SIAM J. Imaging Sciences
Vol. 13, No. 1, pp. 78–107

Automatically Controlled Morphing of 2D Shapes with Textures*

Alexander Tereshin†, Valery Adzhiev†, Oleg Fryazinov†, Felix Marrington-Reeve†, and Alexander Pasko‡

Abstract. This paper deals with 2D image transformations from a perspective of a 3D heterogeneous shape modeling and computer animation. Shape and image morphing techniques have attracted a lot of attention in artistic design, computer animation, and interactive and streaming applications. We present a novel method for morphing between two topologically arbitrary 2D shapes with sophisticated textures (raster color attributes) using a metamorphosis technique called space-time blending (STB) coupled with space-time transfinite interpolation. The method allows for a smooth transition between source and target objects by generating in-between shapes and associated textures without setting any correspondences between boundary points or features. The method requires no preprocessing and can be applied in 2D animation when position and topology of source and target objects are significantly different. With the conversion of given 2D shapes to signed distance fields, we have detected a number of problems with directly applying STB to them. We propose a set of novel and mathematically substantiated techniques, providing automatic control of the morphing process with STB and an algorithm of applying those techniques in combination. We illustrate our method with applications in 2D animation and interactive applications.

Key words. metamorphosis, transfinite color interpolation, space-time bounded blending, signed distance fields

AMS subject classifications. 65S05, 51N20, 68R01

DOI. 10.1137/19M1241581

1. Introduction.

2D imagery is a well-established area dealing with generation and transformation of both raster and vector images. Image dynamics is a more complex field concerned with time-variant image transformations. 2D shape modeling is also considered a well-researched area with its own established set of operations on shapes. However, these areas are not always well connected to each other. It is time to revisit some fundamental problems and algorithms of 2D imagery, taking into account recent advances in 3D modeling and aiming at some specific applications oriented toward particular usage scenarios with distinctive constraints.

Morphing (or metamorphosis) means the visually smooth transition between two given images or shapes. It is a commonly used operation in computer animation, visual effects, and computer art and design, and it is especially relevant for consideration in the above-defined context. There are several well-known solutions for 2D image and shape metamorphosis, all

*Received by the editors February 7, 2019; accepted for publication (in revised form) October 24, 2019; published electronically February 4, 2020.
https://doi.org/10.1137/19M1241581
†The National Centre for Computer Animation, Bournemouth University, Poole, Dorset BH12 5BB, United Kingdom (atereshin@bournemouth.ac.uk, vadzhiev@bournemouth.ac.uk, ofryazinov@bournemouth.ac.uk, i7621149@bournemouth.ac.uk).
‡Skolkovo Institute of Science and Technology, Moscow, 121205, Russia (A.Pasko@skoltech.ru, https://www.skoltech.ru/en).

of which require establishing some form of one-to-one correspondences between two given images, between boundaries of two given shapes, or between their particular features [3, 13].

The general case where no correspondences between two given shapes or their features and no foreground image segmentation are provided has not been fully investigated yet, although several usage scenarios exist. For example,

1. in-between frames generation in a situation where there is no prior information about the source and target frames, such as in a game environment, where characters and assets are procedurally generated on the fly [36];
2. in a live TV show or any streaming online content, which changes dynamically with no time available for establishing correspondences between frames;
3. in interactive applications aimed at nonprofessional users, or at users with specific requirements, where the input can be generic and no correspondences between source and target frames can be established.

All of the above cases are united by a lack of resources (time, information) or skills (cognitive, educational) for establishing correspondences between two given shapes or images. The important thing is that in all mentioned cases there is no need or no opportunity to establish point-to-point or feature-to-feature correspondences. It is made automatically, and no additional in-between frames should be drawn or any parameters of the process changed by the user. In this paper, we deal with the metamorphosis of 2D shapes defined by raster images with various backgrounds. The whole process can be described as heterogeneous object transformations, where a shape and its texture (as a 2D spatial attribute) are changing synchronously and interconnectedly.

The core idea of our method comes from the recent advances in 3D heterogeneous object modeling, which is a rapidly emerging area where geometric shapes are considered in concert with their internal attributes defined for each point of the shape [17, 31]. These attributes represent such physical properties as material, density, and color. In the case of 2D heterogeneous objects considered in the context of the imagery, shape attributes can be reduced to per-point colors of 2D images.

Thus, to obtain morphing between two shapes with no correspondences we utilize a revised version of the metamorphosis technique called space-time blending (STB) [30] for geometry and space-time transfinite interpolation (STTI) for colors/textures [34, 35].

By doing that, we allow for obtaining a smooth transition between source and target images without requirements for shape alignment or topological conformity or for establishing correspondences. The method is computationally inexpensive and artist friendly and can be used in various interactive applications, including educational games.

The contributions of our work can be formulated as follows:

- We formulate the metamorphosis problem in the context of 2D imagery in the most general way from the heterogeneous object modeling point of view, combining the object shape and its properties into a single entity.
- We propose a novel method for solving the problem of metamorphosis between topologically arbitrary 2D shapes with textures. The shapes are represented by signed distance fields [14] in a continuous form. For generating intermediate shapes of the metamorphosis, we apply the STB technique for handling geometry morphing coupled with the STTI method for handling texture morphing.

- We present several novel mathematically substantiated technical solutions for solving the drawbacks of the proposed method for an automatic metamorphosis between 2D shapes.
- The proposed method is based on geometric and algebraic techniques and, unlike the PDE-based approaches, does not use heavy numerical computations.

2. Related works. Out of many existing methods for solving the 2D metamorphosis problem, we will consider two main groups of methods, namely, those that are feature-based and those that are automatically controlled with a particular emphasis on morphing colored or textured objects.

2.1. Feature-based morphing. In this subsection, we consider nonautomatic methods for morphing with additional user control. An early example of an in-between metamorphosis for 2D images can be found in [4], where the authors present a system for morphing between hand-drawn contour shapes with in-between shape generation. For this, the line stroke to stroke mapping along with establishing the correspondences between the shapes is essential for the in-between interpolation. To fully utilize the system, a skeleton must be set up. A similar method was presented in [9] where pixel tiles are moved from one image to another according to the optical flow while changing their colors.

Some attempts to introduce more flexible user control over morphing were undertaken with user-defined feature segmentation. A feature-based image metamorphosis technique was presented in [3] for obtaining smooth transitions between two images. This allows the user to define the pairs of corresponding features in the source image and the target image that are further used for computing in-between frames. This, however, heavily relies on the linear interpolation between shapes, which does not produce acceptable results in most cases. Higher quality interpolation with provided corresponding boundary point pairs is described in [1].

The simplest technique for transforming one digital image into another in terms of in-place morphing (when the input and target shapes are superimposed on one another) is cross-dissolving [37], where one image gradually disappears and another appears through a per-pixel color interpolation. This method can only handle image morphing. According to [18, 39], cross-dissolving without prewarping the initial and target shapes produces a poor result with a double-exposure effect. For obtaining best results, two images should be prewarped to make the shapes similar by specifying control pixels. This leads to the nonautomatic approaches described, for example, in [3, 37, 39].

Dalstein, Ronfard, and van de Panne [7] introduced a feature-based method using the vector animation complex, which is a vector graphics data structure. Their approach uses a space-time concept that is based on a parameterized model for obtaining in-between frames. The suggested method is topology-aware, can work with overlapping objects, and supports coloring of the 2D faces. Unfortunately, this method can produce discontinuous results because of the employed linear interpolation, and it also does not support color/texture transformation between frames.

2.2. Automatically controlled morphing. Averbuch-Elor, Cohen-Or, and Kopf in [2] introduced a data-driven method based on the inner-distance shape context technique to handle geometry transformations and globally affine RGB transformations for color blending. Their

in-between images are created by warping and blending an input image toward a target image to align them. This method cannot handle objects with different topology, and also does not take into account the interior texture of the object.

Gao et al. introduced a data-driven mesh morphing method in [10]. Their method is based on patch-based and linear rotationally invariant coordinates that can handle the deformations of models in a shape collection. This method works with objects of the same topology and does not handle texture or color transformations.

Liao et al. [21] proposed an automatic method for optimization-guided image morphing that also supports the presence of greater dissimilarities between images. The optimization can take into account user-drawn points for better aligning of structural image features. While providing intuitive results, this work does not provide a method for dealing with objects of radically different topology.

A hybrid stroke-based solution [13] deals with the automatic generation of in-betweens for 2D facial cartoon animations. The key idea of the suggested hybridization is to combine the information about the approximate 3D geometry with multiple views of a character's face from key frames to overcome the lack of information. It facilitates automatic establishing of stroke correspondences. However, the strokes can be incorrectly annotated, and this method cannot handle morphing between radically different images such as drawings with dissimilar strokes or shapes.

An interesting approach was introduced by Neumann, Alexander, and Neumann in [27] where a random walk algorithm was applied for obtaining evolutionary image morphing. The proposed algorithm is based on the usage of fitness functions for a per-pixel image transformation using random walks and is only texture-aware.

The most relevant group of methods for automatically controlled metamorphosis is related to optimal mass transport (OMT). The OMT methods provide a solution for the Monge–Kantorovich optimization problem [15]. This solution provides the optimal way for moving a mass distribution from one domain to another with minimal transportation cost. Typically the solution is obtained in the form of the L^2 Kantorovich–Wasserstein distances by solving the differential equation defining the metric. In the context of the metamorphosis method it is important to note that OMT-based methods are parameter free and not feature-based.

In [12] and [40] an intensity-based OMT method was introduced. The process of obtaining the in-between frames assumes computation of the Kantorovich–Wasserstein distances, which are used for generating the warping transportation map between the initial and target images. Then the cross-dissolving method is applied. In [40], to overcome the double-exposure effect caused by cross-dissolving, the authors introduced an intensity penalty term to the mass moving energy functional.

A vector-valued OMT method was introduced in [5]. This method allows handling of the mass flow between vectorial entries across a discrete or continuous space. The authors claimed that their approach is suitable for a number of applications, in particular for color image processing and for morphing between color distributions. They have provided an example of color interpolation for real-life images in the form of photos. However, this is a pure image processing example without taking into account geometry.

The variational OMT approach [22] was applied to grayscale textures with sharp features. A method for topology-aware shape morphing using cluster-based Earth mover's distance

flows was introduced in [23]. Unfortunately, it cannot handle any color or texture morphing. In [19] the author introduced a numerical topology-aware method for dealing with geometric metamorphosis only. It is based on computing the OMT between density in the form of a piecewise linear function and a sum of Dirac masses. The authors in [38] and [6] introduced a topology-aware method for solving the OMT optimization problem using convolutional Wasserstein distances that are approximated using entropic regularization. This method can only handle interpolation between blocks of pure color. In [32] a GPU-based approach for solving the OMT optimization problem was suggested and applied to high-frequency grayscale images. Elsewhere [26] the authors introduced a new solver for computing an approximated OMT that is derivative free and converges within a few iterations. This method is topology-aware and preserves sharp features during metamorphosis but does not support texture or color transformations.

2.3. Summary. Many of the described methods work reasonably well with pure image data (raster arrays or textures) rather than with textured 2D shapes. There are several methods for creating a metamorphosis between two 2D geometric shapes, between images, or between textured shapes. These methods commonly assume establishing correspondences between points or features of the objects. The stroke-based techniques mentioned above are not always intuitive. The most relevant method in the context of automatically controlled metamorphosis is OMT. These methods are parameter free and topology-aware; they can handle the morphing in terms of geometry and images. However, the reviewed works did not provide the implemented examples of metamorphosis between objects with sophisticated textures, especially in the context of applications that we consider paramount in our work.

3. Background. In this section, we outline those specific concepts and methods that will serve as a basis for the development of a theoretical and practical framework for executing automatically controlled morphing between 2D shapes with textures. We will introduce a representational schemes for 2D heterogeneous objects as well as the specific methods called space-time blending (STB) and space-time transfinite interpolation (STTI). These methods will then serve as the basis for a novel method for solving the automatic metamorphosis problem without establishing any correspondences between two images (section 4) as well as for several mathematically substantiated new technical solutions for more visually impressive results (section 5).

3.1. Function representation. The representation that describes geometric objects as closed subsets of n-dimensional Euclidean space E^n with $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ using a a real-valued continuous function $F(\mathbf{x}) \geq 0$ is called function representation (FRep) [28]. FRep has the following characteristics:

$$\begin{cases} F(\mathbf{x}) > 0 & \text{for points inside the object,} \\ F(\mathbf{x}) = 0 & \text{for points on the boundary of the object,} \\ F(\mathbf{x}) < 0 & \text{for points outside the object.} \end{cases}$$

Note that the function changes its sign at the shape boundary. The FRep function can be defined analytically, or with a function evaluated algorithm, or with a discrete scalar field. FRep is closed under a rich set of operations. These means that any operation on FRep objects

produces a resulting FRep object also represented by a continuous real-valued function. There exist many well-developed FRep operations taking single or multiple shapes and producing a new shape in 2D, 3D, or higher dimensional space. In particular, there are set-theoretic operations based on R-functions introduced in [33] that formed the basis for the definition of several types of FRep blending operations [28]. The defining function can be generated for point clouds using interpolating surfaces, shapes with a curve skeleton, mesh surfaces, or voxel arrays. This makes FRep essentially a hybrid representation of volumetric objects. The concept of a heterogeneous object [29], defined as a combination of its geometric shape and the physical properties associated with the object's internal structure, underlies our method. For a 2D heterogeneous object, a color attribute is mapped to each interior point. In the current work we consider 3D space (X, Y, Z), where a point is defined as $\mathbf{x} = (x, y, t)$, with the third coordinate Z defined as a time t.

