A General Method for Overlap Control in Image Warping

Bernard Tiddeman[*], Neil Duffy[†] and GrahamRabey[‡]

March 20, 2000

Abstract

A new general solution for the construction of one-to-one image warping functions is presented. The algorithm takes a set of user specified translations and constructs a set of one-to-one warps by interpolation and scaling. These are concatenated to produce a single one-to-one warping function, which prevents overlap in the resultant warped image.

[*]Perception Laboratory, School of Psychology, University of St Andrews, Fife, KY16 9JP. Telephone +44 (0)1334 463044 Email bpt@st-and.ac.uk

[†]Department of Computing and Electrical Eng., Heriot Watt University, Riccarton, Edinburgh, UK.

[‡]Interface Project, Radiology Department, Stoke Mandeville Hospital, Aylesbury, UK.

1 Introduction

Image warping is the process of deforming a digital image geometrically [1] [2]. The problem of image warping given a set of scattered translation vectors is essentially the problem of interpolating two functions simultaneously. One function is for the translation in the direction of the x-axis and the other for the translation in the direction of the y-axis. Scattered data interpolation problems occur frequently in scientific and engineering problems and many solutions have been proposed. The reader is referred to [3] [4] [5] [6] [7] for excellent reviews.

In image warping, an additional problem to that of interpolation is that of maintaining the one-to-one property of the warping transformation. Each point in the transformed image should correspond to only one point in the original image and vice-versa. If the one-to-one property is not satisfied the warped function will have the appearance of being folded, rather than only stretched.

In Figure 1 three points are mapped according to,

$$(x, y) \rightarrow (f(x, y), g(x, y)) \tag{1}$$

$$(x + \delta x, y) \rightarrow (f(x + \delta x, y), g(x + \delta x, y)) \tag{2}$$

$$(x, y + \delta y) \rightarrow (f(x, y + \delta y), g(x, y + \delta y)) \tag{3}$$

In the limit as $\delta x \rightarrow 0$ and $\delta y \rightarrow 0$ the direction of the vectors $\mathbf{a}$ and $\mathbf{b}$ are given by

$$\mathbf{a} \propto \left(\frac{\partial f}{\partial x}, \frac{\partial g}{\partial x} \right) \tag{4}$$

$$\mathbf{b} \propto \left(\frac{\partial f}{\partial y}, \frac{\partial g}{\partial y} \right) \tag{5}$$

The mapping will not flip when the angle $0 < \alpha < \pi$, or equivalently when $\sin(\alpha) > 0$. The cross-product of $\mathbf{a}$ and $\mathbf{b}$ will be a vector perpendicular to the page, given by

$$\mathbf{a} \times \mathbf{b} = |\mathbf{a}| \, |\mathbf{b}| \sin \alpha \hat{\mathbf{n}} \tag{6}$$

hence the Jacobian determinant, J, of the transformation, given by

$$J = \frac{\partial (f, g)}{\partial (x, y)} = \frac{\partial f}{\partial x} \frac{\partial g}{\partial y} - \frac{\partial f}{\partial y} \frac{\partial g}{\partial x} \tag{7}$$

will have the same sign as $\sin \alpha$. Therefore if the Jacobian determinant passes through zero (becoming negative) the mapping will overlap [8].

This analysis is easily extended to higher dimensions. For example, in three-dimensions the Jacobian is given by the triple scalar product of the partial derivative vectors, $\mathbf{F}_x$, $\mathbf{F}_y$ and $\mathbf{F}_z$, where $\mathbf{F} = (f, g, h)$ is the vector field representing the warping function. The magnitude of the Jacobian at each point will give the volume that an infinitesimal volume element $dxdydz$ is transformed to. If the Jacobian has a negative sign this means that a positive volume element is mapped to a negative volume element, or in other words, that the mapping has flipped.

2 Previous solutions

Previously derived methods for preventing overlap have been specifically designed for a small subset of interpolation methods. For the numerical solution of the thin-plate stiffness equations [9] [10], an additional constraint (in addition to the thin-plate stiffness and the border constraints) can be placed on the solution, forcing the Jacobian determinant to remain positive [11] [12]. For the case of multilevel B-splines [13] a clipping of the spline's control points whose magnitude is above 0.48 can ensure the one-to-one property [14]. For triangulation based warping a method using time-varying triangulation has been proposed [15].

3 A new general solution

The one-to-one property of the change of variables can be preserved using two facts.

