserving its original height. We speculate that in doing so they relied on global context or semantics. (c). Adding one additional anchor handle (e) allows ALUP to capture the human solution.

handle displacements and pairs of outputs, one generated using ALUP reshaping and one generated using an alternative method. Fig 8 summarizes the responses. In comparisons against Poisson deformation ALUP reshaping was preferred 70% of the time, the results were judged on par 14% of the time, with the alternative preferred only 11% of the time. It was preferred by much higher margins against the other two methods, confirming that in the context of reshaping ALUP outputs are drastically better aligned with human expectations than prior art. Our t-test confirmed that these results were highly statistically significant ($p < 0.001$).

In the qualitative study, we visually compare our outputs, as well as those produced by the three methods above, against 15 manual reshaping outputs drawn by the participants in our baseline study (App. A). As the comparisons show, our results are significantly better aligned with those traced by the participants. With one exception, discussed below, our results are visually very similar to those sketched by the participants.

Comparison to Manual Reshaping. We evaluated our outputs against manual reshaping via a comparative study with a similar layout to the above one. Participants were shown three inputs with specified control handle displacements and pairs of outputs, one generated using ALUP reshaping and one manually reshaped by a professional artist to conform to the specified displacements while preserving shape as much as possible. Participants preferred our outputs 60% of the time over the artist-created results, judged us as on par 27% of the time, and preferred the artist results only 13% of the time, confirming that our results are well aligned with viewer expectations.

See the Appendix for discussion of study methodology.

Ablations and Stress Tests. We demonstrate the robustness of our method via ablative studies of the main weight settings used by our method, and by evaluating its ability to perform reshaping that requires extreme compression.

We ablate the three main weight settings used by our method. Fig 10 compares our scheme, which weights E_{normal} and $E_{tangent}$ equally in our overall energy E_{ALUP} (Eqn. 13), against schemes which weigh them by a ratio of 100:1 or 1:100. Fig 11 compares our scheme of setting $\omega_n(u)$ (Eq. 6) against using uniform weights; and Fig 12 compares our scheme of setting $\omega_t(u)$ (Eq. 11) against using uniform weights. Notably our formulation is sufficiently stable that for many inputs alternative weighing schemes make little to no difference; however for the subset they impact the changes they introduce are clearly undesirable.

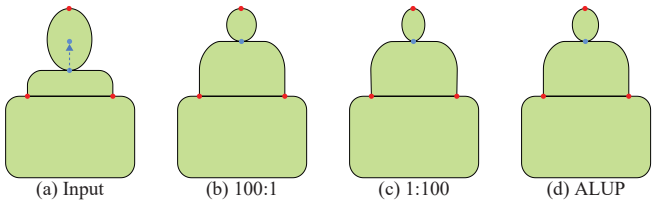


Fig. 10. Emphasizing normal (b) or $\nabla \|\tau_o\|$ (c) preservation by a factor of 100 produces results less aligned with viewer expectations than those produced by our method, which weighs them equally.

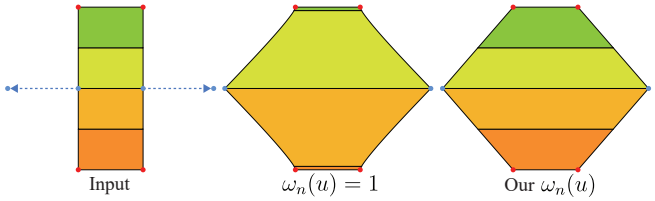


Fig. 11. Setting $\omega_n(u) = 1$ concentrates normal distortion along a small subset of network edges, and leads to significant increase in $\nabla \|\tau_o\|$ error. Our method avoids this behavior.

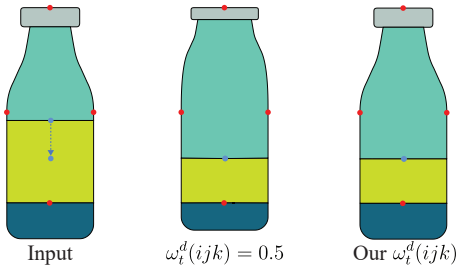
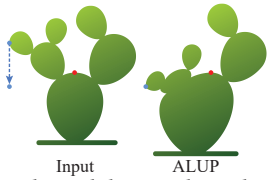


Fig. 12. Setting $\omega_t(u) = 0.5$ everywhere causes the solver to distribute changes in scale uniformly, including across corners and handles, resulting in unexpected changes in parts of the network viewers expect to retain the original scale. Our output is consistent with viewer expectations. Input image adapted from dstarky - stock.adobe.com.



