Motion Estimation

By Colin
Slides courtesy to Steve Seitz

1

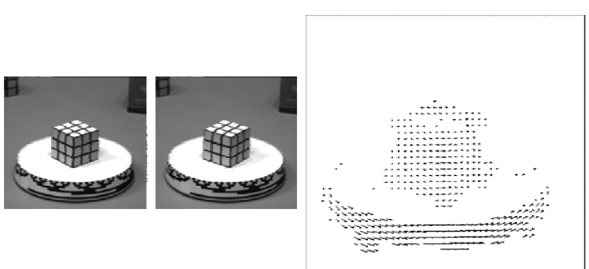
Why estimate motion?

We live in a 4-D world

Wide applications
- Object Tracking
- Camera Stabilization
- Image Mosaics
- 3D Shape Reconstruction (SFM)
- Special Effects (Match Move)

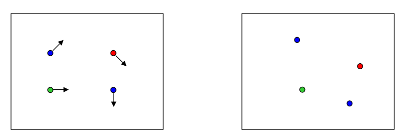
2

Optical flow



3

Problem definition: optical flow



$H(x, y)$ $I(x, y)$

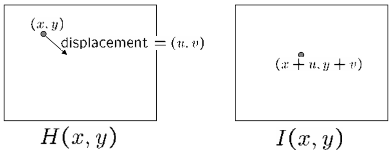
How to estimate pixel motion from image H to image I?
- Solve pixel correspondence problem
  - given a pixel in H, look for nearby pixels of the same color in I

Key assumptions
- **color constancy:** a point in H looks the same in I
  - For grayscale images, this is **brightness constancy**
- **small motion**: points do not move very far

This is called the **optical flow** problem

4

Optical flow constraints (grayscale images)



Let's look at these constraints more closely

- brightness constancy: Q: what's the equation?

  H(x, y) = I(x+u, y+v)
- small motion: (u and v are less than 1 pixel)
  - suppose we take the Taylor series expansion of I:

$$I(x+u, y+v) = I(x, y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \text{higher order terms}$$
$$\approx I(x, y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v$$

5

Optical flow equation

Combining these two equations

shorthand: $I_x = \frac{\partial I}{\partial x}$

$$\begin{aligned} 0 &= I(x+u, y+v) - H(x, y) \\ &\approx I(x, y) + I_x u + I_y v - H(x, y) \\ &\approx (I(x, y) - H(x, y)) + I_x u + I_y v \\ &\approx I_t + I_x u + I_y v \\ &\approx I_t + \nabla I \cdot [u \;\; v] \end{aligned}$$

In the limit as u and v go to zero, this becomes exact

$$0 = I_t + \nabla I \cdot [\tfrac{\partial x}{\partial t} \;\; \tfrac{\partial y}{\partial t}]$$

6

Optical flow equation

$$0 = I_t + \nabla I \cdot [u \;\; v]$$

Q: how many unknowns and equations per pixel?

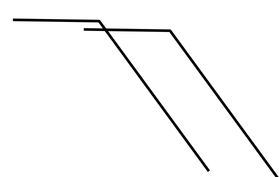
Intuitively, what does this constraint mean?

- The component of the flow in the gradient direction is determined
- The component of the flow parallel to an edge is unknown

This explains the Barber Pole illusion
http://www.sandlotscience.com/Ambiguous/barberpole.htm

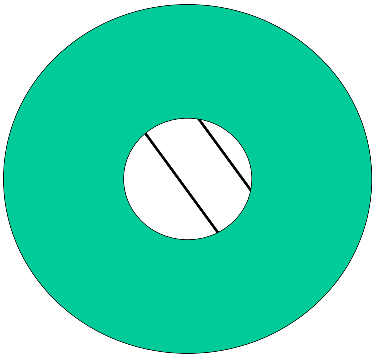
7

Aperture problem



8

Aperture problem



Solving the aperture problem

Basic idea: assume motion field is smooth

Horn & Schunk: add smoothness term

$$\int\int (I_t + \nabla I \cdot [u\ v])^2 + \lambda^2 (\|\nabla u\|^2 + \|\nabla v\|^2)\ dx\ dy$$

Lukas & Kanade: assume locally constant motion

- pretend the pixel's neighbors have the same (u,v)
  - If we use a 5x5 window, that gives us 25 equations per pixel!

$$0 = I_t(\mathrm{p_i}) + \nabla I(\mathrm{p_i}) \cdot [u\ v]$$

- works better in practice than Horn & Schunk

Many other methods exist. Here's an overview:

- Barron, J.L., Fleet, D.J., and Beauchemin, S, Performance of optical flow techniques, *International Journal of Computer Vision*, 12(1):43-77, 1994.

Lukas-Kanade flow

How to get more equations for a pixel?

- Basic idea: impose additional constraints
  - most common is to assume that the flow field is smooth locally
  - one method: pretend the pixel's neighbors have the same (u,v)
    - » If we use a 5x5 window, that gives us 25 equations per pixel!

$$0 = I_t(\mathrm{p_i}) + \nabla I(\mathrm{p_i}) \cdot [u\ v]$$

$$\underset{\substack{A \\ 25\times 2}}{\begin{bmatrix} I_x(\mathbf{p_1}) & I_y(\mathbf{p_1}) \\ I_x(\mathbf{p_2}) & I_y(\mathbf{p_2}) \\ \vdots & \vdots \\ I_x(\mathbf{p_{25}}) & I_y(\mathbf{p_{25}}) \end{bmatrix}} \underset{\substack{d \\ 2\times 1}}{\begin{bmatrix} u \\ v \end{bmatrix}} = - \underset{\substack{b \\ 25\times 1}}{\begin{bmatrix} I_t(\mathbf{p_1}) \\ I_t(\mathbf{p_2}) \\ \vdots \\ I_t(\mathbf{p_{25}}) \end{bmatrix}}$$

RGB version

How to get more equations for a pixel?

- Basic idea: impose additional constraints
  - most common is to assume that the flow field is smooth locally
  - one method: pretend the pixel's neighbors have the same (u,v)
    - » If we use a 5x5 window, that gives us 25*3 equations per pixel!

$$0 = I_t(\mathrm{p_i})[0,1,2] + \nabla I(\mathrm{p_i})[0,1,2] \cdot [u\ v]$$

$$\underset{\substack{A \\ 75\times 2}}{\begin{bmatrix} I_x(\mathbf{p_1})[0] & I_y(\mathbf{p_1})[0] \\ I_x(\mathbf{p_1})[1] & I_y(\mathbf{p_1})[1] \\ I_x(\mathbf{p_1})[2] & I_y(\mathbf{p_1})[2] \\ \vdots & \vdots \\ I_x(\mathbf{p_{25}})[0] & I_y(\mathbf{p_{25}})[0] \\ I_x(\mathbf{p_{25}})[1] & I_y(\mathbf{p_{25}})[1] \\ I_x(\mathbf{p_{25}})[2] & I_y(\mathbf{p_{25}})[2] \end{bmatrix}} \underset{\substack{d \\ 2\times 1}}{\begin{bmatrix} u \\ v \end{bmatrix}} = - \underset{\substack{b \\ 75\times 1}}{\begin{bmatrix} I_t(\mathbf{p_1})[0] \\ I_t(\mathbf{p_1})[1] \\ I_t(\mathbf{p_1})[2] \\ \vdots \\ I_t(\mathbf{p_{25}})[0] \\ I_t(\mathbf{p_{25}})[1] \\ I_t(\mathbf{p_{25}})[2] \end{bmatrix}}$$

Lukas-Kanade flow

Prob: we have more equations than unknowns

$$\underset{25\times2}{A}\ \underset{2\times1}{d} = \underset{25\times1}{b} \longrightarrow \text{minimize } \|Ad - b\|^2$$

Solution: solve least squares problem

- minimum least squares solution given by solution (in d) of:

$$\underset{2\times2}{(A^T A)}\ \underset{2\times1}{d} = \underset{2\times1}{A^T b}$$

$$\underbrace{\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix}}_{A^T A} \begin{bmatrix} u \\ v \end{bmatrix} = - \underbrace{\begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}}_{A^T b}$$

- The summations are over all pixels in the K x K window
- This technique was first proposed by Lukas & Kanade (1981)

13

Conditions for solvability

- Optimal (u, v) satisfies Lucas-Kanade equation

$$\underbrace{\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix}}_{A^T A} \begin{bmatrix} u \\ v \end{bmatrix} = - \underbrace{\begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}}_{A^T b}$$

When is This Solvable?

- **$A^T A$** should be invertible
- **$A^T A$** should not be too small due to noise
  - eigenvalues λ_1 and λ_2 of **$A^T A$** should not be too small
- **$A^T A$** should be well-conditioned
  - λ_1 / λ_2 should not be too large (λ_1 = larger eigenvalue)

14

Eigenvectors of $A^T A$

$$A^T A = \begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} = \sum \begin{bmatrix} I_x \\ I_y \end{bmatrix} [I_x\ I_y] = \sum \nabla I (\nabla I)^T$$

Suppose (x,y) is on an edge. What is $A^T A$? derive on board

- gradients along edge all point the same direction
- gradients away from edge have small magnitude

$$\left(\sum \nabla I (\nabla I)^T\right) \approx k \nabla I \nabla I^T$$

$$\left(\sum \nabla I (\nabla I)^T\right) \nabla I = k \|\nabla I\|^2 \nabla I$$

- ∇I is an eigenvector with eigenvalue $k\|\nabla I\|^2$
- What's the other eigenvector of $A^T A$?
  - let N be perpendicular to ∇I

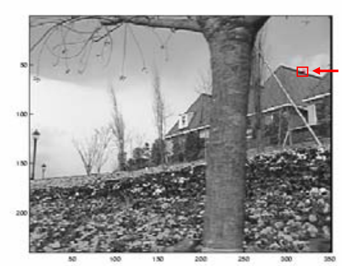
$$\left(\sum \nabla I (\nabla I)^T\right) N = 0$$

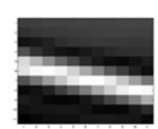
  - N is the second eigenvector with eigenvalue 0

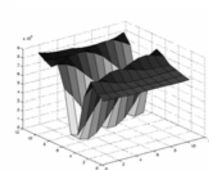
The eigenvectors of $A^T A$ relate to edge direction and magnitude

15

Edge





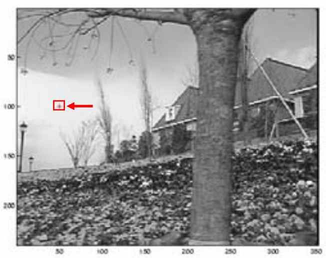


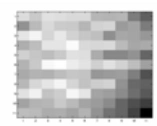
$\sum \nabla I (\nabla I)^T$

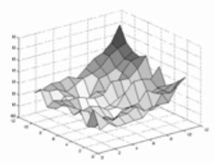
- large gradients, all the same
- large λ_1, small λ_2

16

Low texture region







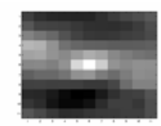
$\sum \nabla I (\nabla I)^T$

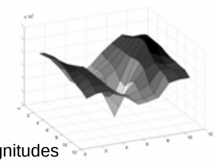
– gradients have small magnitude

– small λ_1, small λ_2

17

High textured region





$\sum \nabla I (\nabla I)^T$

– gradients are different, large magnitudes

– large λ_1, large λ_2

18

Observation

This is a two image problem BUT

- Can measure sensitivity by just looking at one of the images!
- This tells us which pixels are easy to track, which are hard
  - very useful later on when we do feature tracking...

19

Errors in Lukas-Kanade

What are the potential causes of errors in this procedure?

- Suppose A^TA is easily invertible
- Suppose there is not much noise in the image

When our assumptions are violated

- Brightness constancy is **not** satisfied
- The motion is **not** small
- A point does **not** move like its neighbors
  - window size is too large
  - what is the ideal window size?

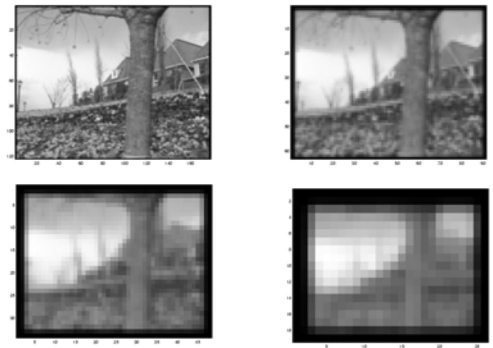
20

Revisiting the small motion assumption

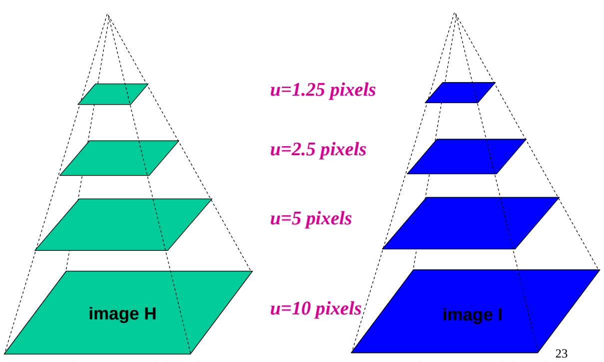
Is this motion small enough?

- Probably not—it's much larger than one pixel (2^{nd} order terms dominate)
- How might we solve this problem?

Reduce the resolution!

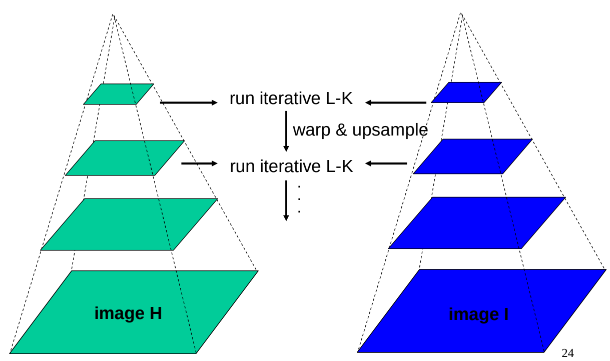


Coarse-to-fine optical flow estimation



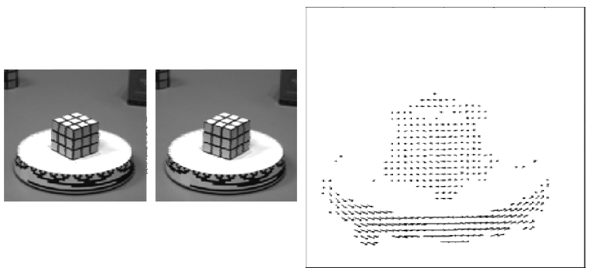
Gaussian pyramid of image H **Gaussian pyramid of image I**

Coarse-to-fine optical flow estimation



Gaussian pyramid of image H **Gaussian pyramid of image I**

Optical flow result



25

Motion tracking

Suppose we have more than two images

- How to track a point through all of the images?
  - In principle, we could estimate motion between each pair of consecutive frames
  - Given point in first frame, follow arrows to trace out it's path
  - Problem: DRIFT
    - » small errors will tend to grow and grow over time—the point will drift way off course

Feature Tracking

- Choose only the points ("features") that are easily tracked
- How to find these features?
  - windows where $\sum \nabla I (\nabla I)^T$ has two large eigenvalues
- Called the Harris Corner Detector

26

Feature Detection

27

Tracking features

Feature tracking

- Compute optical flow for that feature for each consecutive H, I

When will this go wrong?

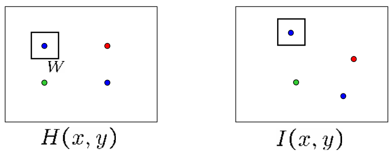
- Occlusions—feature may disappear
  - need mechanism for deleting, adding new features
- Changes in shape, orientation
  - allow the feature to deform
- Changes in color
- Large motions
  - will pyramid techniques work for feature tracking?

28

Handling large motions

L-K requires small motion

- If the motion is much more than a pixel, use discrete **search** instead



- Given feature window W in H, find best matching window in I
- Minimize sum squared difference (SSD) of pixels in window

$$min_{(u,v)} \left\{ \sum_{(x,y) \in W} |I(x+u, y+v) - H(x,y)|^2 \right\}$$

- Solve by doing a search over a specified range of (u,v) values
  - this (u,v) range defines the **search window**

29

Tracking Over Many Frames

Feature tracking with m frames

1. Select features in first frame
2. Given feature in frame i, compute position in i+1
3. Select more features if needed
4. i = i + 1
5. If i < m, go to step 2

Issues

- Discrete search vs. Lucas Kanade?
  - depends on expected magnitude of motion
  - discrete search is more flexible
- Compare feature in frame i to i+1 or frame 1 to i+1?
  - affects tendency to drift..
- How big should search window be?
  - too small: lost features. Too large: slow

30