CSE 252C: Computer Vision III

Lecturer: Serge Belongie
Scribes: Andrew Rabinovich and Vincent Rabaud
Edited by: Catherine Wah

LECTURE 9
Estimating Planar Transformations

9.1. Introduction

We'll begin this lecture by assuming the correspondences between two shapes are known (*e.g.* from the Hungarian method for bipartite matching). As a motivation, let's think of estimating planar transformations as shape matching with deformable templates. We'll begin with affine transformations, which we have never seen before, and then consider a class of transformations that is flexible, and includes affine as a special case.

9.1.1. Shape matching with affine transformations

The basic strategy for shape matching is the following: choose a transformation model, fit the model (using an adequate number of correspondences), and extrapolate the warp to all of $\mathbb{R}^2$. For example, let us look at the simplest transformation model: an affine (linear) transformation with 6 degrees of freedom (rotation, translation, scaling and shear in 2D). The Euclidean/similarity model is arguably simpler, but for non-obvious reasons,

[1]Department of Computer Science and Engineering, University of California, San Diego.

September 20, 2009

estimating them is more difficult. The affine model accounts for mappings from squares to parallelograms; parallel lines are preserved (unlike perspective projection). The model is given by:

$$\boldsymbol{x}' = A\boldsymbol{x} + \boldsymbol{t}, A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}, \boldsymbol{t} = \begin{bmatrix} t_1 \\ t_2 \end{bmatrix}, \tag{9.1}$$

where $\boldsymbol{x} = [x, y]^\top$ in image 1, $\boldsymbol{x'} = [x', y']^\top$ in image 2, for a collection of k points given by $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k$.

We use least squares to estimate A and $\boldsymbol{t}$, and to estimate the affine transformation deterministically. At least 3 correspondences are needed, forming the vertices of a triangle. First, the centroids are subtracted from each point set (the optimal translation vector $\boldsymbol{t}$ is given by the difference between the centroids). Now let's write the least squares problem in matrix form $Y = AX$ or:

$$\begin{bmatrix} x'_1 & x'_2 & \ldots & x'_k \\ y'_1 & y'_2 & \ldots & y'_k \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x_1 & x_2 & \ldots & x_k \\ y_1 & y_2 & \ldots & y_k \end{bmatrix}. \tag{9.2}$$

We then find A that minimizes $\sum_k ||\mathbf{x}'_k - A\mathbf{x}_k||^2$, which in matrix form is $A = YX^\top$. As a side note, by using homogeneous coordinates, the translation component can be absorbed into a single step of solving for A and $\mathbf{t}$ together.

To get the rigid rotation component of A, use the singular value decomposition (SVD): $A = U \sum V^\top$ (here $\sum$ contains the stretching deformation), and we form $R = UV^\top$. We verify that $RR^\top = R^\top R = I$, with $\det(R) = 1$. This is not quite the same as solving for R directly, but it works quite well in practice.

Let's now look at how to find R directly. Assuming centered coordinates, we wish to find $\hat{\theta} = \arg\min_\theta ||X - RY||^2$, where $R = R(\theta)$ (R and θ are interchangeable). Also, note that $||X - RY||^2 = ||X||^2 + ||Y||^2 - 2\mathrm{tr}(X^\top RY)$, so we can instead solve

$$\hat{\theta} = \arg\max_\theta \mathrm{tr}(X^\top RY). \tag{9.3}$$

It can be shown that this is maximized by choosing R such that $X^\top RY$ is symmetric and positive semi-definite, and this occurs for the following choice of R:

$$\hat{R} = XY^\top (YX^\top XY^\top)^{-1/2}. \tag{9.4}$$

What is not obvious is how this method compares to the $R = UV^\top$ method.

9.1.2. Other transformations

There exist many transformations that are more flexible. One avenue is higher order polynomials, but this is not advised because: (i) numerically,

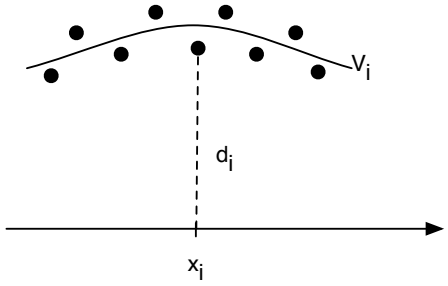

Figure 1. Estimating smooth curve from discretely sampled points.

they are ill-conditioned (unpopular in statistics and computer science communities); and (ii) when dealing with noisy or spurious correspondences, it is not easy to modify the fit to behave in a desirable way. Thus, the preferred solution to modeling flexible coordinate transformations is to use splines.

9.2. Splines

A *spline* is a special kind of function that is used in weighted combination to express a transformation. Classically, splines have been used for curves or surface fitting, so we'll start by establishing a connection between these the two problems.