Robustness to Extreme Deformation. We tested our method on several examples where achieving the user's target deformation requires extreme compression of edge lengths (inset), and were unable to trigger failure cases or numerical instability. We hypothesize that our choice to maximize tangent gradient similarity, combined with division by tangent length (Eqns. 2 and 14 through 17) and the explicit lower bound on weights (Eqn. 6) explicitly discourages the formation of zero-length edges. In our early experiments we explicitly prevented the length of any edge from shrinking below 1% of its original edge length, but we found that this condition was never activated and removed it.

Limitations. Humans often impose semantic constraints which are not articulated when manually reshaping clip-art, based on contextual or aesthetic properties of the input, which our method is not aware of (Fig. 9). Thus if these constraints are not articulated it will not necessarily respect them. In this example, our result can be made identical to the participant traced one by adding one additional anchor handle.

Our method operates on curve networks by discretizing them into polylines. While this is a standard technique for deforming and processing curve networks, conversion and reconversion from a curve network to polylines may cause undesirable changes in curve network topology or curve type. Developing an ALUP reshaping algorithm that operates directly on curves, similar to the work of [Liu et al. 2014], is an interesting area for future work.

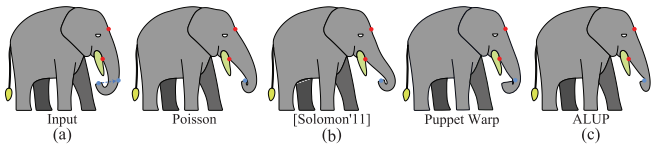


Fig. 13. Reshaping versus Reposing. Humans observers interpret the control point repositioning (left) as reposing the tusk on the elephant; such tasks are well addressed by existing methods (center). Our method (right) is designed for reshaping rather than reposing. Input image adapted from Nataliia - stock.adobe.com.

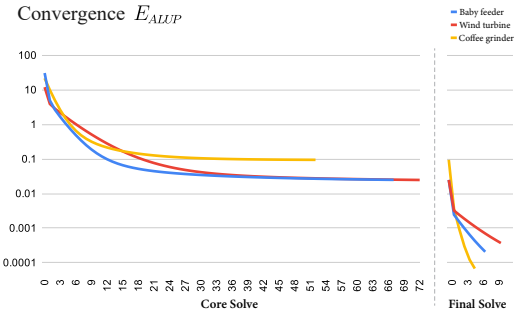


Fig. 14. Convergence of the energy E_{ALUP} (Eqn. 13). Note the rapid convergence of the final solution.

Reshaping versus Reposing. Our ALUP reshaping method is intended as a complement to existing deformation methods. Previous work focuses on deformation that mimics either articulated posing, or physically plausible deformation of shapes (i.e. bending and twisting an iron bar.) These use cases assume that the local geometry of the inputs must be preserved, while the orientation of the geometry can change; they are not designed to address the use case where an artist or designer wishes to reshape the object by changing the relative or absolute scale of its parts. Our method targets this second use case, but does not replace existing methods when it comes to their target use case (Fig. 13).

5 CONCLUSIONS

We presented As-Locally-Uniform-as-Possible (ALUP) reshaping, the first robust method for reshaping vector clip-art images. ALUP is inspired by observations about human expectations of reshaping outcomes, and is demonstrated to be significantly better aligned with user preferences than alternative approaches. Our key technical

contribution is a rigorously formulated discretization of a continuous variational optimization problem, which can be iteratively and efficiently solved with alternating least squares linear solves. The resulting method is simple to implement and suitable for adoption in commercial software packages.

Our work proposes a number of avenues for future research; the first and foremost of these is an extension of our technique to reshaping of human-made 3D content. Our work also raises interesting questions about human perception and expectations when it comes to editing of both 2D and 3D human-made content, specifically the balance humans seek between conflicting structure-preservation preferences when these cannot be jointly preserved.

ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for their helpful comments. We are also deeply grateful to Silver Burla for help with input creation. We acknowledge the support of Adobe and the Natural Sciences and Engineering Research Council of Canada (NSERC) grant RGPIN-2018-03944 ("Broad-Based Computational Shape Design").

