# Sketch Based Image Deformation and Editing with Guaranteed Feature Correspondence

Yaqiong Liu[1], Seah Hock Soon[1], Ying He[1], Juncong Lin[2], and Jiazhi Xia[3]

[1] School of Comoputer Engeering at Nanyang Technological University
[2] Xiamen University, China
[3]Central South Univeristy, China

***Abstract*—The establishment of a good correspondence mapping is a key issue in planar animations such as image morphing and deformation. In this paper, we present a novel mapping framework for animation of complex shapes. We firstly let the user extract the outlines of the interested object and target interested area from the input images and specify some optional feature lines, and then we generate a sparse delaunay triangulation mesh taking the outlines and the feature lines of the source shape as constraints. Then we copy the topology from the source shape to the target shape to construct a valid triangulation in the target shape. After that, each triangle of this triangular mesh is further segmented into a dense mesh patch. Each mesh patch is parameterized onto a unit circle domain. With such parametrization, we can easily construct a correspondence mapping between the source patches and the corresponding target patches. Our framework can work well for various applications such as shape deformation and morphing. Pleasing results generated by our framework show that the framework works well.**

***Index Terms*—image morphing, image deformation, image editing, sketched based editing, 2d shape animation, correspondence mapping.**

## I. INTRODUCTION

Planar shape deformation and animation are ubiquitous in a lot of applications. Examples of ongoing use include digital compositing in CG productions, cel animation and web graphics, etc. Despite the voluminous literature on the subject, there remain many ways in which current 2D techniques can be improved upon.

As the basic operation in many 2D applications, image deformation has been an active research area in computer graphics for a long time. Researches in this area could be roughly classified into two categories. The first one is to deform the space in which the target shape is embedded. The work of [1] used a skeleton to define the deform space. Each point in the shape is associated with a coordinate frame defined by a bone. [2] proposed a moving least squares based image deformation technique. Various linear functions including affine, similarity and rigid transformations were used to create realistic deformation while providing closed-form expressions for each of them. Their work was further enhanced in [3] with a sketch-based interface, fold-over reduction and performance acceleration. [4] generalized barycentric coordinates from real numbers to complex numbers and applied the technique for planar shape deformation. Their method produced results superior to state-of-the-art methods at a small extra cost in computational complexity, in pre-processing time only. In [5], the authors described a novel 2D shape deformation system which generated conformal maps, yet provides the users with a large degree of control over the result. The second kind of deformation technique deforms the shape taking its structure into account. These methods usually adopts some physically based models, mostly mass-spring models, to guide the deformation. Instead of using physically based models, [6] achieved an as-rigid-as-possible deformation effect by geometrically minimizing the distortion associated with each triangle in a mesh. Their technique shared certain similarity with the technique used in [7].

Morphing technique can produce compelling transitions between objects through continuous evolution from the source into the target. It is commonly applied in scientific visualization, film animation and advertising industries. There are two well-understood major issues in morphing. The first is how to conveniently establish a reasonable correspondence mapping between the source and the target, known as the correspondence problem. A popular solution to this problem is to let a user firstly prescribe some feature points [8], line segments [9] on both the source and target images. Then the correspondence for every pixel in the image could be further established through mesh-based techniques, radial basis functions [8], and contouring. The second is how to define the way of transforming the source shape transformed into the target shape, known as the path problem. Usually, in particular in the context of image morphing, the path problem is solved using linear interpolation, though more advanced interpolation schemes can be utilized as well. The path problem for two-dimensional polygons was discussed in [10], where interpolation was performed on edge lengths and angles. In [11], the interior of the polygon as well as its boundary were taken into account. In [12] , a different representation of polygons, with a multi-resolution character was proposed.

A key issue in both image deformation and morphing is to construct a mapping either between the shapes before and after editing (deformation) or between the source and target shapes (morphing). Earlier methods usually defined a certain warping function for such mapping. As shape based editing (where the

Manuscript Received on October 10, 2011
E-mail: {Jianlong.Zhou | Ivan.Lee | Bruce.Thomasg}@unisa.edu.au

domain is a complex shape other than a simple regular domain) becomes popular recently [7], [6], some more recent work tends

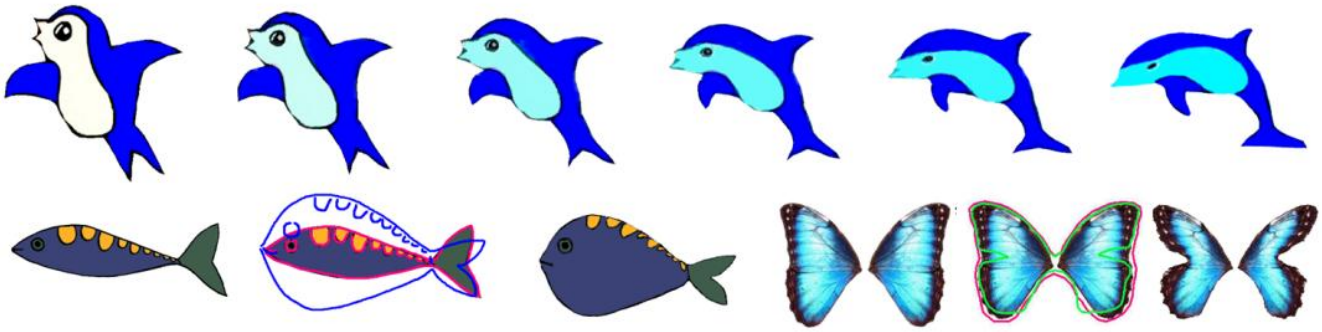

Fig. 1. Feature aligned mapping from 2D shape animation.

