

A new approach to rapid image morphing for lip motion synthesis

Anna Buttfeld

Department of Computer Science & Software Engineering,
The University of Western Australia,
35 Stirling Highway, Crawley, W.A. 6009
email: buttfa02@csse.uwa.edu.au

Abstract

Animating a human face with visual speech in a way that is accurate enough to look natural and aid both conscious and subconscious lipreading is a major goal of animation systems. One method of creating such animations is to collect a set of images of a human subject speaking, then combine these images together in a manner simulating natural speech. Image morphing techniques can be used to create transitions between the static images. A range of morphing techniques have been used in this context, covering traditional techniques and methods that reflect the specific problem domain. In this paper I present a morphing technique that combines the image warping techniques of traditional field morphing with a rapid morphing method developed specifically for facial animation. This new technique avoids complex computation and difficulties involved in optical flow calculation as is used in other methods, and instead uses simple geometrical correspondence to create smooth transitions. This allows for simplicity and flexibility in calculation while preserving the speed of image generation. The technique assumes that the motion of pixels between the two images can be approximated by a vector field; this vector field can then be calculated and stored. New intermediate images can then be generated rapidly by morphing the images along the calculated vector flows.

Keywords: Morphing, Image Metamorphosis, Lip motion synthesis, Facial animation

CR Classification:I.2.7, I.4.8, I.5.4, I.6.8

1 Introduction

Image morphing, or the creation of a smooth animated transition between two different images, is a field of study that is the subject of ongoing research. Morphing has many and varied applications; for example, high quality, computationally expensive morphing is used as a visual effect in multimedia applications, while faster, less precise morphing is used in real-time interactive systems.

An example of the application of morphing is in generating synthetic visual speech. Lip motion synthesis can be achieved by selecting a set of images that cover the visual range of human speech, then using morphing techniques create animated visual speech.

Copyright ©2003, Australian Computer Society, Inc. This paper appeared at Twenty-Sixth Australasian Computer Science Conference (ACSC2003), Adelaide, Australia. Conferences in Research and Practice in Information Technology, Vol. 16. Editor, Michael Oudshoorn. Reproduction for academic, not-for profit purposes permitted provided this text is included.

This paper presents a new efficient image morphing algorithm for visual speech synthesis. The proposed approach improves the existing geometrical morphing technique by employing an interpolation scheme to generate animation. Faster generation of images is achieved by calculating the full geometrical morph between the initial and final frames of the animation, and creating intermediate images by interpolating this morph. This approach is similar to that developed by Ezzat and Poggio (Ezzat & Poggio 1998), who use the optical flow between images to find the total overall morph, and then interpolate this flow to generate the intermediate images. The approach proposed in this paper adapts the image warping algorithm developed by Beier and Neely to use the rapid morphing method described by Ezzat and Poggio.

2 Image morphing

The problem of image morphing is this: how do we create a transition between two images, possibly quite different, so that the transition appears smooth and the motion fluid? Simply fading out the first image while fading in the second produces, in most cases, unsatisfactory results. For the resultant image sequence to appear smooth, there must be some kind of warping of the images so that the initial image appears to transform into the final over time. This is combined with the process of blending the images together to achieve smooth transition of colour and features that weren't specified explicitly.

Thus, every image in a morphed sequence is generated by warping the two end images toward the intermediate configuration and blending these images together to produce a final image. The weighting given to each image when blending them together is adjusted over the course of the sequence; in the first images of the sequence, the first image will have the greatest weight, at the halfway point the two images will have equal weight, and toward the end the final image is given the greatest weight. In this way the first image gradually transforms into the second, with specified features moving smoothly, and the texture of the initial image fading out as the final image fades in.

