Planar Interpolation with Extreme Deformation, Topology Change and Dynamics

YUFENG ZHU, University of British Columbia & Adobe Research
JOVAN POPOVIĆ, Adobe Research
ROBERT BRIDSON, University of British Columbia & Autodesk
DANNY M. KAUFMAN, Adobe Research

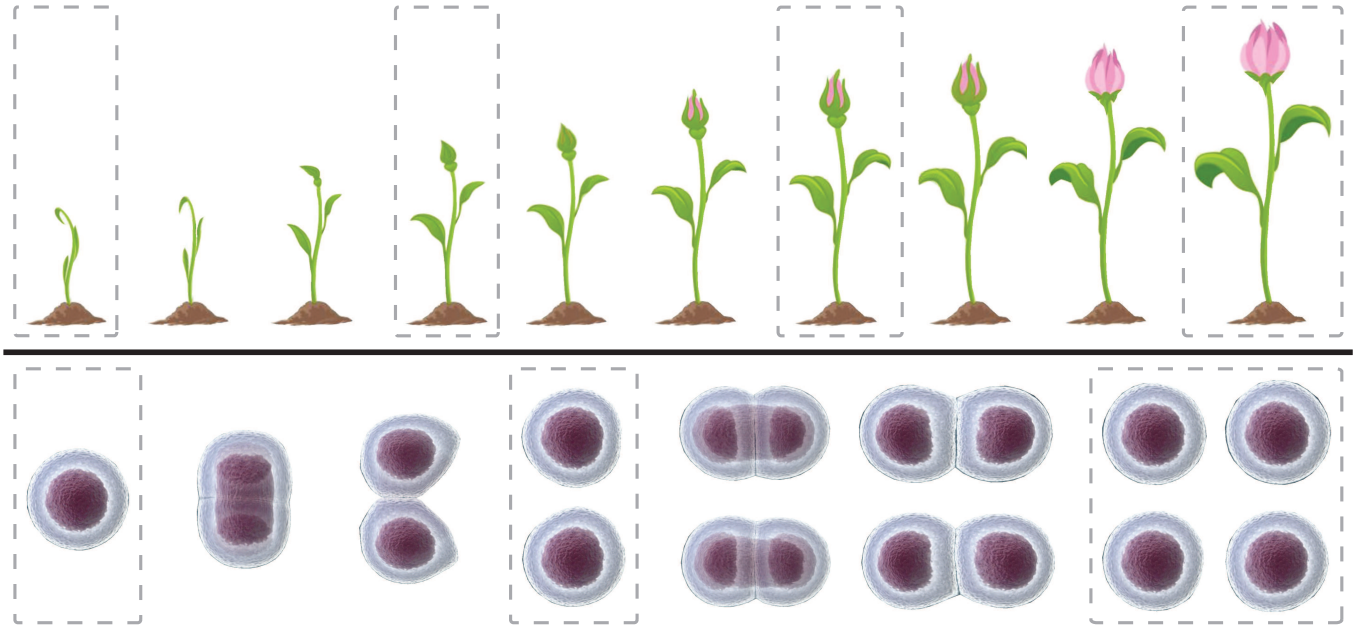


Fig. 1. We interpolate inbetweens (unboxed images) from sparse key drawing shapes (boxed images) across arbitrary topology changes and extreme deformations. Artists define desired correspondences to explore interpolation paths in-between key shapes and add dynamic effects when desired (bottom).

We present a mesh-based, interpolatory method for interactively creating artist-directed inbetweens from *arbitrary* sets of 2D drawing shapes without rigging. To enable artistic freedom of expression we remove prior restrictions on the range of possible changes between shapes; we support interpolation with extreme deformation and unrestricted topology change. To do this, we extend discrete variational interpolation by introducing a consistent multimesh structure over drawings, a Comesh Optimization algorithm that optimizes our multimesh for both intra- and inter-mesh quality, and a new shape-space energy that efficiently supports arbitrary changes and can prevent artwork overlap when desired. Our multimesh encodes specified correspondences that guide interpolation paths between shapes. With these correspondences, an efficient local-global minimization of our energy interpolates n-way between drawing shapes to create inbetweens. Our Comesh Optimization enables artifact-free minimization by building consistent meshes across drawings that improve both the quality of per-mesh energy discretization and inter-mesh mapping distortions, while guaranteeing a single, compatible triangulation. We implement our method in a test-bed interpolation system that allows interactive creation and editing of animations from sparse key drawings with arbitrary topology and shape change.

CCS Concepts: • **Computing methodologies** → *Animation*; *Mesh models*;

Additional Key Words and Phrases: planar interpolation, inbetweening, animation, geometry, deformation, topology change, compatible meshing

ACM Reference format:
Yufeng Zhu, Jovan Popović, Robert Bridson, and Danny M. Kaufman. 2017. Planar Interpolation with Extreme Deformation, Topology Change and Dynamics. *ACM Trans. Graph.* 36, 6, Article 213 (November 2017), 15 pages.
DOI: 10.1145/3130800.3130820

DOI: 10.1145/3130800.3130820

1 INTRODUCTION

Humans draw what's on their mind: they do it to tell stories; they do it to share with others; and they do it to reflect on their thoughts. They use cameras when they record, but they draw when they invent. The software industry has long recognized this need with the development

of comprehensive products for sketching, illustration, and presentation. Brushes, layers, undo, copy, paste, and other tools can make it easier to draw. However, when it comes to drawing *animation*, even the most sophisticated products provide little assistance for changing shape. In this work, we develop a new method to enable artist-driven inbetweening of drawn shapes as an essential steps towards *drawing* animation without having to draw every frame.

Animations change shape. Drawing software requires redrawing each frame. Animation software does not; but it requires *rigging* to articulate shape, and keyframing to change that articulation over time. A drawer wants to draw - not rig or keyframe - but not every frame. This gap inhibits animation. Almost every child draws pictures, but almost none animate.

Making animation more accessible today does not require eliminating drawing. It requires assistance with the creation of inbetween drawings. We focus on enabling intuitive, rig-free control of inbetweening from a set of drawn shapes. Our system delivers an initial animation from just a few sparse key drawings and a set of user-specified correspondences, see Figure 1 and Section 3, and then can be refined with swapped-in and added drawings, e.g., extreme or breakdown poses. This emulates the well established workflow from classical (hand-drawn) animation, but with our software assisting with the interpolation of inbetween drawing shapes.

Inbetweening with drawings should allow artists to communicate as freely as they draw. Drawings can exaggerate and invent without constraint. Thus computational inbetweening with drawings should likewise be unrestricted by rigging and articulation curve constraints. Current computational tools, however, cannot deliver this today because geometric methods do not yet support this implied full freedom of artistic expression. Even the most simple outline drawing shapes reveal three difficult-to-inbetween transformations: 1. large and exaggerated non-uniform stretching and compression; 2. topological change and 3. physics-based dynamics.

Contributions. To enable artistic freedom of expression, we remove these prior restrictions on the range of possible changes between shapes supported by computational interpolation methods. We present a set of contributions that support mesh-based, variational interpolation for interactively creating directed inbetweens from *arbitrary and unrestricted* sets of 2D drawing shapes without rigging. Our goal then is to deliver *pleasing* interpolation that is smooth, stable, and consistent with *extreme deformation, unrestricted topology change, and physics-based dynamic effects.*

Solving this problem led us to two mutually dependent subproblems: 1. an interpolation method for pleasing transitions requires higher quality compatible meshes than previously available and new energies to support these transitions; and 2. compatible meshing must be aware of the interpolation method applied. Our primary contributions address these two subproblems:

- a *Comesh Optimization* algorithm that improves both the mesh and mapping quality of compatible meshes so that we can support pleasing interpolation across unrestricted shape changes; and
- a *Multimesh Interpolation* method that extends variational interpolation [Rumpf and Wirth 2009; Von-Tycowicz et al. 2015] by introducing a consistent, annotated multimesh structure and a new shape-space energy that supports topology change, can prevent shape overlap when and where desired, and add dynamic effects as directed. The resulting interpolation is efficient and enables interactive animation creation and editing. We demonstrate this with a range of challenging animation tasks including production examples in our supplemental video.

2 RELATED WORK

Planar Animation. Planar animation is classical topic in computer graphics. With decades of exploration in the area, a wide range of methods have been developed for 2D animation using strokes [Kort 2002; Whited et al. 2010], triangle meshes [Alexa et al. 2000; Baxter et al. 2009a,b] and vector graphics [Dalstein et al. 2015], to name just a few domains. To enhance planar animation, researchers have also considered occlusion in 2D. Yang et al. [2012] proposed a multi-element 2D shape representation to resolve disjoined and overlapped parts. Sýkora et al. [2010] introduce cracks in the embedding cage by allowing discontinuities in the depth field. Embedding cages can then be separated in two in order to improve image registration accuracy. Nevertheless, these approaches cannot model topology changes such as adhesion effects during interpolated animation. Whited et al.[2010] resolve challenges in layering and support occlusion ambiguities with the help of artists. Research in planar animation has also explored velocity field advection [Li et al. 2013] and data driven strategies [Averbuch-Elor et al. 2016], inference of motion cycles from motion snapshots of individuals captured in a stills [Xu et al. 2008], and beautification of existing drawings via global similarity analysis [Xing et al. 2015]. Here we introduce a novel interpolation method to enable planar animations from sparse key shapes with extreme deformation, topology change and dynamics—to our knowledge this is the first time this has been possible.

Shape Interpolation. Shape interpolation has been widely explored in geometric modeling, computer vision, imaging and computer animation [Wirth et al. 2011]. As in *Euclidean* space, the action of interpolation on *shapes* is induced by definition of a metric such as

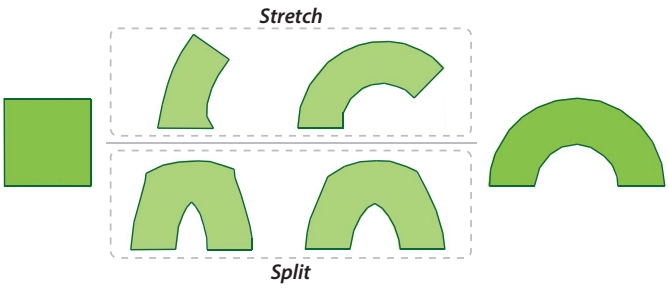


Fig. 2. Animating a square to an arch: depending on taste and storyline, an animator may wish to (1) *stretch* the cube out in anticipation, bridging to the arch; or (2) may instead want to first *split* the cube and then spread to the arch. We provide flexible exploration of the range rather than being restricted to one animation pre-established by a metric as in e.g., optimal transport which captures the *split* but misses the *stretch*.