to establish a mapping to construct a compatible triangulation. In [13,7], they generated a compatible triangulation by mapping the bounding polygons onto a common domain. These methods are conceptually simple but not optimal in terms of the runtime ($O(n^3)$) or in number of Steiner vertices introduced ($O(n^2)$). In [14,15,16], they used a divide-and-conquer strategy to recursively partition the polygon into triangles. These methods are better in time complexity ($O(n^2 \log n)$) and often without Steiner vertices. However, they are algorithmically quite complex.

In this paper, we present a novel system for shape aware image deformation and morphing (Fig. 1). The key component of our system is a novel framework that constructs a correspondence mapping between the shape before and after editing (for deformation), or between the source and the target shapes (for morphing). Given the input image, we firstly extract the outlines of the interested object and specify some feature lines which can help convey the shape. We then construct a triangular mesh through delaunay triangulation taking the outlines and the feature curves as constraints. The triangular mesh is further segmented into patches. After that, each mesh patch is parameterized onto a unit circle domain. With the parametrization, we can easily construct a correspondence mapping between the patches of the source shape and the patches of the target shape for various applications such as shape deformation and morphing. [17] proposed a detail preserving shape deformation method in image editing. Their image editing system decoupled feature position from pixel color generation by resynthesizing texture from the source image to preserve its detail and orientation around a new feature curve location. They constructed a dense correspondence between the source and target images generated by the control curves and then used this correspondence to synthesize texture. Our framework can be applied for image deformation and image morphing and other applications. Both of [17] and our method can preserve the details of the image after image editing. However, our framework supplies more types of feature constraints for users, for example, points constraint (Fig. 3(b)).

## II. OVERVIEW

In this section, we describe the whole framework of our method as shown in Fig. 2. Given the input image $I$ containing an object $O \subset I$, the user firstly extracts the object's outline (Fig. 2(a)) which is defined as a set of non-intersecting simple closed curves. The outline could be easily extracted with available tools such as lazy snapping [18]. The user can also specify some major features of the shape using strokes. Then the system generates a delaunay triangular mesh taking the outlines and the feature lines as constraints (Fig. 2(c)). Each subregion of this triangular mesh is further divided into mesh patches. As we form a one-to-one mapping of the feature constraints of the source shape and the feature constraints of the target shape, there is a one-to-one mapping between each patch of the source triangular mesh and the corresponding patch of the target triangular mesh. We map each mesh patch onto a common circular domain. Equipped with this parametrization, we can conveniently construct a correspondence mapping for 2D shape animations. For shape morphing, we firstly generate the intermediate contours using linear blending or more complex existing methods [10]. We then generate the intermediate triangular mesh from the source and the target meshes with the corresponding mapping constructed by the third step mentioned above. This triangular mesh has similar patch layout as source and target (Fig. 2(d)). Afterwards we fill in the pixels of the intermediate shape. Similarly, each patch of the intermediate shape is also parameterized onto a unit circle. We can finally construct a mapping among the source shape, the intermediate shape, and the target shape, and each pixel of the intermediate shape will be filled by linear interpolation of the color values of the corresponding pixels in the source and the target shapes. For image deformation, we determine the affected area according to a user's operation. Thus, we can get the shape contours of the affected area before and after users' editing. We then update the triangular mesh part of the affected area and parameterize it onto the unit circle. Hence, we can construct the mapping between the shape before and after editing. The pixels will be transferred to the edited part (Fig. 2(e)).

## III. CORRESPONDENCE ESTABLISHMENT

### *3.1 Guaranteed Feature Correspondence*

Our system allows users to specify feature constraints at the

beginning. These feature constraints include boundaries, points, strokes, and circles (Fig. 3). A boundary is absolutely necessary, for we can identify the interested area through this boundary. For image morphing and editing, after specifying feature constraints of the source shape, users

