Tracking Feature Windows

COMPSCI 527 — Computer Vision
Outline

1. Local Motion Estimation
2. Window Tracking
3. The Lucas-Kanade Tracker
4. Good Features to Track
Motion Estimation

- Given: Two (black-and-white) images $f(x)$ and $g(x)$ of the same scene
- Assumption 1: Constant appearance
- Assumption 2: All displacements $d(x)$ between corresponding points are small (more on this later)
- Want: Displacements $d(x)$, wherever they can be computed (more on this later)
- Aperture problem: Image derivatives at $x$ only yield one scalar equation in the two unknowns in $d(x)$
- Therefore, we can only estimate several displacement vectors $d$ simultaneously, under an additional constraint that relates them
Local Motion Estimation

Local Estimation Methods

- Global methods estimate all displacements in an image
- They assume some version of smoothness of motion in space
- Local methods:
  - The image displacement $d$ in a small window around a pixel $x$ is assumed to be constant (extreme local smoothness)
  - Write one constancy-of-appearance equation for every pixel in the window
  - Solve for the one displacement that satisfies all these equations as much as possible (in the LSE sense)
- These techniques are called *(feature)* window tracking methods
Key Questions

• Which windows can be tracked?
• How can these windows be tracked?
• Logically, it’s best to answer the second question first
• The answer to the first question is then “wherever the tracking method works well”
• We’ll come back to the first question later
Window Tracking

• Given images $f(x)$ and $g(x)$, a point $x_f$ in image $f$, and a square window $W(x_f)$ of side-length $2h + 1$ centered at $x_f$, what are the coordinates $x_g = x_f + d^*(x_f)$ of the corresponding window’s center in image $g$?

• $d^*(x_f) \in \mathbb{R}^2$ is the displacement of that point feature

• Assumption 1: The whole window translates

• Assumption 2: $d^*(x_f) \ll h$
General Window Tracking Strategy

- Let \( w(x) \) be the indicator function of \( W(0) \)
- Measure the dissimilarity between \( W(x_f) \) and a candidate window in \( g \) with the loss

\[
L(x_f, d) = \sum_x [g(x + d) - f(x)]^2 w(x - x_f)
\]

- Minimize \( L(x_f, d) \) over \( d \): \( d^*(x_f) = \text{arg min}_{d \in R} L(x_f, d) \)
- The search range \( R \subseteq \mathbb{R}^2 \) is a square centered at the origin
- Half-side of \( R \) is \( \ll h \)
Obvious Failure Points

- Multiple motions in the same window
  
  (Less dramatic cases arise as well)

- Actual motion large compared with $h$
  
  (We’ll come back to this later)
A Softer Window

- Make $w(x)$ a (truncated) Gaussian rather than a box
  \[ w(x) \propto \begin{cases} 
  e^{\frac{1}{2}\left|\frac{x}{\sigma}\right|^2} & \text{if } |x_1| \leq h \text{ and } |x_2| \leq h \\
  0 & \text{otherwise} 
\end{cases} \]
- Dissimilarity $L(x_i, d) = \sum_x [g(x + d) - f(x)]^2 w(x - x_i)$
  depends more on what’s around the window center
- Reduces the effects of multiple motions
- Does not eliminate them
How to Minimize $L(x_f, d)$?

- Method 1: Exhaustive search over a grid of $d$
- Advantages: Unlikely to be trapped in local minima
  
  \[ L(d) \]

- Disadvantage: Fixed resolution
- Accurate motion is sometimes necessary
- Example: 3D reconstruction is ill-posed
- Small errors accumulate: *drift*
- Using a very fine grid would be very expensive
- Exhaustive search may provide a good initialization
How to Minimize $L(x_f, d)$?

- Method 2: Use a gradient-descent method
- Search space is small ($d \in \mathbb{R}^2$), so we can use Newton’s method for faster convergence
- Compute gradient and Hessian of $L(d) = \sum_x [g(x + d) - f(x)]^2 w(x - x_i)$ (omitted $x$ from arguments of $L$ for simplicity)
- Take Newton steps
- Technical difficulty: the unknown $d$ appears inside $g(x + d)$, and computing a Hessian would require computing second-order derivatives of an image, which is available only through its pixels
- Second derivatives of images are very sensitive to noise
The Lucas-Kanade Tracker, 1981

- Instead of computing the Hessian of 
  \[ L(d) = \sum_x [g(x + d) - f(x)]^2 w(x - x_i), \]
  linearize \( g(x + d) \approx g(x) + [\nabla g(x)]^T d \)
- This brings \( d \) “outside \( g \)
- \( L(d) \) is now quadratic in \( d \), and we can find a minimum in closed form
- The derivatives needed for that no longer go through \( g \), since \( d \) appears linearly
- Only differentiate the image once to get \( \nabla g(x) \)
- Since the solution relies on an approximation, we iterate
- This method works for losses that are sums of squares, and is called the *Newton-Raphson method*
Lucas-Kanade Derivation

