

CSci 4968 and 6270
Computational Vision
Lecture 18
Tracking and Motion Estimation

Charles Stewart
Department of Computer Science
Rensselaer Polytechnic Institute

2009/10/29

Overview

- Given are two images, I_0 and I_1 , taken in close proximity both spatially and temporally, and a region \mathcal{N} of image I_0 .
- The problem is to compute the motion of the pixels of \mathcal{N} from the coordinate system of I_0 to the coordinate system of I_1 .
- The mathematical model of the model of the motion could be as simple as translation. Affine models are commonly used. Higher order models are less common.
- The region may range from a small neighborhood (e.g. surrounding a feature) to the entire image.
- For image motion estimation we are likely to be interested in a regularly-sampled grid of regions.
- For tracking applications (and structure from motion), we are likely to choose a set of feature locations. The choice of these locations will be discussed later in these notes.
- The result is called the Lucas-Kanade or Lucas-Kanade-Tomasi (KLT) algorithm.

Small-Scale Translational Motion — The Simplest Case

- We will start with the simplest case where the motion model is just a translation vector $(t_x, t_y)^\top$. This translation, applied to each pixel in the region, is appropriate for small regions and for some types of tracking.
- The objective function is

$$E(t_x, t_y) = \sum_{(x,y) \in \mathcal{N}} w_{x,y} [I_1(x, y) - I_0(x + t_x, y + t_y)]^2 \quad (1)$$

where $w_{x,y}$ is a weight value, such as a Gaussian weight relative to the center of the window.

- Steps involved in the derivation:
 - First-order Taylor expansion of I_0 at (x, y) , with (t_x, t_y) playing the role of Δx and Δy . (This is why we are assuming small motion.)
 - * Result produces what is known as the “optic flow constraint equation” inside the parentheses.
 - Compute the derivative with respect to t_x and t_y .
 - Set the two resulting equations to 0 and solve for t_x and t_y .
 - Switching to vector notation, the result is

$$\mathbf{t} = \left(\sum_{\mathbf{x} \in \mathcal{N}} w_{\mathbf{x}} \nabla I_0(\mathbf{x})^\top \nabla I_0(\mathbf{x}) \right)^{-1} \left(\sum_{\mathbf{x} \in \mathcal{N}} w_{\mathbf{x}} \nabla I_0(\mathbf{x})^\top I_t(\mathbf{x}) \right) \quad (2)$$

where

- * $\nabla I_0(\mathbf{x}) = (I_x(\mathbf{x}), I_y(\mathbf{x}))$ is the intensity gradient at \mathbf{x} , and
- * $I_t(\mathbf{x})$ is the intensity derivative at \mathbf{x} .

Discussing the Result

- We will consider three main issues:
 1. What happens when the inverse involved in solving for \mathbf{t} is not invertible?
 2. What is the effect of linearization?
 3. How to handle higher-order motion models?

Studying the Inverse

- The matrix $\mathbf{M} = \sum_{\mathbf{x} \in \mathcal{N}} w_{\mathbf{x}} \nabla I_0(\mathbf{x}) \nabla I_0(\mathbf{x})^T$ is the summed, gradient outer product of the points in the neighborhood.
- An important concern is whether or not \mathbf{M} is invertible.
- Here are important cases where it is not:
 - If the intensity is constant, then the gradients will all be $\mathbf{0}^T$ and \mathbf{M} will be rank 0.
 - If the intensity is planar, then the gradients will coincide and \mathbf{M} will be rank 1.
 - If there is just a single linear step edge in \mathcal{N} then \mathbf{M} will also be rank 1.

All these are examples of what's known as the *aperture problem* in computer vision.

- For computing image motion, we can't get a true motion when \mathbf{M} is rank 1. Instead, we get the motion along the gradient direction, which is called “normal motion” or “normal flow”.
 - In this case, we use image smoothing operations to combine the estimate of motion within the image region with other nearby motion estimates. There is an entire, large literature on this topic.
 - In general we get only a normal motion when the smaller of the two eigenvalues of \mathbf{M} is too small.
- For tracking applications we can use \mathbf{M} , which can be quickly computed at each pixel in an image, to determine which locations are good for tracking (i.e. most easily matched in the next image):
 - Compute the eigenvalues of \mathbf{M} . Call them λ_1 and λ_2 and assume $\lambda_1 \geq \lambda_2$. By construction, $\lambda_2 \geq 0$.
 - A location has strong contrast if $\lambda_1 + \lambda_2 = \det \mathbf{M}$ is large.
 - A location has consistent variation in all directions if $\lambda_1/\lambda_2 \geq 1$ is relatively small.
 - Various combinations of these measures have been proposed in the vision and medical imaging research literatures.
 - We will discuss the relationship with SIFT features and edge effects in class.

