N-way morphing for 2D Animation

William Baxter, Pascal Barla, Ken-Ichi Anjyo

N-way morphing for 2D animation

Papers ID #8

Abstract

We present a novel approach to the creation of varied animations from a small set of simple 2D input shapes. Instead of providing a new 2D shape for each keyframe of an animation sequence, we instead interpolate between a few example shapes in a reduced pose-space. Similar approaches have been presented in the past, but were restricted in the types of input or range of deformations allowed. In order to address these limitations, we reformulate the problem as an N-way morphing process on 2D input bitmap or vector graphics. Our formulation includes an N-way mapping technique, an efficient, rigidity preserving non-linear blending function, improved extrapolation, and a novel scattered data interpolation technique to manage the reduced pose-space.

The resulting animations are correlated to paths in the reduced pose-space, allowing users to intuitively and interactively control temporal behaviors with simple gestures. We demonstrate our techniques in several example animations.

Keywords: rigid interpolation, multi-way interpolation, morphing, deformation, pose-space, 2D animation

1 Introduction

The creation of attractive 2D animations has always been a skill-demanding and time-consuming process. Software such as ToonBoom© and Adobe Flash© provide some assistance by creating animations from a reduced number of *keyframes*, generating inbetween images automatically in specific cases: for instance, deforming a template shape via handles, or transforming simple geometric primitives. Even with these tools, the tasks of ordering, creating and laying out in time the numerous keyframes that compose an animation is left to users, and requires significant skill and time.

These pressures are leading the animation industry away from traditional hand-drawn techniques towards a pipeline based on parameterized 2D and 3D models. The move to parameterized models reduces production costs, and enables some new possibilities (such as enhanced camerawork in the case of 3D models) but often these gains come at the expense of the real core values of animation, like character and expression, and are at odds with well-understood principles like the importance of strong, easily read silhouettes [1]. Parameterized models inevitably mean loss of freedom and expressiveness. It is simply not possible to encode the freedoms of pencil and paper in any finite number of parameters.

In this paper, we explore an alternative approach for the creation of 2D animations which we believe strikes a compelling balance between the flexibility and expressiveness of hand-drawn animations, and the expedience and cost-savings of parametric models. It addresses many of the limitations of the manual keyframe-based approach by factorizing key-frames into a smaller number of base poses. We identify 2D input shapes as points in a reduced pose-space, and new shapes are created by interpolating from the given data at an arbitrary point in this space. The two main issues of such an approach are thus 1) defining a reduced pose-space that is intuitive and efficient to browse, and 2) defining an N-way interpolation method applicable to arbitrary 2D input shapes. Our work builds on previous techniques that took a similar reduced pose-space approach (e.g. [2, 3, 4]). However, as explained in Section 2, these methods only partially addressed the key issues pertinent to 2D animation.

The main contribution of this paper is thus to adapt and improve various pose-space techniques, with the demands of 2D animation in mind. To this end, we formulate 2D animation as an *N-way morphing* problem: first, base poses are put in correspondence via an *N-way mapping* (Section 3); second, they are manually arranged in a reduced, 2D pose-space, and combined via rigidity-preserving *N-way interpolation* (Section 4). The flexibility of our algorithms allows the creation of a more varied range of 2D animations than previous methods (Section 5) via a technique that is simple for the animator. Moreover, as with other pose-space methods, our approach enables the creation of animations through the interactive exploration of various combinations of base poses, which can be used even by non-animators to create animated results. These techniques also can serve as a means for data reduction by eliminating many keyframes, which can benefit interactive applications like games or animation streaming applications where bandwidth matters.

2 Previous work

The idea of morphing between a set of base poses to produce animations has first been suggested by Alexa et al. [5]. However, their *morphing space* grows in dimensionality with increasing number of base poses, and they provide N-way interpolations via recursive 2-way linear interpolations. This approach is thus neither intuitive nor efficient in practice. The *simplicial families of drawings* of Kovar and Gleicher [6] partly address these limitations by using a tessellated version of pose-space, which enables simple linear N-way interpolation. However, the dimensionality of space still grows with the number of base poses; and linear interpolation often produces undesired results that must be manually discarded.

To address the latter issue, Alexa et al. [7] have introduced an *as-rigid-as-possible interpolation* approach that produces much more plausible animations. It has been improved in the context of 2D animation by Baxter et al. [8], in particular to provide local user control. Both methods are limited to 2-way interpolation, however. Bregler et al. [9] extended the method to more than two dimensions by densely populating pose-space with 2-way rigidly interpolated shapes, and using linear interpolation afterwards. However, as in [6], this approach generates many invalid shapes, and as a result browsable regions must be restricted. Xu et al. [10] propose a 2-way rigid interpolation method and extend it to N-way in one of their examples, but this is done as in [5] by repeated 2-way blends, which is inefficient and leads to an order-dependent result.