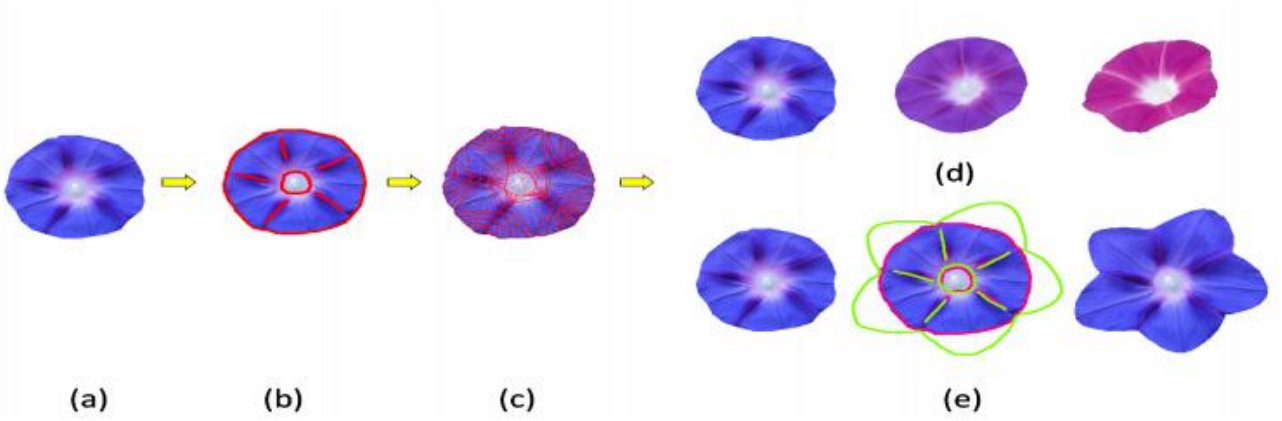


Fig. 2. Flowchart of our 2d shape animation system: (a) source object; (b) feature lines; (c) triangular mesh and patch layout; (d) morphing results sequence: original object, intermediate object, target object; (e) Editing steps: sketched target shape and feature lines, and we obtain the new image.

specify the feature constraints of the target shape in the same order as the source. Therefore, we can get a one-to-one mapping between the feature lines in the source and target shape. After users' operation, our system samples points along each feature constraint with the same quantity automatically. Thus we obtain a one-to-one mapping between a feature point in the source shape and the corresponding feature point in the target shape.

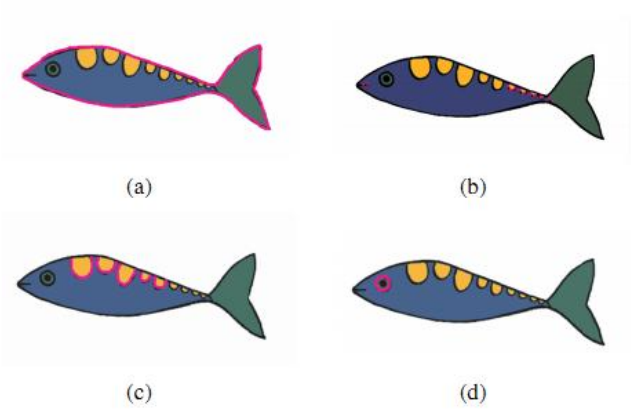


Fig. 3. Different types of feature constraints: (a) boundary; (b) points; (c) strokes; (d) circle.

### *3.2 Patch Correspondence*

Next, our system applies a delaunay triangulation only to the source shape with these sampled points along feature lines. Thus we get a connectivity relationship among these points in the source shape. Then our system automatically connects the corresponding points in the target shape according to the same connectivity. However, only by copying the connectivity topology from the source to the target, we sometimes cannot avoid self-intersection in the triangulation of the target shape. To solve such a problem, our system also allows users to move the position of those sampled points in order to obtain a valid delaunay triangulation of the target shape (Fig. 5(b))(If there are holes in this mesh, each of these polygonal holes is also a rough patch). Therefore, each triangle (or hole) t of the delaunay triangulations can be regarded as a rough patch. As the source and target triangular mesh have the same connectivity topology, they have equal numbers of rough patches and each rough patch in the source shape has one and only one corresponding rough patch in the target shape. Hence, a rough patch-to-patch correspondence has been constructed.

However, each rough patch t is still a relatively large triangle or a polygon. Thus, in addition to the delaunay triangulation which is sparse, we also apply a dense triangulation to each rough patch t of both source and target shape (Fig. 2(c), Fig. 5(c)). This does not change the rough patch-to-patch correspondence. Thus patch correspondence has been constructed (Fig. 4).

## IV. KEY STEPS ANALYSISR

### *4.1 Triangulations*

Our system applies three kinds of triangulations in total. The first triangulation is a Delaunay triangulation applied to the source shape. After copying the connection topology from the source shape to the target shape, we get a rough patch-to-patch correspondence, each large triangle or polygonal hole is a rough patch.we apply a detailed triangulation to each triangle of the Delaunay meshes. We then apply a dense triangulation to each of these rough patches respectively. This is called the second kind of triangulations.

Thus, it appears that we have obtained valid patches based on dense triangulations and obtained a whole dense triangular mesh. However, as we apply a dense triangulation $Tri_1$ to each rough patch separately, there may be different numbers of points on the common edges of two adjacent patches. To solve this problem, we apply a ReTriangulation $Tri_2$ to each patch. The re-triangulation procedure is detailed in Algorithm 1.