Applying image morphing to lip motion synthesis introduces a new set of constraints. In a lip motion synthesis system, the morphing will be concerned with a set of images of visemes (visual phoneme images), typically fewer than twenty, which form the basis of the animation. The images are all of the same subject, taken from the same position; the only difference between the images is the appearance of the mouth region. This means that between two images there is only a small area where new texture appears or is hidden. The limited set of images means that the possible combinations of image transitions are lim-

ited, although the number of intermediate frames required will vary with each sequence. Dividing the work of morphing between a pre-processing stage and an image generation stage is thus both feasible and desirable, and allows for faster run-time generation of images while retaining flexibility in the number of images in the sequence.

2.1 Existing geometrical image warping method

Beier and Neely (Beier & Neely 1992) expounded a method of morphing that they called “field morphing”, which morphs images according to the movement of defined control lines. Significant features in the first image are identified by line segments, as are the corresponding features in the second image. Over the course of the animation sequence, the control lines are moved from their position in the first image to the position in the second image, and the image is warped along with the movement of these lines. The image warping is a geometrical process; the position of every pixel with respect to every line is calculated, and from this the position of the pixel with respect to the shifted control line can be determined. The combined influence of the different control lines over the pixels is determined by a weighting function that specifies how the influence of the control lines decreases with distance.

This method of generating warped images and thus animated sequences is computationally expensive, but is flexible and precise in its application. The expensive arises from the fact that the influence of every control line over every pixel must be calculated and weighted, which means that the time required for the morph increases linearly with the number of control lines. However, this method has the benefit of being simple and intuitive, with precise control over feature specification and a number of ways of modifying the weighting function to adjust the appearance of the morph.

2.2 Optical flow morphing

Different research teams have used morphing to create lip motion synthesis and facial animation systems. The MikeTalk system, developed by Ezzat and Poggio (Ezzat & Poggio 1998), creates morphed image sequences by calculating the flow of pixels between images by using an optical flow technique. Optical flow, originally developed by Horn and Schunck (Horn & Schunck 1981), calculates the change in position of every pixel in the image and stores this as an array of 2-dimensional vectors. However, optical flow is computationally expensive, and most algorithms rely on the change in position of pixels between images being small. Ezzat and Poggio report that this condition was not met for many of the considered image pairs. They circumvented this limitation by calculating the flow via intermediate images, then concatenated these flows to obtain the overall flow. This means that more original images are required beyond those that are the initial and final images in the morphed sequence.

Once the overall pixel flow between the two images has been calculated, the images are warped to the intermediate position by moving the pixels along their flow vectors. A sequence of intermediate images is created in the manner described earlier; warping the two end images toward the intermediate image, then blending the warped images together.

This method is effective, and has been used subse-

quently by other teams developing facial animation applications. For example, Faruque et al. (Faruque, Kapoor, Kate, Rajput & Subramaniam 2001) created a model that includes facial expression animation in addition to lip motion synthesis. The basic principle of computing pixel flow between images, then running the animation rapidly at run time, is a useful model for creating morphed sequences in this context.

3 The proposed approach

The proposed approach uses the process shown in Figure 1 to create a sequence of morphed images.

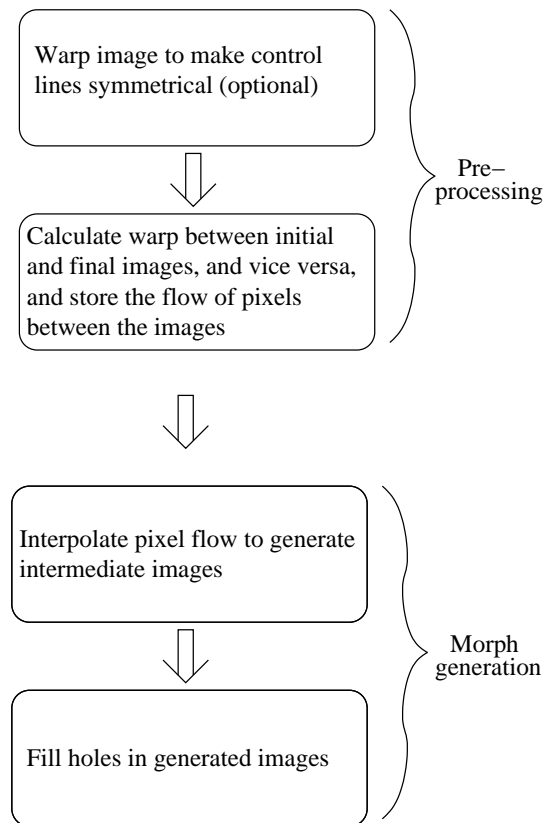


Figure 1: The image morphing process, under the proposed approach.