To the best of our knowledge, no previous work has proposed a pose-space approach which gives intuitive and efficient interpolation behaviors for a set of arbitrary input 2D shapes. Some methods have addressed the problem by restricting the allowable type of inputs. For instance, early work in image morphing [11] used multiple input images, constraining deformation to be applied to a regular grid. More recently, the *latent doodle space* of Baxter et al. [4] proposed an intuitive approach whereby line drawings are interpolated in a pose-space of reduced dimension (called a latent space). The method enables easy browsing and exploration of a subspace of poses. But it is limited to line drawings with the same number of lines and may give unnatural results since curves are linearly interpolated.

Other approaches have dealt with more general shapes by considering deformations of a template model. For instance, the *spatial keyframing* method of Igarashi et al. [12] animates 3D objects composed of simple blobby parts with a pose-space approach. A similar animation technique is employed by these authors to animate 2D shapes using their *as-rigid-as-possible shape manipulation* technique [3]. Given a fixed number of handles, they create multiple poses that are animated by interpolating handles' positions. However, the range of deformation is limited with this approach, as shown in Figure 1. Moreover, it is not always clear how to interpolate the handles to achieve plausible interpolations overall. In contrast, the *mesh inverse kinematics* of Sumner et al. [13] directly interpolates between multiple meshes. However, these are still obtained by deforming a template model, and the pose-space is not browsed directly, but with a non-linear inverse kinematics approach.

As a summary, there are two main issues with previous approaches that strongly limit their application to the context of 2D animation: they impose restrictions either on the type of inputs or on the range of deformations. Dealing with arbitrary shapes as input requires establishing an N-way mapping between them. Unfortunately, existing techniques (e.g., [14, 15, 16]) only permit the creation of 2-way mappings in practice. We propose N-way extensions to the compatible triangulation method of Baxter et al. [16] to overcome this limitation (Section 3). Similarly, as-rigid-as-possible interpolation is defined for a pair of 2D shapes in [7, 10], while N-way interpolation is required in our case. To address this issue, we present an efficient N-way interpolation technique that preserves rigidity and offers improved extrapolation capabilities, along with a novel scattered data technique for organizing a 2D reduced pose-space (Section 4).

3 N-way mapping

We first establish a N-way mapping between the input shapes provided by the user. To this end, we build a compatible triangulation, a process often called the *vertex correspondence* problem: each input shape is tessellated in a way that ensures that each vertex has a corresponding vertex in all other shapes, with consistent neighborhoods. Our approach is inspired from [16] and is presented in Section 3.1. We also discuss another approach to populate pose-space using deformation techniques in Section 3.2. This can be used by itself or to create variations on the base poses obtained by compatible triangulation.

3.1 Mapping via compatible triangulation

Our approach to compatible triangulation is a direct extension of the recent work of [16] to the case of more than two input shapes. Like in their approach, we decompose the problem in three steps: 1) establish correspondences between shape boundaries; 2) simplify these boundaries while keeping their correspondences and ensuring the original shapes are properly embedded; and 3) triangulate the interiors compatibly. The system works both with bitmap input from which we extract a boundary with conventional image processing software, or from shapes directly drawn in vectorial format with a clear outline.

Boundary matching: The outcome of the boundary correspondence algorithm presented in [16] between a pair of shape boundaries B_0 and B_1 is a mapping $f : [0, 1] \to [0, 1]$. The user gives a first correspondence point to initialize the algorithm, and a few more correspondences if the mapping needs to be refined locally. We extend this approach to N-way matching by requiring that all shapes $B_i, i > 0$ be matched to B_0. This way, we make use of B_0 as a common parametrization domain for all input shapes. In practice, B_0 often represents a neutral pose that is located at the origin of pose-space, and bears some resemblance with the reference pose commonly used for character skinning.

Compatible simplification: Next, we need to extend this N-way mapping between boundaries to the interiors using a compatible triangulation algorithm. However, this should not be done at full boundary resolution, otherwise the resulting meshes would contain many tiny triangles, impeding performance in practice. For this reason, Baxter et al. [16] use a compatible boundary simplification algorithm that reduces the number of boundary vertices while maintaining enclosure of the original shape. It consists of a greedy algorithm that removes vertex pairs (one on each boundary) iteratively. At each step, the removed pair is the one that minimizes a quadratic approximation error that retains the enclosing property [16]. The algorithm terminates when the overall error exceeds a user-specified threshold. For N-way simplification, the extension is straightforward: we simply look at the minimum over all the compatible boundaries to decide which set of N corresponding vertices to remove.

Compatible triangulation: After these two steps, we are left with a set of N compatible boundaries at a reasonable resolution. The 2-way approach in [16] works with a divide-and-conquer strategy, by iteratively partitioning boundary polygons in each shape in a compatible way until the remaining polygons are triangles. We summarize the three stages of their approach and the way we extend them to N-way operation below:

- Stage I: Find all pairs of vertices (i, j) in the shapes that have direct line-of-sight, and record these in a matrix as having a distance of 1. In the 2-way algorithm, if a pair is marked in both matrices, then it is a valid compatible partition. For N input shapes, we use N matrices, and apply the same criterion. If no such pair is found, then move to Stage II.