Generally speaking, a coordinate transformation is a pair of functions, one for each coordinate, of the form

$$\boldsymbol{x}' = f_x(x, y), \boldsymbol{y}' = f_y(x, y); \tag{9.5}$$

alternatively, each can be a function of both x and y.

Therefore, solving for a coordinate transformation given a set of correspondences is equivalent to solving two *independent* surface fitting problems, one for the desired x coordinates, and one for the desired y coordinates.

This was also true in the affine case: we were solving for two planes that, together, specify the affine mapping. The elevation coordinates for this surface fit represent the desired x or y coordinates, so displacements in the plane get re-cast as z coordinates above the x-y plane. Therefore, in order to understand the problem of finding planar transformations, we need to understand the basics of surface interpolation.

9.2.1. Discrete case

For simplicity, we'll start with the 1D case, and we will consider discretely sampled curves (then we'll do it for the continuous case). Let's look at Figure **??**: the goal is to fit a smooth curve V to noisy data d at a set of discrete

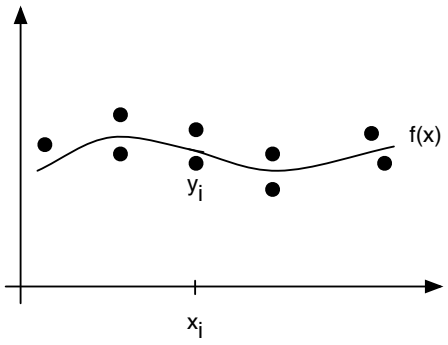

Figure 2. 1D continuous curve fitting.

locations x_i. We'll use a cost functional of the following form to balance between fidelity to the data and smoothness:

$$H[V] = \frac{1}{2}\sum_i (V_i - V_{i+1})^2 + \frac{1}{2}\lambda \sum (V_i - d_i)^2. \tag{9.6}$$

(Notation is from Hertz, Krogh, & Palmer, 1991.) Now let's take the derivative of H with respect to V_i:

$$\frac{\partial H}{\partial V_i} = K(V_{i+1} - 2V_i + V_{i-1}) + \lambda(d_i - V_i). \tag{9.7}$$

Note that the 2 terms in the first set of parentheses will survive, and we'll get a discrete 2nd derivative; in the second set of parentheses, one term survives for the data fidelity component. One way to solve this is to use gradient descent, *i.e.* :

$$V_i^{t+1} = V_i^t - \eta \frac{\partial H}{V_i^t}. \tag{9.8}$$

(Picturing $H[V_1, V_2, \ldots, V_n]$ as a surface in the N-dimensional V-space, we want to take steps downhill until we settle into a minimum.)

9.2.2. Continuous case

This works OK, but there is a more versatile approach to finding the solution that gives more insight into the effect of the choice of smoothness functional. For this purpose, we'll switch over to the continuous case, with yet another notation. For an illustration, refer to Figure **??**. Here we consider

$$[H] = \frac{1}{2}\sum_{i=1}^{N} (f(x_i) - y_i)^2 + \lambda \frac{1}{2}\phi[f] \tag{9.9}$$

and

$$\phi[f] = \frac{1}{2}\int_{\mathbb{R}} \left(\frac{\partial^2 f}{\partial x^2}\right)^2 dx \tag{9.10}$$

(the bending energy).

We appeal to variational calculus to compute the "functional derivative" of H with respect to f and set it equal to zero [1]:

$$\frac{\delta H}{\delta f} = 0. \tag{9.11}$$

First, we need to know that $\frac{\delta f(x)}{\delta f(x')} = \delta(x - x')$, so:

$$\frac{\delta}{\delta f}\frac{1}{2}\sum_{i=1}^{N}(f(x_i) - y_i)^2 = \sum_{i=1}^{N}(f(x_i) - y_i)\delta(x - x_i) \tag{9.12}$$

(this is analogous to the discrete case where only one term survived). Expression (**??**) is only nonzero when $x = x_i$.

Now let's look at the functional derivative of the smoothness functional. Define the linear operator $D \equiv \frac{\partial^2}{\partial x^2}$, then

$$\phi[f] = \frac{1}{2}\int (Df)^2 dx = \frac{1}{2}||Df||^2 = \frac{1}{2} < Df, Df > \tag{9.13}$$

and we have

$$\frac{\delta\phi}{\delta f} = D\hat{D}f = \frac{1}{2}(D\hat{D} + \hat{D}D)f, \tag{9.14}$$

where $\hat{D}$ is the adjoint of D (analogous to the transpose of a matrix $\frac{d}{d\boldsymbol{x}}\boldsymbol{x}^\top A\boldsymbol{x}$). The adjoint of $\frac{\partial^2}{\partial x^2}$ is $\frac{\partial^2}{\partial x^2}$ (it's self adjoint), and the adjoint of $\frac{\partial}{\partial x}$ is $-\frac{\partial}{\partial x}$ (due to odd symmetry of the first derivative).

