Image Deformation (Feature-Based Image Metamorphosis) – Final Report, Computer Graphics Project

SANIDHYA SINGAL (2015085) and HARSHPREET SINGH (2015038), Indraprastha Institute of Information Technology Delhi

We present the final project report for group 10. In this project, we implement the technique of image metamorphosis, as presented originally by Beier-Neely in 1992. The technique allows for smooth transformation between two or more photographed or computer generated subjects, in way that is better than other conventional techniques. We examine the advantages and disadvantages of this technique, and draw the results and conclusions. We support them with several examples.

Additional Key Words and Phrases: Computer Graphics, Computer Animation, Shape Transformation, Image Processing, Linear Interpolation, Image Morphing, Image Warping, Image Cross-dissolve

1 INTRODUCTION

Beier-Neely [1] proposed a new technique for the metamorphosis of one digital image into another. It is a combination of two famous image transforming techniques, *viz.*, image warping and cross-dissolve. The authors refer to this as image *morphing*. In other words, the morph process consists of warping two images so that they have the same 'shape', and then cross-dissolving the resulting images.

1.1 Related Work

Feature-based metamorphosis between two or more images over time is a modern visual technique. Traditional techniques similar to this include optical cross-dissolve. In this technique, one image is faded out while another is simultaneously faded in, in a clever fashion. The subject gets progressively transformed and photographed one frame at a time. Also known by the name of stop-motion animation [1], this technique can give a powerful illusion of continuous metamorphosis, but it requires much skill and is very tedious work.

Another technique called go-motion [1], in which the frame-by-frame subjects are photographed while moving, can also achieve image metamorphosis, but it is very complex and the skills required are even higher. Apart from these, 3D and 2D Computer Graphics techniques are also deployed for the same [1].

1.2 Key Definitions

1.2.1 Image Cross-dissolve. In cross-dissolve between two images, with time, we fade out the first image and fade in the second image. The intermediate image is a weighted average of the two images. For a feature point A in image 1 and a feature point B in image 2, cross-dissolve results in a new feature point F, given by:

$$F = \alpha \cdot A + (1 - \alpha) \cdot B$$

where $\alpha \in [0, 1]$

1.2.2 Image Warping. Warping determines the way in which the pixels in one image should be mapped to the pixels in the other image. It transforms one image into another through rotation, scaling and translation. Warping between two images can be done in two ways. The authors use *reverse* image warping. In this technique, we go through every pixel in the destination image and sample the correct pixel from the source image.

1.2.3 Field Morphing. Field Morphing (or simply, morphing) refers to the combination of generalized image warping with a cross-dissolve between the image elements. In order to morph between two images, we define

corresponding control pixels in source image I0 and destination image I1. We, then, define each intermediate frame I of the metamorphosis by creating a new set of control pixels, by interpolating the control pixels from their positions in I0 to the positions in I1. Both images I0 and I1 are then warped toward the position of the control pixels in I. These two warped images are cross-dissolved throughout the metamorphosis [2].

2 MATHEMATICS OF FIELD MORPHING

We make use of two algorithms: Transformation with one pair of lines and Transformation with multiple pairs of lines. In the interim report, we implemented the former one as shown below [1]. In this report, we implement the latter algorithm.

Algorithm 1 (Transformation with One Pair of Lines)

Consider a pair of lines mapped to each other, one in source image, say $P'Q'$, and other in destination image, say PQ.

(1) for each pixel X in destination image
(2) compute u, v given the line PQ as follows:
(3) $$u = \frac{(X-P)\cdot(Q-P)}{||Q-P||^2}$$
(4) $$v = \frac{(X-P)\cdot Perpendicular(Q-P)}{||Q-P||}$$
(5) compute X' using u, v on the line $P'Q'$:
(6) $$X' = P' + u\cdot(Q'-P') + \frac{v\cdot Perpendicular(Q'-P')}{||Q'-P'||}$$
(7) $$destinationImage(X) = SourceImage(X')$$

The algorithm transforms each pixel coordinate by rotation, translation, and/or scale, thereby transforming the whole image. The scaling is not uniform, but is along the length of the line. Also, shears are not possible to specify. All the transformations based on single line pair are affine.

2.1 Transformation with Multiple Pair of Lines

Multiple pairs of lines specify more complex transformations, as compared to Single pair of lines. Almost any pair of lines result in a non-affine transformation. Still uniform scaling and shearing are not possible [1].