First, the concatenation of two one-to-one maps must be one-to-one, i.e. given two one-to-one mappings

$$(x, y) \rightarrow (f, g) \tag{8}$$

with Jacobian $J_1 > 0$ and

$$(f, g) \rightarrow (u, v) \tag{9}$$

with Jacobian $J_2 > 0$, then the combined mapping

$$(x, y) \rightarrow (u, v) \tag{10}$$

will (by the chain rule of differentiation) have Jacobian

$$J_3 = J_1 J_2 > 0 \tag{11}$$

and so be one-to-one.

Second, an overlapping function of the form

$$(x, y) \rightarrow (x + f, y + g) \tag{12}$$

can be made one-to-one by scaling the functions f and g by an appropriate constant α satisfying $0 \leq \alpha < min(\beta(x, y))$ so that the mapping

$$(x, y) \rightarrow (x + \alpha f, y + \alpha g) \tag{13}$$

is one-to-one. The values $\beta\,(x, y)$ are chosen at each point so that the Jacobian, J, of the rescaled mapping $(x + \beta f(x, y), y + \beta g(x, y))$, given by the equations

$$J = (\beta f_x + 1)\,(\beta g_y + 1) - \beta^2 f_y g_x \tag{14}$$

(where the subscripts denote partial derivatives), is equal to zero . These quadratic equations can be solved at each point in the warping functions for β using finite difference approximations for the partial derivatives. We are only

interested in solutions for which $0 < \beta \leq 1$, because we require the interpolated function to translate the pixels a fraction of the desired shift ($\beta > 0$), without overshooting the target ($\beta \leq 1$). The quadratic equation 14 always passes through $J = 1$ for $\beta = 0$, which leaves only three possible cases (Figure 2):

1) No real roots between 0 and 1. In this case the Jacobian is positive for all $\alpha \in (0, 1]$ so α is not restricted by this point.

2) One real root β between 0 and 1. In this case J is positive for $0 \leq \alpha < \beta$.

3) Two real roots β_0 and $\beta_1$between 0 and 1. In this case α must be less than the smaller root or larger than the larger root for positive J.

Therefore, we set each $\beta(x, y)$ equal to the smallest positive real root on $(0, 1]$ or 1 if there are no roots on (0,1]. The scaling parameter α is chosen to be less than or equal to the smallest value of β in the warping function.

Using these two facts, a simple method for eliminating overlap is to construct a set of one-to-one warps by interpolation and scaling, which is then concatenated into a single one-to-one warp. Interpolation is first carried out using the chosen interpolation method, and then the warping function is scaled by the fraction α. This will translate the border constraints a fraction α of the desired amount. The remaining translations are then interpolated, scaled if necessary and concatenated with the original. This is repeated until $\alpha = 1$ and the border constraints are satisfied or until a user specified number of iterations has been exceeded. A restriction on the number of iterations is necessary because the method is not guaranteed to converge in a finite number of iterations. If convergence is not reached a choice must be made between satisfying the border constraints and overlapping, or having a one-to-one warp and not satisfying the

border conditions. Each iteration involves an interpolation, scaling and concatenation of a partial warp and so the choice of the maximum number of iterations to allow will depend on the speed of the interpolation algorithm used and the speed demands of the application.

An additional user defined parameter can be included in equation 14 to specify a minimum value for the Jacobian greater than zero for each of the partial warps, i.e.

$$(\beta f_x + 1)(\beta g_y + 1) - \beta^2 f_y g_x = J_{min} \tag{15}$$

where J_{min} $(0 \leq J_{min} < 1)$ is the minimum desired value for the Jacobian of each partial mapping. A larger value of J_{min} will lead to a flatter warping function, but will also increase the possibility that the sequence of partial warps will not converge to the specified constraints.