3.2. Signed discrete distance transform. A functionally based object can be represented as a discrete scalar field. For a given closed 2D shape S representing a discrete boundary of some volumetric object O, the most natural way to obtain a defining function for O is to introduce a signed distance field (SDF) [14] in a discrete form. In the current work we choose the convention that the sign of the SDF is positive inside and negative outside of the object O. The absolute value of the SDF at any given point $\mathbf{p}$ is the minimal Euclidean distance from this point to some point $\mathbf{p_s}$ belonging to the surface S_O of the object O: $d(\mathbf{p}, \mathbf{p_s}) = \text{sign}(\mathbf{p}) \cdot \inf_{\mathbf{p_s} \in S_O} d(\mathbf{p}, \mathbf{p_s})$, where $d(\mathbf{p}, \mathbf{p_s})$ is the Euclidean distance between two points.

The most common way to obtain the distance property for an arbitrary discrete scalar field is to use the signed discrete distance transform (SDDT) operation [8, 20]. Formally, the SDDT for a 2D image is a mapping from a binary image $I_b(x, y)$ to a multivalued vector image $I_v(x, y) = (d_x, d_y)$. Here d_x and d_y are distances from the point (x, y) to the image background in the X and Y directions, respectively. The resulting distance to shape S is defined as the Euclidean norm: $D(x, y) = ||d_x^2 + d_y^2||$. If it is essential to obtain a continuous resulting function, the SDDT can be smoothed using a B-spline, bicubic, wavelet, or any other type of interpolation between grid values [16].

3.3. Space-time blending. The main method that we are going to use for the realization of metamorphosis between two geometric shapes defined by SDFs with different topology without any prior knowledge of their correspondences will be space-time blending (STB) [30]. In its essence (see Figure 1), STB is a geometric operation of bounded blending performed in a higher dimensional space. If we consider 2D shapes in that higher dimensional space, the Z axis will be associated with time t. Blending between the initial shape S_1 and the target shape S_2 (see Figure 2) happens in the time interval $t \in [0, 1]$, where, while at $t \leq 0$ only the first shape S_1 exists, then at $t > 0$ it disappears and at time $t = 1$ the second shape appears and exists for any $t \geq 1$. Let us outline the basics of this method.

Let us introduce two input shapes, S_1 and S_2, represented in that higher dimensional space by functions $f_{1_t}(x, y, t)$ and $f_{2_t}(x, y, t)$, respectively. Then the resulting function $F_b(f_{1_t}, f_{2_t}, f_{3_t})$ representing the blending between those two shapes is

$$F_b(f_{1_t}, f_{2_t}, f_{3_t}) = F(f_{1_t}, f_{2_t}) + a_0 \cdot \text{disp}_b(d_r(f_{1_t}, f_{2_t}, f_{3_t})), \tag{3.1}$$

$$F(f_{1_t}, f_{2_t}) = f_{1_t}(x, y, t) + f_{2_t}(x, y, t) + \sqrt{f_{1_t}^2(x, y, t) + f_{2_t}^2(x, y, t)},$$

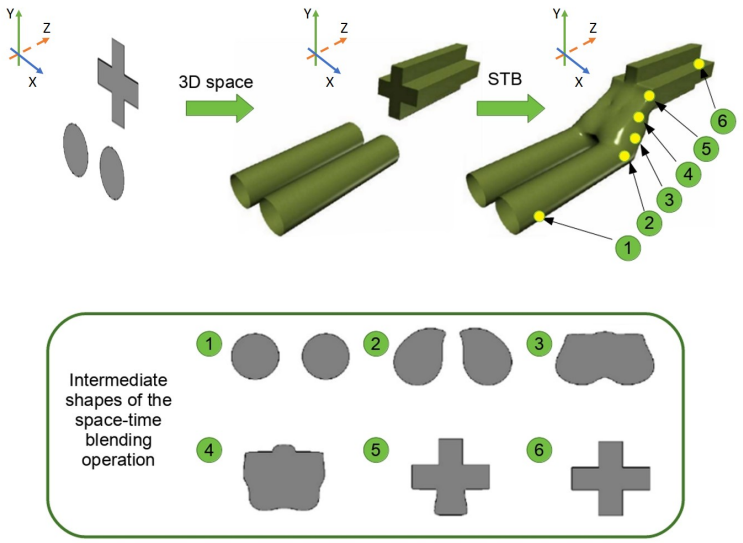

Figure 1. *The concept of space-time blending: two given 2D shapes (two disks and a cross, top left) are extended to 3D space as half-cylinders (top center) with a gap between them. Then a blending union operation is applied, adding material (top right). Intermediate shapes are presented by cross-sections (bottom center).*

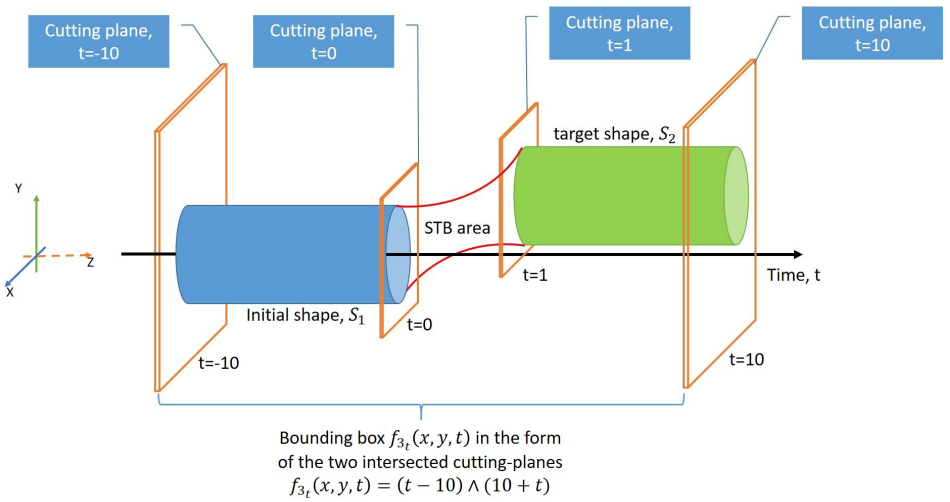

Figure 2. *The space-time blending scheme.*

where $d_r(f_{1_t}, f_{2_t}, f_{3_t})$ is a generalized distance function and $F(f_{1_t}, f_{2_t})$ is a set-theoretical union of two shapes defined by the R-functions introduced by Rvachev [33]. The resulting shape S_2 obtained using the STB function $F_b(f_{1_t}, f_{2_t}, f_{3_t})$ is affected by the bounding solid defined by the function $f_{3_t}(x, y, t)$. The bounding solid is a functionally defined object which restricts the area in which the actual blending happens. In most cases it is convenient to use an intersection of two cutting planes which can be seen in Figure 2. In the case discussed

here, the blending operation will be bounded only along the time axis.

The displacement function $\text{disp}_b(d_r(f_{1_t}, f_{2_t}, f_{3_t}))$ is set up as

$$disp_b(d_r(f_{1_t}, f_{2_t}, f_{3_t})) = \begin{cases} \frac{(1-d_r^2(f_{1_t},f_{2_t},f_{3_t}))^3}{1+d_r^2(f_{1_t},f_{2_t},f_{3_t})}, & d_r(f_{1_t}, f_{2_t}, f_{3_t}) < 1, \\ 0 & \text{otherwise}, \end{cases} \tag{3.2}$$

where $d_r^2(f_{1_t}, f_{2_t}, f_{3_t})$ is defined as

$$d_r^2(f_{1_t}, f_{2_t}, f_{3_t}) = \begin{cases} \frac{d_{r_1}^2}{d_{r_1}^2+d_{r_2}^2}, & d_{r_2} > 0, \\ 1 & \text{otherwise}, \end{cases} \qquad d_{r_1}^2(f_{1_t}, f_{2_t}) = \left(\frac{f_{1_t}(x,y,t)}{a_1}\right)^2 + \left(\frac{f_{2_t}(x,y,t)}{a_2}\right)^2,$$

$$d_{r_2}^2(f_{3_t}) = \begin{cases} \left(\frac{f_{3_t}(x,y,t)}{a_3}\right)^2, & f_{3_t}(x,y,t) > 0, \\ 0 & \text{otherwise}. \end{cases} \tag{3.3}$$

The function $d_{r_1}(f_{1_t}, f_{2_t})$ is a generalized distance function between two shapes $f_{1_t}(x, y, t)$ and $f_{2_t}(x, y, t)$. This function $d_{r_1}(f_{1_t}, f_{2_t})$ provides the algebraic distance measure to both initial S_1 and target shapes S_2. The function $d_{r_2}(f_{3_t})$ controls the influence of the function $f_{3_t}(x, y, t)$ on the overall shape of the blend.

Coefficients $a_0, a_1, a_2, a_3 \in \Re$ are nonzero numerical parameters. To describe their geometrical meaning let us consider the example shown in Figure 3. Let us assume that $f_{1_t}(x, y, t) = x$, $f_{2_t}(x, y, t) = y$, and as a bounding solid we use disk $f_{3_t}(x, y, t) = 1 - \left(\frac{x}{4}\right)^2 - \left(\frac{y}{4}\right)^2$. We apply the STB method to these. The coefficient a_0 defines the total displacement of the blending surface from two initial surfaces S_1 and S_2, defined by $f_{1_t}(x, y, t)$ and $f_{2_t}(x, y, t)$, and controls how much material will be added or subtracted from the overall blend (see Figure 3). Coefficients a_1 and a_2 are proportional to the algebraic distance between the blending surface and the original surfaces S_1 and S_2 defined by $f_{1_t}(x, y, t)$ and $f_{2_t}(x, y, t)$ and control the blend symmetry (see Figure 3). The coefficient a_3 is proportional to the algebraic distance between the blending surface and the surface of the bounding solid (see Figure 3) and controls the influence of function $f_{3_t}(x, y, t)$ on the overall blend.

It is not always intuitive to describe a complex shape using FRep. That is why we suggest using signed distance fields (SDFs) in the discrete form introduced in subsection 3.2 to define both shapes S_1 and S_2. This representation allows one to describe any type of geometrical shape of any complexity in a predictable way.

In the 2D case, the basic STB algorithm can be described as follows:

Step 1. Two shapes, S_1 and S_2, are defined by the functions $f_{1_t}(x, y, t)$ and $f_{2_t}(x, y, t)$. These shapes are represented by scalar fields on the X, Y plane (see Figure 1, top left). Note that both shapes can have radically different topologies.

Step 2. Each shape is used as a cross-section of a half-cylinder in 3D space by extending them in the Z dimension, with a gap in between (see Figure 1, top middle).

Step 3. The blending union operation adding material to the gap is applied to the two half-cylinders (see Figure 1, top right). The blending operation can be user controlled by varying a_i coefficients.

Step 4. The Z dimension is used as time t. The cross-sections of the blending process along the Z axis between the source S_1 and target S_2 shapes are marked with numbers (Figure 1, top right) and are shown at the bottom of Figure 1.

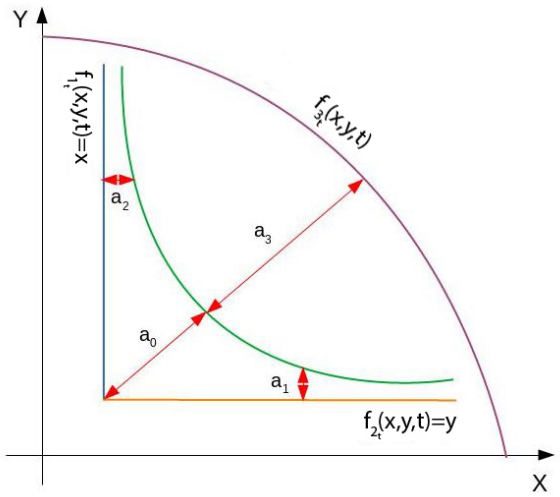

Figure 3. *The intersection of two objects defined by the equations* $f_{1_t}(x,y,t) = x$ *and* $f_{2_t}(x,y,t) = y$, *restricted by the bounding solid represented by the function* $f_{3_t}(x,y,t) = 1 - \left(\frac{x}{4}\right)^2 - \left(\frac{y}{4}\right)^2$ *defining the bounding disk. This is a simple illustration of the geometrical meaning of coefficients* a_i, $i \in [0,3]$.

The implementation in 3D is essentially the same, but raises the shapes into 4D and uses the fourth axis to represent time t.