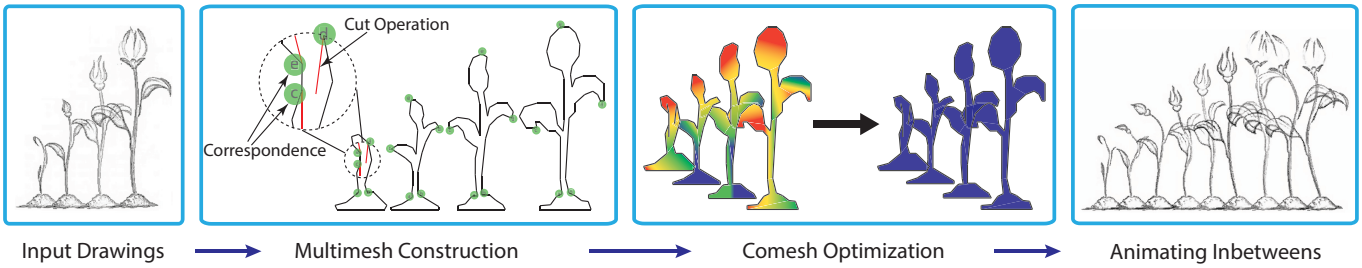


Fig. 3. Our framework takes drawings as a set of images. It constructs our multimesh structure by pairing extracted shapes with annotated cuts and correspondences to build initial embedding shapes for each with compatible meshing. Our Comesh Optimization then improves both mesh and mapping quality of the shape to obtain high-quality, stable interpolations.

Minkowski [Kaul and Rossignac 1992], *Wasserstein* [Solomon et al. 2015], and *Gromov-Hausdorff* [Gromov 2007] distances, as well as local distortion measures [Heeren et al. 2012; Wirth et al. 2011]. For design purposes, interpolation should satisfy artistic intent. Yet, as a geometric task, interpolation between arbitrary planar shapes is ambiguous and underspecified. Consider, for example, the classic exercise of animating a square to an arch. Depending on taste and storyline, we may either wish to stretch the square out in anticipation, bridging to the arch (Figure 2 *stretch*); or we may instead wish to first split the square from the bottom and then spread to the arch (Figure 2 *split*). Our method follows the local distortion framework, as we seek a method that provides control via correspondences over interpolation and topology changes so that users can control interpolation path.

Previous work includes comprehensive review of local distortion approaches [Chen et al. 2013; Von-Tycowicz et al. 2015], but we prefer a categorization into 1. geometric and 2. physical approaches. Geometric approaches rely on purely geometric quantities such as the pullback metric tensor (i.e., rotation-invariant right Cauchy-Green strain); Green strain or edge length; dihedral angles; or a geometric measures of shape distortion [Chen et al. 2013; Kilian et al. 2007; Levi and Gotsman 2015; Martin et al. 2011; Winkler et al. 2010]. Our approach follows variational shape interpolation, based on physical quantities [Chao et al. 2010; Heeren et al. 2014, 2012; Rumpf and Wirth 2009; Von-Tycowicz et al. 2015; Wirth et al. 2009, 2011]. Unlike previous approaches in this family, our final method does not require a full correspondence map from users and supports interpolation with topology changes.

Compatible Meshing. As in traditional mesh-based interpolation, our method requires a *compatible* meshing of all example shapes. Shapes with moderate deformation require no more than existing shape modeling techniques [Barbič et al. 2009]. However, shapes with large and exaggerated deformations and/or topological changes need higher-quality meshes adapted for these changes. We introduce new methods here to efficiently generate high-quality compatible meshes for collections of 2D shapes, given only a sparse set of correspondences.

Locally injective mapping has been extensively studied [Baxter et al. 2009b; Lipman 2012; Poranne and Lipman 2014; Schüller et al. 2013; Surazhsky and Gotsman 2004; Weber and Zorin 2014]. Our problem setting is most similar to those investigated by Weber and Zorin [2014]. Here, however, we focus only on planar shapes arising from drawings. To do so we must solve a problem currently unaddressed by previous methods: guaranteeing local injectivity is necessary but not sufficient to obtain stable, consistent and smooth interpolations over extreme shape changes. Locally injective initializations require additional optimization for both mesh quality and mapping quality.

Prior methods on forming compatible meshes [Baxter et al. 2009b; Surazhsky and Gotsman 2004] similarly observe the need for optimizing meshes after initial connectivity is established. However, these prior methods optimize only the quality of each mesh and do not fix mapping distortions between them. Unfortunately, while optimization of just mesh quality can occasionally be sufficient for small deformation interpolations, it is generally insufficient, see e.g., Figure 8, especially for large deformation interpolations where mapping distortion is a critical issue. For this purpose we motivate and introduce *Comesh Optimization* a new mesh optimization process for high-quality compatible meshing that improves both mesh and mapping quality.

3 PLANAR INTERPOLATION

As shown in Figure 3, we begin with a set of drawings, $i \in [1, m]$, in the plane as input. We then provide an interactive tool to extract drawing shapes and to specify desired correspondences for cuts, openings and boundaries. We next apply our Comesh Optimization algorithm to compute compatible, high-quality meshes, each with n vertices $X_i \in \mathbb{R}^{2n}$, that deliver pleasing interpolations (smooth, stable, and consistent) when minimizing our interpolation energy. With these generated meshes in hand, our framework provides user controls to interactively explore inbetween synthesis by interpolating with varied timings, energy parameters, interpolation weights and artwork sequencings.

3.1 Inbetweening via Variational Interpolation

We compute inbetweens by interpolating variationally [Chao et al. 2010; Heeren et al. 2012; Rumpf and Wirth 2009; Wirth et al. 2009] between our m example artwork shapes. We start by defining a potential $W(X, x)$ to measure the energy of deforming shape x away from a reference shape X. Variational interpolation between just *two* shapes, X_1 and X_2, follows the approximate geodesic path [Chao et al. 2010; Heeren et al. 2012] between them as

$$\bar{x}(w) = \underset{x}{\operatorname{argmin}} (1 - w)\ W(X_1, x) + w\ W(X_2, x), \tag{1}$$

with $w \in [0, 1]$.

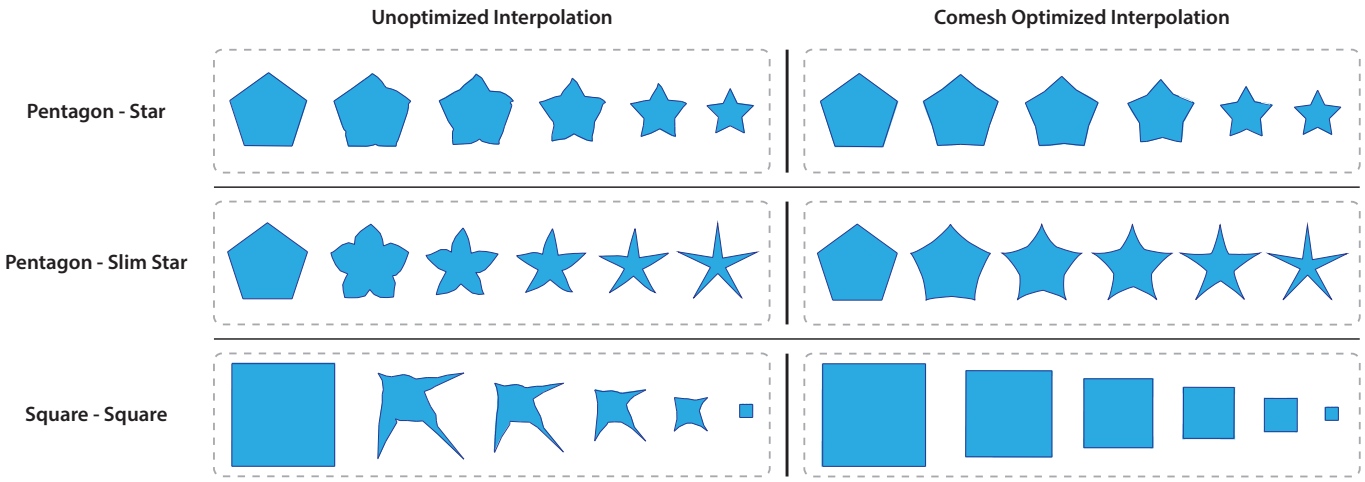


Fig. 4. Interpolated inbetweens generated by LIM parameterization (left column) and Comesh Optimization (right column). As we explore shape variation, e.g., thickness of the star geometry in the two pentagon-to-star examples above, interpolation behavior should vary consistently. Without Comesh Optimization (left) interpolating with an initial LIM parameterization generates unsatisfying animation artifacts, non-smooth interpolation sequences, and inconsistent results for inbetweening. To see these artifacts in animation please see our supplemental videos. After Comesh Optimization (right) both mesh and mapping quality are improved - we obtain low-distortion, smooth and consistent interpolation over varying input shapes; also see our supplemental videos.

Variational interpolation over many shapes is extended by shape averaging [Von-Tycowicz et al. 2015; Wirth et al. 2009]. Constructing a weighted sum of deformation energies,

$$W_I(x, w) = \sum_{i=1}^{m} w_i \, W(X_i, x), \tag{2}$$

we interpolate with the minimization

$$\bar{x}(w) = \underset{x}{\operatorname{argmin}} \, W_I(x, w), \tag{3}$$

varying convex weights $w = (w_1, .., w_m)^T \in \mathbb{R}^m_+$, $\sum_{i=1}^m w_i = 1$. This multishape interpolant is then effectively an equilibrating deformation that balances between weighted contributions of rest shapes. As weights w vary, we move between the influence of multiple shapes to create blended inbetweens.

Inbetweeing. For a set of example artworks $A = \{X_1, .., X_m\}$, a single inbetween is given for each choice of weights $w \in R^m_+$. To build animations, we then can design sequences of inbetweens with animation curves $w(s) \in \mathbb{R}^m_+$, $s \in \mathbb{R}$ applied in $\bar{x}(w(s))$. Keeping just two nonzero entries in w at a time retrieves a traditional keyframe inbetweening, while blending across many nonzero weights at a time establishes complex inbetweens from multiple artworks simultaneously. See Section 6 for details.

Extending Variational Interpolation. Variational interpolation promises a powerful approach for freeform inbetweening with many shapes. However, methods based on variational interpolation have so far been generally restricted to shape spaces with a limited range of expression. Shapes used in these interpolations could neither contradict one another, nor strongly counteract the underlying deformation energy [Martin et al. 2011]; e.g., by extreme deformation, change in topology, or even scaling. See Figure 4 and our supplemental videos for examples.

We address these limitations to enable free-form interpolation between shapes in three steps. First, we introduce here a multimesh structure that ensures a compatible meshing across all shapes and enables annotation of desired correspondences for opening (respectively closing) and cutting (respectively merging) between shapes. Second, we augment the standard elastic measure of deformation in the shape average with additional energies that measure change on boundary: separation/merging and overlap. When minimized in the shape average, these energies now allow interpolation over topology change and, when desired, additionally preserve non-overlap in artworks. Third, we propose a compatible meshing algorithm, *Comesh Optimization*, that critically improves the quality of our multimesh to obtain high-quality, artifact-free interpolations over extreme deformations and topological changes.