- Stage II: Find all the pairs of vertices (i, j) which can be connected via a single internal Steiner point in each shape. Record these in the distance matrix as having a distance of 2. Again check if any pair exists in both matrices, if so, partition with that. The extension to N-way is similar. Move to Stage III in the rare cases this does not work.

- Stage III: Find all pairs of vertices that can be connected using *any* number of steps by running a Floyd-Warshall all-pairs-shortest-paths algorithm on each distance matrix. Choose a vertex pair (i, j) that uses the fewest Steiner vertices across all shapes. For N shapes, find the (i, j) pair that minimizes the number of Steiner vertices over all N shapes.

After a compatible triangulation has been obtained, we use an iterative angle-based smoothing, combined with compatible edge flips to improve the overall minimum angles of triangles as in [15]. It generalizes to N compatible meshes in a straightforward manner.

The important observation is that Baxter et al.'s algorithm generalizes quite easily and efficiently to N-way operation. Most sets of similar shapes can still be triangulated with very few Steiner vertices, as can be seen in Figure 2 (top row). This is not the case with other triangulation methods like the technique of Aronov et. al [14]. A straightforward N-way extension of that method will create $O(|V|^N)$ Steiner vertices, making its use impractical.

3.2 Mapping via deformation

Another convenient way to create compatibly triangulated shapes is simply to deform a base mesh. Any 2D mesh or spatial deformation technique could be used. We have implemented the as-rigid-as-possible deformation technique of Igarashi et al. [3] because it produces results visually similar to as-rigid-as-possible interpolations.

This approach offers some additional flexibility over typical deformation-based animation techniques. When animating using free form deformation, or a handle-based technique, it is typically necessary that the constraints (cage or handles) remain consistent throughout the animation. The same handles and deformers must be used all along, ultimately resulting in many handles to specify at every point of the animation sequence. Our interpolation-based animation, on the other hand, is uniformly applicable to all key poses, regardless of how they were created.

As a result, our rigid interpolation scheme creates pleasing in-between motion between any poses without having to define an interpolation scheme for the underlying parameters that created the deformation. This deformation technique is useful with a single base pose, or for creating variations on several base poses generated via compatible triangulation, as shown in Figure 2 (bottom row) and Figure 3.

4 N-way rigid interpolation

Once input shapes have been put in correspondence, they are arranged as points in a 2D pose-space. The influence of each base pose relative to an arbitrary location in pose-space is then given as a weight $w_i \in [0, 1]$, with $\sum_i w_i = 1$. The problem of interpolating between base poses B_i using weights w_i is often referred to in the geometric interpolation literature as the *vertex trajectory* problem. As-rigid-as-possible interpolation techniques have proven to produce very plausible trajectories, but existing N-way approaches [10, 13] work in N-dimensional pose-spaces. In contrast, our N-way interpolation technique works in a reduced, 2D pose-space. We organize our approach in two steps: 1) for an arbitrary point in our reduced pose-space, we determine the set of weights for each base pose with a sparse interpolation technique (Section 4.1); and 2) we produce a new shape using a weighted combination of base-poses that favors rigid interpolations and extrapolations (Section 4.2).

4.1 Biharmonic weight interpolation

The most straightforward approach to determine the weights of each base pose is to organize them in an N-dimensional space and use linear interpolation. However, as noted previously, this approach greatly reduces the creative and intuitive potential of pose-space approaches because users cannot easily explore the different combinations. For this reason, we prefer to use a two-dimensional pose-space, and we let users position base poses as points in this space, similar to [2, 12].

In such a 2D pose-space, base pose weights can no longer be linearly interpolated. We thus turn to scattered data interpolation techniques to determine the set of weights at a particular location. Techniques such as thin-plate spline radial basis functions (RBF) [17, 12, 4] create smooth (C^2) interpolations, but they have a limitation in our context: they need at least 3 non-collinear base poses. For some animations, though (see Section 5), a linear arrangement of key poses may be exactly what is desired.

We present a new mesh-based interpolation scheme which overcomes this issue and offers some other interesting new possibilities, as well. The basic observation is that minimizing the differences in gradients on a mesh in the least-squares sense serves as a discrete approximation to a biharmonic solution. Thus the result behaves much like a thin-plate spline interpolant [18], but by relying on a least-squares framework it becomes easy to incorporate secondary, even spatially-varying, objective functions. Note that the goal of this interpolation is solely to determine an N-dimensional weight vector to use for interpolating the N input meshes. The mesh interpolation itself will be described in the next section.