3.4. Space-time transfinite interpolation. As we are not only dealing with geometry but also with textures, we need a specific technique for texture transformations. This means that in addition to a geometric shape metamorphosis, it is essential to independently define color and other attribute transformations taking into account the initial and in-between shapes. One possible way was described in [35], where the authors proposed to apply transfinite interpolation [34] to attributes such as color. The main idea of space-time transfinite interpolation (STTI) is that two scalar fields defining an initial and a target shape are used for calculating normalized weights $\omega_1(x,y,t)$ and $\omega_2(x,y,t)$ in (3.4) that are further used for computing intermediate colors.

Let us introduce mathematical expressions for the STTI which we are going to use in the next sections. Let us assume that the initial shape is S_1 and that the target shape is S_2, which are defined by the functions $f_{1_t}(x,y,t)$ and $f_{2_t}(x,y,t)$. The attribute interpolation between single-partitioned objects can then be defined as follows:

$$\mathbf{c}(x,y,t) = \omega_1(x,y,t)\mathbf{c_1} + \omega_2(x,y,t)\mathbf{c_2}, \tag{3.4}$$

$$\omega_1(x,y,t) = \frac{f_{2_t}(x,y,t)}{f_{1_t}(x,y,t) + f_{2_t}(x,y,t)}, \quad \omega_2(x,y,t) = \frac{f_{1_t}(x,y,t)}{f_{1_t}(x,y,t) + f_{2_t}(x,y,t)},$$

where $\mathbf{c_1}$ is the color of the input shape S_1, $\mathbf{c_2}$ is the color of the target shape S_2, $t \in [0,1]$, and $\omega_1(x,y,t)$ as well as $\omega_2(x,y,t)$ are weights. The following constraints should be satisfied:

- at the initial time step, $t = 0$: $\omega_1(x,y,0) = 1; \omega_2(x,y,0) = 0; f_{1_t}(x,y,0) \geq 0$;
- at the final time step, $t = 1$: $\omega_1(x,y,1) = 0; \omega_2(x,y,1) = 1; f_{2_t}(x,y,1) \geq 0$;
- $\omega_1(x,y,t) + \omega_2(x,y,t) = 1$.

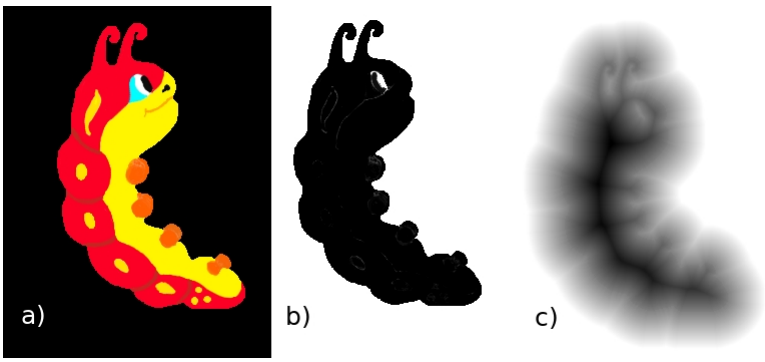

Figure 4. *The process of generating the SDF:* (a) *a textured shape is used as an input;* (b) *the input shape is binarized;* (c) *the binarized shape is used for computing the SDF.*

In the general case, if the given shapes S_1 and S_2 have complex textures with multiple semidisjoint partitions with a constant attribute $\tilde{\mathbf{c}}_\mathbf{j}$ assigned to the jth partition, the contribution of these partitions to the resulting attribute $c_i(x,y)$ at a given external point (x,y) is

$$c_i(x,y) = \frac{\sum_{j=1}^{N} \tilde{w}_j(x,y)\tilde{\mathbf{c}}_\mathbf{j}}{\sum_{j=1}^{N} \tilde{w}_j(x,y)}, \quad \tilde{w}_j(x,y) = \frac{1}{\widehat{f_j}(x,y)}, \quad i = 1,2, \tag{3.5}$$

where N is the number of partitions, $\tilde{w}_j(x,y)$ is the weight of each partition, and $\widehat{f_j}(x,y)$ is a function defining the quadratic Euclidean distance from the external point (x,y) to the pixel in the interior of the object. In the current work we consider textured objects as single-partitioned objects and use (3.5) for computing the interpolated colors $\mathbf{c_1}$ and $\mathbf{c_2}$, where N in this case is the number of pixels inside the textured shape.

4. Automatic metamorphosis between 2D shapes. In this section we describe in a step-by-step manner a novel method for solving the problem of metamorphosis between 2D heterogeneous objects for generating in-between textured shapes without the need to set any correspondences. The suggested method combines the techniques described in section 3. However, as our practical example demonstrates, a basic algorithm for using those techniques in their standard form does not produce completely satisfactory results. We will therefore identify the drawbacks of these techniques.

Let us formulate the problem of metamorphosis between two objects with textures as follows: Given two textured objects defined by SDFs $f_1(x,y)$ and $f_2(x,y)$, we aim at realizing an automatic metamorphosis between those objects with a smooth transition between both geometric shapes as well as their textures without establishing any correspondences between them.

4.1. Description of the basic metamorphosis method. Let us provide a systematic description of the method. Initial and target shapes S_1 and S_2 are represented as 2D images on a monocolored background, e.g., black or white. The steps of the algorithm are as follows:

Step 1. First we convert both input images (see Figure 4(a)) with shapes S_1 and S_2 to binary images (see Figure 4(b)) and then apply one of the SDDT-based methods

[8, 20, 11] to obtain discrete SDFs (see Figure 4(c)) $D_1(\mathbf{p}) : \mathbf{p} \mapsto D_1(\mathbf{p})\ \forall \mathbf{p} \in \Omega$ and $D_2(\mathbf{p}) : \mathbf{p} \mapsto D_2(\mathbf{p})\ \forall \mathbf{p} \in \Omega$ representing those images in a pixel domain Ω where $\mathbf{p} = (x, y)$ is a pixel. The sign of both functions $f_1(x, y)$ and $f_2(x, y)$ changes on the boundary of the binarized shapes.

Step 2. For smoothing the discrete SDFs $D_1(x, y)$, $D_2(x, y)$ and converting them into a continuous representation in the form of the functions $f_1(x, y)$ and $f_2(x, y)$, we apply an interpolation procedure between the discrete SDF values.

Step 3. The next step is applying STB to shapes S_1 and S_2 represented by the SDF functions $f_1(x, y)$ and $f_2(x, y)$ in a continuous form for generating intermediate frames of the metamorphosis. The texture transformation is not considered yet, as STB works only with geometry.

Step 4. To obtain the texture transformations we apply the STTI method described in subsection 3.4 to the input images. We calculate the sum of the color attributes weighted by the distance from the current pixel to other pixels using (3.5) without establishing correspondences. The resulting images can be created with a transparent background and superimposed onto any background image.

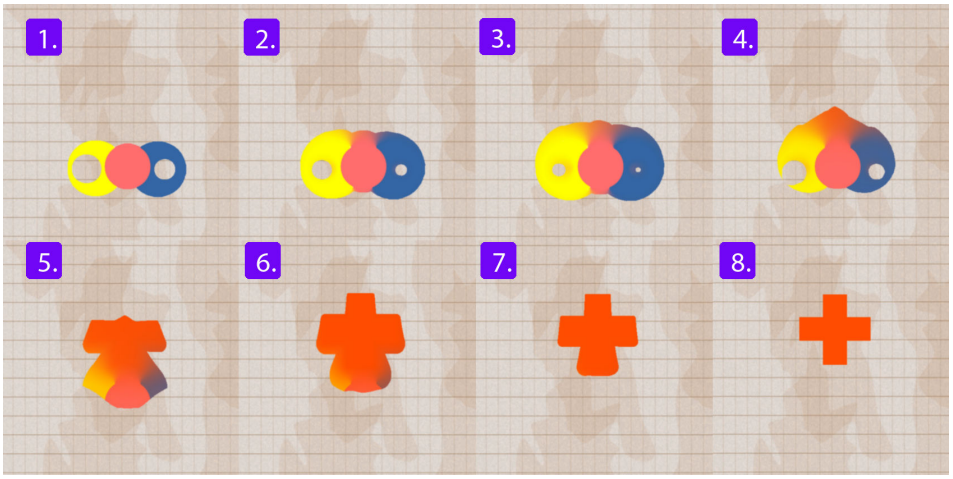

Figure 5. *The result of applying the basic algorithm to compute the metamorphosis between two shapes. See the accompanying supplemental video file* (figure5.mpg [local/web 792KB]).

4.2. Advantages and drawbacks of the method. Figure 5 demonstrates how the suggested method works. It shows a sequence of frames, demonstrating metamorphosis between an initial shape S_1 represented as three overlapping disks with different topology and various colors, and a target shape S_2 which is a red cross. We use (3.1)–(3.3) with manually set coefficients a_0, a_1, a_2, and a_3. As there was no automatic procedure for setting those coefficients, we had to experimentally select the values $a_0 = 2$, $a_1 = 1.3$, $a_2 = 1.5$, and $a_3 = 1$, which resulted in the most acceptable results. The outcome of this example allows us to make the following conclusions regarding advantages and drawbacks of the suggested method. The method has the following obvious advantages:

- It allows for generating in-between shapes in terms of both geometry and textures.

- Initial and target shapes S_1 and S_2 can be of an arbitrary complexity in terms of geometry and topology. In particular, they do not need to have similar shapes and can include disjoint components.
- Initial and target shapes S_1 and S_2 can overlap each other in place or they can be separated in space. The latter is a major issue when attempting cross-dissolving.

The following drawbacks pertain to this method:

- A problematic smooth transition can occur: some in-between shapes (see Figure 5, frames 1–3) appear overly similar, while there is an obvious nonsmooth jumpwise transition between some neighboring shapes (see Figure 5, frames 4–5). This is true for both geometry and color.
- An additional unexpected inflation in some in-between shapes appears (see Figure 5, frames 3, 5).
- The manual control for choosing the working set of parameters is problematic, especially for nonspecialist users. To realize an adequate automatic STB control, one needs to have a set of well-defined default parameters a_0, a_1, a_2, and a_3 in (3.1)–(3.3). However, the method for specifying those parameters, especially taking into account specific features of the initial and target objects, has not been developed yet.

In the next section a number of substantiated solutions for the mentioned problems are described.

5. New techniques enhancing the basic method. In this section we are going to introduce new techniques for solving the drawbacks of the method for automatic metamorphosis between 2D textured shapes introduced in section 4. A solution for these can be achieved by using either one of the following techniques or a combination of them. Let us briefly outline them:

- Half-cylinder smoothing: In subsection 3.3 it was mentioned that both initial and target shapes are considered as cross-sections of half-cylinders in 3D space. As the STB method assumes that half-cylinders are defined using set-theoretical operations, a rapid change in the gradient of the resulting shape in the time interval $t \in [0, 1]$ can appear because of the influence of those operations. This rapid change visually results in a "jump" during shape metamorphosis, which can be seen in Figure 5, frames 4–5. This problem can be solved by smoothing sharp edges of half-cylinders by additionally applying STB to them using (3.1)–(3.3).
- Automatic control of the STB parameters: It is nontrivial for a nonexpert user to select the satisfactory values for coefficients a_0, a_1, a_2, and a_3 defined in (3.1)–(3.3). To simplify this we introduce an algorithm for the automatic control of the coefficients a_0, a_1, a_2, and a_3. For this we use both image processing–based techniques and interval arithmetic–based estimations for the coefficient a_0 as well as exploiting the geometric meaning of the generalized distance $d_{r_1}(f_{1_t}, f_{2_t})$ in (3.3).
- New bounding solid functions: Some undesirable additional inflation of the shape during metamorphosis can appear even with automatic control of the coefficients a_0, a_1, a_2, and a_3. For better shape transformation control between the initial and target objects, we suggest using two specific functions, each defining a new bounding solid. One of these functions defines a truncated cone, and the other defines a truncated pyramid.

- Affine translation: We need to introduce a certain restriction on the distance between initial and target shapes S_1 and S_2. If the distance exceeds the defined limit, STB can produce disjoint in-betweenings. The problem can be solved by applying an affine translation to either initial shape S_1 or to target shape S_2 to satisfy the defined distance limit between them. In addition to this, an affine translation can be used as a shape inflation control while conducting the metamorphosis using the suggested method described in subsection 4.1.

All four proposed techniques will be discussed in detail in this section.

