August 3, 1994 CS-ETR-94-05

Image Metamorphosis using optical flow techniques

Minas E. Spetsakis

Dept. of Computer Science
York University
4700 Keele Street
North York, ONTARIO
CANADA, M3J 1P3
tel. (416) 736-5053 ext. 77886
fax. (416) 736-5872
e-mail minas@cs.yorku.ca

ABSTRACT

This paper presents an application of optic flow estimation to image metamorphosis. It uses a state of the art optical flow algorithm to do image metamorphosis, which is used in video production. The method presented automates the labor intensive process of image animation. It an advanced optic flow algorithm but the rest of it is simple: most of the code fits in the limited space of this summary.

The support of the NSERC (App. No. OGP0046645) is gratefully acknowledged.

1. Introduction

Image metamorphosis or *morphing* is a very useful technique in computer graphics and animation. It is routinely used commercially for special effects. In a recent paper Beier and Neely [2] described how this is done by professional animators at Pacific Data Images and Wolberg [8] how it is done by their colleagues at Industrial Light and Magic. Despite the differences the two methods consist mainly of the following steps.

* The operator specifies via a GUI the deformation. The result is a sparse description of the deformation.
* The system computes the complete deformation field from the sparse data.
* The system warps the image and interpolates the color.
* The cycle is repeated until the result is good.

The procedure can be applied to still images or on individual frames from movies. Major portion of the time is spent on manually specifying the deformation.

2. Using flow to do morphing

In this paper we present a method to estimate the deformation field using optical flow algorithms developed for vision. These algorithms compute the amount of deformation (flow) between successive images in an image sequence. These algorithms are often considered unstable for the simple reason that when used with structure from motion algorithms the end result is unstable. If one is concerned only with the deformation field then several flow algorithms work fairly well [1] in the sense that the needle maps look intuitively correct. It is also relatively straightforward to take two images from the same sequence and transform one into

another. This technique is used by several video compression schemes like MPEG.

Something similar can be done for morphing. Take the images of two people and compute the deformation between them and then use this to morph one into another. The problem is that although the images are not completely different (e.g. both contain faces) the difference is large enough to confuse most flow algorithms [5, 6]. As opposed to the flow algorithms used for structure from motion, algorithms for estimating the deformation for morphing need not worry that much about the aperture problem (the ambiguity of the deformation field when examining a small patch of an image as viewed through an aperture) or accurate detection of discontinuities. But they should work well for pairs of images where the constant intensity assumption is grossly violated and the displacement is large and varying. We tried several algorithms and the one that worked best uses affine model for flow with Gabor filtering. The Gabor filters are particularly good at capturing the motion of shapes without much regard to slow variations of intensity [4] and the affine model can accommodate the varying flow field within the large support of the Gabor filters.

3. Affine flow algorithm

Assume that we have a series of filters g_q for $q = 1..q_{\max}$. We can try to minimize

$$SSE = \sum_q \left\{ I_x^{(g)} u + I_y^{(g)} v + I_t^{(g)} \right\}^2 \tag{1}$$

where the superscript (g) denotes convolution of the image I with the filter g_q, u and v are the components of the flow and the subscripts denote partial derivatives with respect to x, y and t. This is similar to what is used in [3] which gives very stable results with slow varying flow. But since we use Gabor filters with quite big support, what we do in effect is blend constraints

from a large area of the image and we cannot assume that the flow remains constant over the whole area. We can significantly reduce the effects of the the variation in flow by modeling it with a locally affine expression. In this case we have to minimize

$$SSE = \sum_q \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} (I_{aerr})^2 = \sum_q \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \left| \left(I_x^{(g)} u + I_y^{(g)} v + I_x^{(gx)} u_x + I_y^{(gx)} v_x + I_x^{(gy)} u_y + I_y^{(gy)} v_y + I_t^{(g)} \right) \right|^2 \quad (2)$$

We used absolute values because the filters are complex valued Gabor filters. Notice that some of the convolutions are with $g_q(x, y) \cdot x$ etc. The Euler equations for this minimization are

$$\begin{aligned} I_x^{(g)} I^*_{aerr} - \frac{\partial}{\partial x}\left(I_x^{(gx)} I^*_{aerr} \right) - \frac{\partial}{\partial y}\left(I_x^{(gy)} I^*_{aerr} \right) = 0 \\ I_y^{(g)} I^*_{aerr} - \frac{\partial}{\partial x}\left(I_y^{(gx)} I^*_{aerr} \right) - \frac{\partial}{\partial y}\left(I_y^{(gy)} I^*_{aerr} \right) = 0 \end{aligned} \quad (3)$$

where the star superscript denotes complex conjugate and I_{aerr} is defined in Eq. (2). We used the Conjugate Gradient method to solve this linear system.

