CSE 252B: Computer Vision II

Lecturer: Serge Belongie
Scribe: Siddhartha Saha

LECTURE 15
Feature Tracking and Optical Flow

15.1. Introduction

In the previous lecture, we learned about the correspondence problem for the wide baseline case. In this lecture, we will try to tackle the problem of feature matching in the case of small baselines. A typical example of where this approach is useful is when we look at video sequence of images where the motion from one frame to another is very small, on the order of one pixel.

15.2. Optical Flow

This problem is very closely related to the problem of estimating optical flow. We will in fact make use of the equations of optical flow. The only difference is that in the case of optical flow, the algorithm is applied to all the pixels in the image, whereas in the case of small baseline feature tracking, the optical flow algorithm is used only at the feature points. As a first step, an interest point extraction is run, and then the optical flow algorithm is run on those of interest points.

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

May 24, 2004

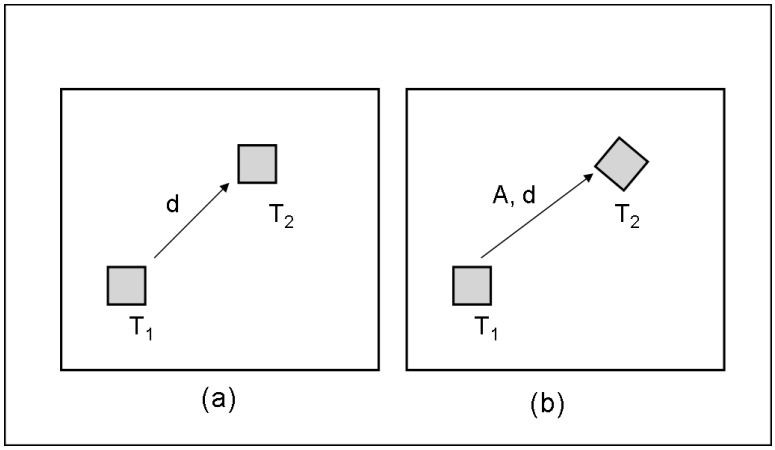

Figure 1. The translational model (2 dof) and affine model (6 dof). The affine model can also effect shear and scaling.

15.2.1. Photometric Assumptions

Brightness constancy is the basic assumption we make before running this algorithm. This means that from one frame to the next, the pixels in the image can move around, but their brightness cannot change. As a consequence, we will not be able to track features through cast shadows.

15.3. Mathematical Model

In this lecture, we will follow the derivation assuming a translational model of motion,

$$I_1(\boldsymbol{x}) = I_2(\boldsymbol{x} + \Delta\boldsymbol{x}) \tag{15.1}$$

as shown in Figure 1(a).

Let Δt be a small increment in time. Let t be the time at which the first image is taken, and at time $t + \Delta t$ the second image is taken. Then for the first image, we have $I_1(\boldsymbol{x}) = I(\boldsymbol{x}(t), t)$, and for the second image, we have $I_2(\boldsymbol{x}) = I(\boldsymbol{x}(t + \Delta t), t + \Delta t)$. This implies the following:

$$I(\boldsymbol{x}(t), t) = I(\boldsymbol{x}(t) + \Delta\boldsymbol{x}(t), t + \Delta t) \tag{15.2}$$

Note that we have removed the subscript from the expression and have expressed it purely in terms of displacements in space and time. We can think of this as describing a single pixel with a given brightness value moving around as time progresses.

Applying the Taylor series expansion around $\boldsymbol{x}(t)$ on the RHS of Equation (15.2), and neglecting higher order terms (which is valid if the displacement

is sub-pixel), we get:

$$RHS = I(\boldsymbol{x}(t), t) + \Delta x \frac{\partial I}{\partial x} + \Delta y \frac{\partial I}{\partial y} + \Delta t \frac{\partial I}{\partial t}$$