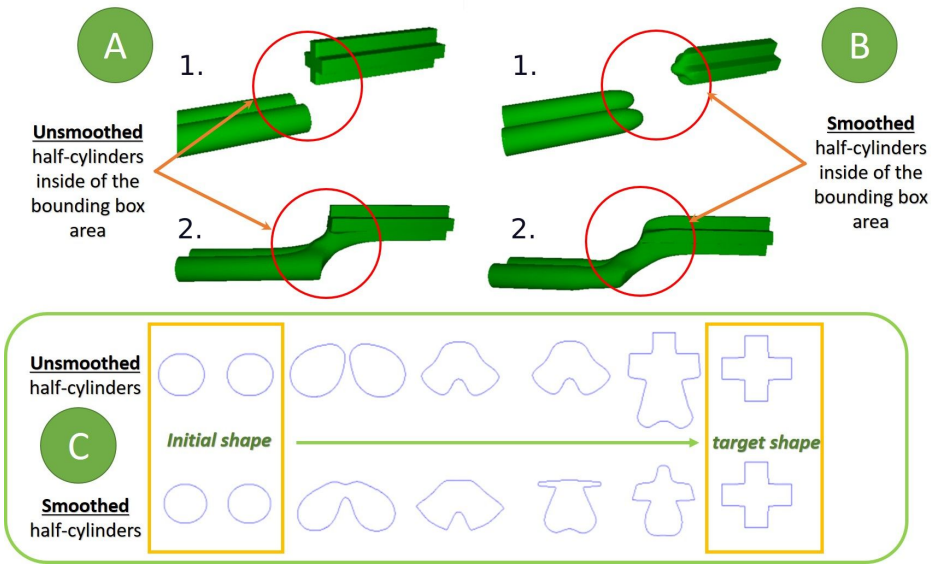

Figure 6. *The blending operation conducted between two circles and a cross that are extended to 3D space.* (A) *The result of the blending operation before applying the smoothing operation.* (B) *The result of the blending operation after applying the smoothing operation.* (C) *In-between cross-sections for the unsmoothed* (C, *top*) *and smoothed cases* (C, *bottom*).

5.1. Smoothing the half-cylinders. One of the drawbacks of STB is the presence of fast transitions or "jumps" in the $t \in [0, 1]$ interval (see Figure 2) in which the most significant shape transformation happens. The cause of this is that when STB is applied to a half-cylinder, the result is bounded by a plane orthogonal to the time axis. The set-theoretic subtraction of the half-space from the infinite cylinder results in a sharp edge for a half-cylinder boundary (see Figure 6, A), and the sharp edge remains a significant feature in the resulting blend (see Figure 6, A (bottom)). To avoid this effect, instead of using the set-theoretic subtraction of the half-space, we will use a bounded blending subtraction. Smoothing can be achieved by applying a blending intersection operation, removing material between the infinite cylinder and the bounding planar half-space for both shapes (see Figure 6, B):

$$F_b(f_{1_t}, f_{2_t}, f_{3_t}) = F(f_{1_t}, f_{2_t}) + a_0 \cdot disp_b(d_r(f_{1_t}, f_{2_t}, f_{3_t})),$$
$$F(f_{1_t}, f_{2_t}) = f_{1_t}(x, y, t) + f_{2_t}(x, y, t) - \sqrt{f_{1_t}(x, y, t)^2 + f_2(x, y, t)^2}, \tag{5.1}$$

where $F_b(f_{1_t}, f_{2_t}, f_{3_t})$ is the resulting blending function defining a smoothed half-cylinder, $\text{disp}_b(d_r(f_{1_t}, f_{2_t}, f_{3_t}))$ is the displacement defined in (3.2)–(3.3), and $F(f_{1_t}, f_{2_t})$ is an FRep set-theoretical intersection operation. If we compare the two bottom pictures, A.2 and B.2 in Figure 6, we notice that smoothing the sharp edges of the cylinders results in a smooth transition between the two shapes (B, bottom).

The smoothing process for initial and target shapes S_1 and S_2 can be described as follows:

$$S_1: \quad f_{1_t}(x,y,t) = f_{SDF_1}(x,y,t); \quad f_{2_t}(x,y,t) = -t; \quad f_{3_t}(x,y,t) = t + P_c, \quad t \geq -P_c, \tag{5.2}$$

$$S_2: \quad f_{1_t}(x,y,t) = f_{SDF_2}(x,y,t); \quad f_{2_t}(x,y,t) = t - 1; \quad f_{3_t}(x,y,t) = P_c - t, \quad t \leq P_c, \tag{5.3}$$

where $f_{1_t}(x,y,t)$ is the SDF function raised into 3D space defining either the initial shape S_1 or the target shape S_2, $f_{2_t}(x,y,t)$ is the function defining a smoothing object as the subtraction of the negative half-space along time-axis t, $f_{3_t}(x,y,t)$ is the bounding function restricting the area where the STB bounding intersection happens, and P_c is the position of the cutting plane on the time-axis t. These functions are substituted in (5.1) for obtaining the smoothed result. For smoothing both the initial shape S_1 as well as the target shape S_2, we need to manually set the coefficients a_0, a_1, a_2, and a_3.

In the lower half of Figure 6 (part C), we compare the results before (C, top) and after (C, bottom) application of the suggested smoothing method to both the initial shape S_1 (two circles) and the target shape S_2 (a cross). As the figure shows, after applying the smoothing operation to the half-cylinders, "jumps" between the frames have disappeared and there are no more similar in-between shapes.

5.2. Automatic control of the space-time blending parameters. Automatic control of the coefficients a_0, a_1, a_2, and a_3 allows us to make our approach more user-friendly. Here we describe the derivation of the expressions for estimating the coefficients a_0, a_1, and a_2. We intend to make these coefficients geometry-dependent for better control of the blending shape in cases of both in-place as well as spaced image morphing. These coefficients should be greater than or equal to 1 to guarantee that a transition between the initial and target shapes is visible. This procedure depends on the geometrical properties of both initial and target objects and the geometrical meaning of the coefficients a_0, a_1, and a_2 described in subsection 3.3.

Let us assume that $a_3 = 1$. Then, according to (3.3), it does not affect the bounding solid defined by the function $f_{3_t}(x,y,t)$. At the initial step we need to find the circumscribed circles around initial shape S_1 and target shape S_2 shown in Figure 7. It is not essential that these two circles overlap. The radii of the circumscribed circles R_1 and R_2 and their center coordinates O_1 and O_2 are known. As the input data is represented by the images, all the calculations are made in a normalized coordinate system: $x = x/I_w$, $y = y/I_h$, where I_w and I_h are image width and height.

The proposed estimation, instead of using the bounded blending function $F_b(f_{1_t}, f_{2_t}, f_{3_t})$ defined in (3.1), relies on a blending equation [30] without time dependence introduced in the

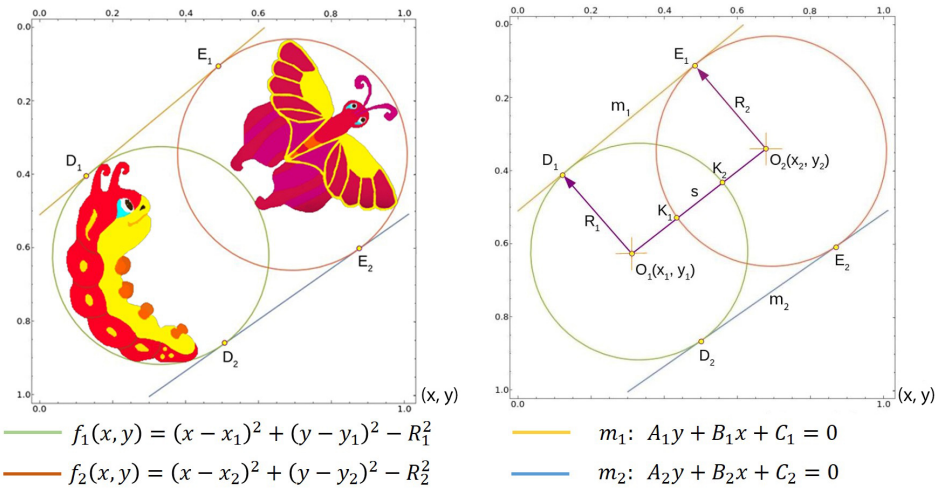

Figure 7. *Geometrical scheme for estimating the coefficients a_0 and $a_1 = a_2$, where S_1 is the initial shape and S_2 is the target shape.*

following form:

$$F_{blend}(f_1, f_2) = f_1(x,y) + f_2(x,y) + \sqrt{f_1(x,y)^2 + f_2(x,y)^2} + \frac{a_0}{1 + d_{r_1}(f_1, f_2)}, \tag{5.4}$$

$$d_{r_1}(f_1, f_2) = \sqrt{\left(\frac{f_1(x,y)}{a_{1,2}}\right)^2 + \left(\frac{f_2(x,y)}{a_{1,2}}\right)^2},$$

where $f_1(x,y)$ and $f_2(x,y)$ are functions with distance property and $d_{r_1}(f_1, f_2)$ is a generalized distance between the initial and target shapes represented by functions $f_1(x,y)$ and $f_2(x,y)$.

The estimation process starts by finding the values for coefficients a_1 and a_2. For simplification let us assume that the blending process will be symmetric. This means that $a_1 = a_2 = a_{1,2}$. As stated in [30], coefficients $a_1 > 0$ and $a_2 > 0$ are proportional to the algebraic distance between the blending surface and the original surfaces S_1 and S_2 defined by functions $f_1(x,y)$ and $f_2(x,y)$.

To estimate coefficient $a_{1,2}$ we suggest the following algorithm: The generalized distance $d^2_{r_1}(f_1, f_2)$ used in (5.4) is equal to the quadratic distance $||s||^2$ (see Figure 7) between two shapes, S_1 and S_2. To calculate this distance we need to solve two equation systems for both circles as well as segment O_1O_2 to define the coordinates of points K_1 and K_2. Then the distance $||s||^2$ between K_1 and K_2 can be defined as

$$||s||^2 = (x_{K_2} - x_{K_1})^2 + (y_{K_2} - y_{K_1})^2. \tag{5.5}$$

Let us assume that the generalized distance $d^2_{r_1}(f_1(x_{K_2}, y_{K_2}), f_2(x_{K_1}, y_{K_1}))$ between shape S_1 and shape S_2 is equal to the quadratic Euclidean distance $||s||^2$ between these two shapes

(see Figure 7). Then we obtain the final estimation for $a_1 = a_2 = a_{1,2}$:

$$a_{1,2} = \sqrt{\frac{f_1(x_{K_2}, y_{K_2})^2 + f_2(x_{K_1}, y_{K_1})^2}{||s||^2}}. \tag{5.6}$$

To estimate coefficient a_0 we suggest using interval arithmetic [25]. Here we provide an outline of the solution for the interval of the coefficient a_0. The whole derivation for this interval is provided in SupplementaryMaterials1.pdf [local/web 215KB]. Let us assume for simplicity that the two input shapes S_1 and S_2, defined by functions f_1 and f_2, respectively, are represented by circumscribed circles (to simplify further, we will use f_1 and f_2). We bound the blending operation defined by (5.4) using two tangential lines, m_1 and m_2 (see Figure 7), which can be written in the general case as

$$m_1(x, y) = A_1 x + B_1 y + C_1, \quad m_2(x, y) = A_2 x + B_2 y + C_2,$$

where A_1, B_1, C_1 and A_2, B_2, C_2 are constants that can be obtained by solving the following system of equations:

$$\begin{cases} A_i x_1 + B_i y_1 + C_i = R_1, \\ A_i x_2 + B_i y_2 + C_i = R_2, \\ A_i^2 + B_i^2 = 1, \end{cases}$$

where $i = 1, 2$, (x_1, y_1) are the coordinates of point O_1, (x_2, y_2) are the coordinates of point O_2, R_1 is the radius of the circumscribed circle with the center point O_1, and R_2 is the radius of the circumscribed circle with the center point O_2.