REFERENCES

Adobe Inc. 2019. *Adobe Illustrator.* https://adobe.com/products/illustrator

Marc Alexa, Daniel Cohen-Or, and David Levin. 2000. As-Rigid-as-Possible Shape Interpolation. In *Proc. SIGGRAPH 2000.* ACM Press/Addison-Wesley Publishing Co., 157–164.

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.

Shai Avidan and Ariel Shamir. 2007. Seam Carving for Content-Aware Image Resizing *(SIGGRAPH '07).* Association for Computing Machinery.

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

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).

Donghyeon Cho, Jinsun Park, Tae-Hyun Oh, Yu-Wing Tai, and In So Kweon. 2017. Weakly-and self-supervised learning for content-aware deep image retargeting. In *Proceedings of the IEEE International Conference on Computer Vision.* 4558–4567.

Ravi Chugh, Jacob Albers, and Mitchell Spradlin. 2015. Program Synthesis for Direct Manipulation Interfaces. *CoRR* (2015).

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.

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.

Michael S Floater. 2003. Mean value coordinates. *Computer aided geometric design* 20, 1 (2003), 19–27.

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.

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

Josef Hoschek and Dieter Lasser. 1993. *Fundamentals of Computer Aided Geometric Design.* A K Peters/CRC Press.

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

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

Inkscape. 2003. *Inkscape.* https://inkscape.org

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.

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.

Tao Ju, Scott Schaefer, and Joe Warren. 2005. Mean value coordinates for closed triangular meshes. In *ACM Siggraph 2005 Papers.* 561–566.

Vladislav Kraevoy, Alla Sheffer, and Craig Gotsman. 2003. Matchmaker: constructing constrained texture maps. *ACM Transactions on Graphics (TOG)* 22, 3 (2003), 326–333.

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.

Sylvain Lefebvre, Samuel Hornus, and Anass Lasram. 2010. By-example Synthesis of Architectural Textures. *ACM Trans. Graph (Proc. SIGGRAPH)* (2010).

Tzu-Mao Li, Michal Lukáč, Michaël Gharbi, and Jonathan Ragan-Kelley. 2020. Differentiable Vector Graphics Rasterization for Editing and Learning. *ACM Transactions on Graphics (TOG)* 39, 6 (2020), 1–15.

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

Songrun Liu, Alec Jacobson, and Yotam Gingold. 2014. Skinning Cubic BéZier Splines and Catmull-Clark Subdivision Surfaces. *ACM Trans. Graph.* 33, 6 (2014).

Ravish Mehra, Qingnan Zhou, Jeremy Long, Alla Sheffer, Amy Gooch, and Niloy J Mitra. 2009. Abstraction of man-made shapes. In *ACM SIGGRAPH Asia 2009 papers.* 1–10.

Seung-Hun Nam, Wonhyuk Ahn, Seung-Min Mun, Jinseok Park, Dongkyu Kim, In-Jae Yu, and Heung-Kyu Lee. 2019. Content-aware image resizing detection using deep neural network. In *2019 IEEE International Conference on Image Processing (ICIP).* IEEE, 106–110.

Daniele Panozzo, Philippe Block, and Olga Sorkine-Hornung. 2013. Designing Unreinforced Masonry Models. *ACM Trans. Graph.* 32, 4, Article 91 (2013), 12 pages.

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

Vidya Setlur, Tom Lechner, Marc Nienhaus, and Bruce Gooch. 2007. Retargeting Images and Video for Preserving Information Saliency. *IEEE Computer Graphics and Applications* 27, 5 (2007), 80–88.

Jonathan Richard Shewchuk. 1996. Triangle: Engineering a 2D quality mesh generator and Delaunay triangulator. In *Workshop on applied computational geometry.* Springer, 203–222.

Denis Simakov, Yaron Caspi, Eli Shechtman, and Michal Irani. 2008. Summarizing visual data using bidirectional similarity. In *2008 IEEE CVPR.* IEEE, 1–8.

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.

Olga Sorkine, Daniel Cohen-Or, Yaron Lipman, Marc Alexa, Christian Rössl, and Hans-Peter Seidel. 2004. Laplacian Surface Editing. In *Proc. EUROGRAPHICS/ACM SIGGRAPH Symposium on Geometry Processing.* ACM Press, 179–188.

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

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.

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.

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

Cem Yuksel. 2020. A Class of C2 Interpolating Splines. *ACM Transactions on Graphics* 39, 5, Article 160 (jul 2020).

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).