The quantity

$$\nabla I(\boldsymbol{x}, t) = \begin{bmatrix} I_x(\boldsymbol{x}, t) \\ I_y(\boldsymbol{x}, t) \end{bmatrix} = \begin{bmatrix} \frac{\partial I}{\partial x}(\boldsymbol{x}, t) \\ \frac{\partial I}{\partial y}(\boldsymbol{x}, t) \end{bmatrix} \tag{15.3}$$

is the spatial derivative of the image (i.e., the image gradient), and the quantity

$$I_t(\boldsymbol{x}, t) = \frac{\partial I}{\partial t}(\boldsymbol{x}, t) \tag{15.4}$$

is the temporal derivative of the image.[1] Since we have assumed brightness constancy, the first order Taylor series terms must vanish:

$$\Delta x \frac{\partial I}{\partial x} + \Delta y \frac{\partial I}{\partial y} + \Delta t \frac{\partial I}{\partial t} = 0 \tag{15.5}$$

Dividing Equation (15.5) by an instant of time Δt, we have

$$\frac{\Delta x}{\Delta t} \frac{\partial I}{\partial x} + \frac{\Delta y}{\Delta t} \frac{\partial I}{\partial y} + \frac{\Delta t}{\Delta t} \frac{\partial I}{\partial t} = 0 \tag{15.6}$$

which in the infinitesimal case reduces to:

$$u \frac{\partial I}{\partial x} + v \frac{\partial I}{\partial x} + I_t = 0 \tag{15.7}$$

or

$$(\nabla I)^\top \boldsymbol{u} + I_t = 0 \tag{15.8}$$

where $\boldsymbol{u} = (u, v)^\top$ denotes the velocity.

Equation (15.8) is known as the Horn-Schunck (H-S) equation. The H-S equation holds for every pixel of an image. The two key entities in the H-S equation are the spatial gradient of the image, and the temporal change in the image. These can be calculated from the image, and are hence known. From these two vectors, we want to find the velocity vector which, when dotted with the gradient, is cancelled out by the temporal derivative. In this sense, the velocity vector "explains" the temporal difference measured in I_t in terms of the spatial gradient. Unfortunately this equation has two unknowns but we have only one equation per pixel. So we cannot solve the H-S equation uniquely at one pixel. This leads us to consider motion measurement inside small neighborhoods or *apertures*.

[1] One approximation for computing I_t is simply to compute the difference of successive frames: $I_t \approx I(\boldsymbol{x}, t) - I(\boldsymbol{x}, t-1)$.

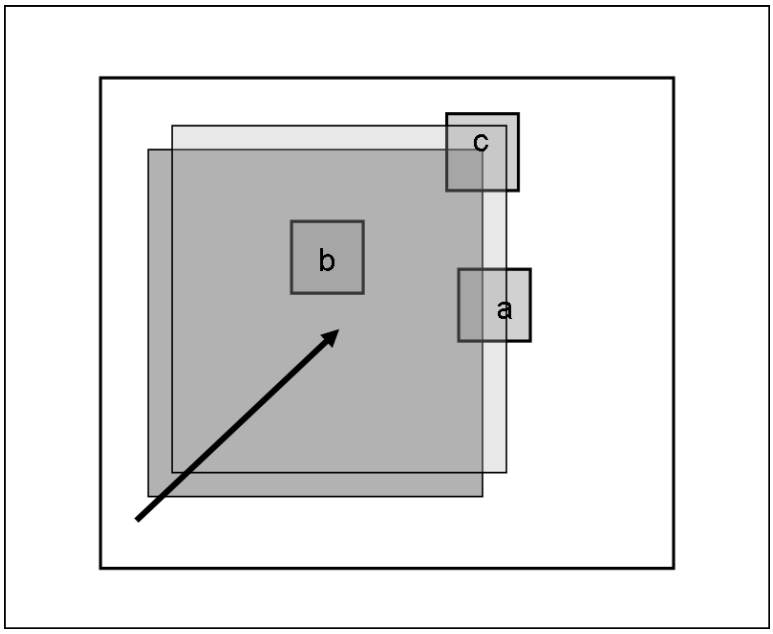

Figure 2. Different positions of an aperture in the image of a dark box over a light background. (a) We can recover only the horizontal motion. (b) No motion information can be recovered (c) Both components of the motion can be recovered.

15.4. Aperture Problem

Consider the black box on the white background in the figure (2). If we consider an aperture (small window) containing only an edge, we will get only one component of flow. Even if there is movement diagonally, only the horizontal component of the movement can be recovered. There is no way to recover any movement in the vertical direction. This holds vice versa for any aperture that contains only a horizontal edge. If the aperture is inside the box, we have the *blank wall problem*, and absolutely no information can be obtained regarding the flow. In the third case, if the aperture is at a corner, both components of the movement can be recovered.

