Supplement to 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

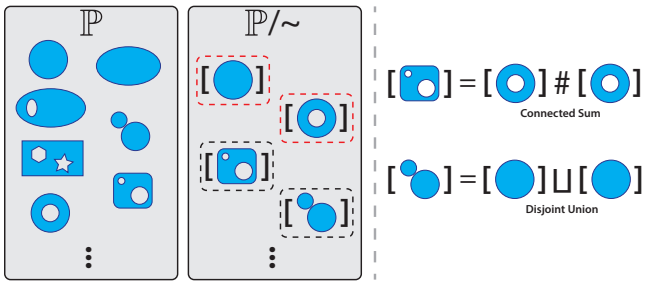

Figure 1: *Non-self intersecting planar shapes can be categorized by defining equivalent relationship as homeomorphism. Two shapes are in the same equivalence class if one can be smoothly morphed to the other. Different equivalence class elements can be converted to the same element under topological operations, connected sum and disjoint union.*

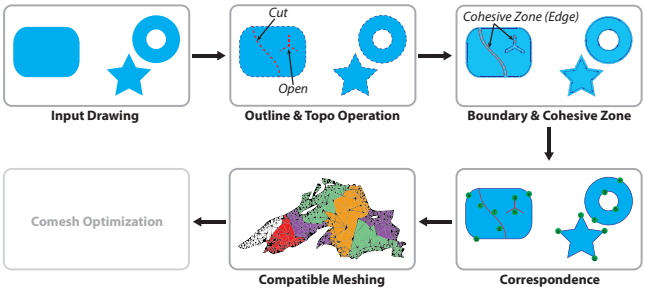

Figure 2: *To generate consistent multimesh structure for given input drawings, we start with specifying outlines, cuts and openings; then construct S^1 equivalent boundaries using directed graph flow algorithm; next mark sparse correspondences; and compatibly mesh all shapes to deliver an initial multimesh solution which will be further improved by comesh optimization.*

1 Constructing Multimesh Structure

Given compact non-self-intersecting planar shapes, there are a pair of ambiguities characterized by topological (in-)equivalence. When shapes are topologically *inequivalent*, one shape *can not* be continuously morphed to the other as there is no bi-continuous map (homeomorphism) between them. Thus we cannot build an isomorphism between the shapes' standard topology. Nevertheless, one can develop a quotient set, $\mathbb{P}/\sim$, by introducing an equivalence relation, defined as homeomorphism, to the set of compact non-self-intersecting planar shapes, $\mathbb{P}$. $\mathbb{P}/\sim$ can then be generated by two elements, equivalence class of disk, $\mathfrak{s}$ (simply connected), and annulus, $\mathfrak{m}$ (multiply connected) under the operations, disjoint union ($\sqcup$) and connected sum ($\#$) as shown in Figure 1. Thus we require two topological operations, cutting and opening, to convert input shapes into an equivalence class in $\mathbb{P}/\sim$. Locations and shapes of these cuts and openings are defined on the domain, which then determine both how and where shapes will open and split.

When shapes are topologically *equivalent*, one shape *can* be smoothly morphed into another. However, there then exists infinitely many bi-continuous maps between them as both correspondences and path can vary. We provide interactive control of specifying boundary correspondences and inbetweening energy parameters to allow additional artistic control.

Equivalence Tool Our interactive equivalence tool allows users to convert input artworks to the same equivalence class by outlining and placing desired cuts and openings. We represent outlines, cuts and openings with a planar straight line graph. Outlines define the boundary of each drawing's embedding shape and are equivalent to a planar circle (S^1) dividing $\mathbb{R}^2$ into two connected components - the bounded inside and the unbounded outside. By specifying orientation of the outline, clockwise or counter-clockwise, users define which connected component embeds the drawings.

As the specified cuts and openings break the simple closed curve property of our outlines, we need to reconstruct boundaries after the *outline & topo operation* (see Figure 2). We treat the resulting soup of outlines, cuts and openings as a directed graph. Orientations of outlines determine directions of their line segments, while edges generated by cuts and openings are bi-directed. The final boundaries are reconstructed by applying a *directed graph flow* algorithm whose details are provided in Section 2. After boundary reconstruction, each line segment belonging to a cut or opening turns into two duplicated edges with opposing directions as shown in Figure 5. We designate each pair of duplicated edges as a *cohesive edge* along which we will assign cohesive energies to help form our inbetweening energy. We then call collections of *cohesive edges* a *cohesive zone*.