4. Image Warping

Warping is an expensive operation because we have to interpolate at every pixel. We used linear interpolation which proved sufficient for the quality of the input images. The warping is done in two stages: first we round the displacement vectors and then we correct by adding the derivatives times the residual of the rounding. The MediaMath [7] code is shown in Fig. 1.*

*MediaMath syntax is very similar to C

```
function linwarp_fimg(im,u,v)
     "Warps the image using linear interpolation"
{
  local im12, im12_x, im12_y, ru, rv, du, dv;
         /* Integer warp the image */
  im12   = intwarp_fimg(im,ru=round_fimg(u),rv=round_fimg(v));
          /* and its derivatives    */
  im12_x = intwarp_fimg(D_x(im),ru,rv);
  im12_y = intwarp_fimg(D_y(im),ru,rv);
  du     = u - ru;
  dv     = v - rv;
                      /* add the derivative times the */
  im12  += im12_x*du;  /* residual of the rounding */
  im12  += im12_y*dv;
  im12;                /* return im12 */
};
```

Fig. 1. Mediamath code for warping using linear interpolation.

It is easy to see that this can be extended to second order interpolation if the results are not satisfactory by computing the second order derivatives and multiplying them by second order monomials of du and dv.

5. Animating the morphing

To produce the morphing we need the displacements from image one to image two and from image two to image one. Then we calculate the intermediate images deforming both images towards each other, so that they “meet” in between and we take a weighted average (cross dissolve). By varying the weights and the amount of deformation we can create a smooth transition from one image to the next. The MediaMath code is shown in Fig. 2.

6. Results

We used the algorithm to morph the images of various people. The image sequences could be viewed using MediaMath and xv (about one image every three seconds) or on an SGI using moviemaker/movieplayer. Nine of the frames are shown on Fig. 3. For comparison

```
uv=Gb_A_flow(im1,im2,:sig=0.0, :lam=10.0, :maxits=10, :solver='ConGrad,
                  :flow_diag_mult=flow_diag_mult,
                  :scls = 3,
                  :scl1 = 0.7,
                  :w_s_prod = 3.0,
                  :oriens=4,
                  :wphase=1.0);

uv_i=Gb_A_flow(im2,im1,:sig=0.0, :lam=10.0, :maxits=10, :solver='ConGrad,
                  ...
                  :wphase=1.0);

xv = spawn("xv", "morph");

for (i=1; i<=10; i++)
{
  local ru, rv, du, dv, uv_s, uv_si, temp;

  uv_s = uv*(i/10.0);
  uv_si= uv_i*((10-i)/10.0);

  im12 = linwarp_fimg(im2,uv_s->u,uv_s->v);
  im12 *= (10-i)/10.0;
  temp = linwarp_fimg(im1,uv_si->u,uv_si->v);
  temp *= i/10.0;
  im12 += temp;

  write_img(im12, "morph",:rescale=nil);
  sleep(1);
  kill(xv, SIGQUIT);
  sleep(1);
};
```

Fig. 2. The MediaMath code to animate the morphing using *xv*.

we show the result of the fifth frame using Horn and Schunck algorithm [5] in Fig. 4. The results with the Gabor-affine algorithm are much better.

7. Conclusions

We presented an approach to morphing that does not require a human operator by using flow techniques from computer vision. We plan to extend the approach to moving images and introduce a simple way for the operator to guide the flow algorithm in cases that the images have parts that are extremely different.