3.2 Multimesh

We build our multimesh structure with a single *shared* mesh topology $O = \{V, E, T\}$ of vertices, V, edges, E, and faces, T, and m *differing*

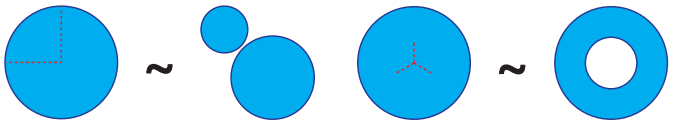


Fig. 5. Non-self-intersecting planar shapes are converted to the same equivalence class, with continuous deformations possible between them, via two topological operations: (a) cutting (respectively merging) and (b) opening (respectively closing). Cut and opening locations are annotated by *cohesive zones* defined in shapes' interior region. Upon discretization, each cohesive zone is a set of cohesive edges.

vertex geometries, $X_i \in \mathbb{R}^{2n}$, cohesive edge sets, $C_i \subset E \times E$, and boundary vertex sets, $B_i \subseteq V$ for all example shapes $i \in [1, m]$. Thus, for example, vertex $v_k \in V$ will have different positions $X_i^k = X_i(v_k)$ and $X_j^k = X_j(v_k) \in \mathbb{R}^2$ in shapes i and j respectively.

To enable topology change we introduce *cohesive edge* sets, C_i, per example shape. These are sets of cohesive edges, C_i^j: edge-edge pairs on the interior (respectively boundary) of each example shape that indicate edge-pair correspondences where a shape will open (respectively close) or cut (respectively merge) for interpolation between topologically *inequivalent* shapes, see Figure 5.

Currently, in our framework, cohesive edge sets are created by artists drawing their desired correspondences. As we will see this enables the exploration of an expressive range of interpolation. Looking ahead we have also designed our multimesh structure and interpolation so that it will similarly function if correspondences are alternately discovered by other interactive or automatic workflows.

Once designated, cohesive edges $C_i^j = \{\{v_a, v_b\}, \{v_c, v_d\}\}$ match generally non-adjacent edges (X_i^a, X_i^b) and (X_i^c, X_i^d) for correspondence in shape i; see Figure 6. These cohesive-zone edge-pairs are then pulled towards and separated away from one another in interpolation via a cohesive energy we construct in Section 5. For each shape i, we then likewise gather all vertices on its boundary into a boundary vertex set $B_i \subseteq V$; we use these vertices to prevent artwork overlap, when desired, during interpolation.

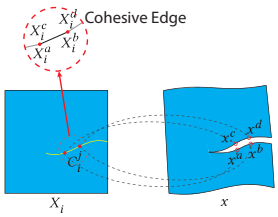


Fig. 6. A cohesive edge, C_i^j, is a pair of collocated edges which can separate apart under deformation. The induced displacement field in X_i will be discontinuous over C_i^j.

We then have multiple compatible boundaries of non-simply connected planar shapes (multiply connected and disjoint). Compatible piecewise linear maps, i.e. triangle meshes, could then be intuitively established in three ways: 1. triangulate one shape and map it to the others [Lipman 2012; Schüller et al. 2013]; 2. compatibly triangulate multiple shapes simultaneously [Baxter et al. 2009b; Surazhsky and Gotsman 2004]; or 3. triangulate each shape separately and then reparameterize them compatibly [Weber and Zorin 2014].

Here we adopt the first approach and use *Triangle* [Shewchuk 2005] to generate a conforming Delaunay triangulation of one shape first. As this meshing step may insert Steiner points along the boundaries, we then upsample boundaries of the other shapes and keep any splits in cohesive edges consistent. Finally, we adopt Schueller et al.'s LIM algorithm [2013] to establish locally injective mappings of the triangle mesh to all other shapes. We note that this approach does not guarantee an initializer for arbitrary cases; we could, for instance, follow the third approach as our workflow can start with results produced by any of the above three strategies. However, in our experiments and for our examples, we have so far found strategy 1 with LIM to be most effective to bootstrap our multimesh construction. For further details on the creation of our multimesh structure with correspondences we refer the interested reader to our supplemental material.

In Section 5 we will construct and detail our deformation energy over meshes, cohesive energy over cohesive edges, and non-overlap energy over boundary vertices. However, the resulting interpolation we will obtain using the default multimesh will lead to unsatisfying animation artifacts during interpolation (see e.g., Figure 4) due to, as we will soon see, both poor intra-mesh element quality and poor inter-mesh mapping quality. In Section 4, we thus first introduce our Comesh Optimization algorithm to improve both the mesh and mapping quality of our multimesh structure to resolve these interpolation artifact problems for freeform shape collections. Here and in the following sections we adopt the notational conventions summarized in Table 1.

Table 1. Notation

Notation	Definition
V	shared mesh vertex set
E	shared mesh edge set
T	shared mesh face set
$n = \|V\|$	number of vertices
$O = \{V, E, T\}$	shared mesh topology
$X_i \in \mathbb{R}^{2n}$	ith example-mesh's vertex positions
m	number of example shapes
$A = \{X_1, .., X_m\}$	set of example-shape meshes
$x \in \mathbb{R}^{2n}$	deformed-mesh vertex positions
$w \in \mathbb{R}_+^m$	interpolation weights
$s \in \mathbb{R}$	timing parameter
$w(\cdot) : \mathbb{R} \to \mathbb{R}_+^m$	animation curve
$\bar{x}(\cdot) : \mathbb{R}_+^m \to \mathbb{R}^{2n}$	variational interpolation
$a_k(\cdot) : \mathbb{R}^{2n} \to \mathbb{R}$	signed area of triangle $t_k \in O$ in input mesh
$C_i \subset E \times E$	cohesive edge set for ith mesh
$B_i \subseteq V$	boundary vertex set for ith mesh
$X_i^j = X_i(v_j) \in \mathbb{R}^2$	vertex $v_j \in V$ in mesh X_i
$x^j = x(v_j) \in \mathbb{R}^2$	vertex $v_j \in V$ in deformed mesh x

4 COMESH OPTIMIZATION

Artifact-free interpolation over changing shapes requires high-quality mappings between well-formed, compatible meshes. *Mesh quality*, per shape, is essential for the stability of minimizations performed on discretized energies for variational interpolation. *Mapping quality*, between shapes, ensures local similarity for high-quality visual interpolations. In this section we propose a *Comesh Optimization* method that jointly improves *both* mesh and mapping quality of initial, inversion-free multimeshes. Our resulting optimized multimeshes then give smooth, stable and consistent interpolations that are visually pleasing over extreme changes in shape.

Goal. Given multiple, compatibly parameterized boundaries of non-overlapping planar shapes we seek meshes with 1. individually well formed elements and 2. bi-directionally *optimal* piecewise-linear maps between all shape pairs.

Input. We bootstrap this process, as described above in Section 3, with an initial, locally injective parameterization of shapes annotated with correspondences. We apply Schüller et al's [2013] LIM algorithm to obtain a single mesh topology, O, for m differing shapes with varying vertex positions, X_i, $i \in [1, m]$. This common connectivity delivers dense mappings from the specified correspondences defined in Section 3.2.

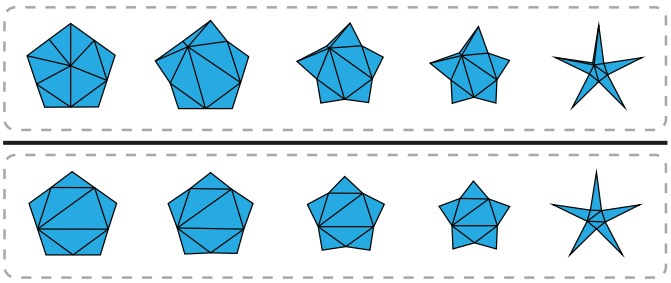


Fig. 7. A low resolution example highlights how initial, locally injective, compatible meshes of example shapes (pentagon and star on far left and right) will produce poor interpolation results (top); note sliver triangles and large distortion maps between triangles. Comesh Optimization of this initial mesh then generates pleasing interpolations (bottom) after improving triangle quality and mapping distortion of the meshes.

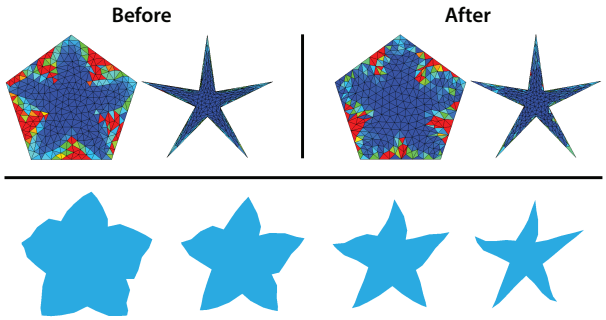


Fig. 8. Optimizing only the mesh quality of example shapes and ignoring map distortion as in [Baxter et al. 2009b; Surazhsky and Gotsman 2004] improves triangle shape but leaves distortion (colormap) largely unimproved (top right). In turn, resulting variational interpolations in between the shapes (bottom row) then suffers from symmetry breaking, inconsistency and instabilities. Compare with our Comesh Optimized results in Figure 4.

Other options for bootstrapping to an initial connectivity are also possible [Baxter et al. 2009b; Surazhsky and Gotsman 2004; Weber and Zorin 2014]. However, while initial meshes found by these alternate approaches are also often locally injective, they too have no guarantee of quality. In practice output meshes from all such methods, including LIM, require additional optimization to avoid interpolation artifacts; see e.g., Figures 4, 7 and 8.

We have so far found LIM to most consistently deliver good initializations for our approach. Nevertheless, on its own, LIM produces unacceptable artifacts upon interpolation, see e.g., Figures 4 and 7 and our supplemental videos. This motivates our Comesh Optimization algorithm for improving the quality of both mesh element shapes and inter-mesh mapping.

4.1 Mesh and Mapping Energies

We measure the *mesh quality of each shape i's triangulation* with Alliez et al.'s [2005] *Optimal Delaunay Triangulation* (ODT) energy,

$$\mathcal{D}(X_i, O) = \frac{1}{4} \sum_{v_j \in V} \|X_i^j\|^2 \Gamma_j(X_i) - q_i, \tag{4}$$

and measure the *map distortion from shape i to shape j* with the *Most-Isometric Parameterizations* (MIPS) energy [Hormann and Greiner 2000],

$$\mathcal{E}(X_i, X_j, O) = \sum_{t_k \in T} a_k(X_i) \left(\frac{\sigma_k^1}{\sigma_k^2} + \frac{\sigma_k^2}{\sigma_k^1} \right). \tag{5}$$

For the variational Delaunay ODT energy above, $\Gamma_j(X_i)$ is the one-ring area of vertex X_i^j while q_i is a constant term for fixed-shape boundaries[1]. For the MIPS energy, $a_k(X_i)$ is the area of triangle t_k, in shape X_i, while σ_k^1, σ_k^2 are the first and second singular values of the 2×2 deformation gradient for triangle $t_k \in T$, when treating shape X_j as a deformation away from shape X_i.

[1]In our setting q_i thus has no effect over gradients with respect to interior vertices, but its inclusion does cancel out gradient components due to boundary vertices, see Alliez et al. [2005].

4.2 Optimizing Solely Mesh or Mapping is Insufficient

Comesh Optimization is built from our observations that 1. only optimizing triangle quality per mesh, as done previously, will not prevent high-distortion mappings between meshes; and 2. only minimizing mapping distortion between meshes, irrespective of choice of distortion measure, will not guarantee good-quality triangulations for stable interpolation.

