# Course notes, CS664, 11/4/04

- Quiz 3 put off by 1 week (until Thursday 11/11), coverage will include OFCE (through previous lecture).

- By next lecture each student should mail me your choice of a paper to write a 1-page report about (if it's available online, please include a URL). Two rules: must be related to the content of 664 and must not be a paper I covered or plan to cover. The second rule is slightly unfair, but hard to avoid...

- Next Thursday in lecture (a week from today) hand in a 1-paragraph project proposal. Projects are done 1 student at a time, and must have a research component (i.e., implementing or comparing existing methods isn't sufficient).

## Parametric models

- Another place where similar optimization methods are used is in global motion estimation with a model (this can be done locally as well).

- Standard model is affine:

$$u(x, y) = a_1 + a_2 x + a_3 y$$

$$v(x, y) = a_4 + a_5 x + a_6 y$$

An affine motion maps between two arbitrary triangles, and can be defined by its action on a triangle.

- Affine motions exactly model a 3D plane under orthographic projection, which is exactly what Birchfield and Tomasi used in the last graph cuts lecture.

- We can plug this into the OFCE (the old case was $a_2 = a_3 = a_5 = a_6 = 0$). At each pixel $i, j$ we get:

$$I_x(x_i, y_i) \cdot (a_1 + a_2 x + a_3 y) + I_y(x_i, y_i) \cdot (a_4 + a_5 x + a_6 y) = -I_t(x_i, y_i)$$

- Now our minimization problem must find 6 parameters rather than 2, but we can still use least squares (or IRLS if we want).

## Other motion optimization problems

- Handling Coarse to fine methods (Pyramid schemes). Reduce the image resolution, estimate motion, then warp. This is kind

of like filling in the bits of the motion vector from high order to low order.

- Tradeoff: small motions may be dwarfed at reduced resolutions, especially when they are not smooth. Consider a speck orbiting a ball. At low resolution, all you see if the ball.

- Layered motion estimation: think of this as inverting animation (Bugs Bunny).

- Standard solution: global parametric motion with IRLS, then fit the outliers. This won't work if there is no majority motion, but often there is.

- This is a very interesting problem, since it's not really a pixel labeling problem any more. Pixels doing very different things, which are in disconnected regions, should ultimately be given the same label.

- Often solved via clustering, which itself is commonly done with

optimization. Compute the local affine motion vectors and cluster in $(a1, a2, a3, a4, a5, a6)$ space.

## Energy functions on curves (or surfaces in 3D images)

- So far we have looked at optimization methods where candidate solutions (things whose energy we minimize) are more or less images. Sometimes called "intrinsic images".

- Optimization also shows up on curves. Or (nearly) equivalently, on surfaces in 3D imagery, i.e. from medical applications. Usually applied to a single image. Today: popular techniques of curve evolution, snakes, level sets.

- Define a curve as $C(s) = (x(s), y(s)), s \in [0, 1]$. The natural energy functions are of the form

$$E(C) = \int_0^1 F(C, C')ds.$$

  Can also use higher order derivatives.

- What is the best curve? Usually we want it to be attracted to certain points in the image but to maintain a somewhat smooth

shape (sound familiar?)

$$E(C) = E_{data}(C) + E_{smooth}(C)$$

Usually $E_{data}$ attracts the curve to high intensity gradients, we will assume this WLOG.

- Smoothness is usually global (here, unlike in stereo, this tends to be reasonable, although you can get corners in real objects). Sometimes there is also a term to make the curve shrink or expand.

- The problem of minimizing $E(C)$ is a variational problem (calculus of variations, minimizing a functional $F$). Usually solved by curve evolution.

- Define $C(s,t), t = \{0, 1, 2, \ldots\}$, where $C(s, 0)$ is our original curve. This curve evolves over time (note: input image is constant, we are not talking about motion!)

- At a given point in the curve there is a tangent vector $T$and a normal vector $N$. Think of the curve being locally approximated by a circle.

- $T$ points along the direction of $(\frac{dx}{ds}, \frac{dy}{ds})$ while $N$ is perpendicular to it (it's actually in the direction of the $\frac{dT}{ds}$. We will by convention make $N$ face inwards.

- $T$ points along the edge of the *osculating circle* and $N$ points to its center. Also note that the size of the osculating circle varies along the curve (it is called the *curvature* $\kappa$.

- $T$ and $N$ essentially define a local coordinate system with respect to which we can talk about curve evolution. The curve evolution equation we will use is

$$\frac{\partial C}{\partial t} = \beta N$$

  Here $\beta$ is not (in general) a constant.