Boundary Tool Next our boundary tool lets users interactively mark correspondences on desired boundaries. Collections of corresponding boundaries across shapes are then compatibly reparameterized to obtain pairwise bijectively mapped boundaries suitable for our initial meshing step.

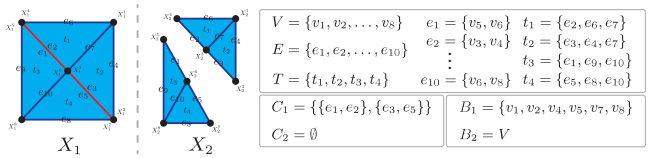

Figure 3: *An illustrative example of multimesh structure for two shapes: both shape share the same mesh topology but may have different cohesive edges, boundary vertice and mesh geometry. Edges colored in red represent cohesive edge.*

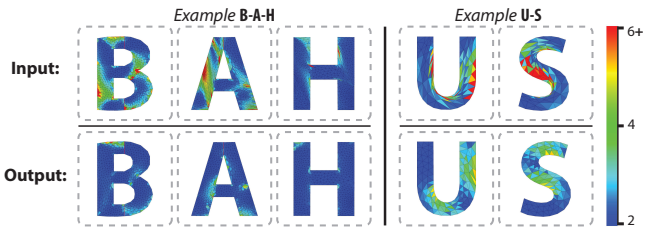

Figure 4: *Comesh Optimization improves both the mapping and meshing quality of the multimesh structure subject to user specified correspondences and cohesive edge constraints and keeps consistent mesh topology throughout the procedure.*

Initial Meshing We now have equivalent shapes with desired cuts, openings and correspondences specified. Next, we augment our shapes with interior meshes. Given multiple compatible boundaries of planar shapes, we use *Triangle* [Shewchuk 2005] to generate a conforming Delaunay triangulation of one shape. As this meshing step may insert Steiner points along the boundaries, we then upsample boundaries of the remaining shapes and ensure that splits in cohesive edges remain consistent. Then, we apply Schueller et al.'s LIM algorithm [2013] to map our first shape's triangle mesh to the remaining example shapes. This will build the initial multimesh structure with mesh topology $\mathcal{O} = \{V, E, T\}$, cohesive edge sets C_i, boundary vertex sets B_i and mesh geometry X_i (see Figure 3 for an illustrative example.). As we will discretize and minimize our inbetweening energy on these meshes, compatible meshes with high quality elements and low distortion mappings between shapes are required. Thus we apply Comesh Optimization to obtain the final reliable meshes, see Figure 4.

2 Directed Graph Flow Implementation

Our interactive equivalence tool allows users to convert example artworks to the same equivalence class by drawing desired cuts and openings. After an editing session, we apply a *directed graph flow* algorithm to construct final boundaries, as the specified cuts and openings break the simple closed curve property of outlines. We treat the resulting soup of outlines, cuts and openings as a directed graph $\mathcal{G}$. Orientations of outlines determine line segment directions, while edges generated by cuts and openings are bi-directed.

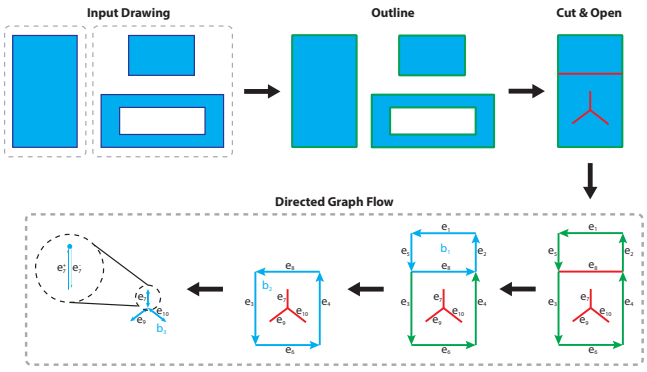

Figure 5: *Cuts and openings (red) will break the simple closed curve property of outlines (green). We apply directed graph flow algorithm to construct final boundaries. We treat soup of outlines, cuts and openings as directed graph, where edges coming from outlines are 1-directed (directed) and edges coming from cuts and openings are bi-directed (undirected). We iteratively look for directed cycles in the graph until every 1-directed edges are visited once and bi-directed edges are visited twice.*

We then apply the *directed graph flow* algorithm detailed in Algorithm 1. Here we iteratively look for closed circular flows (directed cycles) until all 1-directed edges are visited once and all bi-directed edges are visited twice. In detail, here `edge_can_be_visited` returns an edge e^* whose visit counter $c(e^*)$ is not zero, while `next_edge_on_graph` returns the next edge with respect to e^* according to the graph topology and directions field. When there are multiple candidates for next edge, we choose the one clockwise-closest to e^*. After boundary reconstruction, each line segment belonging to a cut or opening turns into two duplicated edges with opposing directions, see Figure 5.