The final intervals for a_0 can be written in simplified form assuming that the intervals for the functions $[f_1] = [f_{1_{\min}}, f_{1_{\max}}]$ and $[f_2] = [f_{2_{\min}}, f_{2_{\max}}]$ are as follows:

$$\begin{cases} a_0 \le (A_1 x + B_1 y + C_1 - (f_1 + f_2 + \sqrt{f_1^2 + f_2^2}))(1 + \frac{1}{a_{1,2}^2}(f_1^2 + f_2^2)), \\ a_0 \ge (A_2 x + B_2 y + C_2 - (f_1 + f_2 + \sqrt{f_1^2 + f_2^2}))(1 + \frac{1}{a_{1,2}^2}(f_1^2 + f_2^2)). \end{cases} \tag{5.7}$$

The inequality system of equations (5.7) can be rewritten using interval arithmetic as

$$\begin{aligned} a_0 \le [&\min[g_{1_{min}} b_{1_{max}}, g_{1_{max}} b_{1_{min}}, g_{1_{min}} b_{1_{min}}, g_{1_{max}} b_{1_{max}}], \\ &\max[g_{1_{min}} b_{1_{max}}, g_{1_{max}} b_{1_{min}}, g_{1_{min}} b_{1_{min}}, g_{1_{max}} b_{1_{max}}]], \\ a_0 \ge [&\min[g_{2_{min}} b_{1_{max}}, g_{2_{max}} b_{1_{min}}, g_{2_{min}} b_{1_{min}}, g_{2_{max}} b_{1_{max}}], \\ &\max[g_{2_{min}} b_{1_{max}}, g_{2_{max}} b_{1_{min}}, g_{2_{min}} b_{1_{min}}, g_{2_{max}} b_{1_{max}}]], \\ [g_1] = [&A_1 x_{min} + B_1 y_{min} + C_1 - f_{s_{max}}, A_1 x_{max} + B_1 y_{max} + C_1 - f_{s_{min}}], \\ [g_2] = [&A_2 x_{min} + B_2 y_{min} + C_2 - f_{s_{max}}, A_2 x_{max} + B_2 y_{max} + C_2 - f_{s_{min}}], \\ [b_1] = &\left[\left(f_{1_{min}} f_{1_{max}} + f_{2_{min}} f_{2_{max}}\right)\frac{1}{a_{1,2}^2} + 1, \left(f_{1_{max}}^2 + f_{2_{max}}^2\right)\frac{1}{a_{1,2}^2} + 1\right], \\ [f_{s_{min}}, f_{s_{max}}] = &\left[f_{1_{min}} + f_{2_{min}} + \sqrt{f_{1_{min}} f_{1_{max}} + f_{2_{min}} f_{2_{max}}}, \right. \\ &\left. f_{2_{max}} + f_{2_{max}} + \sqrt{f_{1_{max}}^2 + f_{2_{max}}^2}\right]. \end{aligned} \tag{5.8}$$

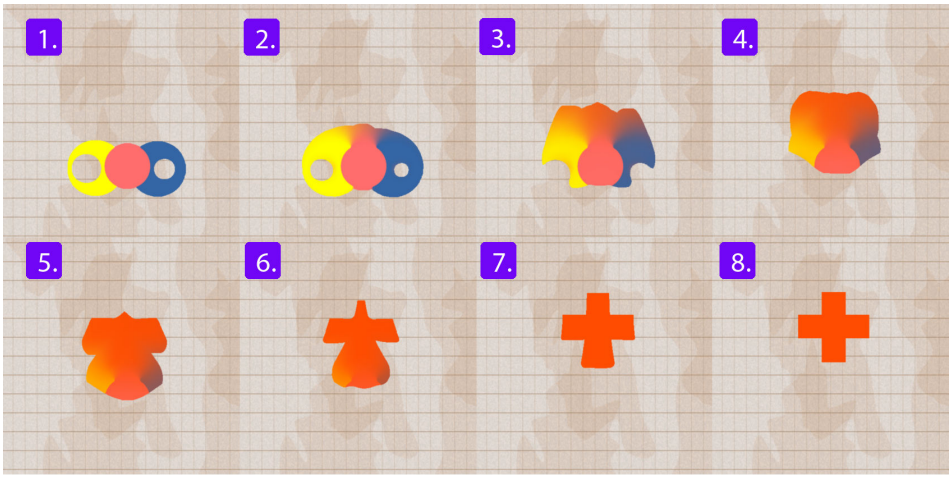

Figure 8. *Blending between shape S_1 consisting of three circles, two of which have a hole, and an orange cross S_2 using the STB and STTI techniques. See the accompanying supplemental video file* (figure8.mpg [local/web 788KB]).

The final interval for a_0 can then be obtained depending on the following cases:

- if $a_0 \leq [a_{0_{\min_1}}, a_{0_{\max_1}}] \wedge a_0 \geq [a_{0_{\min_2}}, a_{0_{\max_2}}]$ and $a_{0_{\min_1}} < 0; a_{0_{\min_2}} < 0$, then the final interval will be $a_0 \in [a_{0_{\max_1}}, a_{0_{\max_2}}]$;
- if $a_0 \leq [a_{0_{\min_1}}, a_{0_{\max 1}}] \wedge a_0 \geq [a_{0_{\min_2}}, a_{0_{\max_2}}]$ and $0 \leq a_{0_{\min_1}} \leq 1; 0 \leq a_{0_{\min_2}} \leq 1$, then the final interval will be $a_0 \in [1, \max(a_{0_{\max_1}}, a_{0_{\max_2}})]$;
- if $a_0 \leq [a_{0_{\min_1}}, a_{0_{\max_1}}] \wedge a_0 \geq [a_{0_{\min_2}}, a_{0_{\max_2}}] = \varnothing$, this may be an overestimation for a_0 and the final interval will be formed as $a_0 \in [a_{0_{\min_1}}, a_{0_{\max_2}}]; a_{0_{\min_1}} \geq 1$.

Figure 8 shows an example of applying the smoothing operation to the half-cylinders with automatic control of the coefficients a_0, a_1, a_2, and a_3. Comparing these results with those shown in Figure 5, generated using the basic algorithm described in section 4, one can conclude that here the transition between initial shape S_1 and target shape S_2 is smooth and there is no additional inflation within in-betweens.

In cases where circumscribed circles are not overlapping and their centers coincide, the estimation procedure for a_0 (5.7) should be slightly modified. The coefficients $a_1 = a_2 = a_{1,2}$ should be set to 1 and inequality (5.7) rewritten as

$$a_0 \leq \left((x - x_i)^2 + (y - y_i)^2 - R_i^2 - (f_1 + f_2 + \sqrt{f_1^2 + f_2^2})\right)\left(1 + \frac{1}{a_{1,2}^2}\left(f_1^2 + f_2^2\right)\right),$$

where $i = 1, 2$, depending on the chosen radius of the biggest circumscribed circle $\max(R_1, R_2)$. In this case the blending area is restricted by the circumscribed circle with the biggest radius. Then, using interval arithmetic, we can obtain the final interval (for more details see

SupplementaryMaterials1.pdf [local/web 215KB])

$$
\begin{aligned}
(5.9)\qquad a_0 \le\ & [\min[f_{c_{min}} b_{1_{max}}, f_{c_{max}} b_{1_{min}}, f_{c_{min}} b_{1_{min}}, f_{c_{max}} b_{1_{max}}],\\
& \max[f_{c_{min}} b_{1_{max}}, f_{c_{max}} b_{1_{min}}, f_{c_{min}} b_{1_{min}}, f_{c_{max}} b_{1_{max}}]],\\
[f_c] =\ & [(x_{min} - x_i)(x_{max} - x_i) + (y_{min} - y_i)(y_{max} - y_i) - \max(R_1^2, R_2^2),\\
& (x_{max} - x_i)^2 + (y_{max} - y_i)^2 - \max(R_1^2, R_2^2)],\\
i = & \begin{cases} 1 & \text{if } \max(R_1^2, R_2^2) = R_1^2, \\ 2 & \text{if } \max(R_1^2, R_2^2) = R_2^2, \end{cases}
\end{aligned}
$$

where the interval for $[b_1]$ was introduced in (5.8).

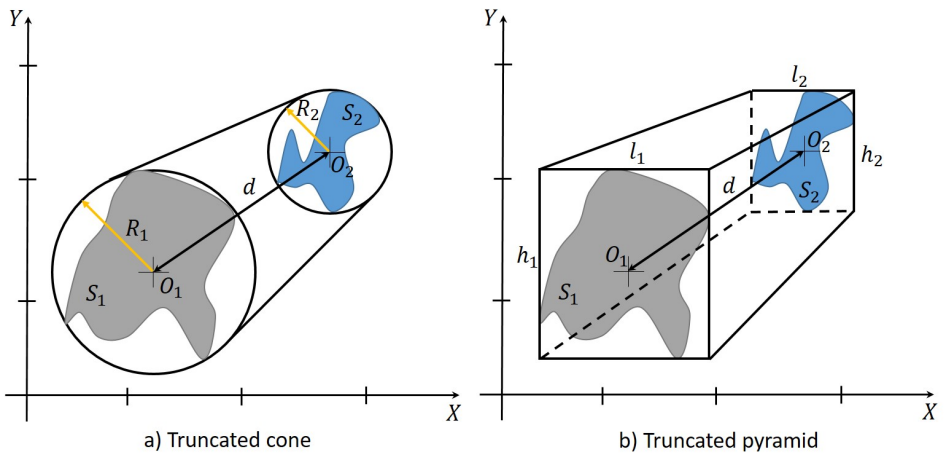

Figure 9. *New functions for defining the bounding solid* $f_{3_t}(x, y, t)$*:* (a) *truncated cone;* (b) *truncated pyramid.* S_1 *is the initial object and* S_2 *is the target object.*

5.3. Truncated cone and truncated pyramid as a bounding solid. An additional inflation caused by the lack of the STB control can be reduced using a different technique. We suggest using a tighter bounding solid $f_{3_t}(x, y, t)$, represented as either a truncated cone or as a truncated pyramid (Figure 9).

We assume that the circumscribed circles have already been obtained for both initial object S_1 and target object S_2. To avoid shape inflation, we specify that the entire metamorphosis process happens within the new bounding solid defined by function $f_{3_t}(x, y, t)$.

Let us introduce a truncated cone with circumscribed circles around initial and target shapes S_1 and S_2 as its two bases. To obtain the defining function $f_{3_t}(x, y, t)_{cone}$ for the truncated cone (Figure 9(a)), a simple linear interpolation between the centers and radii of the circles is applied, where the parameter k is defined in the interval $k \in [0, 1]$, which is

dependent on time t:

$$f_{3_t}(x,y,t)_{cone} = R_m^2 - (x - \widehat{X}_m)^2 - (y - \widehat{Y}_m)^2, \tag{5.10}$$
$$R_m = R_1 + (R_2 - R_1)k, \quad \widehat{X}_m = x_{c_1} + (x_{c_2} - x_{c_1})k,$$
$$\widehat{Y}_m = y_{c_1} + (y_{c_2} - y_{c_1})k, \quad k = \frac{t - t_1}{t_2 - t_1}, \quad t_1 = -10; \quad t_2 = 10.$$

Here (x_{c_1}, y_{c_1}) are the center coordinates O_1 of the circumscribed circle around initial shape S_1, (x_{c_2}, y_{c_2}) are the center coordinates O_2 of the circumscribed circle around target shape S_2, R_1 and R_2 are the radii of the two circumscribed circles, and t_1 and t_2 are the minimum and the maximum values of the interval time t. We use the values recommended for t_1 and t_2 from the article on STB [30].

A truncated pyramid is defined (Figure 9(b)) with circumscribed rectangular bounding boxes for initial and target shapes S_1 and S_2 as its two bases. For this case we need to define each base of the pyramid as a result of the FRep set-theoretic intersection operation and linear interpolation between the widths l_1, l_2, the heights h_1, h_2, and the centers of the rectangles:

$$f_{3_t}(x,y,t)_{pyramid} = (L_m - |x - \widehat{X}_m|) \wedge (H_m - |y - \widehat{Y}|_m), \tag{5.11}$$
$$L_m = l_1 + (l_2 - l_1)k, \quad H_m = h_1 + (h_2 - h_1)k,$$
$$\widehat{X}_m = x_{r_1} + (x_{r_2} - x_{r_1})k, \quad \widehat{Y}_m = y_{r_1} + (y_{r_2} - y_{r_1})k, \quad k = \frac{t - t_1}{t_2 - t_1}, \quad t_1 = -10; \quad t_2 = 10.$$

Here (x_{r_1}, y_{r_1}) and (x_{r_2}, y_{r_2}) are the coordinates of the rectangles' centers O_1 and O_2.

To define a new function for the bounding solid as either a truncated cone or a truncated pyramid we need to use the FRep set-theoretic intersection operation $\wedge$ between either the cone defined by $F_{solid}(x,y,t) = f_{3_t}(x,y,t)_{cone}$ or the pyramid defined by $F_{solid}(x,y,t) = f_{3_t}(x,y,t)_{pyramid}$ with the two time-cutting planes $(t+10)$ and $(10-t)$:

$$f_{3_t}(x,y,t) = F_{solid}(x,y,t) \wedge (t+10) \wedge (10-t). \tag{5.12}$$

In Figure 10 we present the results of applying three different methods with automatic control of the coefficients a_0, a_1, and a_2: (a) is the result of using two half-planes defined by function $f_{3_t}(x,y,t)_{planes} = (t+10) \wedge (10-t)$, (b) is the result of using the truncated pyramid defined by function $f_{3_t}(x,y,t)_{pyramid}$, and (c) is the result of using the truncated cone defined by function $f_{3_t}(x,y,t)_{cone}$. The metamorphosis shape obtained using the two half-planes defined by function $f_{3_t}(x,y,t)_{planes}$ is quite similar to the metamorphosis shape obtained using function $f_{3_t}(x,y,t)_{pyramid}$, set up as a truncated pyramid.

From Figure 10, it follows that

- using the truncated pyramid (see Figure 10(c)) as the new bounding box provides a more visually accurate result compared to that obtained by the two half-plane bounding solid (see Figure 10(a)), but it produces a slightly more inflated result;
- using the truncated cone (see Figure 10(b)) as the new bounding box results in less inflation. The in-between shapes in this case are also quite different. Note that for the truncated cone, better (smoother) results can be obtained when the number of frames in the animation sequence is increased.

