Notes (March 11, 2009)
Geometric Image Warping
CS 6640 Introduction to Image Processing

Ross Whitaker
School of Computing, University of Utah

1 Introduction

These notes discuss the strategy of warping or transforming one image onto another using sets of landmarks or fiducial pairs, which we also call *correspondences*. The transformation maps points in one image, $g(\bar{x})$, to points in another image $f(\bar{x})$, where $\bar{x} = (x, y)$. We can therefore create a warped version of f by sampling the function $\tilde{f}(\bar{x}) = f(T(\bar{x}))$, where T is the transformation. If we were overlay $\tilde{f}$ on g, the correspondences should align.

We denote the set of N correspondences as points $\{(x_1, x_1'), (x_2, x_2'), \ldots, (x_N, x_N')\}$ in g and f respectively. The goal is to find a transformation between two images such that $x_1' = T(x_1)$. To find such a warp we normally parameterize the transformation, so that $T(\bar{x}) = T(\bar{x}, \bar{\beta})$ and $\beta \in \Re^M$. This parameterization effectively restricts the class of transformations, and each member of this class can be viewed as a point β in an M-dimensional space.

The strategy of geometric image warping is to find the β that satisfies the constraints given by the correspondences. Generally, this problem is well posed only if the number of constraints (equations) equals the number of unknowns. For two-dimensional images, each correspondence establishes two constraints (one for x and one for y), and thus the problem is well posed if $2N = M$. Normally we over constrain the problem and solve the least-squares problem

$$\beta = \overset{\text{argmin}}{\beta} \sum_{i=1}^{N} \left(\bar{x}_i' - T(\bar{x}_i; \beta) \right)^2 . \tag{1}$$

2 Radial Basis Functions

Each correspondence establishes an offset of points $(\Delta x_i, \Delta y_i)$ between the coordinates of f and g. The problem of finding T is really a problem of interpolating these offsets so that they cover the image in a smooth way without introducing folds or tears in the image coordinate system. Therefore we can use some tricks from *scattered data interpolation*, which is a field that has studied this problem quite extensively. A *thin plate spline* is a function that minimizes the thin plate bending energy given by

$$\int (f_{xx}^2 + 2f_{xy}^2 + f_{yy}^2) dxdy. \tag{2}$$

Researchers have shown [1] the thin plate spline that satisfies a specific set of constraints has the form

$$f(\bar{x}) = \sum_{i=1}^{N} k_i \phi_i(\bar{x}) + p_2 y + p_1 x + p_o, \tag{3}$$

where

$$\phi_i(\bar{x}) = ||\bar{x} - \bar{x}_i||^2 \lg\left(||\bar{x} - \bar{x}_i||\right), \tag{4}$$

is radially symmetric around the point $\bar{x}_i$ (it is a function of $r_i = ||\bar{x} - \bar{x}_i||$) and is therefore called a *radial basis function.* In order for this function to be optimal (minimize Equation 2), there is the additional constraint that the radial basis function part of the solution should have no constant or linear terms (i.e. this must be entirely captured in the Ps).

We can use the sparse data interpolation of the RBFs to create a smooth transformation from a set of correspondences. We model the transformation as two functions $T^x(\bar{x})$ and $T^y(\bar{x})$, with two different sets of coefficients. This gives $T(\bar{x}) = (T^x(\bar{x}, T^y(\bar{x}))$ and

$$\begin{aligned} T^x(\bar{x}) &= \sum_{i=1}^{N} k_i^x \phi_i(\bar{x}) + p_2^x y_+ p_1^x x + p_o^x && (5) \\ T^y(\bar{x}) &= \sum_{i=1}^{N} k_i^y \phi_i(\bar{x}) + p_2^y y_+ p_1^y x + p_o^y && (6) \end{aligned}$$

where the superscripts in x and y are merely labels for the two sets of coefficients (not exponents).

The parameters (unknowns) of this transformation are the k_is and the Ps—and T is linear in all of these variables. Thus, we say that the transformation is a *linear function* of β, and the solution is given by a linear system $A\bar{z} = \bar{b}$. Each row of this linear system is one constraint, e.g. $T^x(\bar{x}_i) = x'_i$ or $T^y(\bar{x}_i) = y'_i$. The vector $\bar{b}$ consists of all the unknowns, and the coefficients for T^x and T^y don't really interact, so we actually have two separate approximation problems. For convenience, we will put them together into one system, and

[1] F. L. Bookstein, "Principal Warps: Thin-Plate Splines and the Decomposition of Deformations", *IEEE Trans. Pattern Analysis and Machine Intelligence*, (11)6, 1989.

