Massachusetts Institute of Technology

4.s48: Computational Structural Design and Optimization
Caitlin Mueller – Spring 2015

Final project with title:

Design-Space Representation through Blended Transitions

Alexandros Charidis
Massachusetts Institute of Technology
charidis@mit.edu

1. Introduction

In this paper, I present methodologies for shape transformation. In particular, the category of shape transformations that I will be dealing with is commonly known in the Computer Graphics community as shape blending or morphing. Blending or morphing is a computational process that aims to continuously transform or metamorphosize an initial shape, two-dimensional or three-dimensional, to a target shape. The process is popular in the film industry due to the visually appealing “hack” of transforming a figure in an image to that of another image such as transforming the image of a human face to that of another human or animal (Figure 1).

My interest in shape blending originates from questioning what sort of design space can a designer explore between two given designs. I believe that figuring out methods that can support the generation of the space between given designs can aid design exploration at the very beginning, when things “are not settled yet.” This very beginning is sometimes referred to as the conceptual design stage, while a more descriptive term is “pre-parametric” design stage. This stage is important because ideas about material, function and use fluctuate or may have not even fallen into consideration and shape changes in unexpected directions. Shape transformation is also an important subfield of Computer Graphics. Applications that deal with shape analysis, processing or synthesis are vast and they span almost all aspects of the field. I propose to consider Computer Graphics techniques on shape transformations for design space formulation and exploration during the early stages of conceptual design.

In this paper I examine two particular approaches to shape blending. One represents an object as a solid volumetric model and computes its distance field representation. The distance field representation is useful because it can handle topological changes. The second represents the object’s volume as a dissection made of triangles. In this case, I implement a rigidity preservind shape transformation method that transforms one shape into another by minimizing the error between two corresponding vertices. Results of both methods are presented and their usefulness for design applications is discussed.

Ultimately, this paper suggests a body of work that will be useful to practitioners and educators particularly when dealing with collaborative, interdisciplinary research in Computational Design, Optimization and Computer Graphics.

2. Background

Much has been said about morphing both from Architecture and the Computer Graphics community. In Architecture, morphing was introduced somewhere around the end of 1980s as a computational device for form exploration. To the extent of my knowledge, the first attempt to use morphing in an Architectural design setting is reported in [Yessios, 1987]. Rather than creating designs by combining parts in an incremental fashion, morphing starts with two or more complete designs and by applying transformations, one design is mapped onto another through gradual steps. The difficulty of achieving such transitions between complete designs is obvious in Figure 1 as the morphing process introduces artifacts that result in part discontinuity, shape collapsing and overlapping and so on.

Finding methods for overcoming such artifacts and limitations, and coming up with new shape morphing algorithms has been the task of Computer Graphics research until today. The literature on shape morphing is vast and applications span from the film industry, to industrial design, to computer vision and to material science. A distinguished aspect of all research done in shape morphing is the formulation of an appropriate representation of the objects taken into consideration. In fact, the representation of a shape, either in two-dimensions or three-dimensions plays substantial importance on the type and difficulty of the blending algorithm. In theory, there are infinitely many transformations that can take one shape into another, and although slightly outdated, the following sources provide an extensive literature review on the work done in formal shape representation for applications in morphing and transformations [Gomez, 1997; Lazarus and Verroust, 1998;

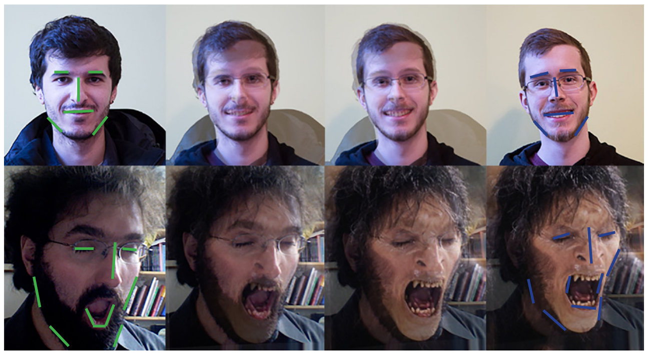

Figure 1: *Feature based morphing between two images. Features such as nose, eyes, head contour, guide the morphing process. The colors of the two faces are blended through linear interpolation.*