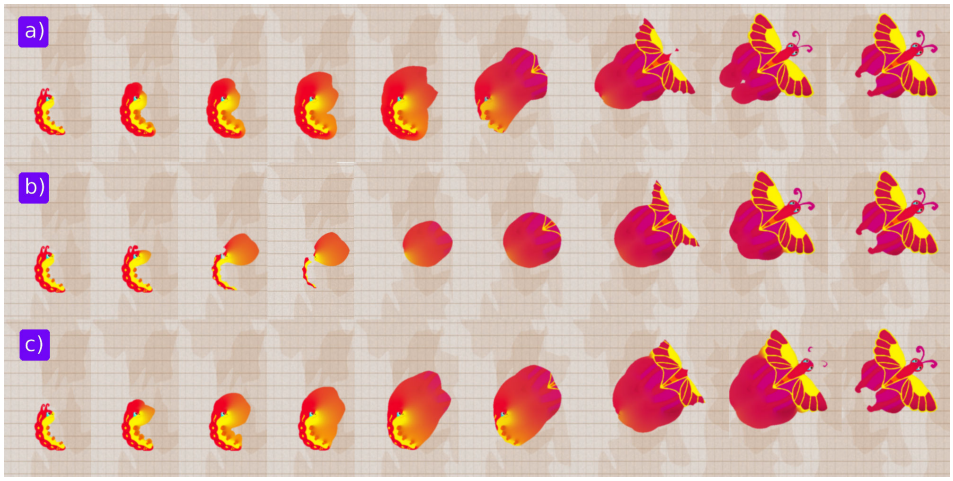

Figure 10. *Blending operation conducted between two shapes, S_1 (caterpillar) and S_2 (butterfly), using* (a) *two half-planes as bounding solid function $f_{3_t}(x,y,t)_{planes}$;* (b) *truncated pyramid as bounding solid function $f_{3_t}(x,y,t)_{pyramid}$;* (c) *truncated cone as bounding solid function $f_{3_t}(x,y,t)_{cone}$. See the accompanying supplemental video files* (figure10a.mpg [local/web 944KB], figure10b.mpg [local/web 1.16MB], and figure10c.mpg [local/web 900KB]).

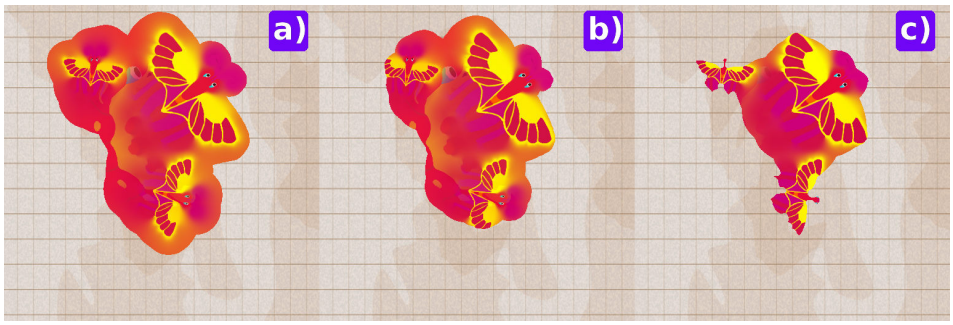

Figure 11. *Blending operation conducted between two shapes, S_1 (caterpillar) and S_2 (three butterflies), in the case of an in-place morphing using* (a) *two half-planes as bounding solid function $f_{3_t}(x,y,t)_{planes}$;* (b) *truncated pyramid as bounding solid function $f_{3_t}(x,y,t)_{pyramid}$;* (c) *truncated cone as bounding solid function $f_{3_t}(x,y,t)_{cone}$. See the accompanying supplemental video files* (figure11a.mpg [local/web 112KB], figure11b.mpg [local/web 116KB], and figure11c.mpg [local/web 108KB]).

The criterion for using one of the suggested new bounding boxes is as follows: If the distance d (see Figure 9) between the centers of two circumscribed circles satisfies the condition $d \leq (R_1 + R_2)$, the blending shape obtained with automatic estimation of coefficients a_i, $i = 0, 1, 2, 3$, might still have some additional inflation. In the case of an in-place animation, for example, when there is one big object transformed into several smaller ones (see Figure 11(a)), it is recommended to apply one of the introduced bounding solids. As seen in Figure 11(b), using the truncated pyramid produces slightly less inflated results compared to the bounding solid defined as two bounding planes in Figure 11(a). Using the truncated cone provides even better results (see Figure 11(c)).

5.4. Affine translation for space-time blending shape control. If two shapes, S_1 and S_2, are placed too far away from each other, there is a chance that disconnections in in-between shapes might appear. To avoid this, shape S_1 can be left static and target shape S_2 should be shifted closer to shape S_1, or vice versa. This can be achieved by applying an affine translation to shape S_2.

The criterion for applying the affine translation is the following: If the distance d (see Figure 9) between the centers of two circumscribed circles satisfies the condition $d \gg (R_1+R_2)$, where R_1 is the radius of the first circle and R_2 is the radius of the second circle, then the two objects S_1 and S_2 are too far from each other. In this case there is a chance that the algorithm for automatic estimation of the coefficients a_i, $i = 0, 1, 2, 3$, will not produce satisfying results, so additional enhancement is essential.

Affine translation allows for reducing the additional shape inflation during the STB metamorphosis. According to the algorithm presented in subsection 4.1, at the first step we need to calculate the SDDT for both input and target shapes S_1 and S_2. Then the geometric centers of the two shapes are superimposed at the initial time step. If two shapes are inside each other, we need to calculate the interval for a_0 according to (5.9). Then we need to check whether the shifted shape S_2 is still within the borders of the circumscribed circle around the static shape S_1. This can be achieved by solving the system of equations for both circles.

Two tangential lines can be drawn to these two circles when this system has two non-coincident real solutions. In that case the algorithm for automatic control of coefficient a_0 described in subsection 5.2 can be used for the rest of the motion of shape S_2. After defining a_0, a_1, and a_2, at each step we apply mapping of the SDDT values from the initial position of shape S_2 to its current position to compute the in-between shapes.

After each SDDT mapping is done, STB is calculated according to (3.1)–(3.3) and an affine translation is applied to the result. The image sequence obtained after applying the affine translation can be seen in Figure 12, where the butterfly (target shape, S_2) is emerging from the caterpillar (static initial shape S_1) while moving toward its initial position. This operation does not produce any additional shape inflation. At each step we use automatic control for recalculation of the coefficients a_0, a_1, and a_2 described in subsection 5.2 and apply a smoothing half-cylinder operation to both shapes as described in subsection 5.1.

6. Implementation and results. In this section we will demonstrate how the proposed method described in section 4 and the techniques introduced in section 5 are applied in combination. We will describe our implementation and discuss its efficiency.

6.1. Algorithmic implementation. We have implemented our algorithm using C++, the OpenCV library for the basic image handling, and OpenMP for multithreading. All examples were computed on a laptop with a 2.6 GHz Intel Skylake 6700 processor and 16 GB of RAM. The implementation of the method is available on GitHub.[1]

Following the basic algorithm described in subsection 4.1, as the first step we need to binarize the input images and use them for calculating the SDDT. We implemented this using the 8-Points Signed Sequential Euclidean Distance Transformation algorithm from [20], which was later improved in [24].

[1] https://github.com/teshaTe/2D-heterogeneous-metamorphosis

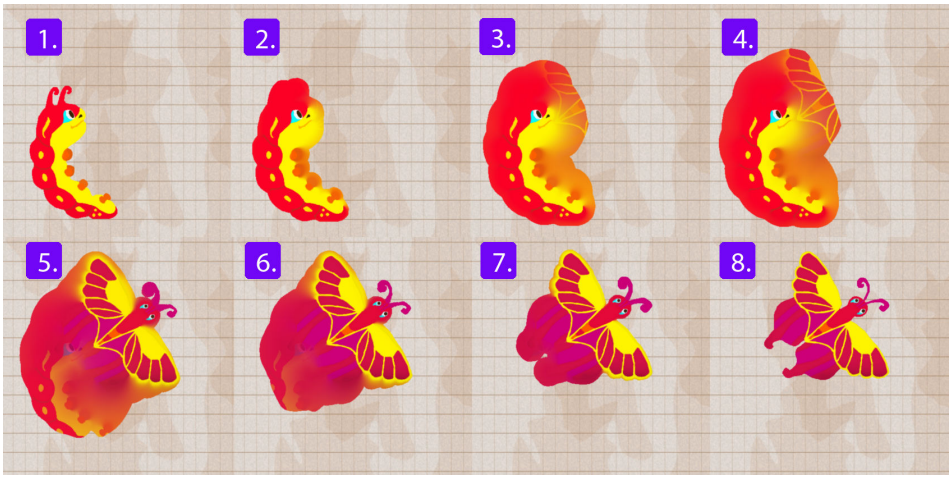

Figure 12. *Blending operation conducted between two shapes, S_1 (caterpillar) and S_2 (butterfly), using affine translation. See the accompanying supplemental video file* (figure12.mpg [local/web 1.01MB]).

Table 1

Comparison table of execution speeds for the image metamorphosis used in the article with timings given as frames per second (FPS). In the case of an affine translation (Figure 12) the final two columns are left empty as intervals a_0 and $a_{1,2}$ are dynamically recalculated at each step.

Execution: Sequential / Parallel / Parallel + Scaled Image					
Figure	Average FPS	Min FPS	Max FPS	a_0 interval	$a_{1,2}$
Fig. 8	1.69 / 3.37 / 18.82	40.19/ 32.33 / 46.25	0.10 / 0.22 / 2.82	[2.49, 3.85]	1.12
Fig. 10					
(a)	0.78 / 1.64 / 13.46	38.98 / 31.08 / 42.38	0.08 / 0.17 / 2.21	[0.48, 3.26]	1.78
(b)	1.23 / 2.42 / 16.18	38.32 / 30.61 / 42.63	0.14 / 0.29 / 3.76	[0.48, 3.26]	1.78
(c)	0.52 / 1.20 / 10.47	35.32 / 27.66 / 40.33	0.07 / 0.17 / 2.22	[0.48, 3.26]	1.78
Fig. 12	0.10/ 0.14 / 2.52	0.09/ 0.10 / 1.63	0.02/ 0.03 /1.07	-	-
Fig. 13	0.71 / 1.88 / 9.36	33.82 / 27.12 / 41.19	0.05 / 0.13 / 0.34	[-20.22, 2.58]	1.00
Fig. 15	0.59 / 1.29 / 12.30	33.70 / 26.88 / 40.20	0.04 / 0.10 / 1.21	[1.75, 2.85]	1.30
Fig. 16	0.49 / 1.26 / 12.72	37.36 / 26.40 / 40.32	0.04 / 0.09 / 1.33	[-21.01, 2.57]	1.00
Fig. 18	1.23 / 2.44 / 20.10	34.28 / 29.62 / 41.10	0.06 / 0.14 / 1.93	[-18.58, 2.06]	1.00
Fig. 19	0.49 / 1.11 / 12.72	37.36 / 28.61 / 40.32	0.04 / 0.09 / 1.33	[3.57, 3.76]	1.00

To create two positive distance fields for input and target shapes S_1 and S_2, which contain the distances to the surfaces, the algorithm employs two grids. The generation procedure for the SDDT can be described as follows:

Step 1. We use the binarized images for initializing both grids: one of the grids is initialized with zeros outside the shape and with a maximum value inside the shape, while another grid is filled with maximum values outside the shape and zeros inside the shape.

Step 2. The grids are traversed multiple times in order to compare the distance written at the current point with the distances stored in its 8 neighbors. If the result of this distance comparison is less than the one stored at the current point, the value is updated.

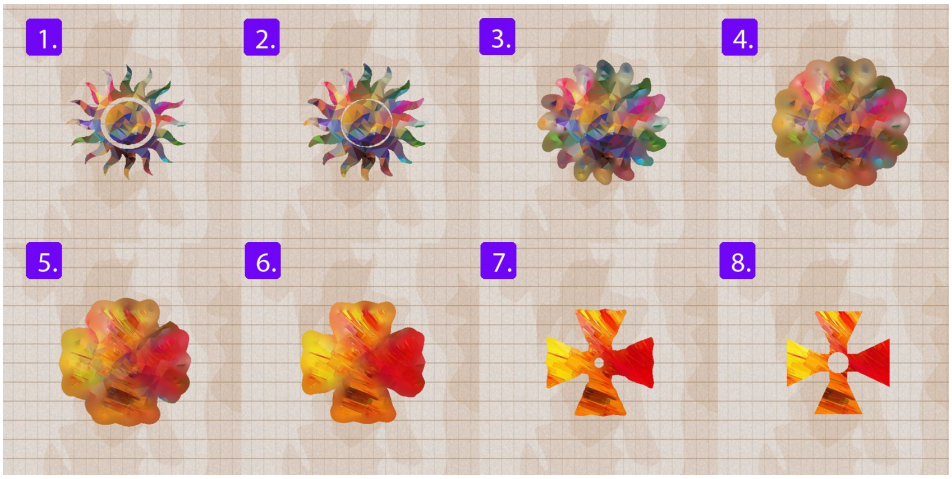

Figure 13. *Metamorphosis between two textured shapes using SDF, STB, and STTI techniques, where initial shape S_1 is a textured sun and target shape S_2 is a textured cross. See the accompanying supplemental video file* (figure13.mpg [local/web 88.0KB]).

