Multiple Landmark Warping Using Thin-Plate Splines

Mark Whitbeck[†] and Hongyu Guo[‡]

Key words: Thin-Plate Splines, AlexWarp, interpolation, applet, Java programming language

Abstract

Image warping plays an extremely important role in face matching and recognition and therefore has wide applications and great significance in computer vision, biomedical imaging and homeland security. One popular warping algorithm developed by Alex Rosen, known as AlexWarp has the drawback that it only allows one pair of landmarks being specified at one time. The consequence of this is that the destination landmarks are displaced in the subsequent warping. We develop an image warping algorithm using thin-plate splines that allows multi-landmarks being specified and warped simultaneously. The space transformation is the solution of minimizing a thin-plate deformation energy. QR decomposition techniques are used to stabilize the solution. We apply the new warping algorithm to various face matching situations.

[†] Department of Computing and Mathematical Sciences, Texas A&M University Corpus Christi, Presenter
Telephone (361) 334-0589 Fax (361) 887-9009 Email mwhitbeck@sci.tamucc.edu

[‡] Department of Computing and Mathematical Sciences, Texas A&M University Corpus Christi
Telephone (361) 825-3492 Fax (361) 825-2795 Email hguo@sci.tamucc.edu

We acknowledge support from Texas Research Development Fund 140100-34025 and
Texas Engineering and Technical Consortium Grant 011161-CS2003

1 Introduction

Warping an image is a transformation which involves mapping pixels from source positions to destination positions [2]. Though there are many particular algorithms for image warping, this paper focuses on implementing an image warping algorithm similar to AlexWarp using thin plate spline interpolation. The basic concept is to allow the user to enter multiple source and destination landmark points versus the one at a time approach used in the AlexWarp applet.

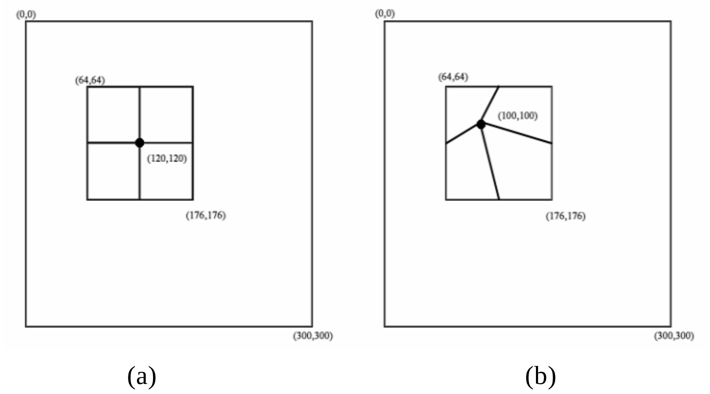
Because there are many drawbacks to the AlexWarp algorithm, this paper concentrates on how an image warping applet can be improved using thin splate interpolation. We will discuss the code requirements necessary for multiple landmark points to be specified at once and compare the results of warped images made with AlexWarp and the thin-plate spline image warping applet.

2 Previous Work

Written and copyrighted in 1996, Alex Rosen developed the Java applet entitled "AlexWarp". Since its creation, the applet has gained popularity among internet users world-wide for its simple and fast method of image warping. Webmasters simply download and add Alex's applet to their existing web-page adding a custom picture.

The AlexWarp applet works by outputting an image to the screen and allowing the user to warp the image by clicking and dragging. When the user provides one pair of landmark points, the applet determines the region to warp, warps it, and then outputs the warped picture.

The image is warped by localizing a transformation area about the landmark points. The size of this area is determined by doubling the distance between the source and target point. Two grids, the size of the transformation area, are then created containing four shapes each. One grid has four shapes centered about the source point, and the other grid has four shapes centered about the target point. The target grid has rectangular shapes and the source grid has polygon shapes. Using linear interpolation, AlexWarp transforms each polygon shape into its corresponding rectangular shape. In essence, four separate warps are performed for one pair of landmark points. This minimizes distortion and pixilation [2], but does not allow the user to warp more than what is inside the transformation area. Figure 1 demonstrates the transformation area for a warp for point (100,100) to point (120,120). Because of localization, the transformation can sometimes create sharp points or cusps. Unless this is what the user intends, this is a very negative aspect of AlexWarp. Figure 2 demonstrates how warping with AlexWarp creates cusps at the transformation target point.