3 Comesh Optimization Energy Gradient

We compute the necessary energy gradients here. Gradient of ODT mesh energy is

$$\frac{\partial \mathcal{D}(X_i, \mathcal{O})}{\partial X_i^j} = \frac{\sum_{t_k \in \tilde{\Gamma}_j} \left(\frac{\partial a_k(X_i)}{\partial X_i^j} \cdot \sum_{v_p \in t_k} \|X_i^j - X_i^p\|^2 \right)}{8\Gamma_j(X_i)}, \tag{1}$$

and gradient of MIPS map energy is

$$\begin{aligned}
\frac{\partial \mathcal{C}(X_i, X_j, \mathcal{O})}{\partial X_j^q} &= \sum_{t_k \in \tilde{\Gamma}_q} \left(\sum_{p=1}^{2} \frac{\partial \sigma_k^*}{\partial \sigma_k^p} \frac{\partial \sigma_k^p(X_i, X_j)}{\partial X_j^q} \right) a_k(X_i), \\
\frac{\partial \mathcal{C}(X_i, X_j, \mathcal{O})}{\partial X_i^q} &= \sum_{t_k \in \tilde{\Gamma}_q} \left(\sum_{p=1}^{2} \frac{\partial \sigma_k^*}{\partial \sigma_k^p} \frac{\partial \sigma_k^p(X_i, X_j)}{\partial X_i^q} \right) a_k(X_i) \\
&\quad + \sigma_k^*(X_i, X_j) \frac{\partial a_k(X_i)}{\partial X_i^q}, \\
\sigma_k^* &= \frac{\sigma_k^1}{\sigma_k^2} + \frac{\sigma_k^2}{\sigma_k^1}
\end{aligned} \tag{2}$$

where X_i^j is the j-th vertex in the i-th shape, namely $X_i(v_j)$, and $\tilde{\Gamma}_j$ is the one-ring triangles of the j-th vertex. Notice $\sigma_k^*(X_i, X_j) = \sigma_k^*(X_j, X_i)$ as MIPS energy is symmetric with respect to the source and target shape under the piecewise linear setting, which can be verified as follows: deformation gradient $F_k(X_i, X_j)$ is a linear map from the tangent space of k-th triangle in shape X_i to tangent space of k-th triangle in shape X_j. For non-degenerate cases, the inverse map $F_k(X_i, X_j)^{-1}$ is the deformation gradient $F_k(X_j, X_i)$ and $\sigma_k^p(X_i, X_j) = \frac{1}{\sigma_k^p(X_j, X_i)}$, $p = 1, 2$.

4 Continuous Energies for Inbetweening

Elastic Energy In the continuous setting, we reuse math notation X and x to represent continuous example and deformed shape manifold respectively. Given the deformation map $f\colon X_i \to x$ from example shape to deformed shape, the ARAP energy is

$$W_D(X_i, x) = \int_{X_i} \kappa \min_{\mathbf{R} \in SO(2)} \|df - \mathbf{R}\|_F^2, \tag{3}$$

where df is the corresponding differential map, or the pushforward, $\mathbf{R}$ is a rotation, κ is a simple, spatially varying, material stiffness, and $\|\cdot\|_F$ is the conventional Frobenius norm.

Algorithm 1 *Directed Graph Flow*

Inputs:
directed graph $\mathcal{G} = \{V, E\}$,
$E = \{e_1, e_2, \cdots, e_f, e_{f+1}, e_{f+2}, \cdots, e_g\}$,
$i \in [1, f]$ is 1-directed, $i \in [f+1, g]$ is bi-directed,
$\mathcal{C} = \{c(e_i)\}, i \in [1, g]$.
Outputs:
boundary list $\mathcal{B} = \{b_i\}$.
Initialize:
$c(e_i) = 1 \ \forall i \in [1, f], c(e_i) = 2 \ \forall i \in [f+1, g]$
$\mathcal{B} = \emptyset, b^* = \emptyset, e^* = e_1$.
while `true` **do**
  if `is_already_in`(b^*, e^*)
    `add_to_boundary_list`$(\mathcal{B}, b^*)$
    $b^* = \emptyset$
    $e^* =$ `edge_can_be_visited`$(\mathcal{C})$
    if $e^* = \emptyset$ **break**
  else
    `add_to_directed_cycle`(b^*, e^*)
    $c(e^*) = c(e^*) - 1$
    $e^* =$ `next_edge_on_graph`$(\mathcal{G}, e^*)$
  end if
end while