The first step is to tessellate the pose-space. Any tessellation will do, though one with evenly-sized triangles is preferred due to the simple second derivative approximation implied by (1), below. We currently tessellate using Shewchuk's Triangle [19], though a regular tessellation would also work. Let $|V|$ represent the number of vertices in the mesh, and $\mathcal{E}(\mathcal{T})$ be the set of triangle pairs that share an edge. The basic equations we minimize are:

$$\sum_{j,k \in \mathcal{E}(\mathcal{T})} \|J_j(W) - J_k(W)\|^2, \tag{1}$$

where $J_j(W)$ is the Jacobian of triangle j expressed as a linear function of the data values, $W \in \mathbb{R}^{|V| \times N}$ (row-vector convention for data values). Let $P_{\{a,b,c\}} \in \mathbb{R}^2$ be the points in the base triangle and $W_{\{a,b,c\}} \in \mathbb{R}^N$ be the corresponding N-dimensional data values in W, then:

$$J_j(W) = \begin{bmatrix} P_a - P_c \\ P_b - P_c \end{bmatrix}^{-1} \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \end{bmatrix} \begin{bmatrix} W_a \\ W_b \\ W_c \end{bmatrix} \tag{2}$$

In addition to the above energy terms, we enforce the N-dimensional data points as constraints at their given locations in the reduced pose-space. In this work we have used symbolic elimination to enforce these constraints, though penalty energies or Lagrange multipliers would also work.

Finally, in order to ensure a non-singular system matrix, we add a secondary objective function with a small weight to encode a preference for zero gradients in the absence of other constraints: $\sum_i \varepsilon \| J_i(W) \|^2$. Without this, the matrix is ill-conditioned given collinear or nearly-collinear constraint points. This extra term also serves to eliminate the unbounded extrapolation which is typical of standard thin-plate spline RBF interpolations. Near the data, the result of this interpolation is visually very similar to thin-plate spline RBF, but far from the data our solution levels out, more like a Gaussian RBF. An example interpolation of several data points, visualized as a color interpolation, can be seen in Figure 3.

4.2 N-way rigid pose interpolation

Our pose-space interpolation technique is an N-way generalization of the 2-way interpolation defined by Alexa et al. [7] using the weights w_i computed in the previous section. It can also be seen as a 2D adaptation of the nonlinear blending technique presented in Sumner et al. [13]. The interpolation involves solving a least-squares problem in which the gradients of the mapping between individual shape triangles are interpolated non-linearly. Alexa et al. [7] observe that best results are obtained by separating the rotation component from the remaining scale and shear components and interpolating these two separately. The shear matrices are interpolated directly, but the rotation is interpolated linearly by angle. This can be generalized to N-way operation by turning the 2-way interpolations of these components into N-way convex mixtures. Let $A_{\tau,i} = R(\alpha_{\tau,i})S_{\tau,i}$ be the polar-decomposed matrix that transforms triangle τ in the base pose to the corresponding triangle in shape i [20]. Here $\alpha_{\tau,i}$ is the rotation angle that rotates the triangle in the base pose into best alignment with the same in shape i, and similarly $S_{\tau,i}$ is the shear matrix. Naturally, if shape i is the base pose, then $\alpha_{\tau,i} = 0$ and $S_{\tau,i} = I$. Given blend weights $w = \{w_i\}$, the per-triangle target transformation matrix $A_\tau(w)$ is computed as

$$\alpha_\tau(w) = \sum_i w_i \alpha_{\tau,i}$$

$$S_\tau(w) = \sum_i w_i S_{\tau,i}$$

$$A_\tau(w) = R(\alpha_{\tau,i}(w))S_{\tau,i}(w).$$

Given these target transforms, the least-squares problem for vertex locations can now be solved exactly as in [7], by minimizing $\sum_\tau \|J_\tau(V) - A_\tau(w)\|^2$. An example result along with a comparison with linear interpolation are shown in Figure 3.

Shear limiting: Extrapolation is usually problematic. In this case, by extrapolation we mean going outside the $(N-1)$-simplex defined by the N input shapes, where $w_i \notin [0,1]$ for one or more w_i. We observe that it is primarily the extrapolation of the shear/scale matrix that leads to problems. Thus we propose shear limiting as a means of improving the ability of mesh interpolation schemes to extrapolate beyond the given data.

Shear limiting can be accomplished by merely clamping each weight to $[0,1]$, then renormalizing. But this leads to discontinuous motion at the extremes. A better approach is to remap the weights using a smooth function, $\phi(t)$.

$$w_i' = \frac{\phi(w_i)}{\sum_j \phi(w_j)},$$

where ϕ is a function satisfying $\phi(0) = 0$, $\phi(1) = 1$, $d\phi/dt|_0 = 0$, $d\phi/dt|_1 = 0$. In our system, we use a Hermite cubic curve. Note that we still use the original, unmodified weights to interpolate rotations. An example of shear limiting can be seen in Figure 4.

Texture blending: Extrapolation is also problematic for texture blending. Colors will look either washed out or dark if extrapolation is used. Another issue is that gradual, linear texture transitions often look poor when applied to inputs with strong, high contrast edges like cartoons. To handle these issues we remap the texture-blending weights using soft clamping plus a variable-width transition, determined by a parameter $\sigma \in [0,1]$. We start from the w'_i weights above, then compute a sharpened transition as

$$\begin{aligned} w^*_i &= \sigma\, \phi\left(w'_i, \frac{\sigma}{2}, 1 - \frac{\sigma}{2}\right) + (1-\sigma)w'_i \\ w''_i &= \frac{w^*_i}{\sum_j (w^*_j)}, \end{aligned}$$

where $\phi(t, a, b)$ is a smoothstep function that maps inputs on $[a, b]$ smoothly to $[0, 1]$. This gives a simple linear transition for $\sigma = 0$ and approaches a step function as $\sigma \to 1$.