Algorithm 2 (Transformation with Multiple Pair of Lines)

(1) for each pixel X in destination image
(2) $DSUM = (0,0)$
(3) $weightsum = 0$
(4) For each line P_iQ_i in destination image
(5) calculate u, v based on P_iQ_i
(6) calculate X_i' based on u, v and $P_i'Q_i'$
(7) calculate displacement $D_i = X_i' - X$ for this line
(8) $dist$ = shortest distance from X to P_iQ_i
(9) $$weight = \left[\frac{length^p}{(a+dist)}\right]^b$$
(10) $DSUM += D_i * weight$

(11) $$weightsum+ = weight$$

(12) $$X' = X + \frac{DSUM}{weightsum}$$

(13) $$destinationImage(X) = sourceImage(X')$$

The algorithm takes a pair of images – source and destination images. We try to transform source image to destination image.

For any pixel X in the destination image, we find the corresponding pixel in the source image. We do so by specifying a pair of lines, one in each image. For a line P_iQ_i in the destination image, we have a corresponding line $P'_iQ'_i$ in the source image. For P_iQ_i, we find the values for u and v (Line (5)), using Algorithm 1. These are the distance of pixel X from the line P_iQ_i as measured along the line and perpendicular to the line, respectively. We use these values of u and v to find the pixel value X'_i in the source image, as measured from line $P'_iQ'_i$ (Line (6)).

The displacement D_i (Line (7)) is the difference between the pixel location in the source and destination image, and a weighted average of the displacement is calculated (Line (10)).

$dist$ is the shortest distance from X to P_iQ_i (Line (8)). It is calculated as: $dist = \begin{cases} ||v|| & \text{if } 0 < u < 1 \\ ||X - P|| & \text{if } u \leq 0 \\ ||X - Q|| & \text{if } u \geq 1 \end{cases}$

In Line (9), $length$ refers to the length of line P_iQ_i. a, b and p are constants and are empirically chosen. We choose $a = 1$, $b = 2$ and $p = 0$. These can be modified to change the relative effect of the lines.

According to the authors of [1], if a is barely greater than zero, then if the distance from the line to the pixel is zero, the strength is nearly infinite. With this value for a, the user knows that pixels on the line will go exactly where he wants them. Values larger than that will yield a more smooth warping, but with less precise control. The variable b determines how the relative strength of different lines falls off with distance. If it is large, then every pixel will be affected only by the line nearest it. If b is zero, then each pixel will be affected by all lines equally. Values of b in the range $[0.5, 2]$ are the most useful. The value of p is typically in the range $[0, 1]$; if it is zero, then all lines have the same weight, if it is one, then longer lines have a greater relative weight than shorter lines.

The displacements D_i's are calculated to each line in the destination image, and a weighted averaging is done, which results in the pixel value X' (Line (12)). The process is repeated for each pixel X in the destination image.

Animation

As per the authors of [1], a morph operation blends between two images, I0 and I1. To do this, we define corresponding lines in I0 and I1. Each intermediate frame I of the metamorphosis is defined by creating a new set of line segments by interpolating the lines from their positions in I0 to the positions in I1. Both images I0 and I1 are distorted toward the position of the lines in I. These two resulting images are cross- dissolved throughout the metamorphosis, so that at the beginning, the image is completely I0 (undistorted because we have not yet begun to interpolate away from the line positions associated with I0). Halfway through the metamorphosis it is halfway between I0 and I1, and finally at the end it is completely I1. Note that there is a chance that in some of the intermediate frames, two lines may cross even if they did not cross in the source images.

Linear Interpolation

Out of the two interpolating methods used by Beier-Neely, we use only one. A pair of lines are linearly interpolated by interpolating their end-points. Although it can result in shrinking of the length of a line during rotation, it is not that big a problem.

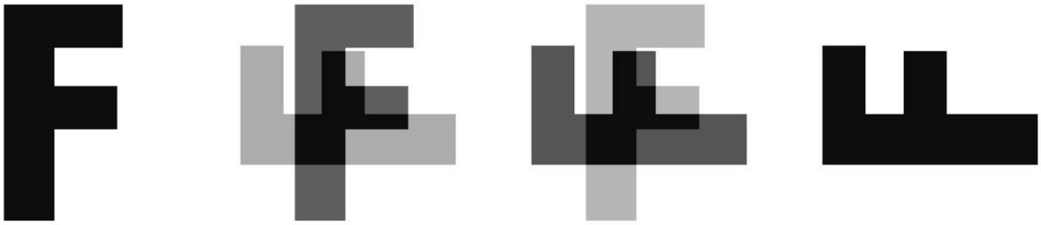
Suppose we have a pair of lines – PQ and $P'Q'$. We find n lines in between these two lines. For ith line, P_i is found by linear interpolation between P and P', and Q_i by linear interpolation between Q and Q', for $i = 1, 2, \ldots, n$. Then, P_iQ_is are the required lines.