Image normalisation

One major advantage of using the Beier-Neely method of defining image warps is that the symmetry of images can be exploited to reduce computation time and data storage. If the control lines in an image are symmetrical, the vector field defining the flow of pixels will also be symmetrical. In this case, the data needed to generate images is halved, and the computation time significantly reduced. If the images are close to symmetrical, they can be normalised with the warping algorithm so that the lines are exactly symmetrical prior to applying the morphing algorithm. This is particularly applicable to facial images, which are usually close to symmetrical but are rendered slightly asymmetrical by facial expression or head orientation. An example of a face image that has been normalised to symmetrical control lines is shown in Figure 2.

Pixel flow calculation

The flow of pixels between the initial and final images, and from the final image to the initial image, is calculated as a pre-processing step and the resultant



Figure 2: Normalising an image to symmetrical control lines.

vector field is used at run time to generate new, intermediate images. This pixel flow is calculated by defining corresponding control lines in the initial and final images, calculating the warp of the initial image control lines to the final image control lines and vice versa. The warp is calculated with the algorithm devised by Beier and Neely (Beier & Neely 1992), the details of which are given later in this paper.

Image generation by pixel flow interpolation

Once the images have been normalised and the flow of pixels between them calculated, new intermediate images can be generated. For each frame of the animation two warped images are generated, a warp from the initial image and a warp from the final image. These images are created by moving every pixel in the image some distance along the flow vectors previously calculated. These images are then blended together to create the animation frame.

Hole filling

In contrast to the morphing algorithm developed by Beier and Neely, not every pixel in the warped images is guaranteed to be calculated. In parts of the image where there is local expansion, holes will appear in the image where no pixels have been set. A hole filling algorithm is required to fill these gaps. This is done by interpolating the values of pixels on either side of the gap across the missing pixels.

3.1 Image warping

The method employed to warp the final image from the initial, and the final to the initial, is the main distinguishing feature in morphing methods. The method I have used is based on the feature correspondence scheme developed by Beier and Neely; it allows feature points in the image to be specified and manipulated with ease, and the image warp to be calculated in a geometric fashion.

To warp an image with this algorithm, important features in the image must be identified with control lines. The desired position of these lines in the final warped image must also be determined. The image is then warped so that the control lines in the original image align with the final control lines.

Transforming an image with one control line

The most simple case of image warping is to warp an image with respect to a single control line. In this situation the warping is geometric, involving scaling, translating, and rotating. The transformation between the original image and the warped image for each pixel can be determined by calculating the position of the pixel with respect to the control line in the warped image, then finding the point in the original image which is in the same relative position with respect to the original control line. Performing the calculation in this direction, working through every pixel in the warped image to determine the pixel in the original image that it corresponds to, ensures that every pixel in the warped image is calculated once and only once.