Minimizing solely distortion, irrespective of distortion measure, results in ambiguity—many mesh solutions are possible and they rarely provide well-shaped triangles; see below. We choose to jointly apply the ODT energy and a cyclic sum (detailed below) of MIPS energies to optimize triangle quality and map distortion respectively. For measuring distortion in Comesh Optimization we initially investigated other energies including the isometric energy from Smith and Schaefer [2015] and ARAP [Sorkine and Alexa 2007]. However, they both resist scaling and so fight the exaggerated scaling behaviors we want to allow. In the end we choose conformal MIPS as it limits angular distortion and allows large scale changes while preventing inversion.

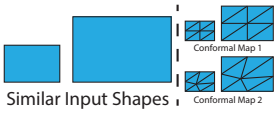


Fig. 9. Ambiguity in optimal, low-distortion, piecewise-linear maps: here both maps minimize the same distortion measure (in this case MIPS) yet Map 1 has better mesh quality.

For very similar shapes, e.g., between squares of similar orientation and size, if we optimize just mesh quality alone, we will sometimes also obtain a close-to-optimal, low-distortion map. However, for a general collection of *different* shapes, optimizing solely mesh quality will produce a set of meshes that are individually well formed yet *work poorly for interpolation* due to large map distortions between them. See for example Figure 10 (ODT), where we optimize solely mesh quality with the ODT energy and so still obtain high distortion; correspondingly, in Figure 8 we see that optimizing just mesh quality introduces unsightly artifacts during interpolation.

On the other hand, if we optimize only map quality, e.g., here with the MIPS energy, our problem is ill defined. Consider Figure 9 where both maps are conformal yet they present different *meshes*. There exists ambiguity in optimally low-distortion piecewise-linear maps

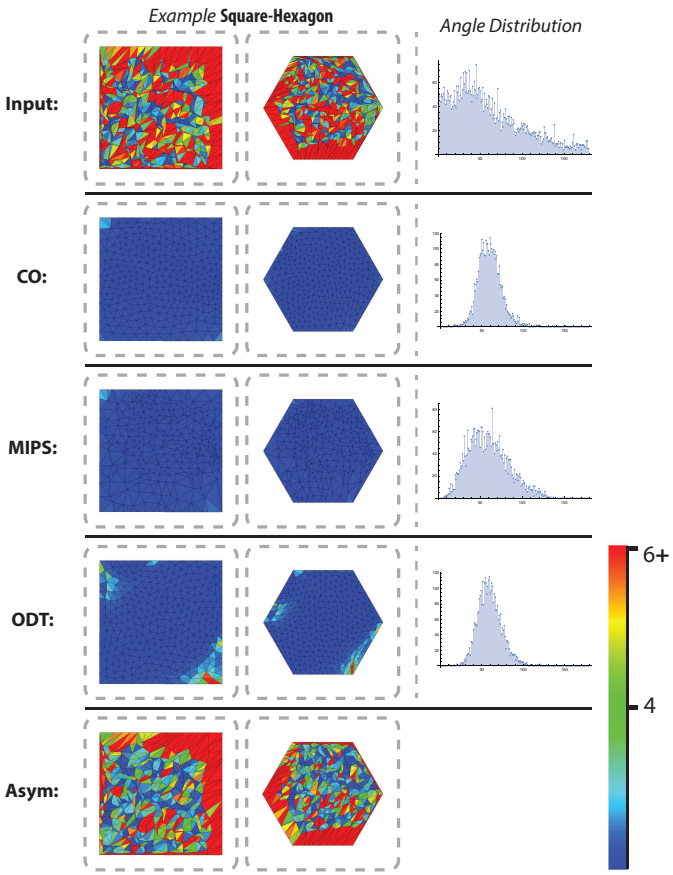


Fig. 10. Here we visualize the change in map quality (colormap) and mesh quality (angle distribution) for a range of multimesh optimization alternatives. Minimizing only MIPS accounts for only distortion and so still obtains poor quality elements (MIPS) while a solution minimizing ODT considers solely mesh quality and so obtains poor mapping quality (ODT). Likewise, minimizing distortion asymmetrically, in this case from just square to hexagon, will still produce poor solutions as in (Asym); compare to the (MIPS) solution that minimizes distortion symmetrically. Our Comesh Optimization seeks a symmetric solution optimizing both per-shape mesh quality and all shape-pair mappings and so obtains both low distortion and well-formed triangle elements (CO).

between shapes and so, unless we are exceedingly lucky, optimizing for map quality alone will not give us well-formed meshes for pleasing interpolation. For example, in Figure 10 (MIPS) optimizing with the MIPS energy we obtain poor-quality meshes even though we generate a low-distortion map.

These issues only worsen as we consider general, multishape interpolations between $m > 2$ shapes and so we design our Comesh Optimization method to optimize both mesh quality with ODT and map quality with MIPS. However, to optimize map quality we must somehow efficiently minimize mapping distortion bi-directionally between all shape pairs from our m example shapes. We address this challenge next.

4.3 Multishape MIPS and Symmetry

As we have many shapes to be compatibly meshed, our goal is distinct from many traditional mesh optimizations task. Each example shape is effectively treated as a *rest shape* for measuring distortion energies in variational interpolation and so requires well-formed mesh elements. At the same time, for measuring mapping quality, each mesh is also a deformed shape with respect to all other example shapes in our collection.

In our setting interpolation is thus not a one-way trip from one shape to another. It can take any direction in shape space with any combination of shapes participating and so our mappings should likewise minimize distortion for all possible paths between shapes. Consider again the simple square-to-hexagon example in Figure 10 where square and hexagon respectively have vertex positions X_{sq} and X_{he}. Following standard distortion minimization we first try optimizing over our target hexagon's vertices, $\min_{X_{he}} \mathcal{E}(X_{sq}, X_{he}, O)$. This, however, gives us the "Asym" solution in Figure 10 where we see that distortion after this optimization remains poor. Why does this happen? The full degrees of freedom in this problem have been ignored: we can and should optimize distortion in all directions by treating each shape as both a *rest* and *deformed* shape when we optimize total mapping quality.

Cyclic Summation. Hence we seek *symmetric*, optimally low distortion maps for both $X_i \rightarrow X_j$ and $X_j \rightarrow X_i$ $\forall i, j \in [1, m]$. For square-to-hexagon this simply amounts to minimizing the sum of $\mathcal{E}(X_{sq}, X_{he}, O)$ and $\mathcal{E}(X_{he}, X_{sq}, O)$ giving us the low distortion solution in the "MIPS" row of Figure 10.

More generally we seek to minimize all possible distortions between example pairs. While this is straightforward to do for two example shapes as above, optimizing this explicitly as an objective over all pairwise distortion measures between m example shapes is prohibitively expensive.

We side-step this potential computational obstruction by observing that MIPS is transitive: if the conformal distortion between X_i and X_j and between X_j and X_k are both small (bounded above), then the conformal distortion between X_i and X_k is also small (bounded above). See our supplemental material for proof and details. This allows us to construct our symmetric, all-pairs distortion energy with just a *cyclic sum* of the MIPS energy over our m example shapes as

$$\sum_{cyc}^{m} \mathcal{E}(X_i, X_j, O) = \mathcal{E}(X_1, X_2, O) + \mathcal{E}(X_2, X_3, O) + \cdots + \mathcal{E}(X_{m-1}, X_m, O) + \mathcal{E}(X_m, X_1, O). \tag{6}$$

In practice we observe stable traversals of the shape space when optimizing with our cyclic MIPS energy, ill-posed results without it, and also confirm in experiment that we obtain the same interpolation behavior even as we vary the order of example shapes in the cyclic summation; see our supplemental videos for details.

4.4 Comeshing Constraints

We have so far focused on constructing a suitable objective for optimizing our multimesh. To maintain prescribed correspondences, i.e., points and cohesive zones, and to maintain consistent mesh topology we must also impose *constraints* on our optimization. We begin

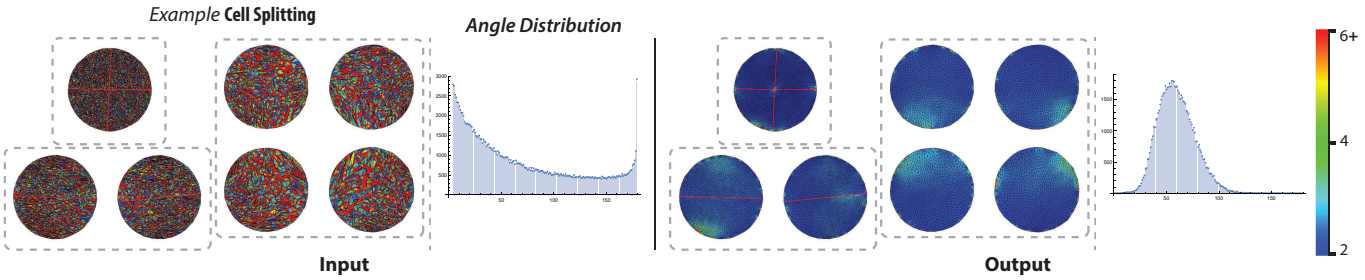


Fig. 11. Comesh optimization improves both map and mesh quality of our multimeshes subject to position and topology constraints. Here we visualize the change in map quality (colormap) and mesh quality (angle distribution) for our cell splitting example. Designated cohesive edges (red) are preserved by both vertex improvement and topology update steps.

by constraining feature points on boundaries and correspondences that can be specified as positional constraints on vertex positions per shape, $\mathcal{P}(\cdot) = 0$. To preserve multimesh structure we *implicitly* constrain mesh topology O to remain consistent across all meshes. We then enforce local injectivity, $a_k(\cdot) > 0$, explicitly so that topology optimization steps (e.g., edge flips and edge collapses, see below) do not cause our inter-mesh mapping energies, (6), to go towards infinity as area becomes small.

Finally, to maintain pre-defined cohesive zones, we constrain edge topology and vertex positions to stay equivalent on both sides of all cohesive zones. Recall that each cohesive edge C_i^j corresponds to two edges in the edge set E, with respective end vertices, $\{X_i^a, X_i^b\}$ and $\{X_i^c, X_i^d\}$ in shape i. We constrain all such pairs to be collocated, during all topological steps with the constraint $\mathcal{Z}(\cdot,\cdot) = 0$, requiring

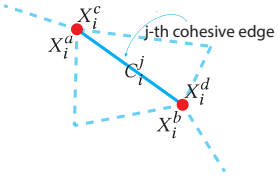


Fig. 12. We require the two pairs of end vertices on a cohesive edge to be collocated.

$$X_i^a = X_i^c, \quad X_i^b = X_i^d. \tag{7}$$

See Figures 11, 12 and 13.