Step 3. Two grids are merged together, and the result is stored in the final SDDT grid.

We then implement the second step described in the basic algorithm from subsection 4.1. Before executing the third step of the basic algorithm, we evaluate the coefficients a_0, a_1, a_2, and a_3 using the algorithm described in subsection 5.2. We also apply the smoothing of the half-cylinder algorithm described in subsection 5.1 to both shapes S_1 and S_2, both represented with SDFs. We can then proceed to the third step of the basic algorithm from subsection 4.1 and calculate the blending between the two shapes. If there is a need to reduce possible unwanted inflation during the STB operation, we suggest using one of the bounding solids defined with functions introduced in subsection 5.3 as a new function $f_{3_t}(x, y, t)$ in (3.1)–(3.3).

Finally, we apply the last step of the basic algorithm to obtain a smooth color interpolation between S_1 and S_2 using the STTI technique described in subsection 3.4.

If it is essential to use the affine translations described in subsection 5.4, these should be applied between the second and third steps of the basic algorithm. Note that coefficients a_0, a_1, a_2, and a_3 should be evaluated at each step of the shape translation using the algorithm described in subsection 5.2.

6.2. Examples. In this subsection we will discuss the results of application of the proposed method, shown in Figures 8, 10, 12, 13, 15, 18, and 19. We provide the corresponding animation sequences in the supplementary video files for those figures. In Table 1 we present the comparative timing results in frames per second (FPS) to demonstrate that interactive frame rates have been achieved.

For the examples shown in Figures 8, 13, 15, 16, 18, and 19 we use the following values as coefficients a_0, a_1, a_2, and a_3 for smoothing half-cylinders (see subsection 5.1), which were obtained from a number of conducted experiments. On the basis of this we suggest the following values for the initial shape S_1: the position for the cutting plane $P_c = 5$ and coefficients $a_0 = -0.3$, $a_1 = a_2 = a_3 = 1$. For the target shape S_2 we suggest the following

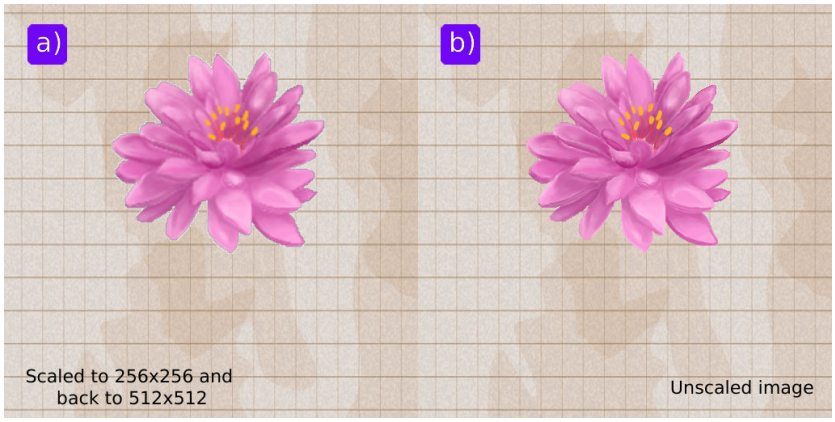

Figure 14. *Comparison of the loss in quality between scaled and unscaled images while conducting metamorphosis;* (a) *image scaled from* 512×512 *to* 256×256 *and after application of STB and STTI scaled back to* 512×512*;* (b) *original picture without scaling.*

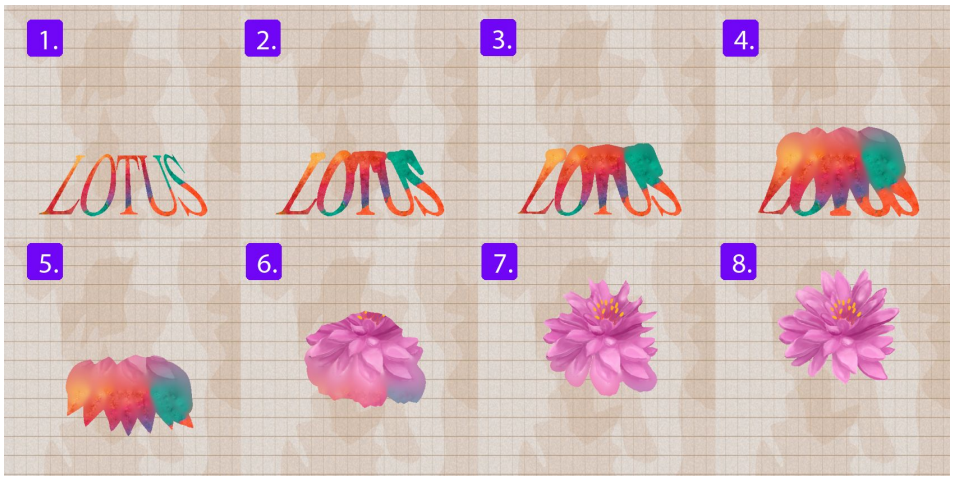

Figure 15. *Metamorphosis between two textured shapes,* S_1 *(text "LOTUS") and* S_2 *(lotus flower), using SDF, STB, and STTI. See the accompanying supplemental video file* (figure15.mpg [local/web 86.0KB]).

values: the position for the cutting plane $P_c = 5$ and coefficients $a_0 = -0.5$, $a_1 = a_2 = a_3 = 1$.

In Figures 8, 13, 15, 16, 18, 19, and 20 we show the results obtained using the combination of the suggested basic method with automatic control of the coefficients a_0, a_1, and a_2 and smoothing of the half-cylinders. As follows from the figures, this combination of the techniques produces good quality in-between shapes with a smooth transition between them and without an additional undesired inflation. From Table 1, it can be seen that this combination of methods works with interactive rates.

In Figure 10 we demonstrate the result obtained using the combination of the suggested basic method with automatic control of the coefficients a_0, a_1, and a_2, smoothing of the half-cylinders, and applying two new bounding solids. As discussed in subsection 5.3 the result will be slightly inflated when using a truncated pyramid, and the shape of the in-betweens

will be different; however, they will be without an additional inflation when using a truncated cone instead. From Table 1 it can be seen that both new bounding solids slow down the calculation, albeit not drastically. For this example on the basis of some experiments that we conducted, we suggest setting up the parameters for automatic control as follows:

- For the truncated cone coefficient $a_3 = 0.03 \cdot \min(R_1, R_2)$: for smoothing of the half-cylinders, the coefficients should be $a_0 = -0.8$, $a_1 = a_2 = a_3 = 1$ for the initial shape S_1, and $a_0 = -0.5$, $a_1 = a_2 = a_3 = 1$ for the target shape S_2.
- For the truncated pyramid coefficient $a_3 = 0.13 \cdot \min(R_1, R_2)$: for smoothing of the half-cylinders the coefficients should be $a_0 = -0.3$, $a_1 = a_2 = a_3 = 1$ for the initial shape S_1, and $a_0 = -0.5$, $a_1 = a_2 = a_3 = 1$ for the target shape S_2.

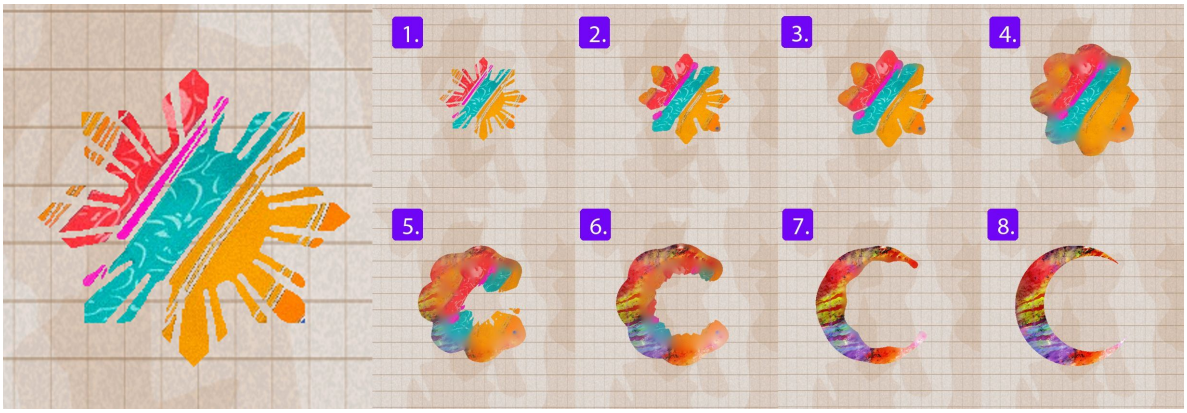

Figure 16. *Metamorphosis between a multiply subdivided, textured star shape S_1 and a textured moon S_2 using the SDF, STB, and STTI techniques for a radical topology change. See the accompanying supplemental video file* (figure16.mpg [local/web 100KB]).

In Figure 12 we demonstrate the result obtained using the combination of the suggested basic method with automatic control for the coefficients a_0, a_1, and a_2, smoothing the half-cylinders technique, and applying an affine translation to the target shape. As the figure shows, applying an affine translation produces the results without any additional inflation. Note that according to Table 1, applying an affine translation can significantly reduce the calculation speed for the whole metamorphosis.

In Figures 15, 16, 18, 19, and 20 we demonstrate that the introduced method with automatically controlled metamorphosis can handle transformations between multiple objects and objects with complex topologies and different textures while simultaneously preserving smoothness of the transition and without undesired inflation. In Figures 16, 18, 19, and 20 we also demonstrate that our method can handle in-place morphing between initial and target shapes, in most cases producing semantically meaningful in-betweens.

In Figure 20 we present the morphing sequence of the textured curved symbol into the textured cross. The texture applied to the curved symbol can be considered as a texture with big changes in color contrast. As can be seen in this figure, our method produces reasonably good results.

In Figures 16 and 19 the initial shapes consist of disjoint, colored patterns that are then

Figure 17. *Metamorphosis between two textured shapes using the suggested method (*(a) *and* (c)*) and cross-dissolving (*(b) *and* (d)*), where the initial shape* S_1 *is the bud and the target shape* S_2 *is the flower. See the accompanying supplemental video files* (figure17a.mpg [local/web 62.0KB], figure17b.mpg [local/web 738KB], figure17c.mpg [local/web 754KB], and figure17d.mpg [local/web 752KB]).

morphed into a target shape. In these examples, using tiled shapes we imitate the partitioning of textured objects and show that our method produces satisfactory color interpolation results without establishing any correspondences between partitions. To make the segmentation of the textured shape a fully automatic method, establishment of correspondences between the partitions is required, which is a matter for future work. In Table 1 we show that the generation of the frame sequences for these examples is relatively fast, even for full image resolution.

With our method we can achieve interactive frame rates by conducting calculations on scaled images and then scaling them back to full size. This results in a loss of image quality of around 15–20 percent, which can be seen in Figure 14.

As a potential application, we consider an educational game for children with cognitive deficits, including those who are severely disabled. The concept is illustrated in Figure 15, and an animation sequence can be seen in the supplementary video file application_example.mpg [local/web 35.2MB]. We have already started working on such a much-needed game with psychologists.

In Figure 17 we compare the results of using our basic method with automatic control of the coefficients a_0, a_1, and a_2 and smoothing of the half-cylinders with results obtained using the cross-dissolving method. From Figure 17 it follows that even though cross-dissolving is a quick linear interpolation, it does not work for gradually changing shapes or for shapes that are placed distantly from each other. Methods working with user-defined correspondences can produce more intuitive image transformations, but using these methods is beyond the scope of this work. In addition, providing sensible correspondences in the case of radically different

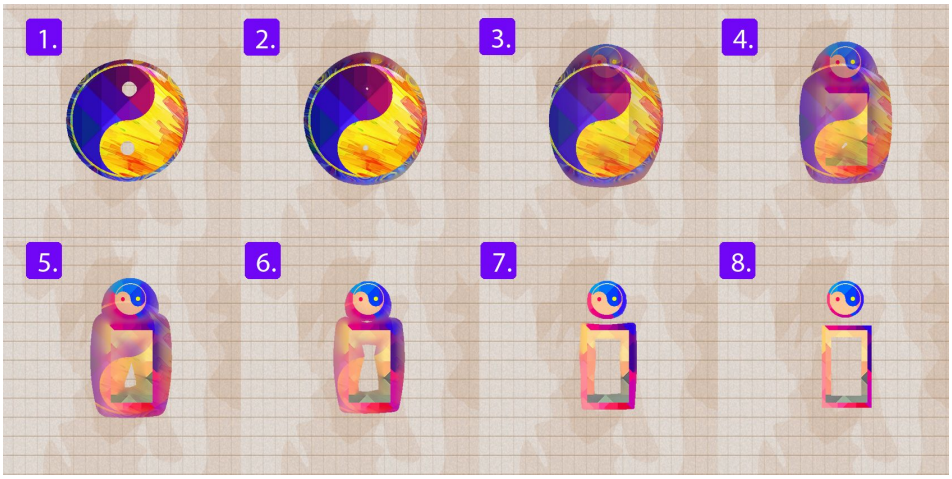

Figure 18. *Metamorphosis between two textured shapes using SDF, STB, and STTI techniques and smoothed cylinders, where the round object is the initial shape S_1 and "i" is the target shape S_2. See the accompanying supplemental video file* (figure18.mpg [local/web 78.0KB]).

object topology is not an easy or obvious task, especially for nonexpert users.