• Let \( \mathbf{d}_t \) the solution at iteration \( t \). We seek \( \mathbf{d}_{t+1} = \mathbf{d}_t + \mathbf{s} \) by minimizing the following over \( \mathbf{s} \) (with \( \mathbf{d}_0 = \mathbf{0} \):

\[
L(\mathbf{d}_t + \mathbf{s}) = \sum_x [g(\mathbf{x} + \mathbf{d}_t + \mathbf{s}) - f(\mathbf{x})]^2 w(\mathbf{x} - \mathbf{x}_f)
\]

• For simplicity, define \( g_t(\mathbf{x}) \overset{\text{def}}{=} g(\mathbf{x} + \mathbf{d}_t + \mathbf{s}) \) so that

\[
g_t(\mathbf{x} + \mathbf{s}) \approx g_t(\mathbf{x}) + [\nabla g_t(\mathbf{x})]^{\mathbf{T}} \mathbf{s} \quad \text{(linearization)}
\]

a quadratic function of \( \mathbf{s} \)
Lucas-Kanade Derivation, Cont’d

- Gradient of
  \[ L(d_t + s) \approx \sum_x [g_t(x) + [\nabla g_t(x)]^T s - f(x)] w(x - x_f) \]
  \[ \nabla L(d_t + s) \approx \sum_x \nabla g_t(x) \{g_t(x) + [\nabla g_t(x)]^T s - f(x)\} w(x - x_f) \]

- Setting to zero yields

\[
\begin{aligned}
\sum_x \nabla g_t(x) [f(x) - g_t(x)] w(x - x_f) &= A \\
\sum_x \nabla g_t(x) \nabla g_t(x)^T w(x - x_f) &= B \\
\end{aligned}
\]
The Core System of Lucas-Kanade

Linear, $2 \times 2$ system

$$A\mathbf{s} = \mathbf{b}$$

where

$$A = \sum_x \nabla g_t(x) \left[ \nabla g_t(x) \right]^T w(x - x_f)$$

and

$$\mathbf{b} = \sum_x \nabla g_t(x) \left[ f(x) - g_t(x) \right] w(x - x_f).$$

- Iteration $t$ yields $\mathbf{s}_t$
- Image $g_t$ is shifted by $\mathbf{s}_t$ by subpixel interpolation
- This shift makes $f$ and $g_t$ more similar within the windows
- Repeat until convergence
- Overall displacement is $\mathbf{d}^* = \sum_t \mathbf{s}_t$
Good Features to Track

- How to select which windows to track?

- We keep solving \( As = b \) where

\[
A = \sum_x \nabla g_t(x)[\nabla g_t(x)]^T w(x - x_f)
\]

- So we want \( A \) to be far from degenerate

- That is, \( \lambda_{\text{min}}(A) \geq \lambda_0 \) for some \( \lambda_0 \)

- The matrix \( A \) keeps changing: \( g_t(x) \overset{\text{def}}{=} g(x + d_t + s) \)

- \( g_t \) is \( g \) shifted to match \( f \)

- So the \( A \) matrices are not far from

\[
A_f(x_f) \overset{\text{def}}{=} \sum_x \nabla f(x)[\nabla f(x)]^T w(x - x_f)
\]

- Pick \( x_f \) for tracking if \( A_f(x_f) \) is far from degenerate

- That is, \( \lambda_{\text{min}}(A_f(x_f)) \geq \lambda_0 \)
Examples of Degenerate $A_f(x_f)$

\[
\nabla f(x) = \begin{bmatrix} g_1(x) \\ 0 \end{bmatrix} \quad \Rightarrow \quad \nabla f(x)[\nabla f(x)]^T = \begin{bmatrix} \gamma_{11}(x) & 0 \\ 0 & 0 \end{bmatrix}
\]

\[
A_f(x_f) = \sum_x \nabla f(x)[\nabla f(x)]^T w(x - x_f) = \begin{bmatrix} a_{11}(x_f) & 0 \\ 0 & 0 \end{bmatrix}
\]
Feature Selection Algorithm

- Compute $\lambda_{\min}(A_f(x))$ for all $x$
  (eigenvalues of $2 \times 2$ matrix can be computed in closed form)
- While $\max \lambda_{\min} > \lambda_0$ and want more points:
  - Pick $\hat{x}$ with highest $\lambda_{\min}$
  - Remove $x$ whose windows contain $\hat{x}$
What if Motion is Large?

Diagram showing feature tracking with arrows indicating motion.