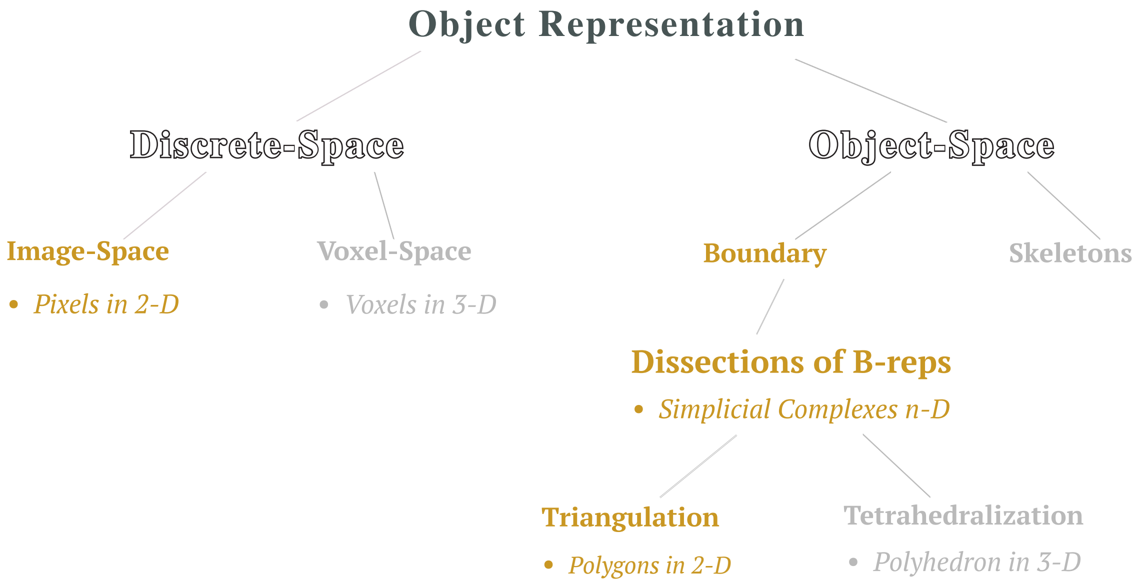

Figure 2: *Left side: Preparation for interpolation of two solid shapes. Binary image is transformed into its distance field representation. The distance field representation is transformed in log-space. Right side: Results from interpolating the start and end shapes for different weights.*

Alexa, 2002].

According to my observations, there are two major branches in formal object representation useful for shape transformation tasks including morphing: discrete-space and object space. A discrete-space representation describes an object as a discretization of space into small parts organized as a matrix such as pixels in the case of images or voxels in the case of three-dimensional solids. An object space representation describes an object as either a boundary or a skeleton. In the case of the boundary representation, objects are represented either with polygons in two-dimensions or polyhedral in three dimensions. In the case of skeletal representation or curve-skeleton representation, an object is represented as a collection of primitives such as "sticks," curves, and patches. Skeletal representations are special in the sense that they capture both explicit (surface mesh) and implicit (zero crossing of implicit function) information about an object [Tagliasacchi 2013]. Figure 2 shows the taxonomy of different representations I observed through literature research. Following is a elaborate analysis specifically concentrated in image morphing and two-dimensional boundary shape transformation. Skeletal based morphing will not be discussed in the context of this paper and hence I omit their literature coverage.

Discrete Space Blending. In discrete space blending, shapes are embedded in two dimensional grid constructs such as images. Any attempt to transform a shape such as the contour that defines the head of a person (see Figure 1) will necessary have to deal with all the pixels in the whole image. In other words, in discrete space blending it is not so easy to distinguish between foreground and background, foreground being a shape of interest, background being the leftover.

To morph one image onto another there is a basic two-fold process: warp generation followed by color interpolation. To generate a warp function a user must specify some features that are common in both images. For example, in [Beier and Neely 1992] warping is done using geometric primitives such as lines that define fields of influence and the displacement of a pixel is a weighted sum of the influence of all the features (see Figure 1 which is an exact implementation of this method). Since the actual matching process is done by the warp function, work in image morphing has been specifically concentrated in developing more elaborate feature specification and warp generation. In particular, some of the proposed methods have been to generalize the idea of features and consider them as just collections of points and then try to fit surfaces that interpolate those points. This idea is based on scattered data interpolation combined with radial basis functions for generating the fitted surfaces [Ruprecht and Müller, 1995; Wolberg 1998]. Warp generation can be also considered as a free form deformation across a hierarchy of control lattices. Control lattices can be constructed using cubic splines that satisfy positional constraints [Wolberg 1998].

Image based techniques produce impressive visual effects but are not quite the right way to approach a problem that gives emphasis on shape information. To blend shapes one needs other representations that are object based.

Object Space Blending. An object based representation essentially treats shapes as collections of vertices and edges. As in image based morphing, shape blending requires solving two problems: the vertex correspondence problem and the vertex path problem [Sederberg et al. 1993; Alexa et al. 2000]. Essentially, research in shape blending focuses on improving either one of these two problems. In the simplest case, the vertex trajectory problem (vertex path problem) can be seen as simply the linear and gradual displacement of a vertex from one location to another location. This approach however will not work in most cases. In [Sederberg et al. 1993] the representation of polygons is based on intrinsic properties such as angles and edge lengths. Rather than vertex location interpolation, the authors propose a geometric algorithm that interpolates edge lengths and angles along with an optimization scheme to ensure that these blended polylines are not self-intersecting. In [Chenet al. 2004] the feature curves of the objects are connected by dependency graphs and the intermediate features are generated by applying this method to the two graphs. In [Shapira and Rappoport 1995] the trajectories of star-shaped pieces are controlled by skeleton via edge-angle interpolation. In [Alexa et al. 2000] rigidity preserving shape interpolation is suggested as the natural human understanding of