4.5 Comesh Optimization

In summary our Comesh Optimization seeks a piecewise-linear map that jointly minimizes our cyclic MIPS and area-scaled ODT energies to find a compatible Delaunay triangulation satisfying position, cohesive zone, and local injectivity constraints,

$$\begin{aligned} \min_{X_1, X_2, \cdots, X_m, O} \; & \sum_{i=1}^{m} \frac{\mathcal{D}(X_i, O)}{\sum_{t_k \in T} a_k(X_i)} + \sum_{cyc}^{m} \mathcal{E}(X_i, X_j, O) \\ s.t. \quad & \mathcal{P}(X_i) = 0, \quad \forall i \in [1, m], \\ & a_k(X_i) > 0, \quad \forall t_k \in T \text{ and } i \in [1, m], \\ & \mathcal{Z}(C_i, X_i) = 0, \quad \forall i \in [1, m]. \end{aligned} \tag{8}$$

Here note that we normalize each shape i's ODT energy by its area $\sum_{t_k \in T} a_k(X_i)$. This scales the energies $\mathcal{D}$ and $\mathcal{E}$ in our objective to common units of squared length.

A note on relative weighting. Comesh Optimization minimizes a multi-objective function formed by the sum of the cyclic MIPS energies for mapping and the area-scaled ODT energies for meshing. Note that we choose to scale both terms equally in our objective above in (8). We base this choice on our prior observation that the space of conformal piece-wise linear mappings is generally ambiguous with respect to mesh quality. This, in turn, suggests that we can find a wide range of conformal mappings with differing mesh qualities. To better understand how the relative weighting of these two terms, mapping and meshing, can actually vary in practice in our results, we experimented with changing their relative weights in our optimizations. For all combinations of finite, *nonzero* weights we observe qualitatively and quantitatively (see our supplemental material) similar results from our Comesh Optimization and so keep them equally weighted in our objective.

4.6 Solving Comesh Optimization

To solve Comesh Optimization we construct an algorithm that minimizes our constrained problem (8) with block, cyclic descent over example shapes $[1, m]$. Our algorithm is detailed in Algorithm 1. Our

ALGORITHM 1: *Comesh Optimization*

Input: multimesh data structure O, $\{X_i\}$, $\{C_i\}$, $\{B_i\}$, $i \in \{1, \cdots, m\}$.
Output: O, $\{X_i\}$, $\{C_i\}$, $\{B_i\}$.
Initialize: `max_outer_itr` = 500, `max_inner_itr` = 100,
`outer_tol` = 10^{-6}, `inner_tol` = 10^{-7},
$[\cdot] : \mathbb{Z} \to \mathbb{Z}/m\mathbb{Z}$.

```
for loop_outer = 0 to max_outer_itr
  total_res = 0
  for i = 1 to m
    inner_res = 0
    for loop_inner = 0 to max_inner_itr
      edge_flip(X_i, O, C_i, B_i)
      edge_split(X_i, O, C_i, B_i)
      sub_res = gradient_descent_step(X_[i-1], X_i, X_[i+1])
      edge_collapse(X_i, O, C_i, B_i)
      if inner_res < sub_res
        inner_res = sub_res
      end if
      if sub_res < inner_tol break
    end for
    total_res = total_res + inner_res
  end for
  if total_res < outer_tol break
end for
```

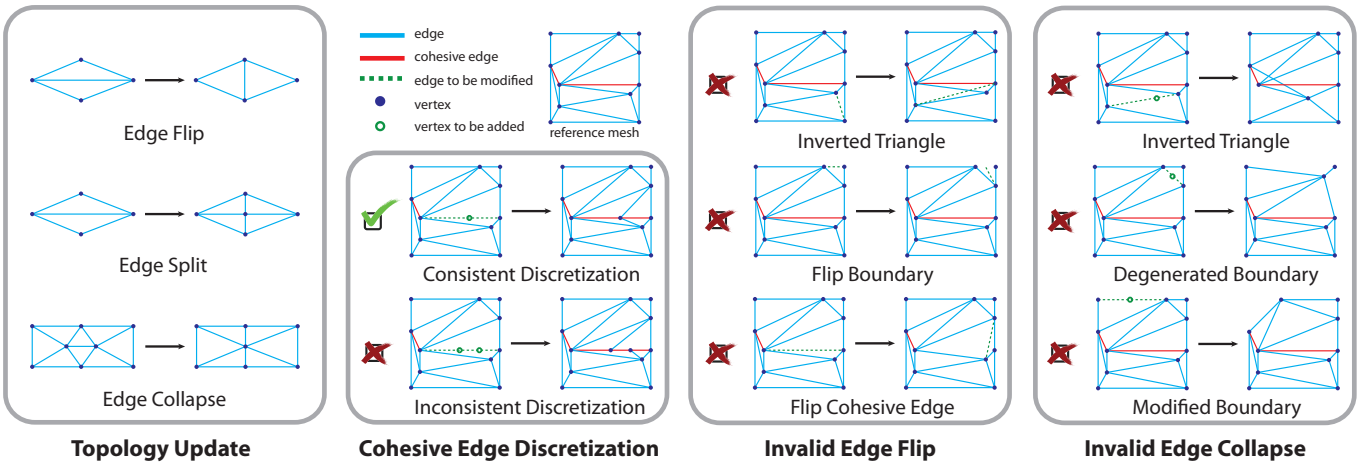


Fig. 13. We apply three local operations to update mesh topology: edge flips, edge splits and edge collapses. These operations are constrained in order to satisfy our Comesh Optimization constraints. Each operation is applied to the multimesh topology shared by all example shape meshes. It propagates to all meshes (or none) to maintain consistent topology across all shapes.

outer loop iterates block-wise, by shape. For each outer iterate on shape $i \in [1, m]$ we solve for an update of shape i's vertex positions X_i and shared multimesh topology O, while holding all other X_j, $j \neq i$ fixed, subject to our compatibility and feasibility constraints. Then, within each inner iteration on shape i, we optimize mesh X_i and topology O applying one step of vertex optimization with *gradient descent*, and three steps of mesh optimization and refinement applying *edge flips*, *edge splits* and *edge collapses*. All four operations are constructed to preserve local mesh constraints and maintain compatible meshing across all m shapes.

Each *vertex optimization* step improves current mesh coordinates by applying a single step of projected *gradient descent* on vertices X_i. The gradient of our objective in (8) with respect to shape, X_i, is $\nabla_{X_i}[\frac{\mathcal{D}(X_i,O)}{\sum_{t_k \in T} a_k(X_i)} + \mathcal{E}(X_i, X_{i+1}, O) + \mathcal{E}(X_{i-1}, X_i, O)]$. For a detailed derivation of this gradient see our supplement. Each gradient descent step applies bisection correction, per vertex, to find a conservative improvement along the gradient that both respects our constraints on vertex position and does not invert triangles in the one-ring neighborhood.

Mesh optimization and refinement applies *edge flips*, *edge splits* and *edge collapses*. *Edge flips* improve mesh connectivity. For each candidate edge, we accept a flip when it both decreases the Delaunay objective over all meshes and respects our constraints in (8). We alternate between evaluating feasibility and evaluating energy decrease. For our feasibility evaluation we check 1. if the flip candidate is a boundary or cohesive edge; and 2. if flipping leads to an inversion and so violates local injectivity (see Figure 13). If either feasibility condition is violated we do not consider the candidate flip further. If, however, the flip is *feasible* we evaluate energy decrease by checking the largest angle in the edge-adjacent triangle pair both before and after the candidate flip is performed over all triangulations. If this largest angle decreases across meshes we flip the candidate edge to improve local mesh quality.

We apply *edge split* and *edge collapse* operations to bound edge lengths and angles to reasonable ranges while preserving our constraints. For each candidate edge, we consider an operation when it is 1. *feasible* with respect to our constraints in (8) 2. out of relative bounds over all shape meshes, and 3. will locally improve the mesh. For our feasibility evaluation we check 1. if the operation leads to an inversion and so violates local injectivity, 2. if the operation would delete a feature point (e.g., ruling out collapse of sharp corners), and 3. if the operation would change the boundary topology (e.g., ruling out collapse of internal edges with both endpoints on the boundary). If any of these feasibility conditions is violated we do not consider the candidate operation further. Feasible edge splits are then applied when the edges relative length is too long in all meshes and the operation would improve the worst edge length and local angles in the immediate neighborhood over all meshes. In turn, feasible edge collapses are applied when the edges relative length is too short in all meshes or if its opposite angles in incident triangles are too small. Please see Figure 13 for illustration and Section 7 for additional evaluation of our Comesh Optimization solver.

5 INTERPOLATION ENERGIES

With the well-formed multimeshes generated by our Comesh Optimization we can now apply physically-motivated quantities as metrics for smooth, stable, and consistent interpolation over extreme changes in shape. We apply an *elastic energy*, $W_D(X_i, x)$, to measure deformation of shape x away from each example shape X_i, and introduce two new energies for interpolation, defined on shape boundaries. We construct a *cohesive energy*, $W_C(C_i, x)$ to achieve topology changes for deformed shape x across example shape cohesive edges C_i, and a *non-overlap* energy, $W_O(B_i, x)$, to avoid overlap of example shape boundaries B_i during variational interpolation, when desired. With these new energies shapes can now also separate, merge together, or even be prevented from overlapping during variational interpolation; see e.g., Figures 1, 14 and 16.

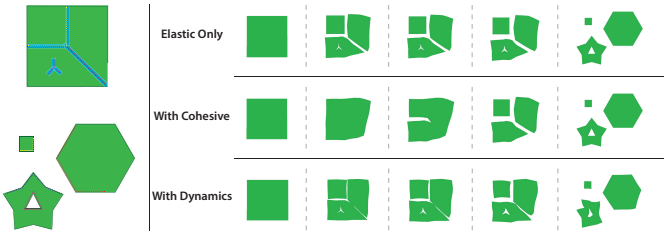


Fig. 14. Our cohesive energy integrates over cohesive zones (blue) that, together with sparse correspondences between shapes, resolve mapping ambiguity to guide interpolation. Our cohesive energy and dynamic blending enable adhesion and separation for topology change as well as secondary physical effects in shape interpolation.

To interpolate between shapes X_i, $i \in [1, m]$, with topology changes and non-overlap, our full interpolation energy is

$$W_I(x, w) = \sum_{i=1}^{m} w_i \Big[W_C(C_i, x) + W_O(B_i, x) + W_D(X_i, x) \Big], \quad (9)$$

with interpolation weights $w = (w_1, ..., w_m)^T \in \mathbb{R}_+^m$. Note that enforcing non-overlap is optional: when we wish *to allow* overlap in shape x, we simply omit the energy $W_O(\cdot, \cdot)$ in Equation (9); see Figure 16. To find an interpolated shape $\bar{x}(w)$ we solve (3) with interpolation energy (9).

5.1 Cohesive Energy

We introduce a new quadratic potential model, a cohesive energy, W_C, that pulls cuts and holes open and closed as we near example keyframes with differing topology in our interpolation.

Recall that *cohesive edge* sets, C_i, are collections of edge-edge pairs. Each pair in shape i, given by $C_i^j = \{\{v_a, v_b\}, \{v_c, v_d\}\}$ matches generally non-adjacent edges (X_i^a, X_i^b) and (X_i^c, X_i^d) for correspondence. They are created on the interior and boundary of example shapes to indicate edge-pair correspondences where an interpolated deformation will open as it moves towards example shapes where edges are separated, and to merge together as the interpolation moves towards example shapes where cohesive edge-pairs are colocated. To do this we augment the standard elastic measure of deformation in the shape average, W_I, with an additional cohesive energy, $W_C(C_i, x)$ for each example shape i, with cohesive edge set, C_i.