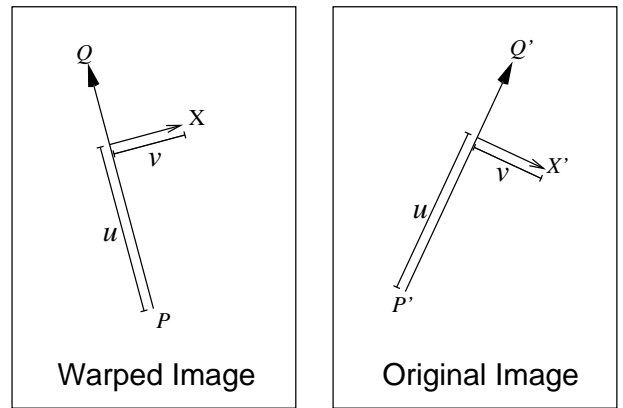


Figure 3: The transformation of point X when a single control line, PQ , defines the transformation between images. As the line PQ moves to $P'Q'$, X moves to X' , maintaining its position relative to PQ .

$$u = \frac{(X - P) \cdot (Q - P)}{\|Q - P\|^2}, \quad (1)$$

$$v = \frac{(X - P) \cdot ((Q - P)^\perp)}{\|Q - P\|}, \text{ and} \quad (2)$$

$$X' = P' + u \cdot (Q' - P') + \frac{v \cdot ((Q' - P')^\perp)}{\|Q' - P'\|} \quad (3)$$

The position of a pixel with respect to a line can be defined uniquely by two variables; in this case we use the following variables:

- The position of the pixel along the line, u , shown in Equation 1. This is a scalar value and represents the fractional distance of the pixel projected onto the control line. The value for this will be 0 and 1 at the beginning and end of the line respectively, $u < 0$ before the beginning of the line and $u > 0$ after the end of the line, and $0 < u < 1$ on the line.
- The perpendicular distance from the line to the pixel, v , shown in Equation 2. This is measured in pixels, so the distance from the line to the pixel is constant in the original and warped images.

These two measures are calculated for the pixel in the warped image, then applied to the control line in the original image to determine which pixel in the original image corresponds to the pixel in the warped image. This is shown in Equation 3. In this equation, *Perpendicular* refers to a vector orthogonal to, and the same length as, the input vector; this can be either the left-hand or right-hand orthogonal vector as long as one is used consistently.

Combining the influence of multiple line segments

More complex warps are constructed by identifying multiple control lines in an image. The warp is then calculated by applying the transformation for each control line on each pixel, then weighting the influence of each control line according to the distance of the pixel from the control line. The equation proposed by Beier and Neely is given as Equation 4, where *dist* is the shortest distance from the pixel to the line.

$$weight = \left[\frac{1}{a + dist} \right]^b \quad (4)$$

The weighting equation contains several variables that change the interaction between control lines.

- The variable *a* controls the influence of control lines over pixels that are very close to the line. If *a* is close to zero, pixels near the a control line will be effected only by that control line, giving precise control. If *a* is larger this precise control is lost but the overall warp will be more smooth.
- Variable *b* controls strength of the control as the distance increases; usually in the range $0.5 < b < 2$, the larger *b* is the more the pixel is only effected by the line closest to it.

3.2 Image normalisation and smoothing

An issue that arises when using this image warping algorithm for normalising images is caused by integer resampling. This occurs when the image is being expanded and more than one pixel in the warped image maps to the same pixel in the original image. On diagonal lines this is apparent as a “staircasing” effect. The effect does not appear when using image warping as a component of the morphing process, since the pixel flow interpolation only assigns one vector in the original image to a single pixel in the warped image, with the holes being filled after the image is generated. This effect can be eliminated by not mapping every pixel in the warped image to a pixel in the original image, but interpolating between pixels in the original image to achieve a smoother image. An example of this process on a simple shape is shown in Figure 5. This operation is more complex than simply rounding to the nearest pixel, so if the effect is not apparent or the quality of the final image is not critical, this process adds unnecessary complexity. However, in cases where the extra computation is warranted the result is far superior.

3.3 Pixel flow calculation

The procedure for calculating the flow of pixels between the initial images to the final image is given below. Pixel flow from the final image to the initial image is calculated in the same way.