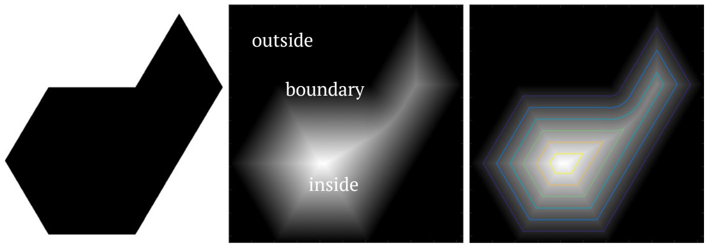

Figure 3: *Distance field representation of a solid two dimensional object. Points (pixels) inside the boundary have positive sign, whereas points (pixels) outside the boundary have negative sign.*

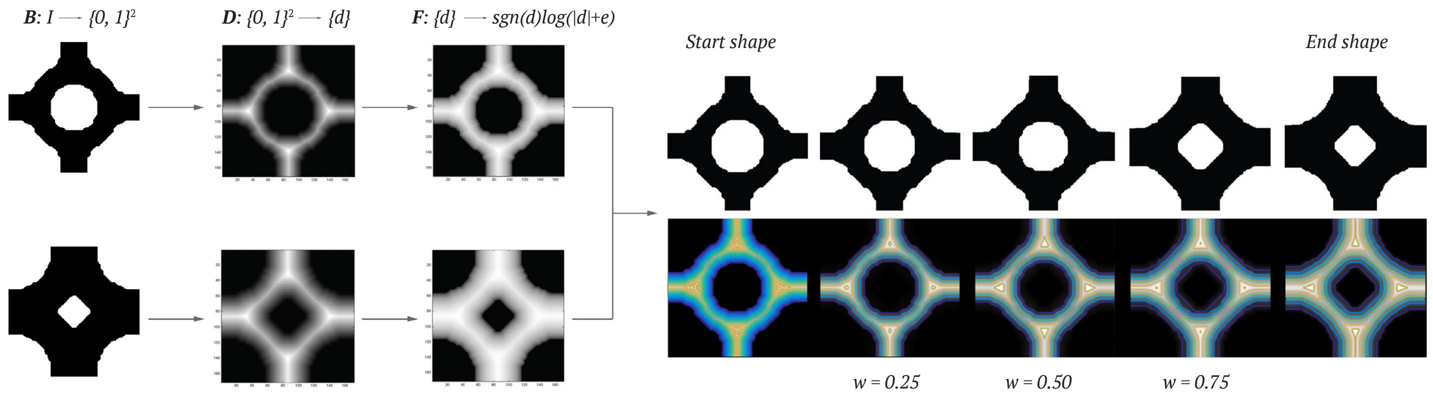

Figure 4: *Left side: Preparation for interpolation of two solid shapes. Binary image is transformed into its distance field representation. The distance field representation is transformed in log-space. Right side: Results from interpolating the start and end shapes for different weights.*

motion according to psychological studies.

3. Background

In this section I present the underlying mathematical principles for computing distance fields of solid objects, and how to smoothly interpolate between the distance fields.

3.1. Distance-field representation and interpolation

The distance field of an object is an implicit representation of the object. It is the set of scalar values that represent at each point of the object's domain, the nearest distance of the point to the object's boundary [Frisken and Perry 2006]. It is a scalar function of the form $dist(x)$ which maps $\mathbf{x} \in \Re^n$ onto $\Re$. In Figure 4, if Ω is the boundary of the object, then Ω is the set of all points with zero distance i.e., $dist(x) = 0$. Every point has a sign that depends on whether it is inside or outside of the objects' boundary. More formally:

$$f(x) = \begin{cases} dist(x, \Omega) & \text{if } x \in \Omega \\ -dist(x, \Omega) & \text{if } x \in \Omega^{\complement} \end{cases} \tag{1}$$

In my case studies, I use the Euclidean distance as the distance function but other distance metrics can be used as well. The computation of distance fields in the simplest case is achieved using the following brute force method: for each element in a discrete set, e.g. pixels in an image, its euclidean distance to all objects is computed and the smallest one is stored. Due to the inefficiency of the brute force approach, other methods have been proposed as explained in [Jones et al. 2006]. In all of my case studies I use MATLAB's built-in routine for computing distance fields.

I use the distance field representation to morph a solid object to another solid object in R^2. Figure 1 shows the overall morphing process for different weights w. The two images are first mapped into the binary domain (e.g. bitmap format) before any further computation takes place. Then I compute the distance fields of the two images. Before interpolation, the distance fields of the images are transformed from real space to logarithmic space using the transformation $sgn(x)\log(|x| + \epsilon)$ where sgn(x) is the sign of the point as determined from (1), and ϵ is a very small value such as 10^{-3} and keeps the values near the surface. After experimentation, the value of ϵ is 1 in all of my case studies. I observed that morphing in logarithmic space produces better results when compared to morphing in real space and [Schumacher et al. 2015] hold a similar position. Finally, I perform linear interpolation on the transformed distance fields of the two images producing the transition shown in Figure 1.

3.2. Rigidity Preserving Shape Interpolation

Rigidity preserving shape interpolation is a well known technique for treating shapes as rigidly as possible during interpolation. The goal is to transform one shape into another by achieving the minimal amount of global or local deformation of the underlying shapes. Following is a detailed explanation of the process and the algorithms I use. Note that the mathematical notation used in this section follows that of [Anjyo and Ochiai 2014]. The authors provide the most recent treatment of mathematical concepts and their notation as they are applied to graphics especially for affine matrix representation and interpolation, which in fact underlie the rigidity preserving technique I present in this section.