The quantity we want to estimate is the velocity vector. But, as shown in Figure (3), if we look at the velocity space, for a given value of $\nabla I = (I_x, I_y)$, there are a family of points which when dotted with ∇I gives $-I_t$ as result. This family of points traces a line in the (u, v) velocity space. To get a solution, we have to consider information from neighboring pixels, each of which will give rise to new constraint lines.

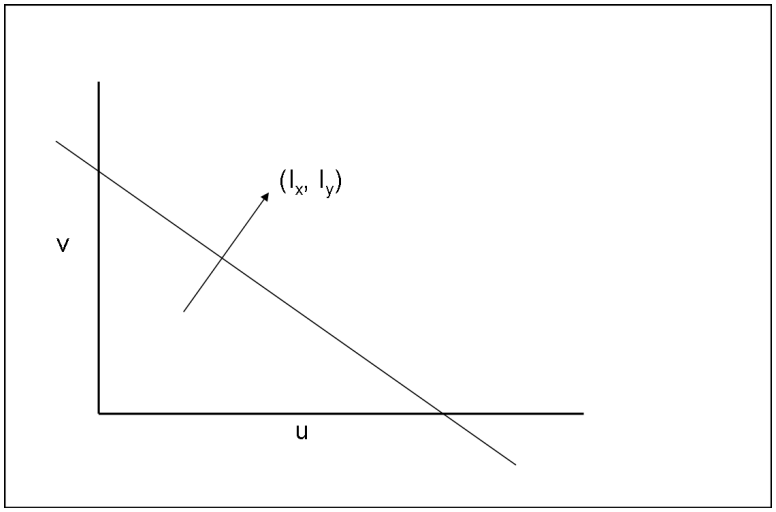

Figure 3. The velocity space and the locus of points which when dotted with ∇I gives $-I_t$ as result.

15.5. One Method of Solution

We will now consider a least squares solution proposed by Lucas and Kanade (1981) (L-K). They assume a translational model and solve for a single velocity vector $\boldsymbol{u}$ that approximately satisfies the H-S equation for all the pixels in a small neighborhood $\mathcal{N}$ of size $N \times N$. In this way, we obtain a highly overconstrained system of equations, where we only have 2 unknowns and N^2 equations.

Let $\mathcal{N}$ denote a $N \times N$ patch around a pixel $\boldsymbol{p}_i$. For each point $\boldsymbol{p}_i \in \mathcal{N}$, we can write:

$$\nabla I(\boldsymbol{p}_i)^\top \boldsymbol{u} + I_t(\boldsymbol{p}_i) = 0 \tag{15.9}$$

Thus we arrive at the over-constrained least squares problem, to find the $\boldsymbol{u}$ that minimizes $\Psi(\boldsymbol{u})$:

$$\Psi(\boldsymbol{u}) = \sum_{\boldsymbol{p}_i \in \mathcal{N}} [\nabla I(\boldsymbol{p}_i)^\top \boldsymbol{u} + I_t(\boldsymbol{p}_i)]^2 \tag{15.10}$$

Due to the presence of noise and other factors (e.g., not all pixels move with the same velocity), the residual will not in general be zero. The least squares solution will be the one which minimizes the residual. We can solve the least squares problem by solving the linear system $A^\top A\boldsymbol{u} = A^\top \boldsymbol{b}$, where

Figure 4. Three frames from the Woody Allen movie *Manhattan*. [from Shi & Tomasi]

Figure 5. The top row showing the speed limit sign board as tracked by the motion model, and the bottom row shows the same images warped by the computed deformation matrices. [from Shi & Tomasi]

$A \in \mathbb{R}^{N^2 \times 2}$ and $\boldsymbol{b} \in \mathbb{R}^{N^2}$ are given by:

$$A = \begin{bmatrix} \nabla I(\boldsymbol{p}_1)^\top \\ \nabla I(\boldsymbol{p}_2)^\top \\ \vdots \\ \nabla I(\boldsymbol{p}_{N^2})^\top \end{bmatrix} \tag{15.11}$$

$$\boldsymbol{b} = \begin{bmatrix} I_t(\boldsymbol{p}_1) \\ I_t(\boldsymbol{p}_2) \\ \vdots \\ I_t(\boldsymbol{p}_{N^2}) \end{bmatrix} \tag{15.12}$$