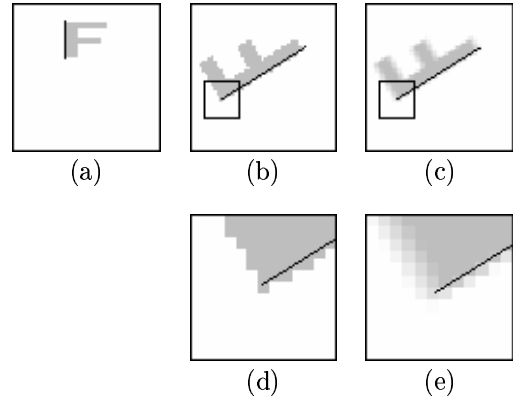


Figure 5: The effect of pixel interpolation on a normalised synthetic image. Image (a) is the original image, a letter F with a single control line situated on its spine as shown. Image (b) is image (a) warped to the new control line without pixel interpolation, while image (c) is the same transformation with interpolation to smooth the image. Image (d) and image (e) are details of image (b) and image (c) respectively. In this case, the interpolated image took approximately 40% longer to process than the uninterpolated image.

For every pixel X in the initial image

```
weightsum = 0
DSUM = [0,0]
```

```
For every line pair, PiQi in the initial image
                    Pi'Qi' in the final image
```

```
calculate u,v based on PiQi
calculate Xi' based on Pi'Qi'
calculate movement Di = Xi' - X
calculate weight based on X and PiQi
weightsum = weightsum + weight
DSUM = DSUM + Di * weight
```

```
forward_pixel_flow(X) = DSUM/weightsum
```

3.4 Generating warped images from pixel flow

Once the pixel flow has been calculated, the intermediate images *forward_warp_image* and *backward_warp_image* are calculated as shown below. The variable *alpha* is the blending parameter, which varies from 0 to 1 over the course of the morph.

```
for every pixel X in initial_image
X' = round(X + (1-alpha) * forward_pixel_flow(X))
forward_warp_image(X') = initial(X)
```

```
for every pixel X in final_image
X' = round(X + alpha * backward_pixel_flow(X))
backward_warp_image(X') = final(X)
```

These two images are then blended together to create the final animation frame, again using the blending parameter *alpha*.

```
blend_image = forward_warp_image * (1 - alpha)
              + backward_warp_image * alpha
```

At its most basic, *alpha* will be linearly increasing from 0 to 1 over the course of the animation, which