(a) (b)

Fig. 1. AlexWarp transformation of pixel (100,100) to (120,120).

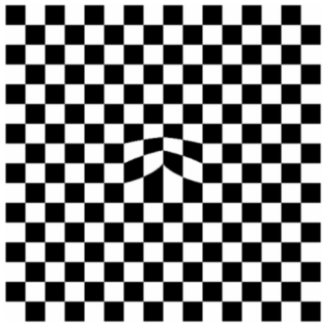


Fig. 2. Example of localization and sharp points.

The most important drawback of the AlexWarp applet is that transformations can only be applied one at a time. When the transformation area of the next warp is within the boundaries of the previous warp, the results of the previous warp are distorted. This can lead to further pixilation and distortion and render very undesirable results. In contrast, however, when multiple warps can be applied at once, the intended result is usually achieved with a much richer and smoother quality.

3 Thin-Plate Splines

The term “thin-plate spline” refers to the bending of a thin sheet of metal [1][3][4][5][6]. The process of using thin-plate splines in image warping involves miminizing a bending energy function for a transformation over a set of given nodes or landmark points. Much like the AlexWarp applet, the user is shown a picture on which they can click and drag to define source and destination points. It also allows the user to enter clamped points, which are points that will not move in the transformation. When the warp button is clicked, all landmark and clamped points will be used to calculate the bending energy function which is then used to interpolate and transform the pixels into a warped image. The resulting image is then outputted to the screen.

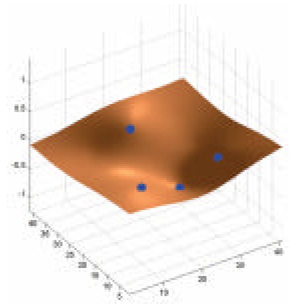


Fig. 3. Bending of a thin sheet of metal.

3.1 The thin-plate spline algorithm

Thin thin-plate algorithm is solved by minimizing the bending energy below [1].

$$\iint_{\mathbb{R}^2} \left(\left(\frac{\partial^2 z}{\partial x^2} \right)^2 + 2\left(\frac{\partial^2 z}{\partial x \partial y} \right)^2 + \left(\frac{\partial^2 z}{\partial y^2} \right)^2 \right) dx\, dy \tag{1}$$

The basis for solving the algorithm is given by a kernel function U as:

$$U = r^2\log(r^2), \text{ where } r = (x_a - x_b)^2 + (y_a - y_b)^2 \tag{2}$$

Given a set of source landmark points, we define P as a matrix of (3 x n) where n is the number of points.

$$P = \begin{bmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ \cdots & \cdots & \cdots \\ 1 & x_n & y_n \end{bmatrix}, 3 \times n; \tag{3}$$

Using the kernel function, we define another matrix K.

$$K = \begin{bmatrix} 0 & U(r_{12}) & \cdots & U(r_{1n}) \\ U(r_{21}) & 0 & \cdots & U(r_{2n}) \\ \cdots & \cdots & \cdots & \cdots \\ U(r_{n1}) & U(r_{n2}) & \cdots & 0 \end{bmatrix}, n \times n; \tag{4}$$

Finally, we define L as a combination of K and P.

$$L = \left[\begin{array}{c|c} K & P \\ \hline P^T & O \end{array} \right], (n+3) \times (n+3) \tag{5}$$

, where ^{T} is the matrix transpose operator and O is 3x3 matrix of zeros.

The matrix L allows us to solve the bending energy equation. Inversing this matrix L on to another matrix Y defined as $Y=(V \mid 0\ 0\ 0)^T$, where V is any n-vector from $(v_1, \ldots, v_n)$, we derive a vector $W = (w_1, \ldots, w_n)$ and the coefficients a_1, a_x, a_y.