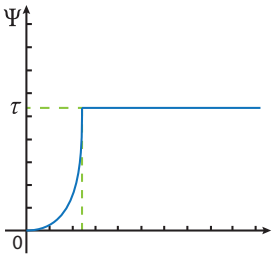


Fig. 15. We measure displacement jump over cohesive edges using a clamped quadratic to model cohesive energy.

Unlike an elastic potential, which integrates over the entire domain, our cohesive energy integrates solely along cohesive-zones. Thus each cohesive energy $W_C(C_i, x)$ integrates across the corresponding cohesive edge set C_i of example shape i. Figure 14 illustrates cohesive zones (blue) needed to separate individual shape (yellow and red; red and purple; yellow and purple) or to open a hole in the interior (pink). Our cohesive potential measures the displacement difference with a quadratic form to enable efficient optimization with a local-global solver; see Section 7.

To construct W_C we start by defining a quadratic measure of *separation* on cohesive edges

$$d(C_i^j, x) = \lambda \begin{bmatrix} x^a - x^c \\ x^b - x^d \end{bmatrix}^T \begin{bmatrix} P & \frac{1}{2}P \\ \frac{1}{2}P & P \end{bmatrix} \begin{bmatrix} x^a - x^c \\ x^b - x^d \end{bmatrix}. \quad (10)$$

Here $\lambda \in \mathbb{R}_+$ enables control of cohesive stiffness and P is an $\mathbb{R}^{2\times 2}$ matrix that provides control over topology change behavior to bias towards greater resistance to shear or stretch. Minimizing separation distance alone draws cohesive-zone edge-pairs together. To enable separation as deformation grows we then construct our cohesive energy by clamping the quadratic, d, to an controlled maximum threshold τ; see Figure 15. Our cohesive energy is then

$$W_C(C_i, x) = \sum_{C_i^j \in C_i} \min\Big(\frac{1}{3} l_i^j d(C_i^j, x),\ \tau \Big). \quad (11)$$

When W_C is minimized in the shape average (9), cohesive-zone edge-pairs are drawn towards each other when our distance measure is below τ, and separate from one another as the threshold is exceeded. Here the multiplier $\frac{1}{3} l_i^j$ is obtained by piecewise-linear discretization of our energy (see our supplement for details) along cohesive edges C_i^j in shape X_i, with

$$l_i^j = |X_i^a - X_i^b| = |X_i^c - X_i^d|. \quad (12)$$

Our shear and stretch control matrix P are then defined by

$$P = \alpha \ \mathrm{Id} + (1 - 2\alpha)\ \hat{x}\hat{x}^T \in \mathbb{R}^{2\times 2}, \text{ with}$$
$$\hat{x} = \frac{x^a + x^c - x^b - x^d}{\|x^a + x^c - x^b - x^d\|}, \quad (13)$$

and $\alpha \in [0, 1]$ giving weighted control of the relative contributions of shearing to stretching in the cohesive energy. Shearing effects becoming more significant as we increase α. Here Id is the 2×2 identity matrix and $\hat{x}$ is the edge-pair averaged direction defined on deformed shape x. Note that our cohesive energy then models both separation and merging behaviors. Separated edge pairs that come close enough to reduce the displacement field difference below thresholds τ are once again drawn together to merge.

5.2 Non-overlap and Deformation Energies

When interpolating it is often desirable to additionally avoid overlapping shapes. For this we need to ensure that boundary vertices B_i do not penetrate the interior of example shape i in deformed shape, x. To do this we construct our non-overlap energy

$$W_O(B_i, x) = \sum_{v_j \in B_i} \mathcal{I}\big(B_i, x(v_j), x\big), \quad (14)$$

with the indicator function

$$\mathcal{I}\big(B_i, x^j, x\big) = \begin{cases} \infty, & \text{if } x^j \in \Omega(x, T, B_i), \\ 0, & \text{else.} \end{cases} \quad (15)$$

Here $\Omega(x, T, B_i)$ gives the interior domain of our mesh at x, composed by the cover of deformed triangles in T, excluding boundary edges. The indicator function, $\mathcal{I}\big(B_i, x^j, x\big)$, then correspondingly evaluates to an extreme penalty of ∞ if a boundary vertex x^j is inside any

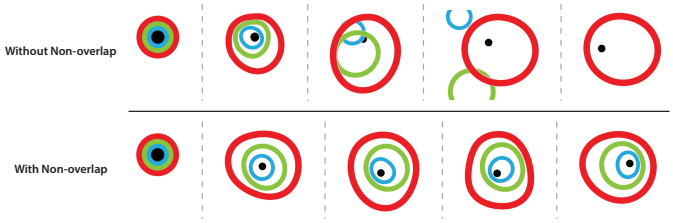


Fig. 16. Applying our non-overlap energy enables variational interpolation to preserve the concentric layout of this circles-in-circles example (bottom). Interpolation *without* our non-overlap energy ignores overlap and so interpolation will then break the concentric layout (top).

triangle in the deformed configuration x. While standard minimization techniques can be challenged by such nonsmooth objectives, within the local-global framework we employ, see Section 7, we can enforce our nonoverlap potential by direct projection in the local solve step [Bouaziz et al. 2014].

Deformation Energy. Finally, To measure *internal* deformation, in this work, we use the discrete, triangle-based as-rigid-as-possible (ARAP) energy [Chao et al. 2010; Liu et al. 2008; Sorkine and Alexa 2007] ,

$$W_D(X_i, x) = \sum_{t_k \in T} \kappa_k a_k(X_i) \|F_k(X_i, x) - R_k(X_i, x)\|_F^2. \tag{16}$$

with $F_k(X_i, x)$ retrieving the deformation gradient of triangle $t_k \in T$ with respect to deformed shape x and rest shape X_i. Here R_k is then the projection of F_k onto rotations and κ_k is the local material stiffness weighted per triangle, $t_k \in T$, for fine-grained control of local resistance to deformation.

6 ANIMATING INBETWEENS

We have so far shown how to interpolate inbetweens from sets of arbitrarily drawn shapes. To build animations we can design sequences of inbetweens with animation curves $w(s) \in \mathbb{R}_+^m$, $s \in \mathbb{R}$. Our animation function is then $\bar{x}(w(s)) = \text{argmin}_x\, W_I(x, w(s))$. Varying $w(s)$ with s gives us control and exploration of timing and spacing, e.g., ease-in and ease-out, while explicitly treating s as a time variable enables the addition of physics-based dynamic effects to our inbetweens.

6.1 Dynamic Effects for Variational Interpolation

Subdividing the animation curve $w(s)$, $s \in [0, S]$ into sequential frames $\{\bar{x}(w(0)), ..., \bar{x}(w(s)),..., \bar{x}(w(S))\}$, we construct a discrete velocity for each frame with finite differencing as $\dot{x}(s) = \frac{1}{h_s}\left(\bar{x}(w(s)) - \bar{x}(w(s-1))\right)$ where h_s is the length of the desired time interval between frames s and $s-1$.

A new, time-stepped position for frame s due *only* to dynamics and thus *without any interpolation* is then

$$x_p(s) = \bar{x}(w(s-1)) + h_s \dot{x}(s-1) + h_s^2 M^{-1} f(s). \tag{17}$$

Here $M = \rho I \in \mathbb{R}^{2n \times 2n}$ is a lumped mass matrix with material density ρ while $f(s)$ adds user specified forces, e.g., gravity, wind, damping and so forth, at time s.

To blend dynamic effects into our interpolation we then construct an additional energy that biases towards a dynamics step for frame s, with a blending parameter $\xi \in \mathbb{R}_+$,

$$W_T(x, \xi) = \frac{\xi}{2h_s^2}(x - x_p(s))^T M (x - x_p(s)). \tag{18}$$

Minimized on its own, $W_T(x, \xi)$ with respect to x this energy applies a forward time step of dynamic motion and ignores interpolation altogether.

To inbetween *with dynamic effects* from frame $s-1$ to frame s, we then compose W_T with the variational average to build a blending between variational interpolation and dynamics effects

$$\bar{x}(w(s), \xi(s)) = \underset{x}{\text{argmin}}\, W_I(x, w(s)) + W_T(x, \xi(s)). \tag{19}$$

Here, the time varying blending parameter $\xi(s)$ balances between interpolated shapes and dynamics-driven deformation. When we set $\xi(s) = 0$, inbetweening returns us shapes $\bar{x}$ that reproduce interpolated frames while, as $\xi(s)$ grows, dynamics pushes us farther from artist-designed shapes.

While some artists may prefer to directly control the blending parameter $\xi(s)$, we also provide a blending controller that automates the process of balancing between dynamic effects and hitting keyframes in example shapes. Effectively, as the difference between the dynamics and and keyframe shapes grow, our controller adaptively drives down $\xi(s)$. Our controller then allows greater physics-driven deformation as it more coincides with drawn keyframe shapes. Artist-level control is then simplified to a single guiding compliance term $\epsilon \in [0, 1]$ that indicates how tightly inbetweening should follow keyframe shapes. See our supplemental materials for details.

7 EXPERIMENTS AND EVALUATIONS

In this section we evaluate the performance of our Comesh Optimization solver; discuss and demonstrate our workflow for creating animations from examples; and present statistics and examples from animation synthesis with our methods. For Comeshing we examine our solver's scaling behavior as we increase mesh resolution, add more example shapes, and as we increase problem difficulty. For animation applications, we examine our method by recreating the benchmark animation tasks from *The 12 Basic Principles of Animation* (see our supplemental) and on a range of challenging animation examples with timing statistics.

7.1 Implementation

All evaluations and experiments are performed on a four-core Intel 3.50GHz CPU with our C++ implementation of the system. Our implementation stores the input multimesh data in an augmented half-edge data structure that we use throughout comeshing and interpolation.

For Comesh Optimization we adopt the fixed parameters in Algorithm 1 for all examples. That is `max_outer_itr` and `max_inner_itr` are set to 500 and 100 respectively, while `outer_tol` and `inner_tol` are set to 10^{-6} and 10^{-7} respectively. For our mesh optimization and refinement we set angles and relative edge length bounds following Brochu et al. [2009].

For interpolation we apply local-global iterations [Bouaziz et al. 2014; Liu et al. 2008] to compute inbetweened animation frames. Our stopping criteria for the solver is set to terminate when the inf-norm of our relative residual is less than 10^{-3}. In our examples we set our cohesive potential stiffness $\lambda = 4 \times 10^3$ while we vary τ (see our

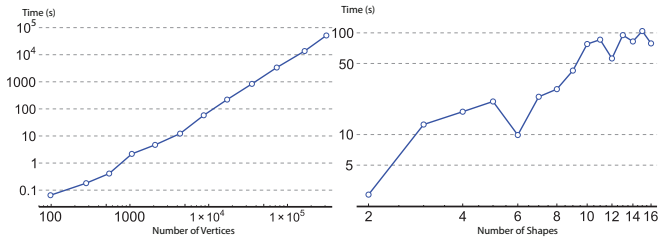
supplemental videos for examples exploring variation with this term). For dynamics we need not explicitly set timesteps, h, material stiffnesses, κ, nor material density, ρ, as our dynamic blending model re-parameterizes these effects with the single user compliance term ϵ; see our supplement for details and Table 3 for parameters used to finalize each animation example. Numerical entries for quadratic terms in the global matrix of our interpolation solve, i.e., P in our cohesive energy (11), are updated in each global step. We then correspondingly update our linear system and numerically factorize once per iteration. Finally, in each local step, along with standard ARAP projections [Bouaziz et al. 2014], wherever non-overlap energies are active intersections are simply back-projected to the closest feasible point while, for the cohesive energy, edge-pair directions $\hat{x}$ are updated.