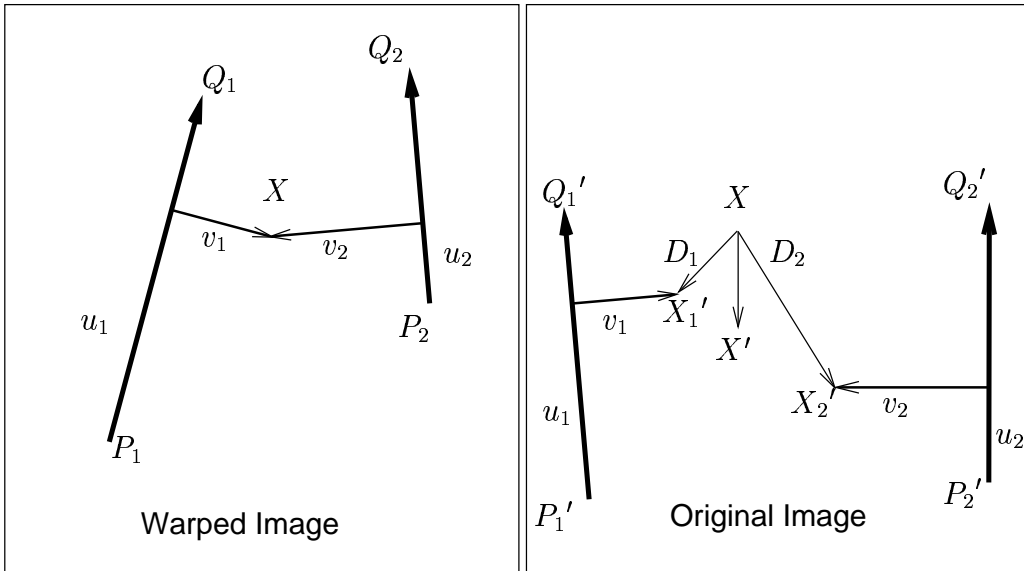


Figure 4: Transformation of point X under the influence of two control lines, P_1Q_1 and P_2Q_2 . Given pixel X in the destination image, the Beier-Neely algorithm calculates the corresponding X' in the source image. X' is calculated in two stages. Firstly it determines X_1' and X_2' , which are the transformations of X under the influence of P_1Q_1 , and P_2Q_2 respectively. Secondly, the final point X' is calculated from the weighted average of the changes in positions between X and X_1' , and between X and X_2' , namely D_1 and D_2 respectively.

gives animation of constant speed. However, alpha can be generated from any function as long as it increases monotonically over the course of the animation. A function that makes the animation proceed more rapidly in the middle of the sequence and more slowly at the beginning and end is more reminiscent of natural motion, as well as having the advantage of moving more quickly through the middle animation frames which are likely to be of the lowest quality. The interpolation I have found most useful is based on a square root function, and is shown in Figure 6.

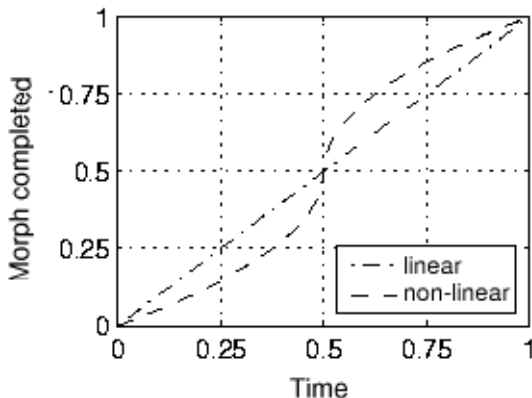


Figure 6: A timing scheme, based on a piecewise square root function, which makes the morph proceed more rapidly at the beginning and end of the sequence.

4 Results

This approach has been tested on many image pairs, including extensive use as part of a lip motion synthesis system. The results produced are encouraging; a sample of the animation of a constructed image is shown in Figure 9, and a lip motion synthesis

sequence in Figure 10. The code for this algorithm was implemented in matlab as a prototype; considerable speedup should be possible by porting to a faster language.

The animation created from the constructed images, Figure 9, shows the details of the animation process and compares the results from the two methods. The forward and backwards warped images are shown in addition to the final blended images. The images produced by the method described in this paper are shown to be equivalent in quality to those produced by the more expensive Beier-Neely process.

The lip motion synthesis example shown in Figure 10 compares the Beier-Neely method and the method described in this paper with actual video images. The pre-processing required in this method is equivalent to generating one image with the Beier-Neely method; after this has been done, generating the images is done rapidly. In contrast, the Beier-Neely method takes the same time to generate every image. In this case, the generation of each Beier-Neely intermediate image and the pre-processing step of the new method took approximately 30 minutes, and the new images generated by vector interpolation took approximately 40 seconds each to generate.

5 Conclusions and future work

This paper presents a morphing algorithm that is effective in creating animation of complex images rapidly. However, there are drawbacks with the method. The process of specifying control lines is time consuming, and adjusting the weight variables to give optimal results takes some practice.

On-going work might include investigating different image warping methods to calculate the vector flow, including testing different methods of defining features. For example, bezier curves might be more appropriate for defining features than line segments.



Figure 7: Detail of intermediate image 2 from Figure 10. Image (a) was produced with the method described in this paper, image (b) is an actual image taken from a video sequence, and the image (c) was produced by morphing with the Beier-Neely method. In this case the method described in this paper produces a result that is visually more similar to the actual image.