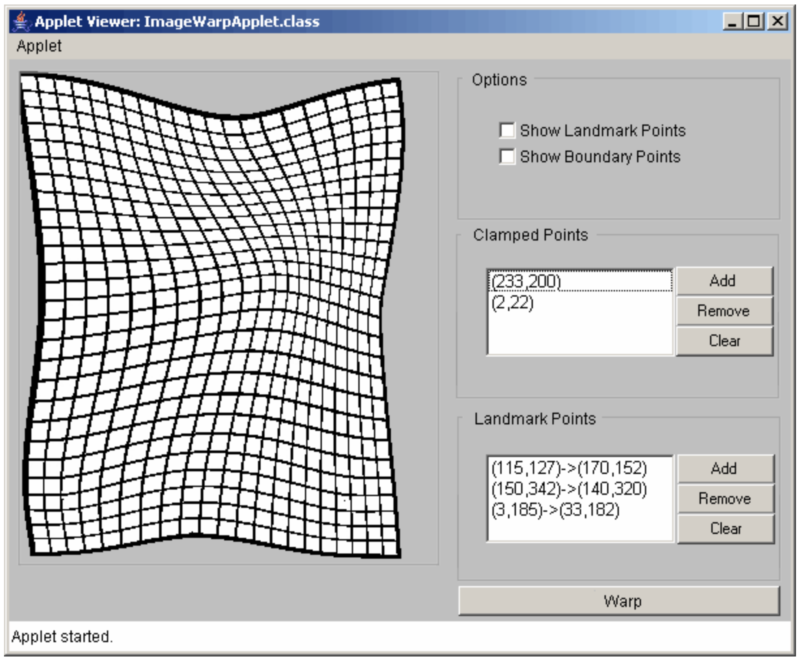
$$L^{-1}Y = (W | a_1 \quad a_x \quad a_y)^T. \tag{6}$$

Using the elements of $L^{-1}Y$, we can define a function $z(x, y)$ everywhere in the plane by the equation:

$$z(x, y) = a_1 + a_x x + a_y y + \sum_{i=1}^{n} w_i U(|P_i - (x, y)|). \tag{7}$$

3.2 Designing the image warping applet

The interface of the thin plate splines image warping applet is much like the AlexWarp applet. The difference is that the user is able to define multiple landmark points at once rather than one at a time. The user is also allowed to enter multiple clamped points as well. To design an interface capable of this feat, three separate classes are created. One class contains the user interface (ImageWarpApplet), one holds the landmark and clamped points entered by the user (ImageWarpPoints), and the final class will actually performs the thin-plate spline interpolation technique on the image (ImageWarpFunction).
The user interface is key to a good image warping algorithm. As seen below, in Figure 4, the interface looks much like the AlexWarp applet with exception of the extra options and buttons. The image is outputted to the screen and the user can define transformation vectors as they please. Each landmark and clamped point defined by the user are held in the ImageWarpPoints class until the warp button is clicked and the ImageWarpFunction performs the actual transformation.



(233,200)
(2,22)

(115,127)->(170,152)
(150,342)->(140,320)
(3,185)->(33,182)

Applet started.

Fig. 4. Thin Plate Splines ImageWarpApplet screenshot.

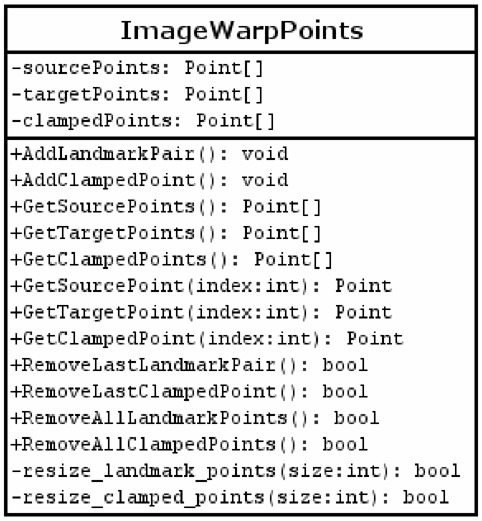
Like most other Applets, the coding of the interface is straightforward; it extends the Java Applet class and implements classes like ActionListener, MouseListener, and MouseMotionListener. At initialization, the ImageWarpApplet class is loaded, the picture and buttons are outputted to the screen, and the interface listens for actions to occur. When buttons are clicked or when mouse is pressed and released, the event listeners take care of the appropriate actions necessary.