The final step is to apply the warping function to the image. This is only carried out once, after concatenation of all the partial warps, to avoid repeated resampling of the image and the corresponding degredation in image quality. The following C-style pseudo-code algorithms give a concise description of the algorithm.

```
/* This function takes two sets of points */
/* and returns the Warp function          */
Warp One_to_one_Interpolate(int numPoints,
         Point *startPos, Point *endPos, float Jmin)
{
    Warp partialWarp, totalWarp;
    int its, maxits=20;
    float α;
```

```
    bool converged=false;

    while (!converged && its<maxits) {
         /* Calculate the partial warp using chosen method */
         partialWarp = interpolate(numPoints, startPos, endPos);
         /* Calculate the scaling factor for this partial warp */
         α = calculate_overlap(partialWarp, Jmin);
         /* Check for convergence */
         if (α==1) converged = true;
         /* Scale the partial warp */
         else partialWarp = scaleWarp(partialWarp, α);
         /* Add partial warp to the total */
         totalWarp = concat(totalWarp, partialWarp);
         /* Reposition start Points for next iteration */
         startPos = shiftPoints(partialWarp, numPoints, startPos);
         its = its+1;
    }
    return totalWarp;
}
```

```
/* This function calculates the scaling factor */
/* required to make the warp one-to-one        */
float calculate_overlap(Warp warp, float Jmin)
{
    int x, y;
    float fx, gx, fy, gy, α = 1, β;
    for(x=0; x<Warp.width; x++)
      for (y=0; y<Warp.height; y++) {
         /* Estimate partial derivatives of warp functions */
         (fx, gx, fy, gy) = estimate_derivatives(warp, x, y);
         /* solve_quadratic(a, b, c) solves ax^2+bx+c=0
                              and returns the two roots */
         (β0, β1) = solve_quadratic(fx*gy-fy*gx, (fx+gy), 1-Jmin);
         /* Check for 3 cases */
         if (0 < β0 ≤ 1 && imag(β0)==0) {
             if (0 < β1 ≤ 1 && imag(β1)==0) {
                 /* Case 3: Two real roots on (0,1], pick smallest*/
                 if (β0 > β1) β = β1;
                 else β = β0;
             }
             /* Case 2(a): β0 only real root (0,1]*/
             else β = β0;
         /* Case 2(b): β1 only real root on (0,1]*/
```

```
            else if (0 < β1 ≤ 1 && imag(β1)==0) β = β1;
            /* Case 1: No real roots on (0,1]*/
            else β = 1;
            /* Check if this is the smallest value of β and store in α */
            if (β < α) α = β;
        }
        return α;
    }
```

Once again this analysis can be extended to higher dimensional cases. For three-dimensions the quadratic equation in β (Equation 14) is replace by a cubic equation in β, which can be solved using a direct method such as Cardan's method. This cubic function will pass through $J = 1$ when $\alpha = 0$ and so choosing the smallest real root on $\alpha \in (0, 1]$ (or 1 if no real roots exist on (0,1]) for β at each point will guarantee to scale the partial warping functions so that they are one-to-one.

4 Results

The method presented is independent of the type of interpolation being performed and so can be combined with many scattered data interpolation procedures. An example image and specified translations (Figure 3) are used to demonstrate the effect of warping with and without overlap control. Figure 4 shows the results of interpolating using Shepard's method [16]. The effects of overlap can be seen, particularly on the central white square, which has been split at the four corners. Figure 5 uses the same interpolation procedure but constrains the warp to be one-to-one using the overlap control method. The central white square now stretches smoothly out to the corners. Figures 6 and 7 demon-

strate an analytically interpolated thin-plate spline warp [17] [18], without and with overlap control respectively. Again the folding of the image is particularly notable around the central white square, with the image completely inverted at the borders when overlap control is not used. The application of the overlap control method to a real image (Figure 8) demonstrates its effect more clearly. Figure 9 shows warping the image with the thin plate spline method without using overlap control and Figure 10 demonstrates the improvement when the overlap control method is used. A three-dimensional example can be seen at http://psych.st-and.ac.uk:8080/people/personal/bpt/overlap.html.

5 Conclusions

The method for overlap control presented in this paper is simple, efficient and effective. Because the method is not dependent on the interpolation method employed it can be combined with many of the interpolation methods already available to give a more robust image warping method.

Acknowledgments

The work reported in this paper would not have been possible without the generous support of Sir James Savile, via the Jimmy Savile Stoke Mandeville Hospital Trust.

References

[1] G. Wolberg, *Digital Image Warping,* IEEE Computer Society Press , Los Alamitos, California (1990).

[2] D. Ruprecht and H. Muller, Image warping with scattered data interpolation. *IEEE Computer Graphics and Applications*, **15**(2), 37-43 (1995).

[3] H. Spath, *Two dimensional spline interpolation algorithms*, A.K. Peters Ltd., Wellesley, Massachusetts (1995).

[4] R. Barnhill, Representation and approximation of surfaces. In *Mathematical software III*, J.R.Rice (Ed.), Academic Press, New York, 68-119 (1977).

[5] R. Franke and G.M.Nielsen, Scattered data interpolation of large sets of scattered data. *International Journal of Numerical Methods in Engineering*, **15**, 1691-1704 (1980).

[6] J. Hoschek and D. Lasser, *Computer Aided Geometric Design*, A.K.Peters Ltd, Wellesley, Massachusetts (1993).