*4.2 Constraint Map*

Let Ps ∈ Os and Pt ∈ Mt be the pair of segmented patches, each of which is a genus-0 surface with only one boundary. We want to find a bijective and smooth map Ø : Ps→Pt . Rather than computing the map directly, we first parameterize Ps to the unit disc using harmonic map, i.e., f : Ps → D such that ∆ f = 0 and f maps the boundary of Ps to the boundary of D using arc length parametrization, f (∂Mi) = ∂D. Similarly, we also parameterize Pt to the unit disc using harmonic map g : Pt → D. More details of discrete harmonic map could be Found at [19].

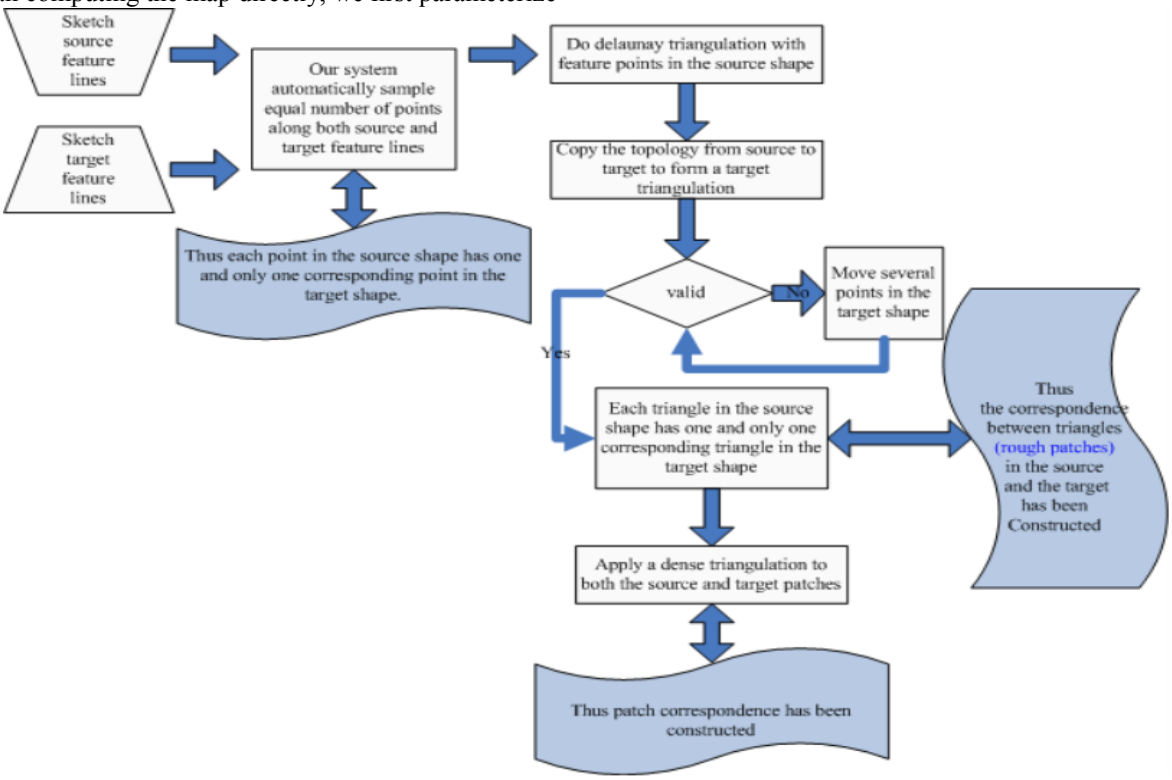


Fig. 4. How patch correspondence is established.

Then we seek a smooth map between two unit discs $h : D \rightarrow D$. This map $h$ is also computed using harmonic map $\Delta h = 0$ and the boundary condition is set as follows: Let $s_1$, $s_2$ and $s_3$ be the sample points on $\partial M_i$, and $f(s_j) \in \partial D$, $j = 0, 1, 2$ be the images on the boundary of unit disc. Similarly, let $g(s'_j) \in \partial D$ be the images of the sample points $s'_j \in P_i$. Then we require the function h maps $f(s_j)$ to $g(s'_j)$, i.e., $h \circ f(s_j) = g(s'_j)$, $j = 0, 1, 2$. The images for the points between $f(s_j)$ and $f(s_{(j+1)\%3})$, $j = 0, 1, 2$, are computed using arc length parametrization.

Finally, the correspondence mapping between the two patches is given by the composite map $\emptyset = f \circ h \circ g^{-1}$ as shown in the following commutative diagram:

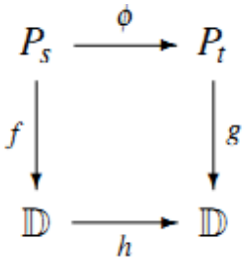


Fig. 6. Demonstrates the results of the constraint map. User can take full control of the correspondence using simple sketches as the boundary condition.