The following Java style pseudocode defines what happens when the Warp button is clicked in the applet:
(note: myImage is defined as Java.awt.Image and myTransformationPoints is defined as ImageWarpPoints)

```
Function ActionPerformed( event )
   if event is equal to "Warp" then WarpImage()
End Function


Function WarpImage()
   Declare thinPlate as new ImageWarpFunction( myImage.width, myImage.height)
   Return thinPlate.Warp( myTransformationPoints, myImage )
End Function
```

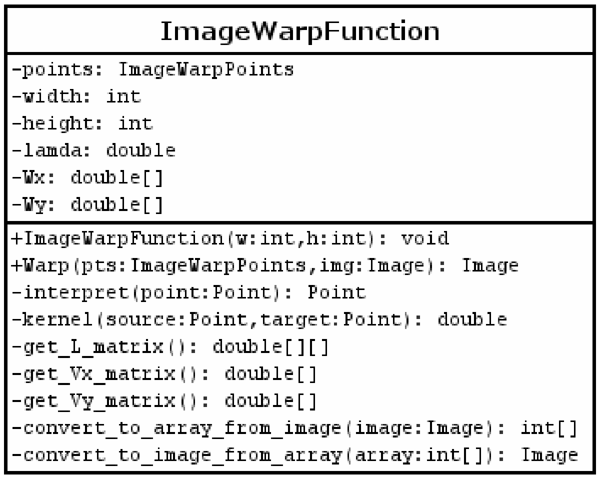
As mentioned earlier, the ImageWarpPoints class holds all of the source and target landmark points as well as any clamped points entered by the user. The public functions of the class allow for points to be added or removed from the private arrays containing the source, target, and clamped points. The UML diagram for this class is shown in Figure 4.



```
ImageWarpPoints
-sourcePoints: Point[]
-targetPoints: Point[]
-clampedPoints: Point[]
+AddLandmarkPair(): void
+AddClampedPoint(): void
+GetSourcePoints(): Point[]
+GetTargetPoints(): Point[]
+GetClampedPoints(): Point[]
+GetSourcePoint(index:int): Point
+GetTargetPoint(index:int): Point
+GetClampedPoint(index:int): Point
+RemoveLastLandmarkPair(): bool
+RemoveLastClampedPoint(): bool
+RemoveAllLandmarkPoints(): bool
+RemoveAllClampedPoints(): bool
-resize_landmark_points(size:int): bool
-resize_clamped_points(size:int): bool
```

Fig. 5. UML diagram for ImageWarpPoints class.

The final element for the thin-plate splines image warping applet is the ImageWarpFunction class. This class implements the thin-plate splines interpolation algorithm and actually performs the warp on the image. Because the algorithm involves solving an inverse on a matrix, the JavaNumerics class was used.
(JavaNumerics is available for download at http://math.nist.gov/javanumerics/)



```
ImageWarpFunction
-points: ImageWarpPoints
-width: int
-height: int
-lamda: double
-Wx: double[]
-Wy: double[]
+ImageWarpFunction(w:int,h:int): void
+Warp(pts:ImageWarpPoints,img:Image): Image
-interpret(point:Point): Point
-kernel(source:Point,target:Point): double
-get_L_matrix(): double[][]
-get_Vx_matrix(): double[]
-get_Vy_matrix(): double[]
-convert_to_array_from_image(image:Image): int[]
-convert_to_image_from_array(array:int[]): Image
```

Fig. 6. UML diagram for the ImageWarpFunction class.