The process starts by describing the two initial two-dimensional shapes as dissections made of polygons. In the case of two dimensional shape blending, dissections are triangulations and they need to be compatible between the initial and the target shapes; there needs to be a one-to-one correspondence in their respective dissections (bijective triangle-to-triangle mapping). Hence, some manual pre-processing is needed to establish two compatible triangulations but automatic methods exist as well [Baxter et al. 2009]. In fact for complicated shapes with large amounts of polygons implementing an automatic method might be a better path to follow. In that case, the user would need to establish boundary vertex correspondences and an appropriate triangulation algorithm would dissect the two shapes in a compatible fashion based on the vertices specified by the user.

Suppose that the two initial shapes are comprised by a single triangle. Let the source triangle's vertices be $P_T = (\vec{p_1}, \vec{p_2}, \vec{p_3})$ and the target triangle's vertices be $Q_T = (\vec{q_1}, \vec{q_2}, \vec{q_3})$ where vertices with same index correspond. To achieve least distorting shape interpolation, local and global techniques are developed as follows.

Local case. An affine transformation maps the initial triangle to the target triangle:

$$\hat{\mathbf{A}} = \begin{pmatrix} q_1^x & q_2^x & q_3^x \\ q_1^y & q_2^y & q_3^y \\ 1 & 1 & 1 \end{pmatrix}^{-1} \begin{pmatrix} p_1^x & p_2^x & p_3^x \\ p_1^y & p_2^y & p_3^y \\ 1 & 1 & 1 \end{pmatrix} \tag{2}$$

and it is of the form $Aff^{+}(2) = \left\{ \begin{pmatrix} M & d \\ 0 & 1 \end{pmatrix} | M \in GL^{+}(2), d \in \mathbb{R}^2 \right\}$,

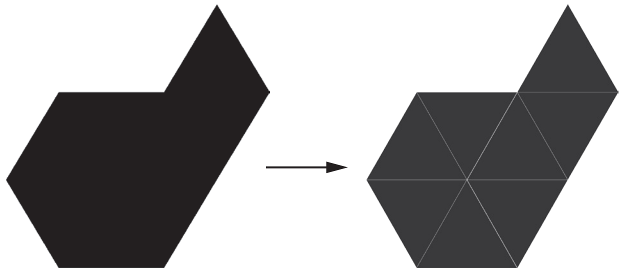

Figure 5: *The representation used for rigidity preserving shape interpolation is a triangulation, i.e. the dissection of a solid object in polygons.*

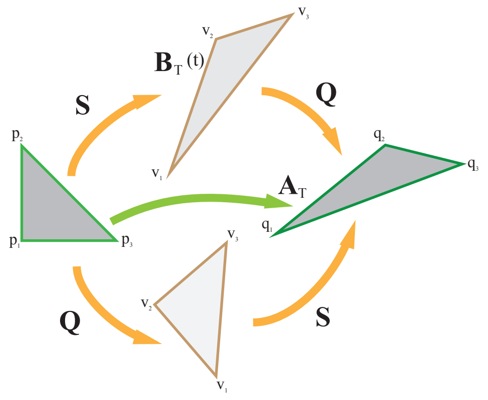

Figure 6: *Polar decomposition decomposes affine matrix M into a rotation Q and a shear-scale part S. Q and S are interpolated separately to map the initial triangle to the target one.*

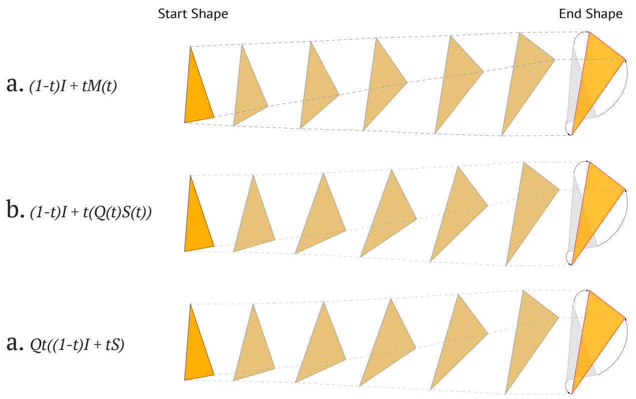

Figure 7: *Triangle homotopies: (a) Direct affine matrix interpolation. Vertices travel in straight lines (b) Linear homotopy after polar decomposition (c) Interpolation based on the homotopy (4).*

where M is a 2x2 orthogonal matrix which is called the linear part and d is a vector in R2 and is called the translational part. M belongs to the general linear group of transformations with positive determinants $GL^+(n) = \{M \in GL(n) | detM > 0\}$. This means that M should not encode reflections but only pure rotations. If the determinant is found to be negative then the transformation "flips" the triangle and therefore some necessary processing has to take place to remove the reflection.