Cohesive Energy Deformation map f gives a displacement field over X_i which can be discontinuous along artist-specified cuts and opens, called cohesive zone $C_i \subset X_i$ which is Lebesgue measure 0 subset of X_i, like graphs or nets. Here we reuse the math notation C in the continuous setting. Cohesive energy is designed to measure the gap of discontinuity along C_i,

$$W_C(C_i, x) = \int_{C_i} \mathcal{E}(\llbracket\, x(v)\, \rrbracket), \quad v \in C_i. \tag{4}$$

We reuse the math notation v to represent point lying on C_i in the continuous setting. The jump operator $\llbracket\cdot\rrbracket$ evaluates the displacement difference between both sides, $x^+(v)$ and $x^-(v)$, of cohesive zone,

$$\llbracket\, x(v)\, \rrbracket = x^+(v) - x^-(v), \quad v \in C_i. \tag{5}$$

The energy density $\mathcal{E}$ measures the squared norm of this difference to compute the amount of energy accumulated in the zone. Separation occurs when the accumulated energy exceeds the separation threshold τ,

$$\begin{aligned} \mathcal{E}(\llbracket\, x\, \rrbracket) &= \begin{cases} \tilde{\tau}(\llbracket\, x\, \rrbracket), \ \tilde{\tau} < \tau \\ \tau, \text{ else.} \end{cases}, \\ \tilde{\tau}(\llbracket\, x\, \rrbracket) &= \lambda \left\| \llbracket\, x\, \rrbracket \right\|^2, \ 0 \le \lambda, \tau. \end{aligned} \tag{6}$$

To further control cohesive energy, biasing towards greater resistance to shear or alternatively to stretch, we decompose shear and stretch measures of the displacement jump by redefining

$$\tilde{\tau}(\llbracket\, x\, \rrbracket) = \lambda \left[(1-\alpha) \|\llbracket\, x\, \rrbracket\|_{/\!/}^2 + \alpha \|\llbracket\, x\, \rrbracket\|_{\perp}^2 \right], \ \alpha \in [0, 1]. \tag{7}$$

Here $\|\cdot\|_{/\!/}$ and $\|\cdot\|_{\perp}$ are norms measuring input vector's tangential and normal component with respect to the direction $\hat{x}$ of deformed edge pair, defined in Equation (14), while a change of weights, α, decides their relative contribution.

Non-overlap To avoid overlapping during inbetweening, we need to ensure that points $x(v)$ on deformed shape boundaries ∂x do not penetrate the interior. We re-purpose the indicator function from contact mechanics [Kane et al. 1999]

$$\mathcal{I}(x(v)) = \begin{cases} 0, & \text{if } x(v) \cap (x \setminus \partial x) = \emptyset, \\ \infty, & \text{else.} \end{cases}, \quad v \in \partial X_i. \tag{8}$$

Here the indicator function $\mathcal{I}$ maps boundary point $x(v), v \in \partial X_i$ to an extreme penalty energy ∞ if it lies in the interior of the deformed shape x and to 0 otherwise.

We then integrate up the indicator function over each example shape's boundary to create an energy that ensures boundary points on shape X_i avoid overlap in deformed configuration x

$$W_O(\partial X_i, x) = \int_{\partial X_i} \mathcal{I}\big(x(v)\big). \tag{9}$$

5 Cohesive Energy Discretization

In the discrete setting, cohesive zone is modeled by pairs of overlapping triangle edges, cohesive edge. Since we adopt P1 element to discretize displacement field, displacement jump $[\![x]\!]$ also varies linearly along cohesive edges. For a cohesive edge element C_i^j, we can parameterize displacement jump over it by

$$\begin{aligned} [\![x(u)]\!] &= (1-u)\ [\![x(v_0)]\!] + u\ [\![x(v_1)]\!] \\ 0 \le u \le 1, \ \ \partial C_i^j &= \{\{v_0^+, v_0^-\}, \{v_1^+, v_1^-\}\}. \end{aligned} \tag{10}$$