*4.3 Pixel Operation*

After the correspondence mapping between two patches has been constructed, we join all the patches together to form one single triangular mesh for both the source and target shapes, and thus we can get the source mesh S and the target mesh T. At the same time, the correspondence mapping between the source mesh S and the target mesh T can be also obtained.

*1) Shape Morphing:* For shape morphing, to calculate the intermediate shapes between the source shape and the target shape, we need two steps. In the first step, we calculate the outlines of the intermediate shapes. In the second step, we calculate each pixel of the intermediate shape. To determine each point in each intermediate shape, we use the following steps.

Step 1: determine the triangle mesh *M* of the intermediate shape according to the correspondence mapping between the source mesh and the target mesh.

Step 2: for each point *p* in the intermediate shape, determine the specific triangle *t* in which it locates in the intermediate mesh.

Step 3: calculate the barycentric coordinates of *p* about

triangle *t*.

Step 4: find a pair of corresponding points of *p* separately in the corresponding triangle $t_1$ in the source mesh *S* and the corresponding triangle $t_2$ in the target mesh *T*. Let $p_1$, $p_2$ denote these two points respectively.

Step 5: use linear interpolation of the pixel values of $p_1$ and $p_2$ to determine the pixel value of *p*.

These are ordinary steps. Our system gets pleasing results such as human faces, flowers, leaves, fruits, cartoon characters and some other shapes, as shown in Fig. 10.

*2) Shape Editing:* For shape editing, there is no need to calculate the intermediate shape. We can just transfer the pixel values in the source shape to the corresponding pixels in the target shape according to the correspondence mapping between the source mesh *S* and the target mesh *T*. Our system also gets pleasing results as shown in Fig. 11.

**Algorithm 1:** ReTriangulation

**Input**: the initial delaunay mesh without inner holes *S*, initial inner hole patch *H*

**Output**: the new triangular mesh after ReTriangulation *S'*

**for** *each triangle face(a rough patch)* $t_0 \in S$ **do**
  $t_0'$ : the corresponding patch after $Tri_1$;
  *OldBound* : the set of boundary points of $t_0'$ ;
  *InnerPoints* : the points inside $t_0'$;
  *NewBound* : the set of new boundary points;
  $NewBound = \emptyset$;
  **for** $\forall$ *half-edge* $E \in t_0$ **do**
    **if** *E is not a boundary edge* **then**
      $v_{start}$, $v_{end}$ : the two end points of *E*;
      *V* : a set containing all the points along this boundary from $v_{start}$ to $v_{end}$ in $t'$;
      $E_{OP}$ : the opposite half-edge of *E*;
      $t_1$ : the face which $E_{OP}$ belongs to;
      **if** $t_1$ *has already been ReTriangulated* **then**
        $t_1''$ : the corresponding mesh patch after ReTriangulation $Tri_2$;
        $V'$ : the boundary points set from $v_{start}$ to $v_{end}$ in $t_1''$;
        $V = V'$;
        /* Update the boundary to the common boundary edge; */
      **end**
      **else**
        $t_1'$ : the corresponding mesh patch after the first patch triangulation $Tri_1$;
        $V'$ : the set of the boundary points from $v_{start}$ to $v_{end}$ in $t_1'$;
        Let $V = V \bigcup V'$;
      **end**
      $NewBound = NewBound \bigcup V$;
      /* Update boundary points; */
    **end**
  **end**
  Let $NewPoints = NewBound \bigcup InnerPoints$;
  Do Delaunay Triangulation to *NewPoints*;
**end**
**for** *each inner hole H (initial delaunay hole)* **do**
  $H'$ : the corresponding triangular mesh after $Tri_1$;
  *OldBound* : the set of boundary points;
  *InnerPoints* : the set of inner points of $H'$;
  *NewBound* : the set of new boundary points after $Tri_2$;
  $NewBound = \emptyset$;
  **for** $\forall$ *boundary edge E of H* **do**
    $V_{start}$, $V_{end}$ : the two end points of *E* ;
    *F* : the face *s.t.* $E \in F$ ;
    $F''$ : the corresponding patch mesh of *F* after $Tri_2$ ;
    *V* : the set of the boundary points from $V_{start}$ to $V_{end}$;
    $NewBound = NewBound \bigcup V$;
    /* Update boundary points; */
  **end**
  let $NewHolePoints = NewBound \bigcup InnerPoints$;
  Do Delaunay Triangulation with set *NewHolePoints*;
**end**

## V. APPLICATIONS

We now show how to use the correspondence mapping to realize various 2D shape animation effects taking shape morphing and shape deformation as examples.