7.2 Comeshing

Mesh Resolution. We first investigate the scaling of our Comesh Optimization solver's performance as we increase problem size in terms of number of vertices. In Figure 17 (a) we test the scaling of our Comesh Optimization solver with square-to-hexagon examples increasing vertex count up to well over 300K vertices and see that our solver scales efficiently. Statistics for all examples in the plot are fully detailed in our supplemental material. An example of which, at a mesh resolution of over 17K vertices, is shown in Figure 20 left. Here our unoptimized solver converges in under 4 minutes while achieving a low-distortion mapping with high-quality elements.

Number of Example Shapes. We next investigate scaling behavior of our Comesh Optimization solver as we increase problem size in terms of the number of provided example shapes In Figure 17 (b) we see that the Comesh solver also scales efficiently as we increase the number of input shapes from 2 to 16 for our walking character example in Figure 20, right. As detailed in the supplemental material, the solver remains convergent in under 2 minutes, while achieving a low-distortion mapping with high-quality elements, see Figure 20 right.

Problem Difficulty. We then investigate the performance of our Comesh solver as we increase *problem complexity*. In Figure 18 we illustrate a series of examples where we increasingly thin out the spokes in the star shape example to increase complexity. As shown



(a) mesh scale **vs** timing (b) mesh number **vs** timing

Fig. 17. Comesh Optimization statistics: our Comesh solver has efficient scaling as we increase both (a) mesh resolution and (b) number of example shapes used to build our interpolation.

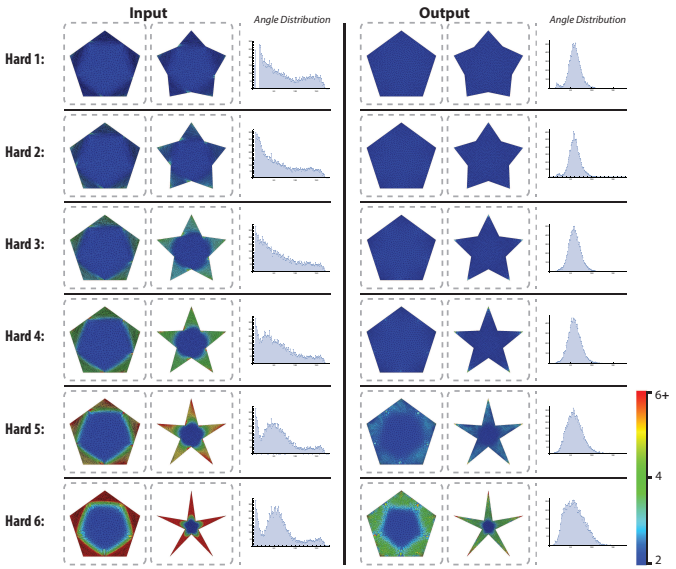


Fig. 18. Comesh Optimization examples over increasing problem complexity. Here our Comesh solver consistently produces low-distortion mappings and high-quality elements as we increase variation between input shapes. We visualize input/output map distortions and angle distributions here and compare the improvement of our resulting Comesh Optimized interpolations for *hard* 3 and *hard* 6 examples over unoptimized LIM initializations in Figure 4.

in the left column of Figure 18, this introduces increasingly more distorted and challenging Comesh Optimization problems. In the right column of Figure 18 we show the low-distortion mappings and high-quality elements we compute with our Comesh solver, while in the top two rows of Figure 4 we compare the improvement of our Comesh Optimized interpolations for *hard* 3 and *hard* 6 examples over unoptimized LIM initializations. Finally, we observe that similar input shapes generally require greater time to optimize; for timing statistics see our supplemental. Currently, we conjecture that this may be due to similar input shapes having greater ambiguity in conformal mapping and so requiring greater effort to converge; we plan to investigate this further.

Comesh Solver Statistics. In Table 2 we detail the performance of our Comesh Optimization solver on our animation examples, see our supplemental videos. Maximum iteration and timing were 29 outer iterations with 18 seconds of compute time in our prototype code. In all example so far, with the exception of our *alligator* example, we observe that Comesh Optimization additionally reduces the number of triangles from the input, but not significantly, as its goal is refining topological connectivity of the initial meshes.

7.3 Animation

Control. Our system exposes a small number of parameters for interactive exploration of animations. These terms are listed in Table 3 and are summarized here. The blending compliance parameter ϵ, indicates the desired balance between dynamic effects and example shape conformance; it is used to control how tightly interpolated dynamics match keyframe shapes; smaller values give interpolations

Table 2. We list the comesh optimization statistics for our video examples: number of example shapes, input/output mesh information like number of vertex, triangle and cohesive edge; inner iteration number shows the total loops of geometrical and topological optimization while outer iteration number shows the number of outer loops; flip, collapse and split timing show the amount of time spent on respective topological operation while total timing shows the overall cost of the whole algorithm.

	Shape(#)	Ver(#/#)	Tri(#/#)	CoE(#/#)	Inner(#)	Outer(#)	Flip(s)	Collapse(s)	Split(s)	Total(s)
aligator	4	297/358	485/593	174/162	1051	19	1.722	1.615	0.857	7.589
cell splitting	3	573/570	978/972	142/142	572	15	1.009	0.547	0.487	4.078
chinese character	2	277/277	382/379	10/10	36	4	0.023	0.012	0.012	0.119
circle in circle	2	201/201	241/241	126/126	70	3	0.024	0.012	0.013	0.137
continental drift	4	271/269	362/357	104/104	2565	29	8.841	5.539	1.067	18.322
growing flower	4	163/159	213/205	24/24	54	5	0.023	0.018	0.028	0.104
flying bird	3	226/226	265/265	0/0	60	7	0.028	0.018	0.016	0.141
stylized walk	16	132/132	161/161	0/0	243	6	0.311	0.166	0.166	0.824
walk star	8	89/89	93/93	24/24	243	7	0.085	0.049	0.051	0.284

Table 3. We list the animation statistics and parameter settings for the video examples, including maximal iteration number per frame, maximal time spent on a single local-global step, compliance parameter ϵ, cohesive zone parameter τ, model control α, damping paramter μ as well as material type (homogeneous or inhomogeneous) and with or without animation curve tuning.

	Iteration(#)	Local Step(s)	Global Step(ms)	ϵ	τ	α	μ	Uniform Material	Animation Curve
aligator	47	0.194	2.113	0.02	1e5	0.7	0.5	✓	✓
cell splitting	173	0.202	6.144	1.7	2e6	0.2	1.0	✓	✗
chinese character	38	0.133	1.307	0	2e5	0.5	-	✓	✗
circle in circle	52	0.007	2.106	1.5	4e5	0.5	1.8	✓	✗
continental drift	79	0.085	1.817	0	2e5	0.5	-	✓	✗
growing flower	53	0.089	0.592	0	2e5	0.5	-	✓	✗
flying bird	43	0.052	0.502	1	-	-	0.5	✗	✓
stylized walk	40	0.038	0.441	0	-	-	-	✓	✓
swirl	406	0.04	0.273	0	-	-	-	✓	✗
walk star	36	0.027	0.286	0	6e5	0.2	-	✓	✓

that matches the keyframes better while larger value bias towards more exaggerate dynamics. The cohesive threshold, τ, controls the effective stickiness of separation and merging during interpolation. Our mode control parameter, α, then determines whether stretch or shear dominates the topology change. The damping force scaling term, μ, is enabled when dynamics are applied and helps guide how excited secondary dynamics remain over time. Finally, we enable the ability to paint multiple material stiffnesses, κ, with a brush interface to control the degree to which shapes resist deformation. Interaction with all of these user parameters is illustrated in our demo session in the supplemental videos.

Fig. 19. Trumperfly: interpolation from a sparse set of production frames enables experimentation and variation.

Range. Our method generates interpolated animations for a range of challenging animation problems consisting of extreme deformation, topology changes and dynamics. Some of these examples are illustrated in Figure 1. Our system enables the generation of a wide variety of animations from sparse input drawings. As a baseline test we reproduce a set of benchmark examples from the *12 Basic Principles of Animation*, as shown in the supplemental. For this task we use a small number of input drawings; always less than five. In our supplemental videos we additionally duplicate a professional, hand-drawn animation that includes extreme transitions with topology change and extreme deformations; see Figure 19 for sample interpolated frames. We use just a sparse subset of the frames and then demonstrate freeform experimentation on the resulting interpolation to interactively explore animation variations without having to draw new frames.

Statistics. Our interpolation provides interactive feedback so that artists can quickly iterate in creating final animations. Performance statistics for interpolating our animation examples (both with and without dynamics) are summarized in Table 3 with a maximum of less than 10ms per local-global iteration across all examples in our unoptimized code.

We have presented an interpolation system that allows artists to interactively animate and interpolate inbetweens from arbitrary key

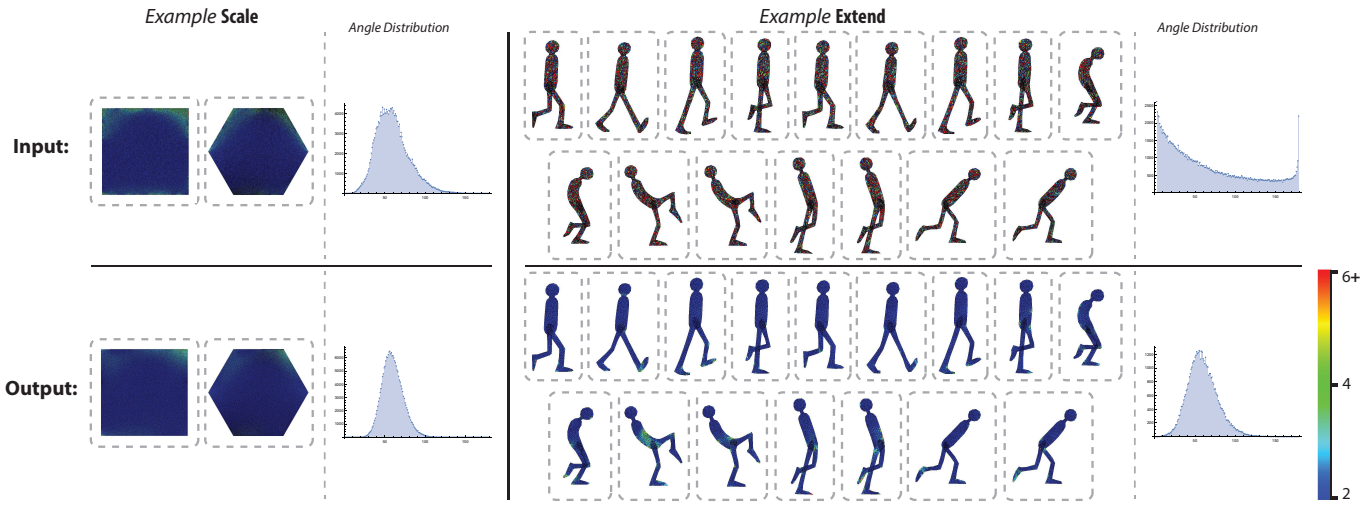


Fig. 20. Our Comesh Optimization algorithm scales well to large size meshes (left) and large number of inputs (right). In both cases, our method efficiently generates meshes with high-quality elements and maps.

drawing shapes. We demonstrate that this framework enables exploration of a wide range of 2D animations from sparse drawing inputs. Our method provides artists the flexibility to specify desired correspondences and topology changes, and to create and edit inbetweening for animation. To achieve artifact-free interpolation, we introduce a Comesh Optimization method and solver that builds consistent meshes across drawings while respecting correspondences and cuts. Our optimization algorithm improves both the energy discretization on each mesh, by enhancing element quality, as well as the mapping distortion across meshes. Given our Comesh Optimization, we then introduce two boundary energies, a cohesive energy and a non-overlap energy. Finally, we demonstrate how to add dynamic secondary motion effects to our keyframe interpolation.