This is an inhomogeneous least squares problem, and the solution is given by $\boldsymbol{u} = (A^\top A)^{-1} A^\top \boldsymbol{b}$ or $\boldsymbol{u} = A^+ \boldsymbol{b}$, where A^+ is the pseudoinverse of A. The symmetric matrix $A^\top A \in \mathbb{R}^{2\times 2}$ is the second moment matrix, which we saw previously in Förstner corner detection. As with corner detection, the solution is unique when $A^\top A$ is rank 2.

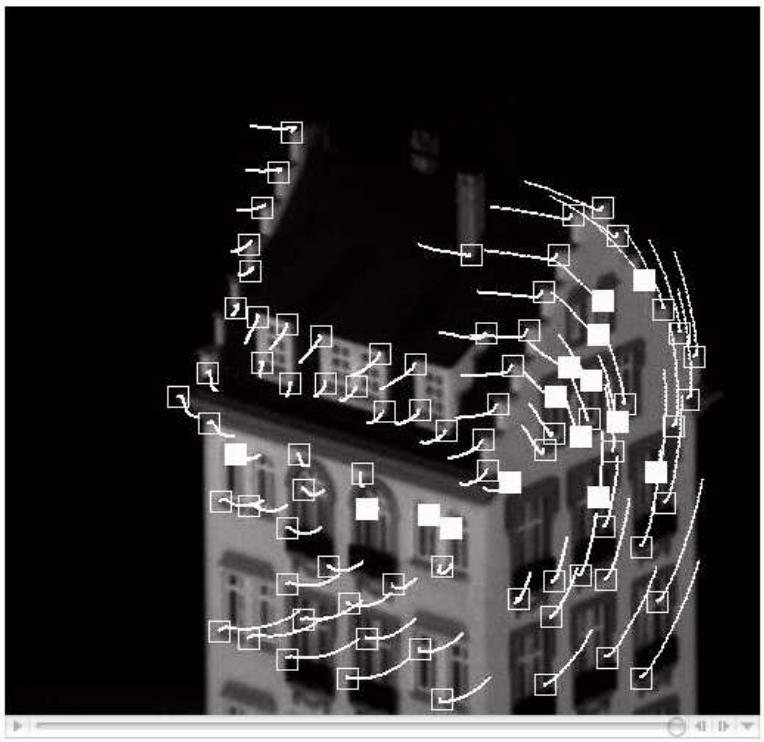

Figure 6. This photo shows the tracked movement of image patches of a building over a number of successive frames in a movie. The patches are tracked using an affine model. [Tommasini et al.]

15.6. Improvements Over the Translational Model: Affine Model

When the translational model that we have assumed so far is not sufficient then we need to use a more powerful motion model, like affine or planar homography. In practice, the affine model works pretty well if there is not too much projective distortion. In the affine case, the form of u and v is as follows:

$$u(x,y) = a_1 + a_2 x + a_3 y \tag{15.13}$$

$$v(x,y) = a_4 + a_5 x + a_6 y \tag{15.14}$$

Unlike the translational model, which would only include a_1 and a_4, we have 6 unknowns here. The rest of the derivation proceeds by plugging this form of u and v into the expression for Ψ and minimizing it w.r.t. the parameters $a_1, \ldots, a_6$. The problem can still be solved with a pseudoinverse, but the counterpart to the second moment matrix is bigger, and the aperture problem is more subtle.

15.7. Discussion

One question that came up is what should we use for the size of $\mathcal{N}$? It could be anywhere from one pixel to the whole image. As a rule of thumb, a lot of people use L-K with 5×5 or 7×7, but it's more appropriate to set this as a percentage of the image dimensions. The scale selection issues that arise in interest point detection also apply in feature tracking.

While it might seem strange to make $\mathcal{N}$ the whole image, this is exactly what is done in so-called "direct methods" for optical flow estimation. For example, in an aerial video sequence, the optical flow field for successive frames might have a good fit to an affine motion model, in which case the results of regressing on $a_1, \ldots, a_6$ would be better than letting L-K run in little local windows all across the image.