5 Discussion and future work

We have presented several techniques for creating 2D animations using N-way morphing. Figures 2–4 show various results from our techniques. The quality of animations is difficult to convey through still images, so we encourage the reader to also view the examples in our supplemental video.

We have demonstrated how with a few fairly small changes to existing methods, one can generate much richer 2D animations than was previously possible with morphing. We believe these techniques will prove useful for a variety of purposes: for animating crowds or secondary characters in 2D feature animations, for low-bandwidth, controllable expressive avatars for networked virtual worlds, and for the animatics used in the pre-visualization of feature films.

However, there are still many ways to improve upon these results. For the user, a way to directly specify N-way boundary correspondences, instead of pair-wise correspondences, would simplify the workflow. Our compatible triangulation has difficulty with anything but simple polygons as input (loops and degenerate slivers are not accepted). Interior correspondences are also currently problematic. Furthermore we make no attempt to warp textures to improve the blending, relying only on the coarse geometry to achieve approximate correspondence. We would like to address these issues in future work.

Another major limitation of the current work is the reliance on rigidity-preserving interpolation. Objects which are rigid in 3D do not generally preserve that rigidity in 2D projection. Hence, our animations tend to be limited to motions in the camera plane. More flexible controls over local scaling are needed to overcome this. Another significant limitation is that it is difficult to use drawings with overlapping parts, such as an arm drawn in front of the chest. One could manually separate these drawings into distinct layers, but this is tedious. We are investigating ways to automate this process.

Future directions: Control over continuity is an important aspect of animation creation; however, with only the ability to morph between two shapes at a time, works such as [7, 16] were limited in their ability to create long sequences of smooth, keyframed animations. To create a continuous curve, you must gather contributions from more than two samples. Our new techniques enable this, thereby opening the door to longer, more complex morphing-based animations in the future.

The ability to add objective functions to our least-squares biharmonic scattered data scheme opens new possibilities, such as locally controlling the behavior of interpolation via user-interactions. Also the mesh based nature of the algorithm means it could be applied to interpolation problems with more complex topology than a 2D plane. We plan to investigate such extensions.

References

[1] Richard Williams. *The Animator's Survival Kit.* faber and faber, 2001.

[2] Peter-Pike J. Sloan, III Charles F. Rose, and Michael F. Cohen. Shape by example. In *SI3D '01: Proceedings of the 2001 symposium on Interactive 3D graphics*, pages 135–143, New York, NY, USA, 2001. ACM Press.

[3] Takeo Igarashi, Tomer Moscovich, and John F. Hughes. As-rigid-as-possible shape manipulation. *ACM Trans. Graph.*, 24(3):1134–1141, 2005.

[4] William Baxter and Ken-ichi Anjyo. Latent doodle space. *Computer Graphics Forum*, 25(3):477–485, 2006.

[5] Marc Alexa and Wolfgang Mller. The morphing space. In *Proceedings of WSCG 99, Plzen*, pages 329–336, 1999.

[6] Lucas Kovar and Michael Gleicher. Simplicial families of drawings. In *UIST '01: Proceedings of the 14th annual ACM symposium on User interface software and technology*, pages 163–172, New York, NY, USA, 2001. ACM Press.

[7] Marc Alexa, Daniel Cohen-Or, and David Levin. As-rigid-as-possible shape interpolation. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 157–164, New York, NY, USA, 2000.

[8] William Baxter, Pascal Barla, and Ken Anjyo. Rigid shape interpolation using normal equations. In *Proc. Non-Photorealistic Animation and Rendering*, 2008.

[9] Christoph Bregler, Lorie Loeb, Erika Chuang, and Hrishi Deshpande. Turning to the masters: motion capturing cartoons. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 399–407, New York, NY, USA, 2002. ACM Press.

[10] Dong Xu, Hongxin Zhang, Qing Wang, and Hujun Bao. Poisson shape interpolation. In *SPM '05: Proceedings of the 2005 ACM symposium on Solid and physical modeling*, pages 267–274, New York, NY, USA, 2005. ACM Press.

[11] George Wolberg. Image morphing: A survey. *The Visual Computer*, 14(8):360–372, 1998.

[12] Takeo Igarashi, T. Moscovich, and John F. Hughes. Spatial keyframing for performance-driven animation. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 107–115, New York, NY, USA, 2005. ACM Press.

[13] Robert W. Sumner, Matthias Zwicker, Craig Gotsman, and Jovan Popović. Mesh-based inverse kinematics. *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*, pages 488–495, 2005.

[14] B. Aronov, R. Seidel, and D. Souvaine. On compatible triangulations of simple polygons. *Computational Geometry: Theory and Applications*, 3:27–35, 1993.