After computing the map in (2) I interpolate the translational part and linear part separately. The interpolation of the translational part can be neglected. I focus on the interpolation of the linear part. As described in [Kaji et al. 2012], a homotopy of the linear transformation M is a collection of matrices M(t) parameterized by time $t \in \mathbb{R}$ such that M(0) = I and M(1) = M, where I is the identity matrix. A linear homotopy is given by:

$$M^L(t) = (1-t)I + tM \tag{3}$$

However, this linear homotopy is not good for several reasons. During interpolation, the triangles might collapse or shrink or flip producing unnatural interpolation. In the literature, the following properties seem to be the desirable ones [Shoemake and Duff 1992; Alexa et al. 2000; Kaji et al. 2012]:

- The transformation should be symmetric with respect to t.
- The rotational angle(s) should change linearly.
- The triangle should not reflect.
- The resulting vertex paths should be simple.

Instead, in all the following resources [Shoemake and Duff 1992; Baxter et al. 2008; Alexa et al. 2000; Rossignac and Vinacua 2011; Kaji et al. 2012; Kaji and Ochiai 2014] different homotopies are used. Those homotopies are based on the polar decomposition of the matrix M into orthogonal and symmetric components. The linear part of the affine transformation is decomposed into an orthogonal part Q and a symmetric semi-definite positive part S. The orthogonal part encodes rotations or reflections (which need to be avoided) and the symmetric part encodes shear and scale (called stretch matrix according to [Shoemake and Duff 1992]). Polar decomposition according to [Alexa et al. 2000] can be derived from singular value decomposition (SVD) as follows:

$$svd(M) = Udiag(S)V^T \Rightarrow U(V^TV)diag(S)V^T \Rightarrow (UV^T)(Vdiag(S)V^T)$$

I can observe that $Q = UV^T$ and $S = Vdiag(S)V^T$(this is also called a spectral decomposition). In [Shoemake and Duff 1992] another method is proposed for computing the polar decomposition which is based on an iterative minimization process. In all of my examples, I use the approach described in (4) and hence I omit the explanation of the iterative process.

Having the polar decomposition of M I can compute the homotopy as:

$$M^L(t) = R_{t\theta}((1-t)I + tS) \tag{4}$$

Another homotopy is proposed in [Kaji et al. 2012; Kaji and Ochiai 2014] which is based on the concept of the exponential map:

$$M^E(t) = Q^t_\theta S^t \tag{5}$$

where $S^t = \exp(tlogS)$. In my examples I have not implemented homotopy (5) and I only use homotopies (3) - for purposes of illustration - and (4). Practically, the homotopy in (4) is obtained by spherically interpolating the rotational part Q using its quaternion representation [Vince 2011], linearly interpolating the stretch part S and multiplying the resultant matrices as in (4) for $t \in [0,1]$. Homotopy (4) produces visually better results than direct matrix interpolation as can be observed from Figure and satisfies the criteria discussed earlier.

Global interpolation. To achieve least distorting global interpolation between the target and the initial shapes, one can follow two approaches. The first approach is through direct minimization of a quadratic error functional [Alexa et al. 2000] and the second using normal equations [Baxter et al. 2008]. I follow the later approach due to its compactness and ease of implementation. The two approaches are equivalent [Nealen 2004; Baxter et al. 2008].

As a general process, global transformation is achieved by assembling local transformations, as described in the previous section, and finding the intermediate vertex trajectories by minimizing a quadratic error function:

$$E(t) = \sum_{T=1}^{M} \left\| B_T(t) - A_T(t) \right\|^2 \tag{6}$$

where $B(t) = \{B_i(t) \in Aff(2) | i \in \{1, 2, ..., m\}\}, (t \in \mathbb{R})$ a collection of affine maps that relate the initial vertices P_T to unknown vertices. By expressing the minimization problem in (6) in terms of least squares normal equations, a clear structure of the linear system is derived the implementation of which becomes quite straightforward:

$$E(t) = \sum_{T=1}^{M} a_T \left\| \mathbb{P}_T \mathbb{V} - A_T(t) \right\|^2$$

where a_T is the area of each triangle, $\mathbb{V}$ is a 2xN matrix containing the unknown interpolated vertex locations at time t, $\mathbb{P}_T$ a sparse 2xN matrix and A_T is (2). Expression (7) is equivalent to the following formulation using normal equations:

$$(\mathbb{P}^T \mathbf{W} \mathbb{P})\mathbb{V} = \mathbb{P}^T \mathbf{W} \mathbb{A} \tag{7}$$