7. Conclusions and future work. In this paper we have presented a novel theoretical and practical approach for dealing with 2D image transformations in a dynamic context, namely, such a popular time-variant image transformation as a metamorphosis. Our method provides a smooth transition between source and target textured objects with different topologies without requirements for shape alignment or establishing any correspondences.

Most other methods do not provide automatic metamorphosis without establishing some form of correspondences between two given objects or their features. In the context of automatically controlled metamorphosis, the most relevant methods with which to make a comparison are the OMT-based methods, which do handle the morphing in terms of both geometry and images—even if the published works do not provide many examples of metamorphosis between textured objects. Our method can produce the in-between frames of reasonable (though not always the highest) quality. It is computationally light and oriented toward nonprofessional users in the context of mainly artistic applications, especially requiring interactive rates, such as games and live streaming.

The proposed basic algorithm relies on a combination of three relatively new techniques borrowed from 3D modeling of heterogeneous objects. These are signed distance functions (SDFs), space-time blending (STB), and space-time transfinite interpolation (STTI). SDF techniques serve as a representation tool for functionally based objects. The STB technique serves to generate a smooth sequence of in-betweens in terms of their geometry, and the STTI method produces visually convincing metamorphosed textures within those intermediate geometric shapes.

Subsequently we have identified a number of drawbacks inherent in our basic algorithm and proposed several mathematically substantiated techniques to address these. These techniques allow the automatic generation of visually smooth transitions between initial and target shapes

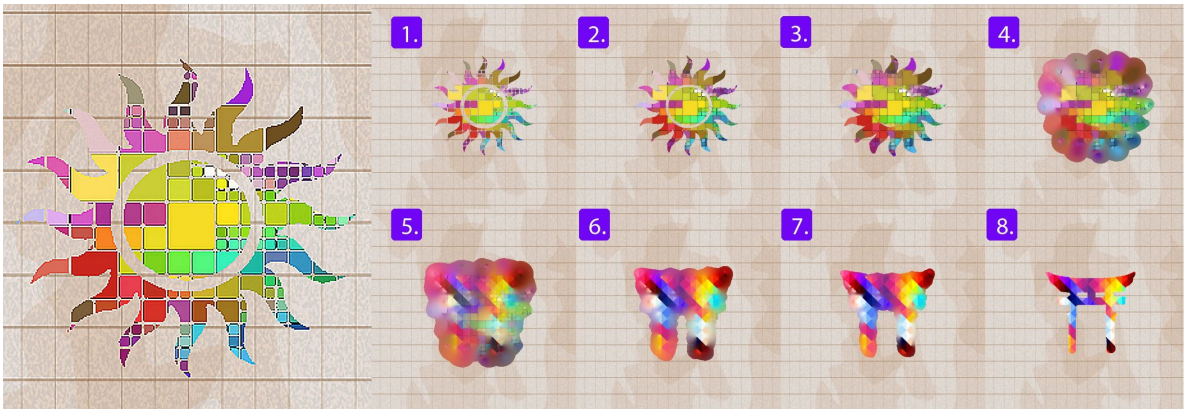

Figure 19. *Metamorphosis between two textured shapes, S_1 (tiled textured sun) and S_2 (textured ark), using SDF, STB, and STTI techniques for the case of in-place morphing. See the accompanying supplemental video file* (figure19.mpg [local/web 82.0KB]).

Figure 20. *Metamorphosis between two textured shapes, S_1 (curved shape with high-frequency texture) and S_2 (textured cross), using SDF, STB, and STTI techniques for the case of in-place morphing. See the accompanying supplemental video file* (figure20.mpg [local/web 90.0KB]).

without "jumps." They also reduce an additional inflation in some intermediate shapes which commonly appears during the metamorphosis process. These new techniques can be applied separately and in combination. Overall, this is the first implementation of the metamorphosis effect using a combination of these methods: converting images to SDFs, subsequently computing STB between their carrying shapes, and applying STTI to their colored textures.

We have also conducted numerous experiments to find the most suitable values for parameters on which various morphing effects depend. Finally, we have implemented a number of representative test examples proving that the approach does work for in-place morphing and spaced images morphing and that it provides interactive frame rates, which is important for many emerging applications.

This work could be expanded in the future by exploring alternative approaches for interpolating attributes between two heterogeneous objects as well as by introducing an automatic feature-based color segmentation to achieve more sophisticated interpolation between textured objects. In order to achieve further improvements, it would be beneficial to focus on other areas that were not touched upon in this work, such as optimization of the color averaging algorithm or the use of a quad-tree or similar data structures for acceleration. It is also possible to implement the functionality in 3D.

Acknowledgments. The authors would like to thank the reviewers for their detailed, constructive, and helpful comments. Special thanks to Dr. Eike Falk Anderson for his comprehensive help in improving the manuscript.

REFERENCES

[1] M. Alexa, D. Cohen-Or, and D. Levin, *As-rigid-as-possible shape interpolation*, in Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '00), ACM Press, Addison-Wesley, 2000, pp. 157–164.

[2] H. Averbuch-Elor, D. Cohen-Or, and J. Kopf, *Smooth image sequences for data-driven morphing*, Comput. Graph. Forum, 35 (2016), pp. 203–213.

[3] T. Beier and S. Neely, *Feature-based image metamorphosis*, SIGGRAPH Comput. Graph., 26 (1992), pp. 35–42.

[4] N. Burtnyk and M. Wein, *Interactive skeleton techniques for enhancing motion dynamics in key frame animation*, Commun. ACM, 19 (1976), pp. 564–569.

[5] Y. Chen, T. T. Georgiou, and A. Tannenbaum, *Vector-valued optimal mass transport*, SIAM J. Appl. Math., 78 (2018), pp. 1682–1696, https://doi.org/10.1137/17M1130897.

[6] M. Cuturi and G. Peyré, *A smoothed dual approach for variational Wasserstein problems*, SIAM J. Imaging Sci., 9 (2016), pp. 320–343, https://doi.org/10.1137/15M1032600.

[7] B. Dalstein, R. Ronfard, and M. van de Panne, *Vector graphics animation with time-varying topology*, ACM Trans. Graph., 34 (2015), pp. 145:1–145:12.

[8] P.-E. Danielsson, *Euclidean distance mapping*, Computer Graphics Image Process., 14 (1980), pp. 227–248.

[9] M. Eisemann, B. De Decker, M. Magnor, P. Bekaert, E. De Aguiar, N. Ahmed, C. Theobalt, and A. Sellent, *Floating textures*, Comput. Graph. Forum, 27 (2008), pp. 409–418.

[10] L. Gao, S.-Y. Chen, Y.-K. Lai, and S. Xia, *Data-driven shape interpolation and morphing editing*, Comput. Graph. Forum, 36 (2017), pp. 19–31.

[11] G. J. Grevera, *Distance Transform Algorithms and Their Implementation and Evaluation*, Springer, New York, 2007, pp. 33–60.

[12] S. Haker, L. Zhu, A. Tannenbaum, and S. Angenent, *Optimal mass transport for registration and warping*, Internat. J. Comput. Vision, 60 (2004), pp. 225–240.

[13] B. Jin and W. Geng, *Correspondence specification learned from master frames for automatic inbetweening*, Multimedia Tools Appl., 74 (2015), pp. 4873–4889.

[14] M. W. Jones, J. A. Baerentzen, and M. Sramek, *3D distance fields: A survey of techniques and applications*, IEEE Trans. Visual. Comput. Graphics, 12 (2006), pp. 581–599.

[15] L. Kantorovich, *On a problem of Monge*, Uspekhi Mat. Nauk, 3 (1948), pp. 225–226.

[16] G. D. Knott, *Interpolating Cubic Splines*, Progr. Comput. Sci. Appl. Logic 18, Birkhäuser Basel, 2000.

[17] X. Y. Kou and S. T. Tan, *Heterogeneous object modeling: A review*, Comput. Aided Des., 39 (2007), pp. 284–301.

[18] S. Lee, G. Wolberg, and S. Y. Shin, *Polymorph: Morphing among multiple images*, IEEE Comput. Graphics Appl., 18 (1998), pp. 58–71.

[19] B. Lévy, *A numerical algorithm for L_2 semi-discrete optimal transport in 3D*, ESAIM Math. Model. Numer. Anal., 49 (2015), pp. 1693–1715.

[20] F. Leymarie and M. Levine, *Fast raster scan distance propagation on the discrete rectangular lattice*, CVGIP: Image Understanding, 55 (1992), pp. 84–94.
[21] J. Liao, R. S. Lima, D. Nehab, H. Hoppe, P. V. Sander, and J. Yu, *Automating image morphing using structural similarity on a halfway domain*, ACM Trans. Graph., 33 (2014), pp. 168:1–168:12.
[22] J. Maas, M. Rumpf, and S. Simon, *Transport based image morphing with intensity modulation*, in Scale Space and Variational Methods in Computer Vision, F. Lauze, Y. Dong, and A. B. Dahl, eds., Springer, Cham, 2017, pp. 563–577.
[23] Y. Makihara and Y. Yagi, *Earth mover's morphing: Topology-free shape morphing using cluster-based EMD flows*, in Computer Vision – ACCV 2010, R. Kimmel, R. Klette, and A. Sugimoto, eds., Springer, Berlin, Heidelberg, 2011, pp. 202–215.
[24] R. Mitton, *Signed Distance Fields*, 2009, http://www.codersnotes.com/notes/signed-distance-fields/.
[25] R. E. Moore, R. B. Kearfott, and M. J. Cloud, *Introduction to Interval Analysis*, SIAM, Philadelphia, 2009, https://doi.org/10.1137/1.9780898717716.
[26] G. Nader and G. Guennebaud, *Instant transport maps on 2D grids*, ACM Trans. Graph., 37 (2018), pp. 249:1–249:13.
[27] A. Neumann, B. Alexander, and F. Neumann, *Evolutionary image transition using random walks*, in Computational Intelligence in Music, Sound, Art and Design, J. Correia, V. Ciesielski, and A. Liapis, eds., Springer, Cham, 2017, pp. 230–245.
[28] A. Pasko and V. Adzhiev, *Function-based shape modeling: Mathematical framework and specialized language*, in Automated Deduction in Geometry, F. Winkler, ed., Springer, Berlin, Heidelberg, 2004, pp. 132–160.
[29] A. Pasko, V. Adzhiev, and P. Comninos, eds., *Heterogeneous Objects Modelling and Applications: Collection of Papers on Foundations and Practice*, Springer-Verlag, 2008.
[30] G. I. Pasko, A. A. Pasko, and T. L. Kunii, *Bounded blending for function-based shape modeling*, IEEE Comput. Graphics Appl., 25 (2005), pp. 36–45.
[31] W. Regli, J. Rossignac, V. Shapiro, and V. Srinivasan, *The new frontiers in computational modeling of material structures*, Computer-Aided Design, 77 (2016), pp. 73–85.
[32] T. u. Rehman, G. Pryor, and A. Tannenbaum, *Fast multigrid optimal mass transport for image registration and morphing*, in Proceedings of the British Machine Vision Conference, N. M. Rajpoot and A. H. Bhalerao, BMVA Press, 2007, pp. 11.1–11.10.
[33] V. Rvachev, *Methods of Logic Algebra in Mathematical Physics*, Naukova Dumka, Kiev, Ukraine, 1973 (in Russian).
[34] V. Rvachev, T. Sheiko, V. Shapiro, and I. Tsukanov, *Transfinite interpolation over implicitly defined sets*, Comput. Aided Geom. Design, 18 (2001), pp. 195–220.
[35] M. Sanchez, O. Fryazinov, V. Adzhiev, P. Comninos, and A. Pasko, *Space-time transfinite interpolation of volumetric material properties*, IEEE Trans. Visual. Comput. Graph., 21 (2015), pp. 278–288.
[36] R. M. Smelik, T. Tutenel, R. Bidarra, and B. Benes, *A survey on procedural modelling for virtual worlds*, Comput. Graph. Forum, 33 (2014), pp. 31–50.
[37] A. R. Smith, *Planar 2-pass texture mapping and warping*, SIGGRAPH Comput. Graph., 21 (1987), pp. 263–272.
[38] J. Solomon, F. de Goes, G. Peyré, M. Cuturi, A. Butscher, A. Nguyen, T. Du, and L. Guibas, *Convolutional Wasserstein distances: Efficient optimal transportation on geometric domains*, ACM Trans. Graph., 34 (2015), pp. 66:1–66:11.
[39] G. Wolberg, *Image morphing: A survey*, The Visual Computer, 14 (1998), pp. 360–372.
[40] L. Zhu, Y. Yang, S. Haker, and A. Tannenbaum, *An image morphing technique based on optimal mass preserving mapping*, Trans. Image Process., 16 (2007), pp. 1481–1495.