[15] Vitaly Surazhsky and Craig Gotsman. High quality compatible triangulations. *Engineering with Computers*, 20(2):147–156, April 2004.

[16] William Baxter, Pascal Barla, and Ken Anjyo. Compatible embedding for 2d shape animation. Technical Report OLMTRE-2008-001, OLM Digital, 2008.

[17] Holger Wendland. *Scattered Data Approximation*. Monographs on Applied and Computational Mathematics. Cambridge University Press, 2004.

[18] Grace Wahba. *Spline Models for Observational Data*. SIAM, 1990.

[19] Jonathan Richard Shewchuk. Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator. In Ming C. Lin and Dinesh Manocha, editors, *Applied Computational Geometry: Towards Geometric Engineering*, volume 1148 of *Lecture Notes in Computer Science*, pages 203–222. Springer-Verlag, May 1996. From the First ACM Workshop on Applied Computational Geometry.

[20] Ken Shoemake and Tom Duff. Matrix animation and polar decomposition. In *Proc. Graphics Interface*, 1992.

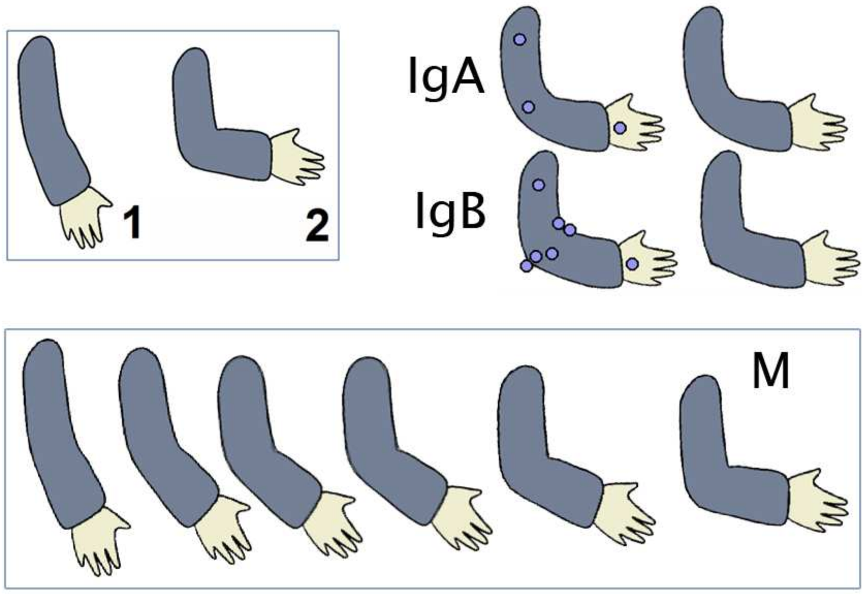

Figure 1: **Morphing vs deformation for animation.** Deformation techniques generally have a difficult time achieving a specific silhouette. Here 1 and 2 are the desired silhouettes. Handle-based deformation is simple, but with a small number of handles (IgA), it cannot match the desired target shape. With more handles (IgB) it comes closer, but managing the handles becomes cubmersome for the user. With our morphing approach (M), the desired silhouettes can be matched exactly.