3 RESULTS AND OBSERVATIONS

In the interim project report, we had shown the results obtained by the techniques of image cross-dissolve and image warping (transformation with a single pair of lines). These are summarized below:

3.1 Image Cross-dissolve

Following images indicate the process of cross-dissolve between two images:



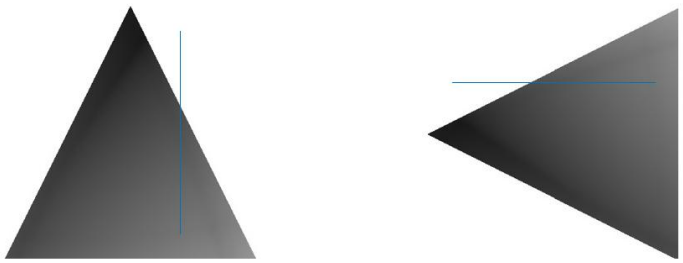
Source Image | Intermediate Image 1 | Intermediate Image 2 | Destination Image

Fig. 1. Cross-Dissolve between two images

3.2 Image Warping (Transformation with Single Pair of Lines)

3.2.1 Image has only Binary Colour Values. Following are two sets of images, indicating the process of image warping:

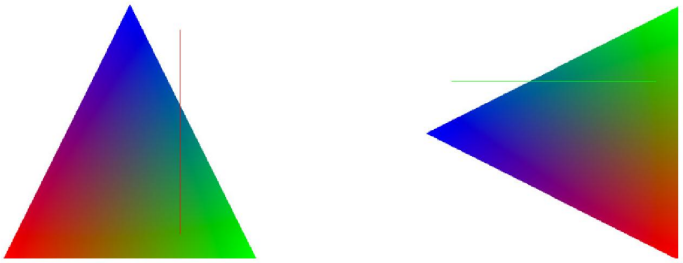
Source Image (a) | Destination Image (a)



Source Image (b) Destination Image (b)

Fig. 2. Image Warping in two sets of Images

3.2.2 RGB Image. Following images indicate the process of image warping:



Source Image Destination Image

Fig. 3. Image Warping in RGB Images

3.3 Transformation with Multiple Pair of Lines

3.3.1 Transformation with Two Pairs of Lines. We begin with transforming source image into destination image (not known to us), by specifying two lines in source image and corresponding lines we wish to see in the destination image. The destination image is built from the source image. Following are results obtained:

Source Image Destination Image

Fig. 4. Transformation with Two Pairs of Lines

3.3.2 Linear Interpolation. Here, we have two pairs of lines, one each for source and destination images. We now linearly interpolate between each pair of lines, as described earlier to get 10 intermediate line pairs. These are shown below:

Source Image Intermediate Image 1 Intermediate Image 2 Intermediate Image 3

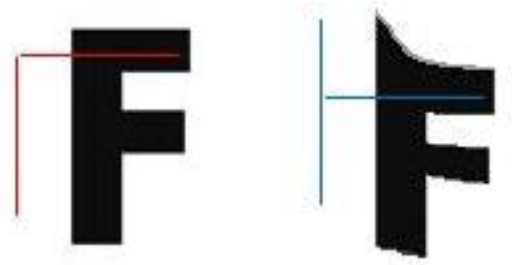
Intermediate Image 4 Intermediate Image 5 Intermediate Image 6 Intermediate Image 7

Intermediate Image 8 Destination Image

Fig. 5. Linear Interpolation on Transformation with Two Pairs of Lines

3.4 Issues with Feature-Based Image Metamorphosis

3.4.1 Ghosting. Between the lines, sometimes unexpected interpolations are generated. The algorithm tries to guess what should happen far away from the line segments; sometimes it makes a mistake. This problem usually manifests itself as a "ghost" of a part of the image showing up in some unrelated part of the interpolated image, caused by some unforeseen combination of the specified line segments [1].



Source Image Destination Image

Fig. 6. Issue of Ghosting

3.4.2 Cross Over. If the value a is set to zero there is an undefined result if two lines cross. Each line will have an infinite weight at the intersection point.