Looking ahead we believe that a complete system for drawing planar animation should also support layered drawings. A few of our examples use two layers (near and far) and rely on interpolation to preserve coherence. More elaborate drawing arrangements will likely be enhanced by multilayer support: techniques to extract layers from drawings and techniques to attach them together explicitly. Comesh Optimization could also be further improved with a smooth spline-defined boundary. As is, a feature point, such as a sharp corner, on the boundary will not move during optimization. Likewise, we expect that the numerical solver for our Comesh Optimization could be enhanced with improved convergence.

Our approach could be extended to 3D. Our main contributions, *Comesh Optimization* and our *Multimesh Interpolation* with boundary energies appear to generalize quite easily to 3D. However, we also depend on topology optimization and piecewise linearly injective mappings, which remain challenging to extend reliably and efficiently in 3D.

Finally, we note that in this work we do not address the interesting and important question of how to meaningfully automate correspondences for artists. Instead, we sought to expand the expressive range of interpolation so that artists can accomplish more with less, such as complex animation of cell splitting with only three keyframes. In doing so, we also recognized that correspondence plays a key semantic role and designed a solution that could work for correspondences discovered by either interactive or automatic workflows. As detailed here and illustrated in our videos our current method uses interactively defined correspondences from artists. Optimal transport based methods are one interesting avenue of future exploration for automation. Currently, however, as we illustrate in Figure 2, such methods generally choose one path and not always the desired one. Thus we see an investigation of automated and semi-automated methods for providing correspondences remains an avenue of exciting future work here.

ACKNOWLEDGEMENTS

We thank Tim Luecke and the team at *The Late Show with Stephen Colbert* for valuable discussion and use of the *Trumperfly* assets.

REFERENCES

Marc Alexa, Daniel Cohen-Or, and David Levin. 2000. As-rigid-as-possible Shape Interpolation. *Proc. ACM SIGGRAPH*.

Pierre Alliez, David Cohen-Steiner, Mariette Yvinec, and Mathieu Desbrun. 2005. Variational Tetrahedral Meshing. *Proc. ACM SIGGRAPH* 24, 3 (2005).

Hadar Averbuch-Elor, Daniel Cohen-Or, and Johannes Kopf. 2016. Smooth Image Sequences for Data-driven Morphing. *Computer Graphics Forum* 35, 2 (2016).

Jernej Barbič, Marco da Silva, and Jovan Popović. 2009. Deformable Object Animation Using Reduced Optimal Control. *Proc. ACM SIGGRAPH* 28, 3 (2009).

William Baxter, Pascal Barla, and Ken Anjyo. 2009a. N-way Morphing for 2D Animation. *Comput. Animat. Virtual Worlds* 20 (2009).

W.V. Baxter, P. Barla, and K.-i. Anjyo. 2009b. Compatible Embedding for 2D Shape Animation. *IEEE Transactions on Visualization and Computer Graphics* 15, 5 (2009).

Sofien Bouaziz, Sebastian Martin, Tiantian Liu, Ladislav Kavan, and Mark Pauly. 2014. Projective Dynamics: Fusing Constraint Projections for Fast Simulation. *Proc. ACM SIGGRAPH* 33, 4 (2014).

Tyson Brochu and Robert Bridson. 2009. Robust Topological Operations for Dynamic Explicit Surfaces. *SIAM Journal on Scientific Computing* 31, 4 (2009).

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

Renjie Chen, Ofir Weber, Daniel Keren, and Mirela Ben-Chen. 2013. Planar Shape Interpolation with Bounded Distortion. *Proc. ACM SIGGRAPH* 32, 4 (2013).

Boris Dalstein, Rémi Ronfard, and Michiel van de Panne. 2015. Vector Graphics Animation with Time-varying Topology. *Proc. ACM SIGGRAPH* 34, 4 (2015).

Mikhail Gromov. 2007. *Metric structures for Riemannian and non-Riemannian spaces*. Springer Science & Business Media.
B. Heeren, M. Rumpf, P. Schröder, M. Wardetzky, and B. Wirth. 2014. Exploring the Geometry of the Space of Shells. *Computer Graphics Forum* 33, 5 (2014).
B. Heeren, M. Rumpf, M. Wardetzky, and B. Wirth. 2012. Time-Discrete Geodesics in the Space of Shells. *Computer Graphics Forum* 31, 5 (2012).
K. Hormann and G. Greiner. 2000. *MIPS: An Efficient Global Parametrization Method*. Defense Technical Information Center.
Anil Kaul and Jarek Rossignac. 1992. Solid-interpolating deformations: Construction and animation of PIPs. *Computer Graphics Forum* 16, 1 (1992).
Martin Kilian, Niloy J. Mitra, and Helmut Pottmann. 2007. Geometric Modeling in Shape Space. *Proc. ACM SIGGRAPH* 26, 3 (2007).
Alexander Kort. 2002. Computer Aided Inbetweening *(NPAR)*.
Zohar Levi and Craig Gotsman. 2015. Smooth Rotation Enhanced As-Rigid-As-Possible Mesh Animation. *IEEE Transactions on Visualization and Computer Graphics* 21, 2 (2015).
Guiqing Li, Liang Yang, Shihao Wu, Wenshuang Tan, Xinyu Chen, and Chuhua Xian. 2013. Planar Shape Interpolation Using Relative Velocity Fields. *Computer Graphics Forum* 37, 5 (2013).
Yaron Lipman. 2012. Bounded Distortion Mapping Spaces for Triangular Meshes. *Proc. ACM SIGGRAPH* 31, 4 (2012).
Ligang Liu, Lei Zhang, Yin Xu, Craig Gotsman, and Steven J. Gortler. 2008. A Local/Global Approach to Mesh Parameterization. In *Proceedings of the Symposium on Geometry Processing (SGP '08)*.
Sebastian Martin, Bernhard Thomaszewski, Eitan Grinspun, and Markus Gross. 2011. Example-based Elastic Materials. *Proc. ACM SIGGRAPH* 30, 4 (2011).
Roi Poranne and Yaron Lipman. 2014. Provably Good Planar Mappings. *Proc. ACM SIGGRAPH* 33, 4 (2014).
Martin Rumpf and Benedikt Wirth. 2009. A Nonlinear Elastic Shape Averaging Approach. *SIAM J. Img. Sci.* 2, 3 (2009).
Christian Schüller, Ladislav Kavan, Daniele Panozzo, and Olga Sorkine-Hornung. 2013. Locally Injective Mappings. *Computer Graphics Forum* 32, 5 (2013).
Jonathan Shewchuk. 2005. Triangle. (2005). http://www.cs.cmu.edu/~quake/triangle.html
Jason Smith and Scott Schaefer. 2015. Bijective Parameterization with Free Boundaries. *Proc. ACM SIGGRAPH* 34, 4 (2015).
Justin Solomon, Fernando de Goes, Gabriel Peyré, Marco Cuturi, Adrian Butscher, Andy Nguyen, Tao Du, and Leonidas Guibas. 2015. Convolutional Wasserstein Distances: Efficient Optimal Transportation on Geometric Domains. *Proc. ACM SIGGRAPH* 34, 4 (2015).
Olga Sorkine and Marc Alexa. 2007. As-rigid-as-possible Surface Modeling. In *Proceedings of the Fifth Eurographics Symposium on Geometry Processing (SGP '07)*.
V. Surazhsky and C. Gotsman. 2004. High Quality Compatible Triangulations. *Eng. with Comput.* 20, 2 (2004).
D. Sýkora, D. Sedlacek, S. Jinchao, J. Dingliana, and S. Collins. 2010. Adding Depth to Cartoons Using Sparse Depth (In)equalities. *Computer Graphics Forum* 29, 2 (2010).
Christoph Von-Tycowicz, Christian Schulz, Hans-Peter Seidel, and Klaus Hildebrandt. 2015. Real-Time Nonlinear Shape Interpolation. *ACM Trans. Graph.* 34, 3 (2015).
Ofir Weber and Denis Zorin. 2014. Locally Injective Parametrization with Arbitrary Fixed Boundaries. *Proc. ACM SIGGRAPH* 33, 4 (2014).
B. Whited, G. Noris, M. Simmons, R. Sumner, M. Gross, and J. Rossignac. 2010. BetweenIT: An Interactive Tool for Tight Inbetweening. *Computer Graphics Forum* 29, 2 (2010).
T. Winkler, J. Drieseberg, M. Alexa, and K. Hormann. 2010. Multi-Scale Geometry Interpolation. *Computer Graphics Forum* 29, 2 (2010).
Benedikt Wirth, Leah Bar, Martin Rumpf, and Guillermo Sapiro. 2009. Geodesics in Shape Space via Variational Time Discretization. In *Energy Minimization Methods in Computer Vision and Pattern Recognition*. Vol. 5681.
Benedikt Wirth, Leah Bar, Martin Rumpf, and Guillermo Sapiro. 2011. A Continuum Mechanical Approach to Geodesics in Shape Space. *International Journal of Computer Vision* 93, 3 (2011).
Jun Xing, Li-Yi Wei, Takaaki Shiratori, and Koji Yatani. 2015. Autocomplete Hand-drawn Animations. *Proc. ACM SIGGRAPH Asia* 34, 6 (2015).
Xuemiao Xu, Liang Wan, Xiaopei Liu, Tien-Tsin Wong, Liansheng Wang, and Chi-Sing Leung. 2008. Animating Animal Motion from Still. *Proc. ACM SIGGRAPH Asia* 27, 5 (2008).
Wenwu Yang, Jieqing Feng, and Xun Wang. 2012. Structure Preserving Manipulation and Interpolation for Multi-element 2D Shapes. *Computer Graphics Forum* 31, 7pt2 (2012).