As v^+ and v^- are collocated in example shape X_i, we use v instead to denote them. Given piecewise constant cohesive stiffness λ, the edge element's cohesive energy below threshold τ is then

$$\begin{aligned} W_C(C_i^j, x) &= \int_{C_i^j} \mathcal{E}(\llbracket\, x(u)\, \rrbracket), \quad u \in [0, 1], \quad \bar{\tau} < \tau \\ &= \lambda \int_{C_i^j} \|\llbracket\, x(u)\, \rrbracket\|^2 \\ &= \lambda \int_0^{l_i^j} \|\llbracket\, x(u)\, \rrbracket\|^2 dl. \end{aligned} \tag{11}$$

l_i^j is the edge length of C_i^j, which is linearly parameterized as

$$l(u) = u \cdot l_i^j, \qquad 0 \le u \le 1. \tag{12}$$

Then

$$\begin{aligned} & W_C(C_i^j, x) = \lambda \int_{l(0)}^{l(1)} \|\llbracket\, x(u)\, \rrbracket\|^2 dl(u) \\ &= \lambda \int_0^1 \|\llbracket\, x(u)\, \rrbracket\|^2 d(u \cdot l_i^j) \\ &= \lambda l_i^j \int_0^1 \|(1-u)[\![x(v_0)]\!] + u[\![x(v_1)]\!]\|^2 du \\ &= \lambda l_i^j \{\|[\![x(v_0)]\!]\|^2 (\frac{1}{3}u^3 - u^2 + u) + \|[\![x(v_1)]\!]\|^2 (\frac{1}{3}u^3) \\ &\quad + [\![x(v_0)]\!]^T [\![x(v_1)]\!] (-\frac{2}{3}u^3 + u^2)\}\Big|_0^1 \\ &= \frac{1}{3}\lambda l_i^j \{\|[\![x(v_0)]\!]\|^2 + \|[\![x(v_1)]\!]\|^2 + [\![x(v_0)]\!]^T [\![x(v_1)]\!]\}. \end{aligned} \tag{13}$$

In order to have control over connecting/separating mode behavior, we further split the displacement jump $[\![x]\!]$ into tangential and normal component with respect to $\hat{x}$. Even though v^+ and v^- are collocated in example shape X, it's not necessarily true for $x(v^+)$

and $x(v^-)$ in deformed shape x. For the consistency of math notation convention, we use $x^+(v)$ and $x^-(v)$ instead to represent $x(v^+)$ and $x(v^-)$ respectively. Thus $\hat{x}$ of C_i^j is defined as

$$\hat{x} = \frac{x^+(v_0) + x^-(v_0) - x^+(v_1) - x^+(v_1)}{\|x^+(v_0) + x^-(v_0) - x^+(v_1) - x^-(v_1)\|} \tag{14}$$

under the piecewise linear setting. Then we can define $\|\cdot\|_{/\!/}$ and $\|\cdot\|_\perp$ with respect to $\hat{x}$ as follows,

$$\begin{aligned} [\![x]\!] &= [\![x]\!]_{/\!/} + [\![x]\!]_\perp, \\ [\![x]\!]_{/\!/} &\coloneqq \hat{x}[\![x]\!]^T\hat{x},\ [\![x]\!]_\perp \coloneqq [\![x]\!] - \hat{x}[\![x]\!]^T\hat{x}, \\ \|[\![x]\!]\|_{/\!/} &\coloneqq \|[\![x]\!]_{/\!/}\| = (\hat{x}[\![x]\!]^T\hat{x})^T(\hat{x}[\![x]\!]^T\hat{x}) \\ &= \hat{x}^T[\![x]\!][\![x]\!]^T\hat{x} = [\![x]\!]^T(\hat{x}\hat{x}^T)[\![x]\!], \\ \|[\![x]\!]\|_\perp &\coloneqq \|[\![x]\!]_\perp\| = [\![x]\!]^T(I_{2\times 2} - \hat{x}\hat{x}^T)[\![x]\!]. \end{aligned} \tag{15}$$

By adopting a relative weighting parameter $\alpha \in [0,1]$ to control shear/stretch contribution and substituting Equation (15) into Equation (11), we get

$$W_C(C_i^j, x) = \lambda \int_{C_i^j} (1-\alpha)\cdot \|[\![x(u)]\!]\|_{/\!/}^2 + \alpha \cdot \|[\![x(u)]\!]\|_\perp^2. \tag{16}$$

Following similar derivations above, we finally achieve the discretization of cohesive energy with control over connecting/separating mode behavior.

6 Transitivity of MIPS

We optimize cyclic summation, instead of full graph summation, of MIPS energy in our comesh algorithm to account for inter-mesh quality. This is due to the fact that MIPS is transitive: if the conformal distortion between X_i and X_j, as well as X_j and X_k are both small, then the conformal distortion between X_i and X_k is also small. Before we provide the proof for it, we first prove a lemma that will be used later.