[7] L. Schumaker, Fitting surfaces to scattered data. *Approximation Theory II*, C. Chui, L. Schumaker and G.Lorenz (Eds.), Wiley, New York, 203-268 (1976).

[8] G.H. Meisters and C. Olech, Locally one-to-one mappings and a classical theorem on Schlicht functions. *Duke Mathematics Journal*, **30**, 63-80 (1963).

[9] D. Terzopoulos, The computation of visible surface representations. *IEEE Trans. Pattern Analysis and Machine Intelligence*, **10**(4), 417-439 (1988).

[10] D. Terzopoulos, Multilevel computational processes for visual surface reconstruction. *Computer Vision, Graphics and Image Processing*, **24**, 52-96 (1983).

[11] S.-Y. Lee, K.-Y. Chwa, J. Hahn and S.Y. Shin, Image morphing using deformable surfaces. In *Proceedings of Computer Animation '94,* IEEE Computer Society Press, Geneva, Switzerland , 31-39 (1994).

[12] P. Litwinowicz and L. Williams, Animating images with drawings. *Computer Graphics (SIGGRAPH '94 Proceedings)*, 409-412 (1994).

[13] S.-Y. Lee, G. Wolberg and S.Y. Shin, Scattered data interpolation with multilevel B-splines. *IEEE Transactions on Visualization and Computer Graphics*, **3**(3), 229-244 (1997).

[14] S.-Y. Lee, K.-Y. Chwa, S.Y. Shin and G. Wolberg G., Image metamorphosis using snakes and free-form deformations. *Computer Graphics (SIGGRAPH '95 Proceedings)*, 439-448 (1995).

[15] K. Fujimura and M. Makarov, Foldover-Free Image Warping. *Graphical Models and Image Processing*, **60**(2), 100-111 (1998).

[16] D. Shepard, A two dimensional interpolation function for irregularly spaced data. *Proc. ACM 23rd National Conference*, 517-524 (1968).

[17] J. Duchon, Splines minimizing rotation-invariant semi-norms in Sobolev spaces. In *Multivariate Approximation Theory*, Chui C., Schumaker L. and Ward J. (Eds.), Birkhauser, Basel, Switzerland , 85-100 (1975).

[18] J. Meinguet, Multivariate interpolation at arbitrary points made simple. *Zeitschrift fur Angewandte Mathematik und Physik (ZAMP),* **30**, 292-304 (1979).

Figure Captions

Figure 1

A mapping function will be one-to-one provided the angle α is between 0 and π. This is only true if the cross product of the vectors **a** and **b** points out of the page. Taking the limit as $\delta x \to 0$ and $\delta y \to 0$ this becomes equivalent to the Jacobian of the transform remaining positive.

Figure 2

Examples of the three possible cases for equation 14. From left to right: a) Case 1, no real roots on (0,1]. b) Case 2 one real root on (0,1]. c) Case 3 two real roots on (0,1].

Figure 3

The example translations (left) and image (right) used in the warping examples. The four corner points are moved from the inner square to the outer square, while the remaining eight outer square edge points remain fixed.

Figure 4

The warping function interpolated using Shepard's method without overlap control (left) and the resulting transformed image (right). The deformed grid (left) shows where the pixels values are collected from in the original image, and so is effectively the inverse of the deformed image (right).

Figure 5

The example problem of figure 4 solved with Shepard's method with overlap control ($J_{min} = 0.2$, converged in 3 iterations).

Figure 6

The example warping problem interpolated using analytical thin-plate splines without overlap control.

Figure 7

The example problem solved using analytical thin-plate splines with overlap control ($J_{min} = 0.2$, converged in 3 iterations).

Figure 8

An example real image with the control points displayed as circles and the desired translation defined by the arrows.

Figure 9

The example real image warped using the thin plate spline method without overlap control.

Figure 10

The example real image warped using the thin plate spline method with overlap control ($J_{min} = 0.2$, converged in 3 iterations).