When the ImageWarpFunction class is initialized, the private variables width and height are set by the input parameters *w* and *h*. In order to perform the transformation, the Warp function is called with two parameters, *pts* and *img*. The *pts* variable is of type ImageWarpPoints, and contains all source, target, and clamped points. The *img* variable is of type Java.awt.Image and contains the picture to be warped. The Warp function converts the image into a one-dimensional array of HSB values (Hue, Saturation, Brightness), solves the bending energy equation, and interprets every pixel to return a new warped image.

3.3 Comparison

When comparing the two algorithms, one must consider how the AlexWarp applet only allows for one transformation at a time. The example below illustrates this drawback, as well as demonstrates the negative results due to localization and creation of cusps inherent to the AlexWarp algorithm. In Figure 7, we see two shapes. One is the original image (a), and the other is the desired result (b). The objective is to warp the original image into the desired result. For the purpose of illustrating this approach, only five warp landmark pairs will be used. The corners of the image will be stretched out, two on the top and three on the bottom. When using AlexWarp, we see that these five warps produce undesirable results. Notice the sharp points, pixilation, and localization of each warp in Figure 8.

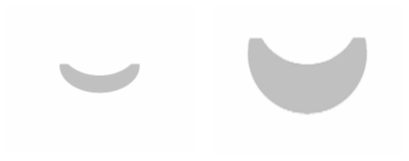


Fig. 7. Transformation example. On the left we see the original image (a), and on the right we see the desired result after the transformation (b).

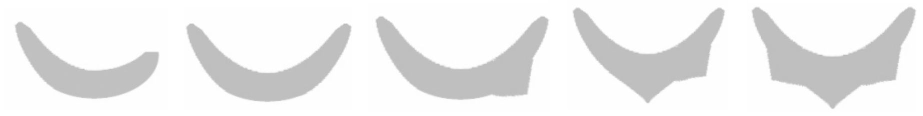


Fig. 8. Five step approach necessary to warp the image using AlexWarp. Notice the sharp points and non smooth result in Step 5.

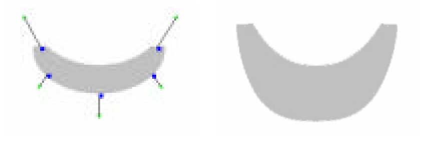


Fig. 9. Two step approach for thin plate splines image warping applet. Image (a) is the source image with the landmark points for the warp and image (b) is the transformation after clicking the Warp button.

The example on this page and the next shows how the thin-plate splines image warping algorithm produces more desirable results than the AlexWarp algorithm. In the AlexWarp section of Figures 10 - 12, we clearly see how the AlexWarp algorithm takes no attempt to avoid the creation of cusps. In Figure 10, for instance, the localization of each warps caused Clinton's lips to become very pointy versus bigger. Same as with Bush's chin and Gore's ear loaf in Figures 11 and 12. In all three Figures, however, the objectives were more easily accomplished without distortion, pixilation, or the creation of cups by using the thin-plate splines algorithm.

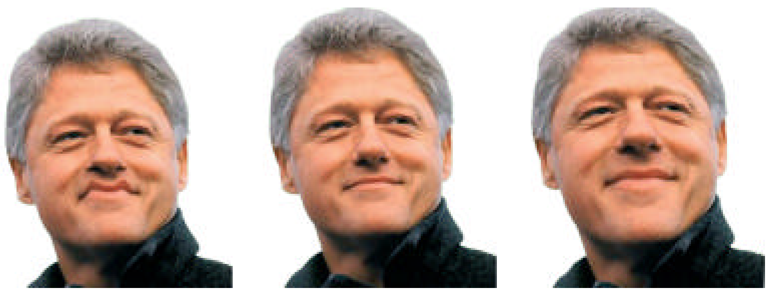


Fig 7. Objective: make Clinton's lips bigger with two warps.