Lemma 1. *For matrix $A, B \in \mathbb{R}^{n\times n}$, the following inequality holds,*

$$\|AB\|_F \le \|A\|_F\|B\|_F. \tag{17}$$

Proof. According to the definition of matrix A's *Frobenius* norm, we have

$$\|A\|_F^2 = \sum_{i,j}|a_{i,j}|^2 = \sum_i \|A_{i*}\|_2^2 = \sum_j \|A_{*j}\|_2^2, \tag{18}$$

where A_{i*} is the i-th row vector and A_{*j} is the j-th column vector. Due to *Cauchy-Schwarz* inequality, we have

$$\|Ax\|_2^2 = \sum_i |A_{i*}x|^2 \le \sum_i \|A_{i*}\|_2^2\|x\|_2^2 = \|A\|_F^2\|x\|_2^2, \tag{19}$$

where x is a column vector. Thus we have

$$\begin{aligned} \|AB\|_F^2 &= \sum_j \|AB_{*j}\|_2^2 \le \sum_j \|A\|_F^2\|B_{*j}\|_2^2 \\ &= \|A\|_F^2 \sum_j \|B_{*j}\|_2^2 = \|A\|_F^2\|B\|_F^2. \end{aligned} \tag{20}$$

By taking square root of the inequality's both sides, we prove the lemma. □

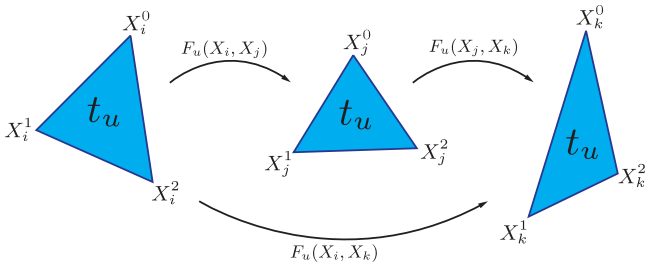

Figure 6: *MIPS energy has transitivity property. It is enough to optimize the cyclic summation instead of the full graph summation. We prove this property by considering chain of maps between single triangle t_u from three shapes, X_i, X_j and X_k.*

Next we show the transitivity of MIPS, which is defined as

$$\begin{aligned} \sigma_u^*(X_i, X_j) =& \frac{\sigma_u^1}{\sigma_u^2} + \frac{\sigma_u^2}{\sigma_u^1} \\ =& \|F_u(X_i, X_j)\|_F\|F_u^{-1}(X_i, X_j)\|_F, \end{aligned} \tag{21}$$

where $F_u(X_i, X_j)$ is the deformation gradient of triangle t_u by deforming from X_i to X_j.

Lemma 2. *If the MIPS distortion between X_i and X_j as well as X_j and X_k are both small (bounded above), then the MIPS distortion between X_i and X_k is also small (bounded above).*

Proof. We consider the distortion of a single triangle, t_u, among three shapes X_i, X_j and X_k as shown in Figure 6 and assume the following upper bounds hold,

$$\begin{aligned} \sigma_u^*(X_i, X_j) =& \|F_u(X_i, X_j)\|_F\|F_u^{-1}(X_i, X_j)\|_F \le p, \\ \sigma_u^*(X_j, X_k) =& \|F_u(X_j, X_k)\|_F\|F_u^{-1}(X_j, X_k)\|_F \le q. \end{aligned} \tag{22}$$

As the deformation gradient $F_u(X_i, X_j)$ is a linear map from t_u in X_i to t_u in X_j, it can be defined as

$$\begin{aligned} F_u(X_i, X_j) &= A_u(X_j)A_u^{-1}(X_i), \\ A_u(X_i), A_u(X_j) &\in \mathbb{R}^{2\times 2}, \end{aligned} \tag{23}$$

for non-degenerate triangles. $A_u(X_i)$ and $A_u(X_j)$ are defined as

$$\begin{aligned} A_u(X_i) &= \left[X_i^1 - X_i^0, X_i^2 - X_i^0\right], \\ A_u(X_j) &= \left[X_j^1 - X_j^0, X_j^2 - X_j^0\right]. \end{aligned} \tag{24}$$

Thus we can rewrite Equation (22) as

$$\begin{aligned} &\|A_u(X_j)A_u^{-1}(X_i)\|_F\|A_u(X_i)A_u^{-1}(X_j)\|_F \le p, \\ &\|A_u(X_k)A_u^{-1}(X_j)\|_F\|A_u(X_j)A_u^{-1}(X_k)\|_F \le q. \end{aligned} \tag{25}$$