Effect of Linearization

- The solution discussed so far is just an approximation: it is the solution **given** the Taylor expansion. It is not the solution to the original problem.
- To solve the original problem, we need to iterate starting from an initial translation vector \mathbf{t} ($\mathbf{0}$ or the previous estimate for the feature location):

1. Taylor expansion to linearize about $\mathbf{x} + \mathbf{t}$.
2. Solve for $\Delta\mathbf{t}$.
3. Assign $\mathbf{t} = \mathbf{t} + \Delta\mathbf{t}$.
4. Repeat until $\Delta\mathbf{t}$ is sufficiently small.

- The objective function before linearization is

$$E(\Delta\mathbf{t}) = \sum_{\mathbf{x} \in \mathcal{N}} w_{\mathbf{x}} [I_1(\mathbf{x}) - I_0(\mathbf{x} + \mathbf{t} + \Delta\mathbf{t})]^2 \quad (3)$$

- The update value $\Delta\mathbf{t}$ is

$$\Delta\mathbf{t} = \left(\sum_{\mathbf{x} \in \mathcal{N}} w_{\mathbf{x}} \nabla I_0(\mathbf{x} + \mathbf{t})^\top \nabla I_0(\mathbf{x} + \mathbf{t}) \right)^{-1} \left(\sum_{\mathbf{x} \in \mathcal{N}} w_{\mathbf{x}} \nabla I_0(\mathbf{x} + \mathbf{t})^\top (I_1(\mathbf{x}) - I_0(\mathbf{x} + \mathbf{t})) \right) \quad (4)$$

- Importantly, in each iteration,
 - The image gradient is being computed at location $\mathbf{x} + \mathbf{t}$, a new location (and potentially subpixel location) each time.
 - The intensity difference $I_1(\mathbf{x}) - I_0(\mathbf{x} + \mathbf{t})$ is also based on differences in changing image locations.

More General Motion Models

- Consider a more general motion $\mathbf{T}(\mathbf{x}; \mathbf{a})$ where \mathbf{a} is the vector of parameters in the motion model.

- For translation, \mathbf{a} is a two-component vector, and

$$\mathbf{T}(\mathbf{x}; \mathbf{a}) = \mathbf{x} + \mathbf{a} \quad (5)$$

- For affine motion, \mathbf{a} is a six-component vector, and

$$\mathbf{T}(\mathbf{x}; \mathbf{a}) = \begin{pmatrix} (1 + a_1) & a_2 & a_3 \\ a_4 & (1 + a_5) & a_6 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \quad (6)$$

Reasons for this new (to us) version of the affine model will be discussed in class

- The overall objective function becomes

$$E(\mathbf{a}) = \sum_{\mathbf{x} \in \mathcal{N}} w_{\mathbf{x}} [I_1(\mathbf{x}) - I_0(\mathbf{T}(\mathbf{x}; \mathbf{a}))]^2 \quad (7)$$

- The same iterative scheme as just discussed is applied. Now, the update objective function becomes

$$E(\Delta \mathbf{a}) = \sum_{\mathbf{x} \in \mathcal{N}} w_{\mathbf{x}} [I_1(\mathbf{x}) - I_0(\mathbf{T}(\mathbf{x}; \mathbf{a} + \Delta \mathbf{a}))]^2 \quad (8)$$

- We take a first-order Taylor expansion at \mathbf{a} to obtain a least-squares function of $\Delta \mathbf{a}$ and solve from there.
- If we have time in class, we will work through the derivation, a good exercise in the use of the chain rule.

Summary

- Region-based computation of low-order motion models based on differences and gradients of intensity
 - Gaussian smoothed images and even gradient magnitude images can be used in place of I_0 and I_1 .
- Incremental steps of linearization and estimation, returning to the images to recompute derivatives.
- More general, more efficient, and symmetric techniques exist in the literature.
- Used for image motion, optical flow, and tracking.