### *5.1 Shape Morphing*

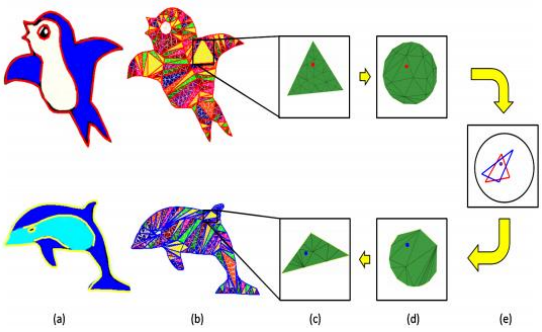


Fig. 5. Realization of morphing with our correspondence mapping method. (a) Morphing shapes; (b) Triangular meshes. Delaunay triangulation applies to both source and target images. Each of the triangles is a triangular patch. (c)The corresponding patches and corresponding pixels; (d) The parametric domains of patches and the image points of the corresponding pixels; (e) determine the pixel correspondence in the parametric domain.

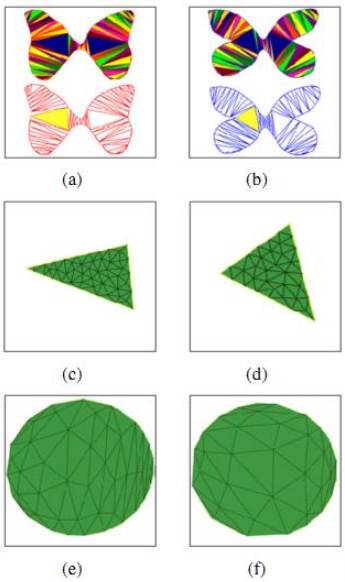


Fig. 6. Constraint mapping of the two patches. (a) source patch layout and take the yellow patch for example; (b) target patch layout and target corresponding example patch; (c) details of this yellow patch; (d) target corresponding patch; (e) parameterize this patch to circular domain; (f) a circular mesh parameterized from the target corresponding patch.

There are two steps in the shape morphing. We firstly generate the outlines and the feature curves of the intermediate shape. This can be easily done with linear interpolation or existing more complex methods like [10]. In the second step, we then fill in the pixels of the shape. For

each pixel of the intermediate shape, we firstly find the triangle it locates in, and compute its barycentric coordinate. We can then find its image in the parametrization domain with the same triangle and same barycentric coordinate. After that, we can find the corresponding pixels in the source and target that have the same image in the parametrization domain. We first determine the source (or target) triangle the image point located in and calculate the barycentric coordinate in the triangle. We can similarly find the pixel in the source (target) shape in a reverse way with the same triangle and same barycentric coordinate. Finally, given the colors of the pixels ps, pt, denoted as c(ps) and c(pt) respectively, the final color of pixel p, c(p), is set to be:

$$c(p) = w \cdot c(p_t) + (1-w) \cdot c(p_s)$$

### 5.2 Image Editing

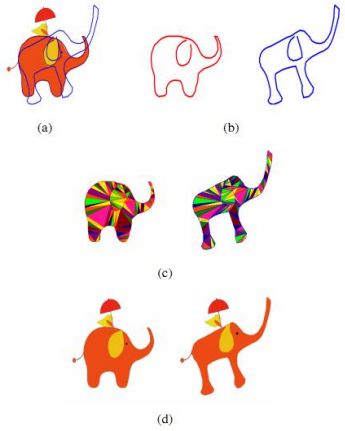


Fig. 7. Implementation of image deformation with our method. As shown in (a), a user sketches the original feature and the target feature on one single image. (b) Our system treats these feature lines the same as the morphing process. (c) shows the delaunay triangulation of the source and target object. (d) shows the deformation result.

Our system also provides a similar sketching interface for image editing [20]. Users simply draw a new boundary shape as the interested editing area. They could also draw interior feature lines as other feature constraints. The system then constructs a triangular mesh of both the original shape and the new shape and parameterize each patch of these two triangular meshes onto a unit circle. Hence, the system obtains a correspondence mapping between the source shape and the target shape. Finally, the interior of the new shape will be filled by the pixels from the original shape through the correspondence mapping. Details of image editing are shown in Fig. 7.

Fig. 8. Our method works well on bunny ears.

## VI. RESULTS ESTIMATION

We have implemented a prototype system of the described method in C++ running on a workstation with an Intel Xeon 2.67-GHz CPU and 8 GB RAM. Our method is conceptually simple and easy to implement. It is also quite efficient. Sparse linear systems and 2D triangulation are involved. The time statistics of the main steps are listed in Table I. Although simple, our method is quite effective and provides pleasing results as shown in Fig. 1. More morphing examples could be found in Fig. 10, and deformation examples could be found in Fig. 11.

Compared to [17], our method can also work well on some figures such as long bunny ears(Fig. 8) with user specified feature constraints. The difference between our method and the method of [17] is that our framework can work well on most of the figures with smooth texture, while [17] applies image synthesis to smooth neighboring patches of figures with uneven texture.