Source Image Destination Image

Fig. 7. Cross-over when $a = 1$

3.5 Animation

Following images depict the animation:

Source Image

Intermediate Image 1

Intermediate Image 2

Intermediate Image 3

Intermediate Image 4

Intermediate Image 5

Intermediate Image 6

Intermediate Image 7

Intermediate Image 8 Intermediate Image 9 Destination Image

Fig. 8. Animated Sequence of Images

4 DISCUSSIONS

4.1 Advantages

The new method of feature-based image metamorphosis provides a better transformation from one image to another as compared with other conventional methods like stop-motion animation and go-motion animation. It is also preferred over 2D and 3D Computer Graphics techniques.

According to Beier-Neely [1], this technique has one big advantage over the mesh warping technique: it is much more expressive. The only positions that are used in the algorithm are ones the animator explicitly created. For example, when morphing two faces, the animator might draw line segments down the middle of the nose, across the eyes, along the eyebrows, down the edges of the cheeks, and along the hairline. Everything that is specified is moved exactly as the animator wants them moved, and everything else is blended smoothly based on those positions. Adding new line segments increases control in that area without affecting things too much everywhere else.

4.2 Disadvantages

The algorithm has several disadvantages. The biggest of them are slow speed and global control. Because it is global, all line segments need to be referenced for every pixel. Other disadvantages include the issue of ghosting and cross-over as shown earlier. Ghosting is caused when the algorithm tries to guess what should happen far away from the line segments, and it makes a mistake. Cross-over problem occurs when $a = 0$, due to which, there is an undefined result if two lines cross. Each line will have an infinite weight at the intersection point.

5 FUTURE WORK

Feature-based image metamorphosis is indeed very slow. So, better techniques can be invented to increase the speed. Also, we can have better algorithms with local control and not global control, as is the case with this algorithm. *Ghosting* and problems in linear interpolation can also be removed for improved future works.

6 CONCLUSION

Feature-based Image Metamorphosis is a useful visual animation tool that transforms one image to another in a smooth fashion. The animation can generate an image of a subject that looks like a real subject, but is actually a combination of the subjects in the source and destination images. Metamorphosis can also be done among more than two images to generate an animated video. This is quite dramatic when compared with metamorphosis between only two images. Furthermore, we might use moving images instead of still ones, as used in Michael Jackson's video – *Black or White*.

7 EVALUATION OF SUBMISSION

7.1 Milestones Covered

Proposed Milestones:

(1) Understand the techniques of image warping and cross-dissolve among image elements.
(2) Implement field morphing techniques provided in the text to achieve feature-based image metamorphosis.
   (a) Implement algorithm for transformation with one pair of lines.
   (b) Implement algorithm for transformation with multiple pair of lines.
   (c) Achieve morphing among more than two images (as shown in the video), if possible. (BONUS)
(3) Discuss advantages, disadvantages and results of the technique, and compare with those given in the text.

(1) and (2a) were achieved in the interim project submission. We have completed (2b) and (3) in our final project submission. The BONUS part (2c) has been done as well.

7.2 Individual Contribution

(1) Sanidhya Singal (2015085): Field Morphing Algorithms for Transformation with Multiple Pair of Lines and Single Pair of Lines (Image Warping); Algorithm for Animation Sequence; Literature Survey
(2) Harshpreet Singh (2015038): Algorithm for Cross-Dissolve between two images; Debugging, Evaluation and Performance of Code; Algorithm for Linear Interpolation; Literature Survey

Note: No third party libraries or codes have been used in the project.

7.3 Files

Image Cross-dissolve. `crossdissolve.m` and `intermediate.m`

Image Warping (Transformation with Single Pair of Lines). `imageMorphing.m`, `calc_U.m`, `calc_V.m`, `calc_Xd.m` and `singleLineMorph.m`

Transformation with Multiple Pair of Lines. `imageMorphing2.m`, `calc_U.m`, `calc_V.m`, `calc_Xd.m`, `crossDissolve.m`, `findDist.m`, `getControlPoints.m`, `interpLines.m` and `multiLineMorph.m`

Please refer to README.txt for instructions on running the code and viewing the outputs.

REFERENCES

[1] Thaddeus Beier and Shawn Neely. 1992. Feature-based image metamorphosis. SIGGRAPH Comput. Graph. 26, 2 (July 1992), 35-42. DOI: https://doi.org/10.1145/142920.134003 https://www.cs.princeton.edu/courses/archive/fall00/cs426/papers/beier92.pdf
[2] http://wikieducator.org/Image_Morphing