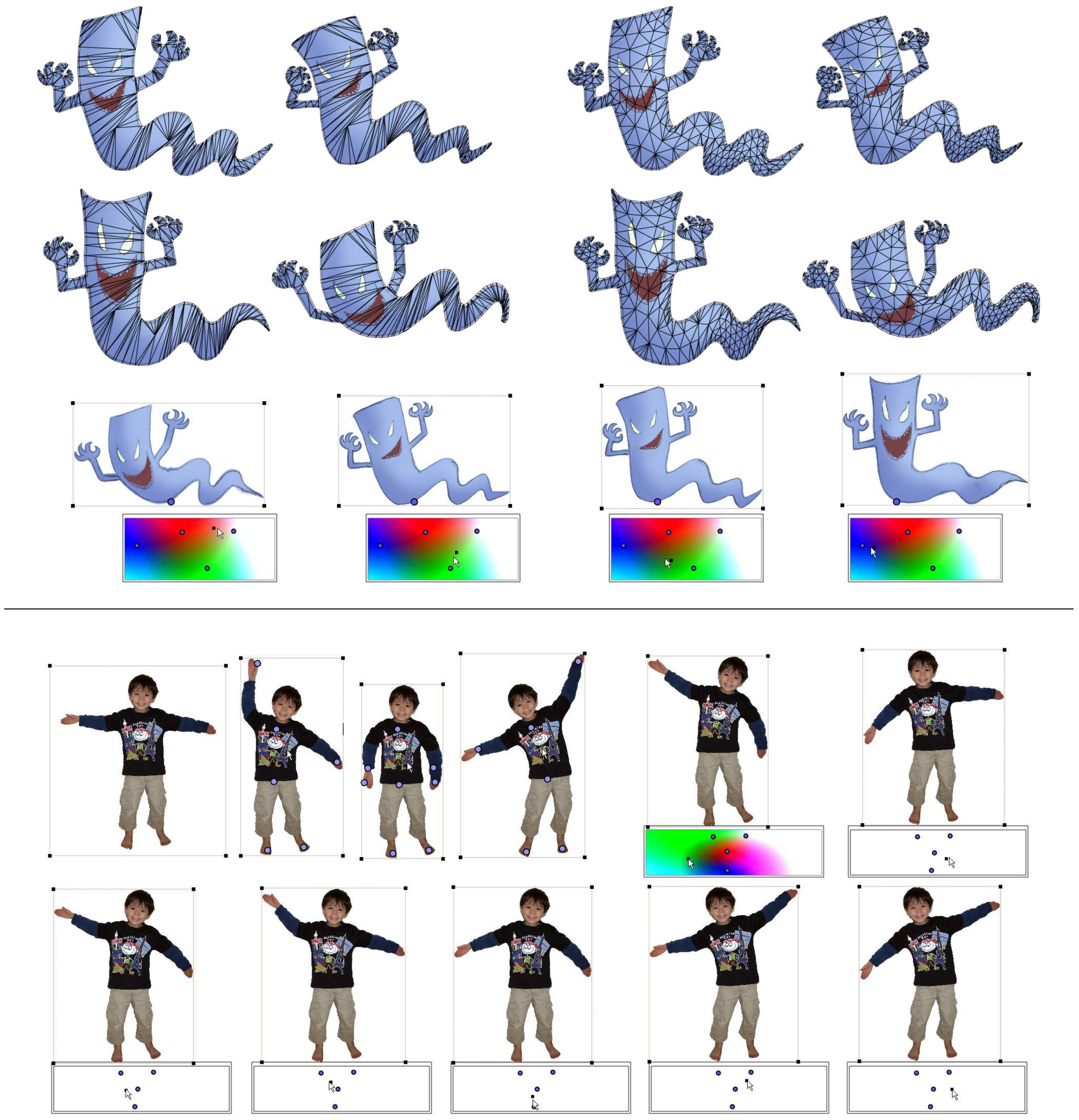

Figure 2: **N-way mapping.** Top: Mapping via compatible tessellation. Four tesselated shapes generated by our algorithm (left). The algorithm added just 7 Steiner vertices. Meshes are then smoothed and refined compatibly (right). Bottom: Mapping via deformation. Four deformations of an input image (top left) provide trivial mappings. Novel poses generated with our technique are shown for both sets of inputs.