where

$$\mathbb{P} = \left[P_1^T ... P_M^T\right]^T$$

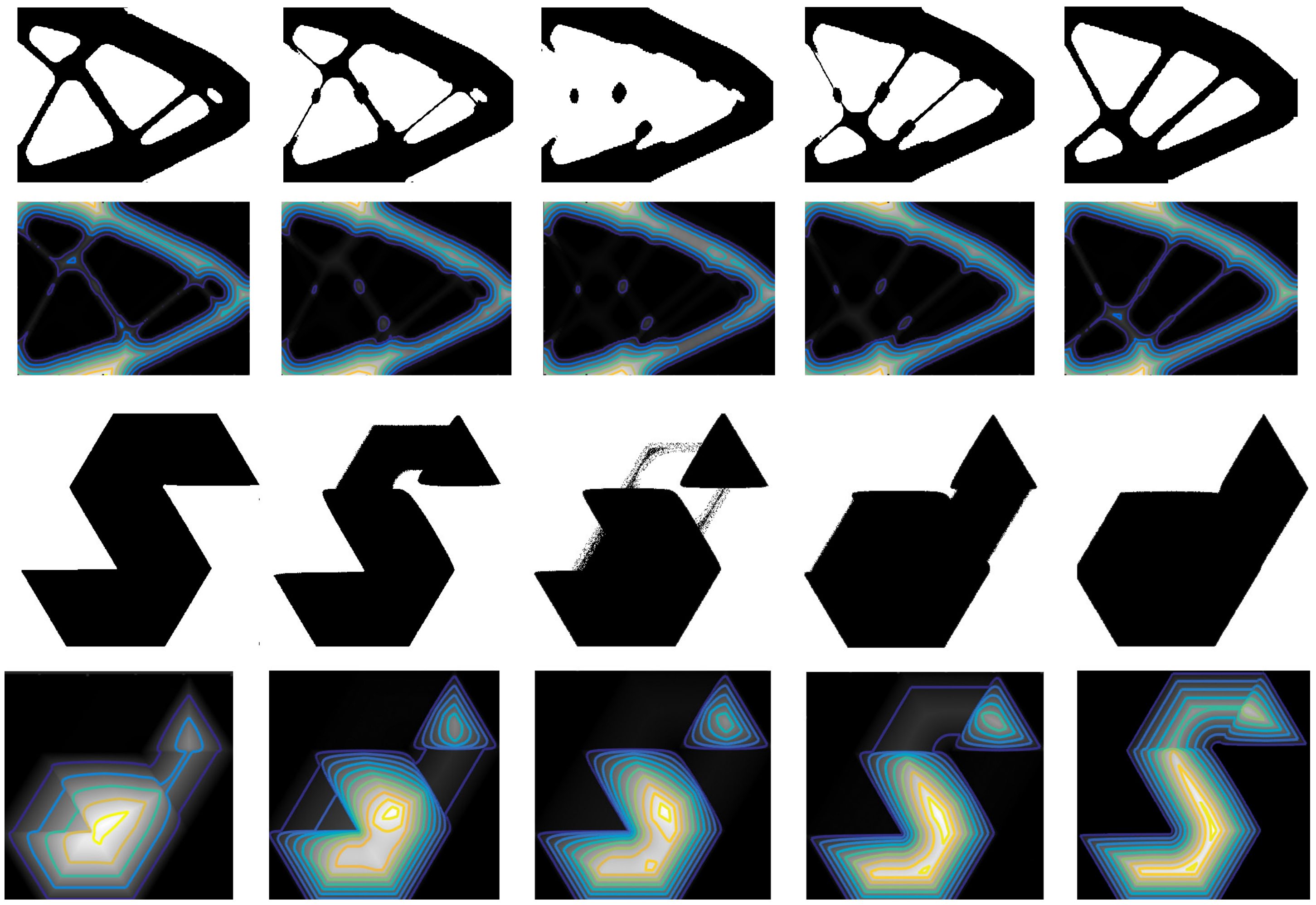
Figure 8: *Left side: Preparation for interpolation of two solid shapes. Binary image is transformed into its distance field representation. The distance field representation is transformed in log-space. Right side: Results from interpolating the start and end shapes for different weights.*

$$\mathbb{A} = \left[A_1^T ... A_M^T\right]^T$$
$$W = diag(\left[a_1 ... a_M\right])$$

The implementation of this method is done in the JAVA language. All necessary routines for matrix decomposition, operations, interpolation and display are custom. Rendering and printing is done using the graphics pipeline of Processing API.

4. Results

I tested Distance Field interpolation and rigidity preserving interpolation on images that represent solid shapes, trusses and continuums. Both methods have their advantages and drawbacks depending on the application. The representation of a shape, as mentioned earlier, plays principal importance to the success or failure of an approach.

In Figures 4 and 8 I show examples of applying distance field interpolation. In Figure 4 the transition from the initial shape to the target shape is smooth and creates no unwanted artifacts. In Figure 8 on the other hand the top row shows two continuums with the same number of loads but in different positions. The representation of the shape as a distance field does not take into account topological transitions as a designer would want to. It does not take into account part continuity and local displacements resulting in holes and weird artifacts. A similar case is the bottom row although the distance field creates a correct boolean subtraction of the un-common areas between the source and target shapes. However, the transition from D to S is unnatural especially when thresholding is applied to go back from distance fields to R^2. I conclude that distance field interpolation becomes a very powerful tool when one needs to morph between topologically similar objects and when not many local displacements are needed.