TABLE 1. TIME STATISTICS OF THE MAIN STEPS IN OUR FRAMEWORK.

| Item | Timing (second) | | |
|---|---|---|---|
| | 208 vertices | 381 vertices | 486 vertices |
| The Second Triangulation | 0.062 | 0.073 | 0.081 |
| Patch mapping | 0.070 | 0.082 | 0.098 |

Fig. 9. Example triangulations with different resolutions. The numbers of vertices are 208, 381, 486 respectively.

## VII. CONCLUSION

In this paper, we present a simple yet efficient method that uses guaranteed feature correspondence to realize sketch based image morphing and image editing. For each of the two shapes that need a correspondence mapping between them, we generate a triangular mesh taking the outlines and the feature lines of the shape as constraints. The triangular mesh is further divided into patches. We then parameterize each patch onto a unit circular domain. With the parametrization, we can build the corresponding mapping of each pair of corresponding patches, and then build the corresponding mapping of the whole shapes. The efficiency and efficacy of this method are shown through various morphing and deformation examples.

The main contribution of this paper is that we proposed a method that constructs a guaranteed patch correspondence to edit images from user-controllable feature constraints in an intuitive manner. Thus in computer animation area, there has been a new method that can be used to realize image editing and deformation. Users are able to control the number of the feature curves and feature types. Many results indicate that users do not need to sketch as many feature curves as

possible. Approximate feature constraints are enough.

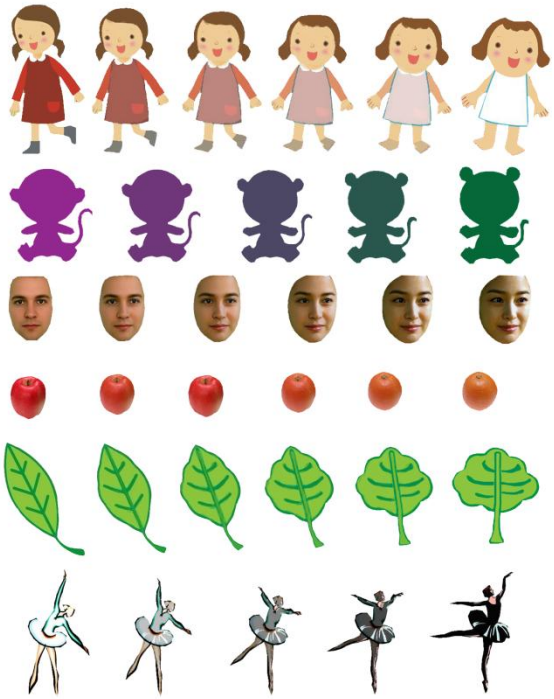

Fig. 10. Morphing results generated with our method (Color Plate 7)

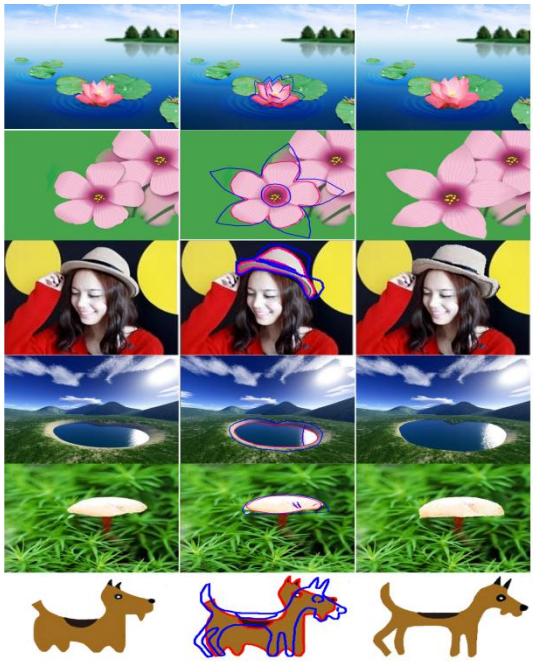

Fig. 11. Deformation examples designed with our framework (Color Plate 8)

However, as mentioned in Section 3.2, copying topology from source to target may cause self-intersection. In practice, this is mainly depended on the feature constraints. Different number of sampled points along the feature lines lead to different triangulations, while in most of the cases, the target triangulation is valid. Thus our system allows users to move the position of the feature points easily and freely if required, to guarantee a valid triangulation. However, this operation may take up users' time. Normally, users can only spend less than one minute to move the position of the feature points because in such occasions, there are only several triangles that intersects each other.

In the future, we will continue focusing on this topic to work out a solution that could avoid self intersection and enable the automatic detection of the feature lines. Meanwhile, we will try to extend the method to 3D animation/morphing, in particular, for the shapes of different topology.

## REFERENCES


[1] J. Lewis, matt Cordner, and N. Fong, "Pose space deformation: a unified approach to shape interpolation and skeleton driven deformation," in Proc of *ACM SIGGRAPH, 2000*, pp. 165–172.

[2] S. Schaefer, T. McPhail, and J. Warren, "Image deformation using moving least squares," *ACM Transacitons on Graphics*, 25(3), pp. 533–540, 2006.