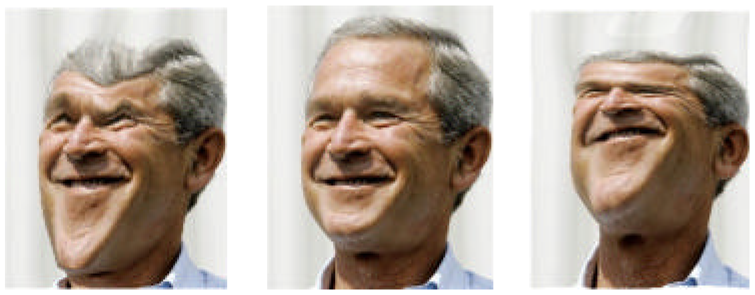


Fig 8. Objective: squish Bush’s head and extend size of chin.

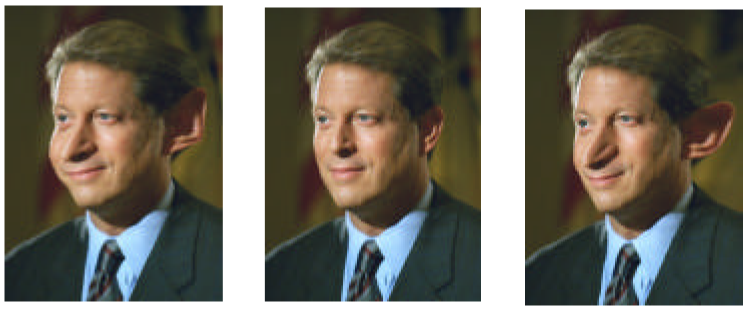


Fig 9. Objective: extend Gore’s nose and ear without altering facial features.

4. Conclusion

We have presented an approach and implementation to an image warping applet similar to AlexWarp. By utilizing QR decomposition techniques provided by the JavaNumerics math class, Bookstein’s bending energy equation can be minimized and the thin plate spline algorithm can be applied to a digital picture. The advantages of the this technique is that it overcomes the problems inherent to the AlexWarp algorithm. Multiple high quality warps can be produced at once versus the one at a time approach, and problems like localization and creation of cusps are easily avoided.

After comparing several warps made using AlexWarp and the thin plate splines image warping applet, it is easy to understand why the drawbacks in AlexWarp are detrimental to an artist’s ability to be creative. Using the thin plate splines image warping applet, constraints on the warping area can be made using clamped points, multiple warps can be generated at once by specifying multiple landmark pairs of points, and the user has the ability to add or remove landmark/clamped points as they please. These three fundamental features are key to providing a true workspace for facial warping. Complimenting this with an image warping algorithm like the one presented in the paper, and it requires only minimal efforts from the user to render high quality results. The comparisons of warps made in each program, AlexWarp and the thin plate splines image warping applet, show that unintentional results are more likely to occur in AlexWarp due to it’s lack of interface and ability to warp images.

Future ideas to extend the functionality of this image warping applet include using different kernel functions, implementing facial recognition techniques, and/or predefining user selectable clamps. These future add-ons will help in producing higher quality results and will even help make the warping process a little easier. Because of the versatility of the Java programming language, many add-ons can be created and many different applications can be made using this applet as a basis.

5. References

[1] Bookstein, F.L. (1989). Principle Warps: thin plate splines and the decomposition of deformations. *IEEE Transactions on Pattern Analysis and Machine Intellegence*, 16, 460-468.

[2] Glasbey, C.A and Mardia, K.V. (1998). A Review of Image Warping Methods. *Journal of Applied Statistics*, 25, 155-171.

[3] Wahba, G. (1979). How to smooth curves and surfaces with splines and cross-validation. *Proc. 24th Conference on the Design of Experiments*. US Army Research Office 79-2, 167-192

[4] Trouve, A. (1998) Diffeomorphisms groups and pattern matching in image analysis. *International Journal of Computer Vision*, 28-3, 213-221.

[5] Joshi, S.C. and Miller, M. M. (2000). Landmark matching via large deformation diffeomorphisms. *IEEE Transaction on Image Proccessing*, 9-8, 1357-1370.

[6] Nagel H.H. (1983). Displacement vectors derived from second order intensity variations in image sequences. *Comput. Graph. Image Process*. 21: 85-117.