I tested rigidity preserving interpolation on three cases. The first two cases are shown in Figure 9. The source and target trusses have the same topology but different geometry. The transformation taking one onto the other is successful in both cases. In Figure 10 I represent letters D and S as triangulations but the morphing process is unsuccesful. This is because I did not create a correct one-to-one mapping between the triangles of the two shapes. The truth is that this method lacks automation specifically in the stage of triangle correspondence. A user needs to determine all triangle correspondences in order for the morph to be successful. Such is the case in the trusses of Figure 9. As a conclusion, rigidity preserving interpolation seems quite similar to a parametric design scheme. However, the method I present is quite different from having to determine all the variables needed to achieve the transition apart from the fact that the user needs to specify a bijective mapping between the tesselations of the two shapes (compatible triangulation scheme explained earlier). I believe that the design space in this case is based on a transition, i.e. an affine transformation that maps one shape onto the other. The designer then controls an interpolation parameter that controls the amount of mapping from one design onto the other. In this sense, this method is succesfful in illustrating the concept of this paper which is the representation of the design space between two designs in terms of transitions.

5. Conclusion

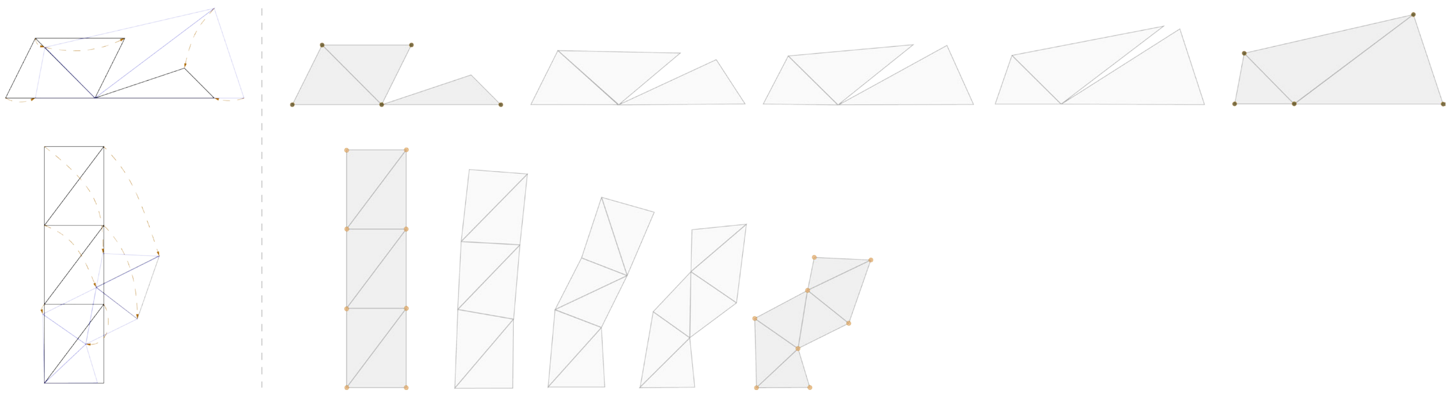

Figure 9: *Morphing between two topologically similar trusses using the rigidity preserving method.*

In this paper I articulated the concept of a design space based on transitions or transformations from one design state onto another. I developed two methods that illustrate the concept namely distance field interpolation and rigidity preserving interpolation and I show examples of designs created with the two methods.

Limitations and future work. As a future work I believe that the following are of particular importance. The rigidity preserving method can be expanded a lot more to facilitate better control and objects with a lot more triangles. At this stage, the determination of the bijective map between two dissections is cumbersome and it depends from the user. Automatic compatible triangulation is a future goal.

A more appropriate representation would allow two designs to transition one onto another but facilitating topological change as well. For this purpose, a part-aware and structure aware representation is a way to achieve smooth and continuous transitions between topologically dissimilar designs. Apart from generating transitions that take into account shape, considering structural or material properties is an important extension. Filtering out generated objects that are not plausible (e.g. structure wise) can be an important addition.

Moreover, different error functions can be used apart from the one used in the examples of this paper. Different weightings schemes and constraints can create much more intersting design spaces. From the technical side, some improvements such as the representation of rigid motion with dual quaternions might create better control of the transitions.

Last, creating a user interface and determining what sort of interaction will the user need with a transition based system will be an important next step.

References

Alexa, M., Cohen-Or, D., and Levin, D. 2000. As-rigid-as-possible shape interpolation. *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., 157–164.

Alexa, M. (2002) Recent advances in mesh morphing. Compuvter Graphics Forum, 21, 2, 173-198.

Alhashim, I., Li, H., Xu, K., Cao, J., Ma, R., and Zhang, H. 2014. Topology-varying 3D shape creation via structural blending. ACM Transactions on Graphics (TOG) 33, 4, 158.

Anjyo, K. and Ochiai, H. 2014. *Mathematical basics of motion and deformation in computer graphics*. Morgan & Claypool Publishers.

Baxter, W., Barla, P., and Anjyo, K. 2008. Rigid shape interpolation using normal equations. *Proceedings of the 6th international symposium on Non-photorealistic animation and rendering*, ACM, 59–64.

Baxter III, W.V., Barla, P., and Anjyo, K. 2009. Compatible embedding for 2d shape animation. *Visualization and Computer Graphics*, IEEE Transactions on 15, 5, 867–879.

Beier, T. and Neely, T. (1992) Feature-based image metamorphosis. ACM Transactions on Graphics, 26, 2, 35-42.