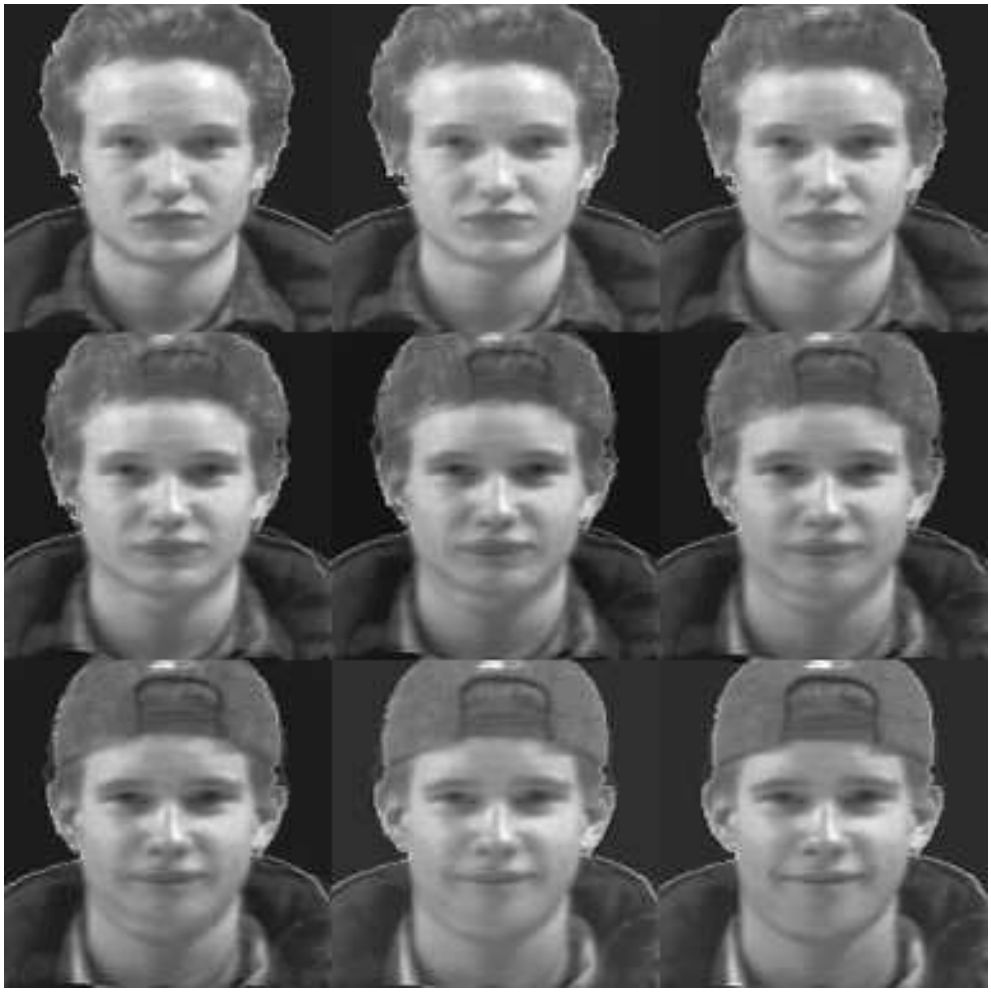

Fig. 3. Nine images of one young male morphing into another.

Fig 4. The fifth image using Horn and Schunck algorithm.

References

1. J. L. Barron, D. J. Fleet, and S. S. Beauchemin, *Performance of Optical Flow Techniques*, RPL-TR-9107, Robotics and Perception Lab, Queen's University (July 1993).

2. T. Beier and S. Neely, "Feature Based Image Metamorphosis," *SIGRAPH*, pp. 35-42 (1992).

3. Hsiao Jing Chen, Yoshiaki Shirai, and Minoru Asada, "Obtaining optical flow with multi orientation filters.," *CVPR*, (1993).

4. D. J. Fleet and A. D. Jepson, "Computation of component image velocity local phase information," *Intl' Journal of Computer Vision* **5** pp. 77-104 (1990).

5. B. K. P. Horn and B. G. Schunck, "Determining Optical Flow," *Artificial Intelligence* **17** pp. 185-204 (1981).

6. B. Lucas, *Generalized Image Matching by the Method of Differences*, PhD Dissertation, Dept. of Computer Science, Carnegie Mellon University (1984).

7. Minas Spetsakis, "MediaMath: A reasearch environment for vision research," *Vision Interface*, pp. 118-126 (1994).

8. G. Wolberg, *Digital Image Warping*, IEEE Computer Society Press Monograph (1990).