we therefore have

$$\begin{pmatrix} B & 0 \\ 0 & B \end{pmatrix} \begin{pmatrix} k_1^x \\ k_2^x \\ \vdots \\ k_N^x \\ p_2^x \\ p_1^x \\ p_0^x \\ k_1^y \\ k_2^y \\ \vdots \\ k_N^y \\ p_2^y \\ p_1^y \\ p_0^y \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ x'_1 \\ x'_2 \\ \vdots \\ x'_N \\ 0 \\ 0 \\ 0 \\ y'_1 \\ y'_2 \\ \vdots \\ y'_N \end{pmatrix} \tag{7}$$

where the first three constraints (the zeros) for the xs and ys cover the constraint that the RBF part can have no constant or linear terms and B is an $(N+3) \times (N+3)$ matrix. B encodes evaluations of the RBFs at the fiducial points $\bar{x}_i$. If we let ϕ_{ij} be $\phi(\bar{x}_i - \bar{x}_j)$, we have

$$B = \begin{pmatrix} x_1 & x_2 & \dots & x_N & 0 & 0 & 0 \\ y_1 & y_2 & \dots & y_N & 0 & 0 & 0 \\ 1 & 1 & \dots & 1 & 0 & 0 & 0 \\ \phi_{11} & \phi_{12} & \dots & \phi_{1N} & y_1 & x_1 & 1 \\ \phi_{21} & \phi_{22} & \dots & \phi_{2N} & y_2 & x_2 & 1 \\ \vdots & & & & & & \\ \phi_{N1} & \phi_{N2} & \dots & \phi_{NN} & y_N & x_N & 1 \end{pmatrix} \tag{8}$$

Thus, solving the linear system given in Equation 7 gives the coefficients needed to construct a smooth transformation $T(\bar{x})$ that maps all of the correspondences correctly.

3 Image Mosaicing

Two images that are produced from either a pure rotation of an observer or arbitrary views of a planar object are related to one another by a 2D perspective transformation. We denote two corresponding points in homogeneous coordinates as $(x, y, 1)$ and $(x', y', 1)$. In perspective mappings, two points are equivalent if they are identical to within a scale factor. That is, x, y, z is equivalent to x', y', z' if there exists some λ such that $x = \lambda x'$, $y = \lambda y'$, and $z = \lambda z'$. Notice this is the same as saying that two points are equivalent if $x/z = x'/z'$ and $y/z = y'/z'$. The *perspective transformation* relating two points is a 3×3 matrix multiplication followed by the division (normalization) described above. That is

$$\begin{pmatrix} x^* \\ y^* \\ z^* \end{pmatrix} = P \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{9}$$

Thus, when we are trying to find perspective transformations, must find the 3×3 matrix that maps sets of control points (in homogeneous coordinates) to within a scale factor. We denote the elements of this linear mapping as $P = p_{ij}$:

$$P = \begin{pmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & 1 \end{pmatrix} \tag{10}$$

The linear part of the mapping is:

$$\begin{aligned} x^* &= p_{11}x + p_{12}y + p_{13} && (11) \\ y^* &= p_{21}x + p_{22}y + p_{23} && (12) \\ z^* &= p_{31}x + p_{32}y + 1 && (13) \end{aligned}$$

Because the mapping is defined to within only a constant scale factor, we set the lower right corner to 1 (in order to avoid a homogeneous linear system—there are other ways to handle this). I.e. $p_{33} = 1$.

$$\begin{aligned} x' &= \frac{p_{11}x + p_{12}y + p_{13}}{p_{31}x + p_{32}y + 1} && (14) \\ y' &= \frac{p_{21}x + p_{22}y + p_{23}}{p_{31}x + p_{32}y + 1} && (15) \end{aligned}$$

This is a nonlinear relationship but if we multiply both sides by the denominator, we have

$$\begin{aligned} p_{31}xx' + p_{32}yx' + x' &= p_{11}x + p_{12}y + p_{13} && (16) \\ p_{31}xy' + p_{32}yy' + y' &= p_{21}x + p_{22}y + p_{23} && (17) \end{aligned}$$

If we rearrange the terms we have the following linear relationship

$$\begin{aligned} p_{31}xx' + p_{32}yx' - p_{11}x - p_{12}y - p_{13} &= -x' && (18) \\ p_{31}xy' + p_{32}yy' - p_{21}x - p_{22}y - p_{23} &= -y' && (19) \end{aligned}$$

If we take multiple correspondences (x_i, y_i) and (x'_i, y'_i) we have the following linear system

$$\begin{pmatrix} -x_1 & -y_1 & -1 & 0 & 0 & 0 & x_1x'_1 & y_1x'_1 \\ -x_2 & -y_2 & -1 & 0 & 0 & 0 & x_2x'_2 & y_2x'_2 \\ & & & \vdots & & & & \\ -x_N & -y_N & -1 & 0 & 0 & 0 & x_Nx'_N & y_Nx'_2 \\ 0 & 0 & 0 & -x_1 & -y_1 & -1 & x_1y'_1 & y_1y'_1 \\ 0 & 0 & 0 & -x_2 & -y_2 & -1 & x_2y'_2 & y_2y'_2 \\ & & & \vdots & & & & \\ 0 & 0 & 0 & -x_N & -y_N & -1 & x_Ny'_N & y_Ny'_N \end{pmatrix} \begin{pmatrix} p_{11} \\ p_{12} \\ p_{13} \\ p_{21} \\ p_{23} \\ p_{23} \\ p_{31} \\ p_{32} \end{pmatrix} = \begin{pmatrix} -x'_1 \\ -x'_2 \\ \vdots \\ -x'_N \\ -y'_1 \\ -y'_2 \\ \vdots \\ -y'_N \end{pmatrix} \tag{20}$$

Solving this system gives us P. To actually implement the transformation (e.g. in image warping), is a two step process. You first multiply each coordinate (homogeneous) by the matrix P and then divide by w.