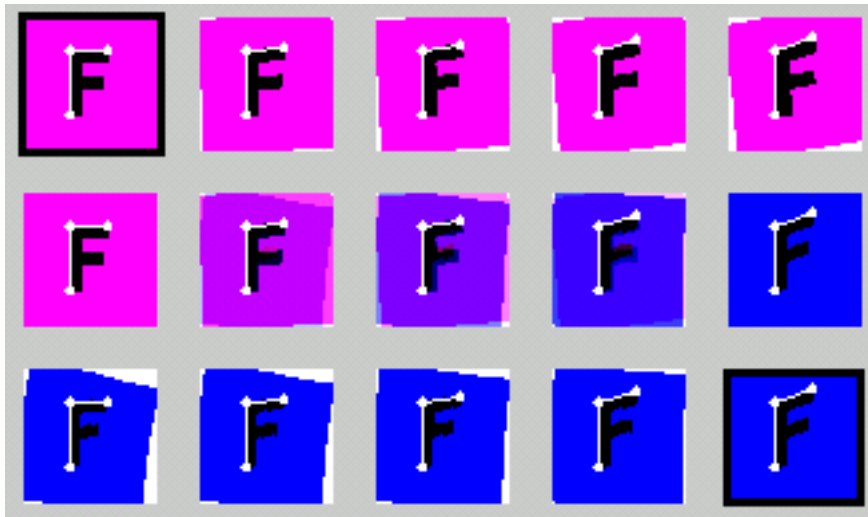


Figure 8: Vertical and skewed letter F morphed with the Beier-Neely method, using two control lines. The top row shows the forward warped images, the bottom row shows the backward warp, and the middle line is the combined morph. The control lines are marked in white and original images bordered in black.

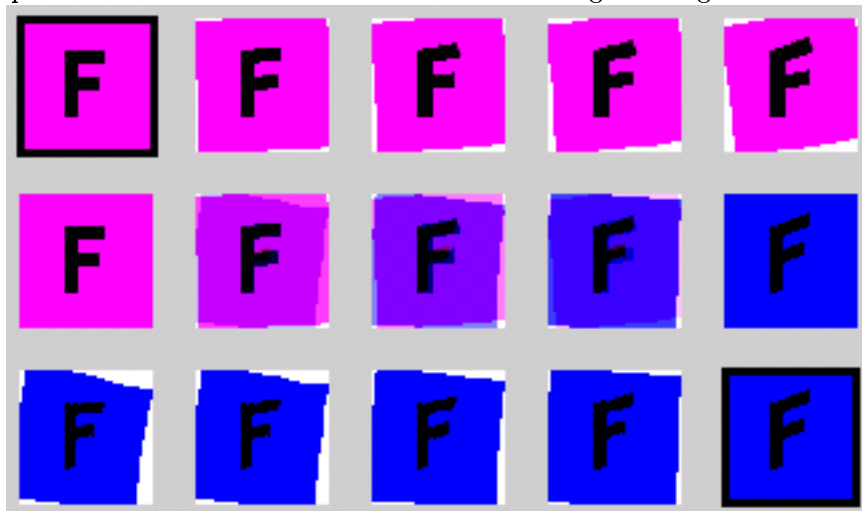


Figure 9: Images generated by morphing with the new method. The image generated are of a similar quality to those generated by the Beier-Neely method, but the computation is far less expensive.