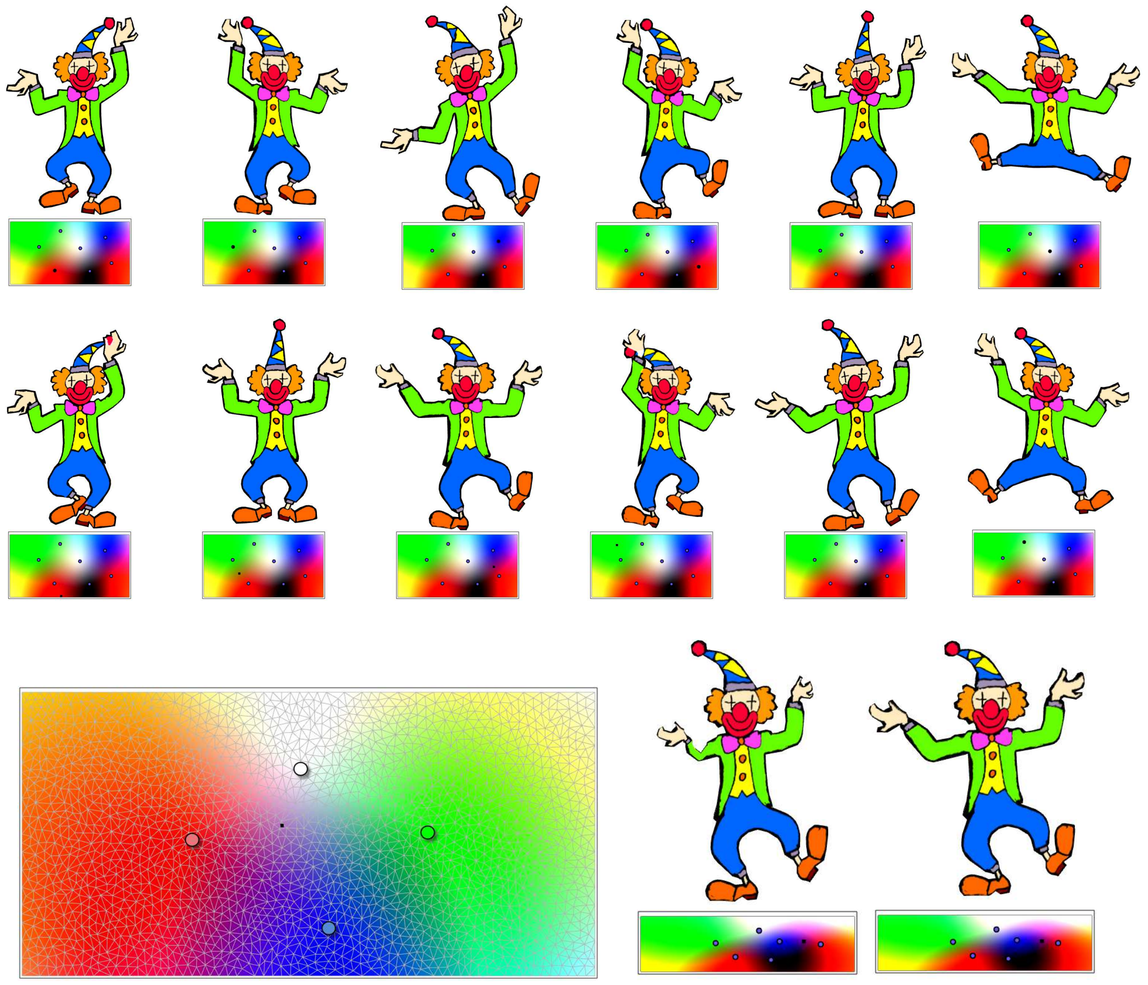

Figure 3: **N-way interpolation.** Top row: two input shapes (left) plus four additional shapes obtained by deformation (right) constitute the base poses animating the clown. Middle row: Novel poses generated with our approach. Bottom row: scattered data interpolation using our mesh-based biharmonic least squares formulation (left), and a visual comparison of linear and rigidity-preserving interpolation (right).

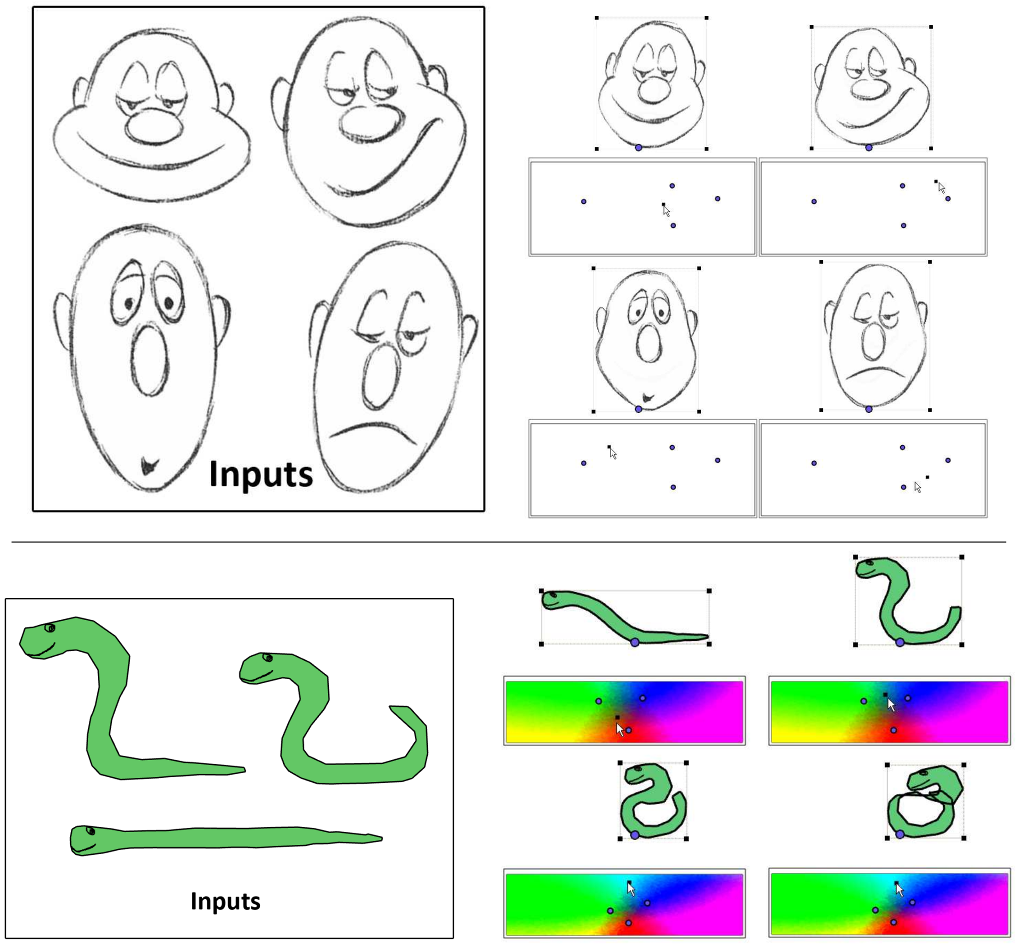

Figure 4: **Additional results.** Top: interpolation results from four different drawings of a character's face. Bottom: interpolation of a vector-graphics snake; note how rigidity-preserving interpolation produces plausible intermediate poses. Extrapolation gives more plausible results with shear limiting than without (bottom right).