Therefore, $\frac{\delta\phi}{\delta f} = \frac{\partial^4 f}{\partial x^4}$, so

$$\frac{\delta H}{\delta f} = \sum_{i=1}^{N}(f(x_i) - y_i)\delta(x - x_i) + \lambda\frac{\partial^4 f}{\partial x^4} = 0 \tag{9.15}$$

$$\Rightarrow \frac{\partial^4 f}{\partial x^4} = \sum_{i=1}^{N}\frac{y_i - f(x_i)}{\lambda}\delta(x - x_i) = \sum_{i=1}^{N} c_i\delta(x - x_i), \tag{9.16}$$

where $c_i = \frac{y_i - f(x_i)}{\lambda}$.

[1] Some background material on variational calculus is available on the class website.

9.2.3. Radial basis functions

Now we have a partial differential equation (a weighted sum of impulses) for f and we need to solve for it. By linearity, if we solve the PDE for a single impulse, we can add up the solutions with weights to get the full solution, which suggests the method of Green's functions. The Green's function corresponding to this differential operator is the solution to $\frac{\partial^4 g}{\partial x^4} = \delta(x)$. We can solve this just by integrating both sides 4 times: $g(x) = |x|^3$. After four derivatives we are left with $\delta(x)$ times a constant.

This specific Green's function or *radial basis function* (RBF) is called a cubic spline, and our solution to the original problem has the form:

$$f(x) = \sum_{i=1}^{N} c_i g(x - x_i), \tag{9.17}$$

a weighted sum of splines centered at the data points; we need to solve for the c_i's. There is one thing we forgot, however, and that is that certain functions are "invisible" to $\phi[f]$, in particular, constants and ramps, so our solution actually should have the form:

$$f(x) = \sum_{i=1}^{N} c_i g(x - x_i) + \sum_{\alpha=1}^{k} d_\alpha \Psi_\alpha(x), \tag{9.18}$$

where $\Psi_1(x) = 1$, $\Psi_2(x) = x$, for $k = 2$.

Writing down this equation for each $x_i, i = 1, \ldots, N$, we get the following linear system:

$$\begin{bmatrix} G + \lambda I & \Psi \\ \Psi^\top & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{c} \\ \boldsymbol{d} \end{bmatrix} = \begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{0} \end{bmatrix}, \tag{9.19}$$

where $(\boldsymbol{y})_i = y_i$, $(\boldsymbol{c})_i = c_i$, $(\boldsymbol{d})_i = d_i$, $(G)_{i,j} = g(x_i - x_j)$ (the spline is evaluated between all pairs of points), and $(\Psi)_{\alpha i} = \Psi_\alpha(x_i)$. Then we can solve for the c_i's and d_i's by inverting the big matrix.

What is the effect of λ? If λ is small, we get an exact interpolation through the points. If it's very large, it overwhelms G and something interesting happens: the solution becomes purely linear, *i.e.* it reduces to affine (try this at home). As λ is varied from $0 \to \infty$, we get a continuum of curve fits that range from perfectly straight to maximally "wiggly."

9.2.4. Thin plate splines

Now let's go back to the 2D case. In 2D, the generalization on the bending energy becomes:

$$\phi[f] = \frac{1}{2} \int \left(\frac{\partial^2 f}{\partial x^2}\right)^2 + \left(\frac{\partial^2 f}{\partial x \partial y}\right)^2 + \left(\frac{\partial^2 f}{\partial y^2}\right)^2 dx dy \tag{9.20}$$

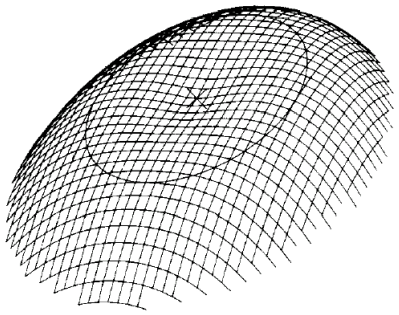

Figure 3. Fundamental solution of the biharmonic equation: a circular fragment of the surface $z(x, y) = -r^2 \log r^2$ viewed from above. The X is at $(0, 0, 0)$; the remaining zeros of the function are on the circle of radius 1.

and the corresponding Green's function is the "thin plate spline" (TPS), $g(x, y) = r^2 \log r$, $r^2 = x^2 + y^2$, as shown in Figure **??**.

Of all interpolants passing through a set of data points, the one obtained via TPS is the one that has the minimum bending energy. Finally, to estimate a planar transformation, we solve for two interpolated surfaces: one for x', and one for y'.