Figures

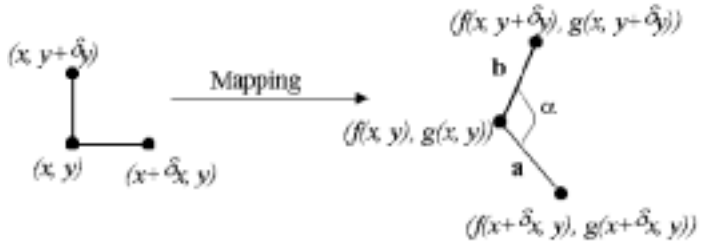

Figure 1:

Figure 2:

Figure 3:

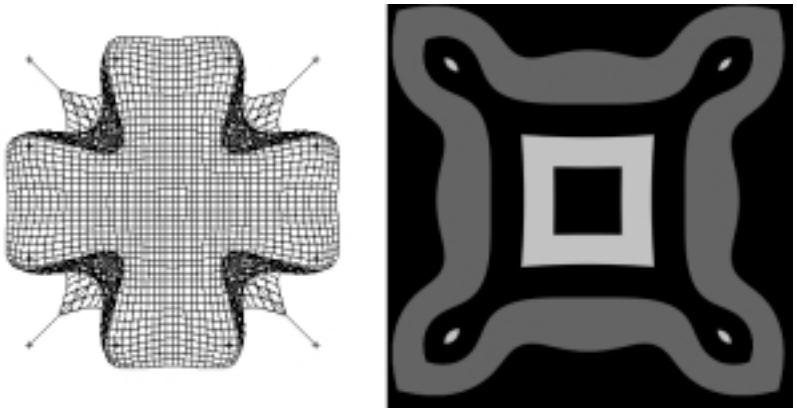

Figure 4:

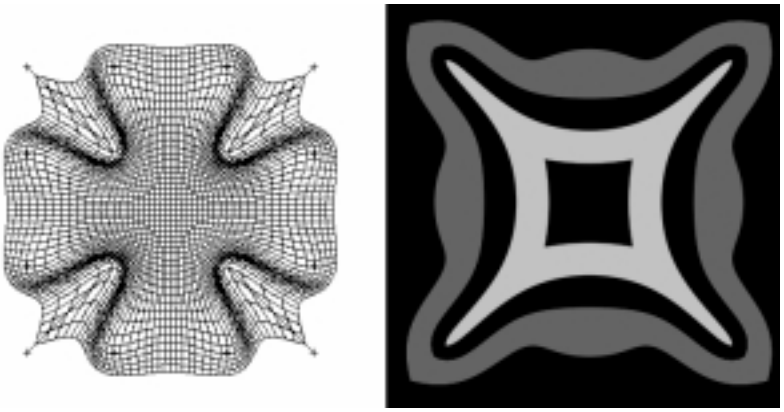

Figure 5:

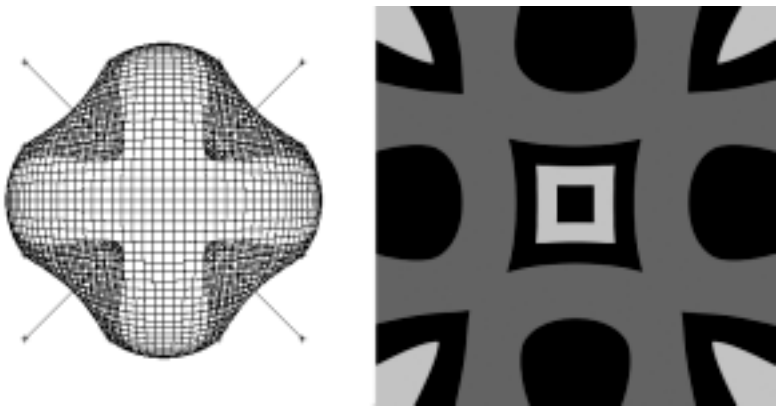

Figure 6:

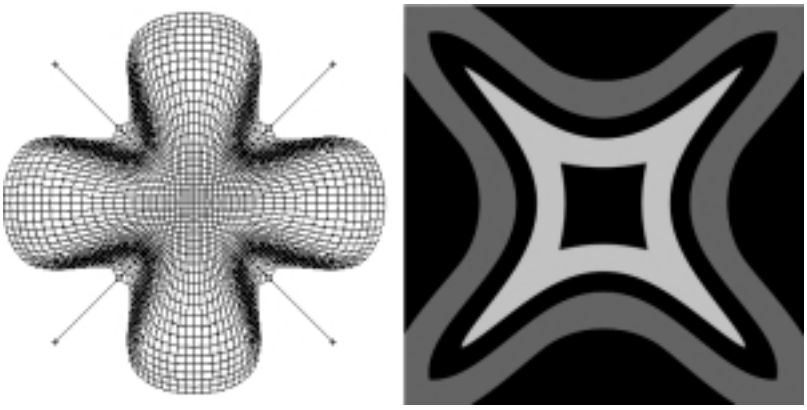

Figure 7:

Figure 8:

Figure 9:

Figure 10: