motion and optical flow

Jason Lawrence (some slides by Szymon Rusinkiewicz)
CS 651, Spring 2007: Computer Vision
moving to multiple images

- until now: processing single images
- multiple images
  - stereo-based scene reconstruction
  - video
  - multiple cameras at multiple times...
what causes changes in images?

- moving / deforming objects
- illumination / reflectance
- camera sensitivity (?)
why is this important?

- very rich cue about scene structure
- object avoidance / detection
- material / illumination properties
related applications

- **2D**
  - feature / object tracking
  - segmentation based on motion
- **3D**
  - shape extraction
  - motion capture
applications in signal processing and computer graphics

- image panoramas
- automatic image morphing
- reconstruction of 3D models for rendering
- capturing articulated motion for realistic animation
- video compression ("motion estimation")
key problem

- everything comes down to finding correspondences

[Tomasi and Kanade]
correspondence

- small displacements
  - differential algorithms
  - gradients in space / time
  - dense correspondence estimates
  - most common with video
- large displacements
  - matching algorithms
  - sparse correspondence estimates
  - most common with multiple cameras / stereo
output of correspondence algorithms

- for each point in image $i$, displacements to corresponding locations in image $j$
  - in video: “motion field”
  - in stereo: “disparity”

[Tomasi and Kanade]
computing motion field

- **basic idea:** a small portion of the image ("local neighborhood") shifts position

- **assumptions**
  - no / small changes in reflected light
  - no / small changes in scale
  - no occlusion or disocclusion
  - neighborhood is correct size (aperture problem)
actual vs. apparent motion

- if assumptions violated, can still use the same methods: apparent motion
- result: optical flow (vs. ideal motion field)
- most obvious effects:
  - aperture problem: can only obtain motion perpendicular to edges
  - errors near occlusions
Aperture problem

- too big: confused by multiple motions
- too small: only get motion perpendicular to edge
computing optical flow: preliminaries

- image sequence $I(x,y,t)$
- uniform discretization along $x,y,t$
  (think “cube” of data)
- differential framework: compute partial derivatives along $x,y,t$ by convolving with derivative of Gaussian (isolate “edges” in both space and time)
computing optical flow: preliminaries

- dense motion field:

\[ I(x, y, t_i) \]

\[ v_i(x, y) \]

\[ I(x, y, t_{i+1}) \]
computing optical flow: image brightness constancy

- basic idea: a small portion of the image ("local neighborhood") shifts position
- brightness constancy assumption

\[
\frac{dI}{dt} = 0
\]
computing optical flow: image brightness constancy

✦ this does not say the image remains the same brightness!
✦ total vs. partial derivative
✦ apply chain rule:

\[
\frac{dI(x(t), y(t), t)}{dt} = \frac{\partial I}{\partial x} \frac{dx}{dt} + \frac{\partial I}{\partial y} \frac{dy}{dt} + \frac{\partial I}{\partial t}
\]
computing optical flow: image brightness constancy

• given optical flow: \( \mathbf{v}(x, y) = \left( \frac{dx}{dt}, \frac{dy}{dt} \right) \)

\[
\frac{dI(x(t), y(t), t)}{dt} = 0
\]
\[
\frac{\partial I}{\partial x} \frac{dx}{dt} + \frac{\partial I}{\partial y} \frac{dy}{dt} + \frac{\partial I}{\partial t} = 0
\]
\[
(\nabla I)^T \mathbf{v} + I_t = 0
\]
computing optical flow: discretization

- look at some neighborhood $N$:

$$\sum_{(i,j) \in N} (\nabla I(i,j))^T v + I_t(i,j) = 0$$

$$A v + b = 0$$

Assumption:

$$A = \begin{bmatrix}
\nabla I(i_1,j_1) \\
\nabla I(i_1,j_2) \\
\nabla I(i_1,j_3) \\
\cdot \\
\cdot \\
\cdot \\
\nabla I(i_n,j_n)
\end{bmatrix}$$

$$b = \begin{bmatrix}
I_t(i_1,j_1) \\
I_t(i_1,j_2) \\
I_t(i_1,j_3) \\
\cdot \\
\cdot \\
\cdot \\
I_t(i_n,j_n)
\end{bmatrix}$$
computing optical flow: least squares

- overconstrained linear system
- solve by least squares

\[ A \mathbf{v} + \mathbf{b} = 0 \]

\[ A^T A \mathbf{v} = -A^T \mathbf{b} \]

\[ \mathbf{v} = -\left( A^T A \right)^{-1} A^T \mathbf{b} \]
computing optical flow: stability

- has solution unless $C = A^T A$ is singular

$$C = [\nabla I(i_1, j_1) \ldots \nabla I(i_n, j_n)]$$

$$C = \begin{bmatrix}
\sum_n I_x^2 & \sum_n I_x I_y \\
\sum_n I_x I_y & \sum_n I_y^2
\end{bmatrix}$$

where have we seen this before?
computing optical flow: stability

- corner detector!
- C is singular if uniform or edge
- use eigenvalues of C:
  - evaluate stability of optical flow computation
  - find good places to compute optical flow (finding good features to track)
  - [Shi-Tomasi] “Good Features to Track”
computing optical flow: improvements

✦ assumption that optical flow is constant over neighborhood not always good

✦ decreasing size of neighborhood => C more likely to be singular

✦ alternative: weighted least-squares
  - points near center = higher weight
  - still use larger neighborhood
computing optical flow: weighted least squares

- let \( W \) be a matrix of weights

\[
A \rightarrow WA \\
b \rightarrow Wb
\]

\[
v = -(A^T A)^{-1} A^T b
\]

\[
v_w = -(A^T W^2 A)^{-1} A^T W^2 b
\]
computing optical flow: improvements

• what if windows are still bigger
• adjust motion model: no longer expect lateral translation (constant within window)
• popular choice: affine model

[Shi and Tomasi]
computing optical flow: affine motion model

- translational model:

\[
\begin{bmatrix}
  x_2 \\
  y_2
\end{bmatrix}
= \begin{bmatrix}
  x_1 \\
  y_1
\end{bmatrix} + \begin{bmatrix}
  t_x \\
  t_y
\end{bmatrix}
\]

- affine model:

\[
\begin{bmatrix}
  x_2 \\
  y_2
\end{bmatrix}
= \begin{bmatrix}
  a & b \\
  c & d
\end{bmatrix} \begin{bmatrix}
  x_1 \\
  y_1
\end{bmatrix} + \begin{bmatrix}
  t_x \\
  t_y
\end{bmatrix}
\]

- solved as before, but 6 unknowns instead of 2
computing optical flow: improvements

- iterative refinement
  - early iterations: use larger window (Gaussian) to allow more motion
  - late iterations: use less blur to find exact solution, lock on to detail
  - analogous to Newton’s method
Lucas-Kanade method

iterate:

1. set $\sigma = \text{large (e.g., 3 pixels)}$

2. set $I' \leftarrow I_1$

3. set $v \leftarrow 0$

4. repeat while $SSD(I', I_2) > \tau$

   1. $v + = \text{optical flow}(I' \rightarrow I_2)$
   2. $I' \leftarrow \text{warp}(I_1, v)$

5. after $n$ iterations, set $\sigma = \text{small (e.g., 1.5 pixels)}$
Lucas-Kanade method

- I’ always holds warped version of
  - best estimate of
- gradually reduce threshold
- stop when difference between I’ and I2 small
- simplest difference metric = sum of squared differences (SSD)
image warping

* given a coordinate transform $x' = h(x)$ and a source image $f(x)$, how do we compute a warped image $g(x') = f(h(x))$:
Forward warping

- send each pixel $f(x)$ to its corresponding location $x' = h(x)$ in $g(x')$
- what if pixel lands “between” two pixels?

[Szeliski]
forward warping

- send each pixel $f(x)$ to its corresponding location $x' = h(x)$ in $g(x')$
- what if pixel lands “between” two pixels?
- answer: add “contribution” to several pixels, normalize later (splatting)
reverse warping

✦ get each pixel $g(x')$ from its corresponding location $x = \text{inv}(h)(x')$ in $f(x)$

✦ what if pixel comes from “between” two pixels?
reverse warping

- get each pixel $g(x')$ from its corresponding location $x={}^\text{inv}(h)(x')$ in $f(x)$
- what if pixel comes from “between” two pixels?
- answer: resample color value from interpolated (prefiltered) source image
optical flow applications

video frames

[Feng and Perona]
optical flow applications

[Image: Diagram showing optical flow and depth reconstruction.

- Optical flow: A series of arrows indicating the direction and magnitude of movement.
- Depth reconstruction: A scatter plot showing the X and Z coordinates in space.

[Feng and Perona]
optical flow applications

obstacle detection: unbalanced optical flow
optical flow applications

obstacle avoidance:
keep optical flow balanced
optical flow applications

obstacle avoidance:
keep optical flow balanced
video matching
Video Matching