Che, W., Yang, X., and Wang, G. 2004. Skeleton-driven 2D distance field metamorphosis using intrinsic shape parameters. Graphical Models 66, 2, 102–126.

Cohen-Or, D., Solomovic, A., and Levin, D. 1998. Three-dimensional distance field metamorphosis. ACM Transactions on Graphics (TOG) 17, 2, 116–141.

Jones, M.W., Baerentzen, J.A., and Sramek, M. 2006. 3D distance fields: A survey of techniques and applications. Visualization and Computer Graphics, IEEE Transactions on 12, 4, 581–599.

Kaji, S., Hirose, S., Sakata, S., Mizoguchi, Y., and Anjyo, K. 2012. Mathematical analysis on affine maps for 2D shape interpolation. Proceedings of the 11th ACM SIGGRAPH/Eurographics conference on Computer Animation, Eurographics Association, 71–76.

Kaji, S. and Ochiai, H. 2014. A concise parametrization of affine transformation. submitted for publication 52, 53.

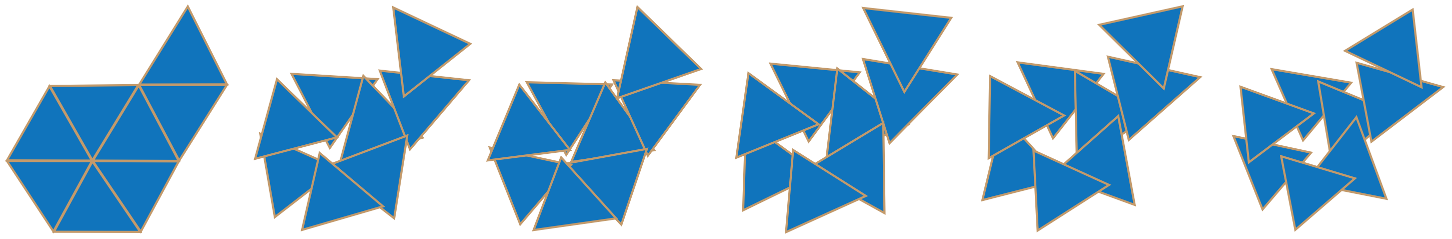

Figure 10: *Lack of a good bijective mapping between the start and end design will create an unplausible transition from one design to another. Automatic ways of determining the mapping will create less cumbersome and error prone mappings.*

LAZARUS, F. and VERROUST, A. 1998. Three-dimensional metamorphosis: a survey. The Visual Computer 14, 8, 373–389.

LEE, S.-Y., CHWA, K.-Y., HAHN, J.K., AND SHIN, S.Y. 1996. IMAGE MORPHING USING DEFORMATION TECHNIQUES. JOURNAL OF VISUALIZATION AND COMPUTER ANIMATION 7, 1, 3–23.

NEALEN, A. 2004. AN AS-SHORT-AS-POSSIBLE INTRODUCTION TO THE LEAST SQUARES, WEIGHTED LEAST SQUARES AND MOVING LEAST SQUARES METHODS FOR SCATTERED DATA APPROXIMATION AND INTERPOLATION. URL: HTTP://WWW. NEALEN. COM/PROJECTS 130, 150.

FRISKEN, S.F. and PERRY, R.N. 2006. Designing with distance fields. ACM SIGGRAPH 2006 Courses, ACM, 60–66.

HANSON, A. 2006. Visualizing quaternions. Morgan Kaufmann ; Elsevier Science [distributor], San Francisco, CA : Amsterdam ; Boston.

ROSSIGNAC, J., AND VINACUA, A. 2011. STEADY AFFINE MOTIONS AND MORPHS. ACM TRANSACTIONS ON GRAPHICS (TOG), 30, 5, 116.

RUPRECHT, D. AND MÜLLER, H. 1995. IMAGE WARPING WITH SCATTERED DATA INTERPOLATION. IEEE COMPUTER GRAPHICS AND APPLICATIONS 2, 37–43.

SCHUMACHER, C., BICKEL, B., RYS, J., MARSCHNER, S., DARAIO, C., GROSS, M. 2015. MICROSTRUCTURES TO CONTROL ELASTICITY IN 3D PRINTING. ACM TRANSACTIONS ON GRAPHICS (TOG) 34, 4, 136.

SEDERBERG, T.W., GAO, P., WANG, G., and MU, H. 1993. 2-D shape blending: an intrinsic solution to the vertex path problem. Proceedings of the 20th annual conference on Computer graphics and interactive techniques, ACM, 15–18.

SHAPIRA, M. and RAPPOPORT, A. 1995. Shape blending using the star skeleton representation. *IEEE Transactions on Computer Graphics and applications*, 15, 2, 44-50.

SHOEMAKE, K. and DUFF, T. 1992. Matrix animation and polar decomposition. Proceedings of the conference on Graphics interface, Citeseer, 258–264.

VINCE, J. 2011. Quaternions for Computer Graphics. Springer London, London.

WOLBERG, G. 1998. Image morphing: a survey. The visual computer 14, 8, 360–372.

YESSIOS, C. I. 1987. A fractal studio. *Proceedings of The Association of Computer Aided Design in Architecture*, North Carolina State University.