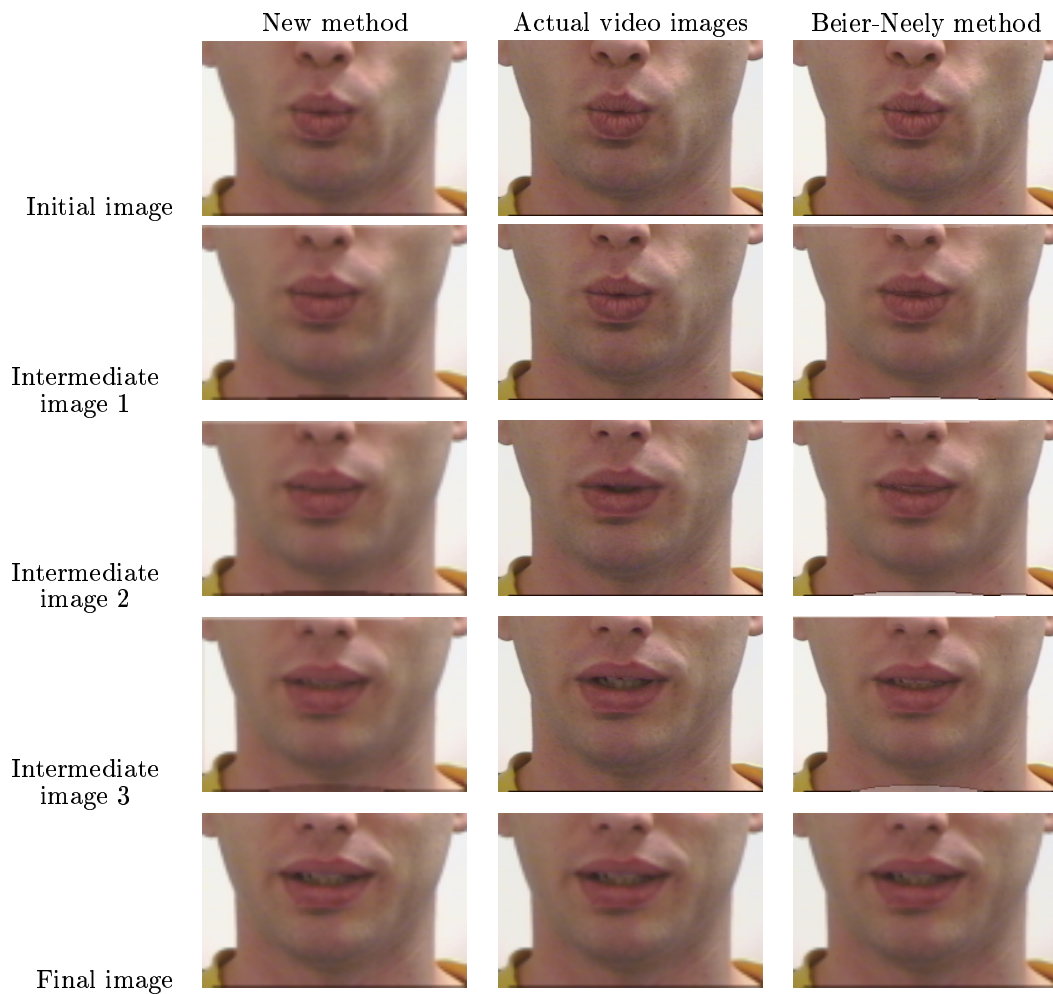


Figure 10: An example of lip motion synthesis morphing. The middle column shows actual images, taken from a video stream of the subject saying “win”. The first column includes two original images, the initial and final images, with three images interpolated using the method described in this paper. The third column also contains two real images and three generated images, but these images were generated using the Beier-Neely algorithm.

References

- Beier, T. & Neely, S. (1992), 'Feature-based image metamorphosis', *Computer Graphics* **26**(2), 35–42. ACM SIGGRAPH. Proceedings of SIGGRAPH '92. Available at: <http://www.hammerhead.com/thad/morph.html>.
- Ezzat, T. & Poggio, T. (1998), Miketalk: A talking facial display based on morphing visemes, *in* 'Proceedings of IEEE computer animation '98', pp. 96–102.
- Faruque, T. A., Kapoor, A., Kate, R., Rajput, N. & Subramaniam, L. V. (2001), Audio driven facial animation for audio-visual reality, *in* 'ICME 2001, Proceedings of IEEE International conference on multimedia and expo'.
- Horn, B. K. P. & Schunck, B. G. (1981), 'Determining optical flow', *Artificial Intelligence* **16**(1–3), 185–203.