Then we have

$$\begin{aligned} &\|A_u(X_j)A_u^{-1}(X_i)\|_F\|A_u(X_i)A_u^{-1}(X_j)\|_F \\ &\cdot\|A_u(X_k)A_u^{-1}(X_j)\|_F\|A_u(X_j)A_u^{-1}(X_k)\|_F \le pq, \\ \Rightarrow\ &(\|A_u(X_k)A_u^{-1}(X_j)\|_F\|A_u(X_j)A_u^{-1}(X_i)\|_F) \\ &\cdot(\|A_u(X_i)A_u^{-1}(X_j)\|_F\|A_u(X_j)A_u^{-1}(X_k)\|_F) \le pq. \end{aligned} \tag{26}$$

According to Lemma 1, we have

$$\begin{aligned} &(\|A_u(X_k)A_u^{-1}(X_j)A_u(X_j)A_u^{-1}(X_i)\|_F) \\ &\cdot(\|A_u(X_i)A_u^{-1}(X_j)A_u(X_j)A_u^{-1}(X_k)\|_F) \le pq, \\ \Rightarrow\ &\|A_u(X_k)A_u^{-1}(X_i)\|_F\|A_u(X_i)A_u^{-1}(X_k)\|_F \le pq, \\ &\Rightarrow \sigma_u^*(X_i, X_k) \le pq. \end{aligned} \tag{27}$$

Thus the MIPS distortion between X_i and X_k is also bounded. □

7 Automated Blending of Dynamic Effects

We propose a simple blending control method to balance between hitting keyframes and satisfying dynamics. Instead of utilizing external force to control dynamics like other space-time optimization approach, our blending method makes use of numerical stiffness depending on timestep size h_s, material stiffness κ and material density ρ. We automate the blending of dynamics by reparameterizing the blending weight ξ with a user set compliance control ϵ,

$$\xi(s) = \frac{\epsilon \cdot h_s^2 \cdot \kappa}{\rho \cdot (\delta(s) + 1)}, \tag{28}$$

which cancels out the influence of timestep size, material stiffness and material density to turn a multi-parameter control into a single-parameter control. $\delta(s)$ measures deformation between the previous frame $\bar{x}(w(s-1))$ and the target inbetweened shape without dynamics $\Phi\big(\mathcal{A}, w(s)\big)$,

$$\delta(s) = W_D(\Phi\big(\mathcal{A}, w(s)\big), \bar{x}(w(s-1))). \tag{29}$$

As the difference between the dynamic and non-dynamic inbetween frames grow, $\delta(s)$ increases and the inbetweening $\bar{x}(w(s))$ is driven towards the interpolated shape without dynamics $\Phi\big(\mathcal{A}, w(s)\big)$.

References

KANE, C., REPETTO, E., ORTIZ, M., AND MARSDEN, J. 1999. Finite element analysis of nonsmooth contact. *Comp. Meth. Appl. Mech. Engrg. 180 (1999).*

SCHÜLLER, C., KAVAN, L., PANOZZO, D., AND SORKINE-HORNUNG, O. 2013. Locally injective mappings. *Computer Graphics Forum 32*, 5, 125–135.

SHEWCHUK, J., 2005. Triangle.

8 Experiment Statistics

In this section, we provide detailed statistics tables corresponding to the comesh optimization evaluation discussed in the paper.

8.1 Relative Weighting of Energies in Comesh Optimization Objective

Table 1: *We take the square-to-hexagon example and evaluate the impact of relative contribution of the ODT mesh and the cyclic MIPS mapping energies in the objective on our Comesh Optimization's solution by varying relative weighting as (1 - weight) for meshing and weight for map energy. The starting map and mesh energy are 36.394 and 0.516 respectively. Note that except for this experiment we use an equal weighting (relative weight of 0.5) in all of our examples and evaluations.*

Weight	Ver(#/#)	Total(s)	Map	Mesh
0.13	467/358	2.329	2.114	0.494
0.21	467/331	1.881	2.095	0.494
0.33	467/340	2.034	2.065	0.494
0.44	467/332	3.473	2.054	0.494
0.5	467/345	2	2.056	0.494
0.62	467/337	2.245	2.052	0.494
0.7	467/335	1.186	2.055	0.494
0.85	467/327	1.263	2.050	0.495
0.93	467/360	1.072	2.046	0.495

8.2 Mesh Resolution

Table 2: *We evaluate the scaling behavior of comesh optimization solver by refining mesh resolution of square-hexagon example. As the resolution is doubled, we observe almost linear scaling behavior in terms of timing.*