[3] Y. Weng, X. Shi, H. Bao, and J. Zhang, "Sketching mls image deformations on the gpu," *Computer Graphics Forum*, 27(7), pp.1789–1796, 2009.

[4] O. Weber, M. Ben-Chen, and C. Gotsman, "Complex barycentric coordinates with applications to planar shape deformation," *Computer Graphics Forum*, 28(2), 2009.

[5] O. Weber and C. Gotsman, "Controllable conformal maps for shape deformation and interpolation," *ACM Transacitons on Graphics*, 29(4), 2010.

[6] T. Igarashi, T. Moscovich, and J. F. Hughes, "As-rigid-as-possible shape manipulation," *ACM Transacitons on Graphics*, 24(3), pp. 107–116, 2005.

[7] M. Alexa, D. Cohen-Or, and D. Levin, "As-rigid-as-possible shape interpolation," in Proc of *ACM SIGGRAPH 2000*, 2000, pp. 157–164.

[8] N. Arad and D. Reisfeld, "Image warping using few anchor points and radial functions," *Computer Graphics Forum*, 14(1), pp. 35–46, 2003.

[9] T. Beier and S. Neely, "Feature-based image metamorphosis," in Proc of *ACM SIGGRAPH*, vol. 26, 1992, pp. 35–42.

[10] T. W. Sederberg, P. Gao, G. Wang, and H. Mu, "2d shape blending:an intrinsic solution to the vertex path problem," in Proc of *ACM SIGGRAPH*, vol. 27, 1993, pp. 15–18.

[11] M. Shapira and A. Rappoport, "Shape blending using the star-skeleton representation," *IEEE Computer Graphics and Applications*, vol. 15, pp. 44–50, 1995.

[12] E. Goldstein and C. Gotsman, "Polygon morphing using a multiresolution representation," in *Proc of Graphic Interface, 1995*, pp. 247–254.

[13] A. Tal and G. Elber, "Image morphing with feature preserving texture," *Computer Graphics Forum, 18(3)*, pp. 339–348, 1999.

[14] V. Surazhsky and C. Gotsman, "High quality compatible triangulations," *Engineering with Computers, 20(2)*, pp. 147–156, 2004.

[15] H. Gupta and R. Wenger, "Constructing piecewise linear homeomorphisms of simple polygons," *Journal of Algorithms*, 22(1), pp. 142–157, 1997.

[16] W. Baxter, P. Barla, and K.-i. Anjyo, "Compatible embedding for 2d shape animation," IEEE Transactions On Visualization and Computer Graphics, 15(5), pp. 867–879, 2009.

[17] H. Fang and J. Hart, "Detail preserving shape deformation in image editing," ACM Transactions on Graphics (TOG), vol. 26, no. 3, pp. 12–es, 2007.

[18] Y. Li, J. Sun, C.-K. Tang, and H.-Y. Shum, "Lazy snapping," ACM Transacitons on Graphics, vol. 23, no. 3, pp. 303–308, 2004.

[19] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle, "Multiresolution anaylsis of arbitrary meshes," in Proc of *ACM SIGGRAPH, 1995*, pp. 173–182.

[20] M. Eitz, O. Sorkine, and M. alexa, "Sketch based image deformation," in Proc of *Proceedings of Vision, Modeling and Visualization, 2007*, pp.135–142.

**Yaqiong Liu** is currently a PhD candidate in School of Comoputer Engeering at Nanyang Technological University. She received her B.S. in Computer Science & Technology and Financial Management from Tianjin University in China. Her research interests are computer graphics, image animation, information retrieval and data mining.

**Hock Soon Seah** is a full professor of School of Computer Engineering at Nanyang Technological University. He is director of GameLab in School of Computer Engineering. His research areas include computer vision research in tracking, extraction of camera trajectory, and 3D reconstruction from image sequences for augmented reality and automating 2D/3D computer graphics for the film/video industry. For more information, please go to page http://www.ntu.edu.sg/home/ashsseah/.

**Ying He** is currently an assistant professor of School of Computer Engineering at Nanyang Technological University since 2006. He received his Ph.D. and M.S. in Computer Science from Stony Brook University, USA. He received his M.S. and B.S. in Electrical Engineering from Tsinghua University, China. His research interests fall into visual computing. He is particularly interested in the problems which require geometric analysis and computation. He leads the Geometric Modeling and Processing group.

**Juncong Lin** is currently a faculty at Xiamen University, China. He was a research fellow in the geometric modeling group in the School of Computer Engineering at Nanyang Technological University from Oct 2009 to Sep 2011. He received his B.S degree in Environmental Engineering and Ph.D. degree in Computer Science from Zhejiang University. His primary research interests are computer graphics and user interface.

**Jiazhi Xia** is currently a faculty at Central South Univeristy, China. He received his Ph.D. in Computer Engineering from Nanyang Technological University in Jun. 2011. He received his M.S. and B.S. in Computer Science from Zhejiang University, China.