Ver(#/#)	Inner(#)	Outer(#)	Total(s)
98/98	59	5	0.065
278/277	50	7	0.181
545/544	50	8	0.41
1057/1046	131	13	2.181
2087/2074	127	13	4.691
4331/4296	117	10	12.243
8649/8611	209	17	58.011
17131/16963	361	18	220.362
35386/35114	311	21	838.025
73072/71870	382	26	3358.401
165107/164638	447	33	13650.724
311522/311183	495	34	51253.368

8.3 Number of Example Shapes

Figure 7: *It is easy to blend dynamics or secondary motion into the inbetweening using our method to create more expressive and vivid animation. In this flying bird example, we also use inhomogeneous material by making bird body and wing skeleton much stiffer than head, feet and end of wings region.*

Table 3: *We evaluate the scaling behavior of comesh optimization solver by increasing number of meshes in the walking character example. As the number of meshes increases, we also observe almost linear scaling behavior in terms of timing.*

Shape(#)	Ver(#/#)	Inner(#)	Outer(#)	Total(s)
2	1639/1514	81	6	2.566
3	1639/1509	419	13	12.548
4	1639/1482	482	12	16.776
5	1639/1483	554	16	21.146
6	1639/1482	197	7	9.894
7	1639/1446	471	16	23.526
8	1639/1427	529	12	27.953
9	1639/1425	825	11	42.558
10	1639/1380	1273	24	77.621
11	1639/1380	818	14	85.71
12	1639/1357	643	13	56.014
13	1639/1342	742	12	94.933
14	1639/1314	843	18	82.283
15	1639/1337	723	11	104.171
16	1639/1357	883	16	78.754

8.4 Problem Difficulty

Table 4: *We evaluate the performance behavior of comesh optimization solver by increasing the problem complexity of pentagon-star example. The solver produces consistently good solutions as we increase variations between pentagon and star. We also observe that similar input shapes require more time to converge.*

	Ver(#/#)	Inner(#)	Outer(#)	Total(s)
hard 1	1683/1516	3149	96	82.906
hard 2	1683/1546	1391	53	40.188
hard 3	1683/1556	479	13	14.508
hard 4	1683/1574	484	22	14.347
hard 5	1683/1539	205	9	6.444
hard 6	1683/1376	233	9	9.46

9 Animation Results

Our method can generate high quality results for difficult animation problems consisting of extreme deformation (see Figure 8) and dynamics (see Figure 7). Our system is easy to use and capable of generating large variety of animation. To verify this claim, we reproduce the benchmark examples from *12 basic principles of animation*, as shown in Figure 9, using very few input drawings (at most 4).

Figure 8: *Our inbetweening method uses local-global solver and interpolates shapes with extreme deformation, also shown by Chen et al., without any artifacts.*

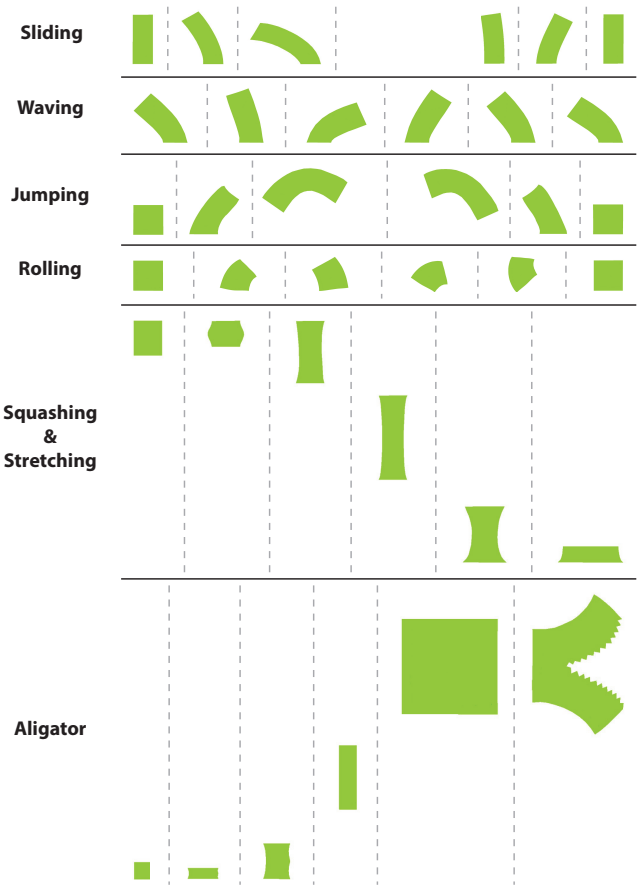

Figure 9: *We evaluate our animating method by reproducing animation benchmark tasks from the 12 basic principles of animation. We achieve almost the same animation results using just few input drawings (2 to 4).*