

## Motion Estimation

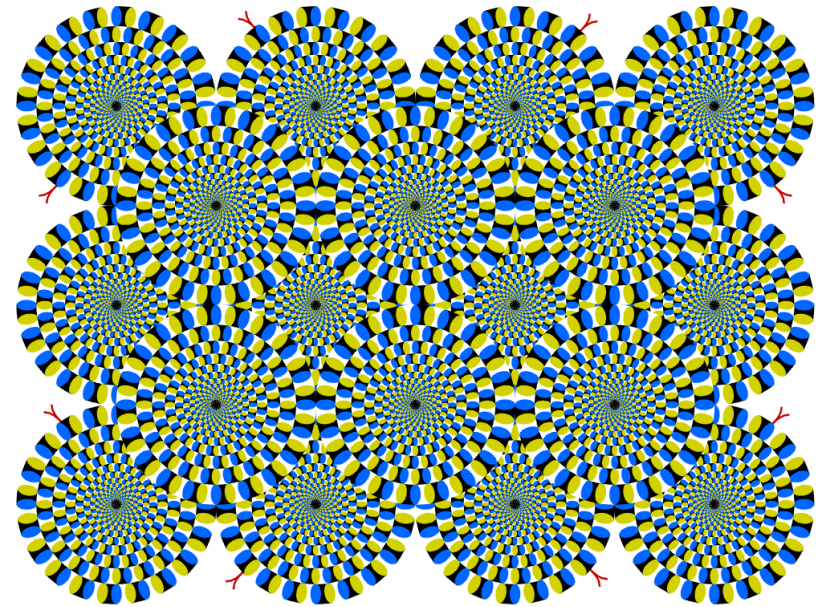
---

[http://www.sandlotscience.com/Distortions/Breathing\\_Square.htm](http://www.sandlotscience.com/Distortions/Breathing_Square.htm)

[http://www.sandlotscience.com/Ambiguous/Barberpole\\_Illusion.htm](http://www.sandlotscience.com/Ambiguous/Barberpole_Illusion.htm)

### Today's Readings

- Trucco & Verri, 8.3 – 8.4 (skip 8.3.3, read only top half of p. 199)
- [Newton's method Wikipedia page](#)



Copyright A.Kitaoka 2003

## Why estimate motion?

---

### Lots of uses

- Track object behavior
- Correct for camera jitter (stabilization)
- Align images (mosaics)
- 3D shape reconstruction
- Special effects
- [Video slow motion](#)
- [Video super-resolution](#)

## Motion estimation

---

Input: sequence of images

Output: point correspondence

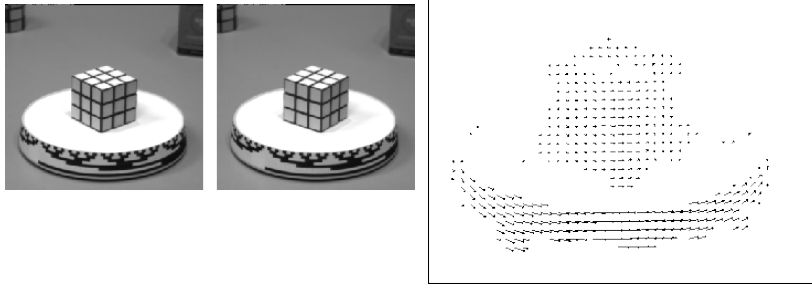
### Feature tracking

- we've seen this already (e.g., SIFT)
- can modify this to be more efficient

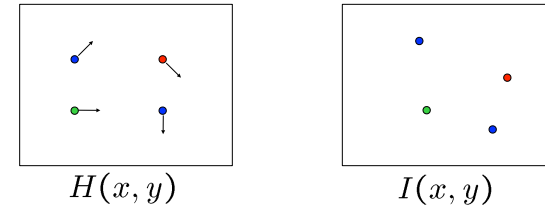
### Pixel tracking: "Optical Flow"

- today's lecture

## Optical flow



## Problem definition: optical flow



How to estimate pixel motion from image H to image I?

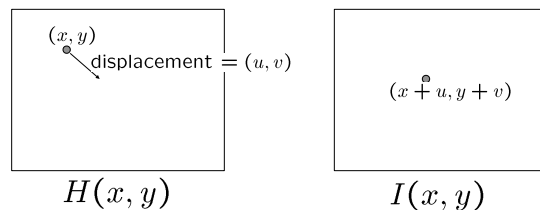
- Solve pixel correspondence problem
  - given a pixel in H, look for **nearby** pixels of the **same color** in I

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

## Optical flow constraints (grayscale images)



Let's look at these constraints more closely

- brightness constancy: Q: what's the equation?
- small motion: (u and v are less than 1 pixel)
  - suppose we take the Taylor series expansion of I:

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

$$\approx I(x, y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v$$

## Optical flow equation

Combining these two equations

$$0 = I(x+u, y+v) - H(x, y) \quad \text{shorthand: } I_x = \frac{\partial I}{\partial x}$$

$$\approx I(x, y) + I_x u + I_y v - H(x, y)$$

$$\approx (I(x, y) - H(x, y)) + I_x u + I_y v$$

$$\approx I_t + I_x u + I_y v$$

$$\approx I_t + \nabla I \cdot [u \ v]$$

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

$$0 = I_t + \nabla I \cdot \left[ \frac{\partial x}{\partial t} \ \frac{\partial y}{\partial t} \right]$$

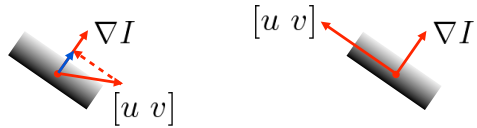
## Optical flow equation

---

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

Q: how many unknowns and equations per pixel?

Intuitively, what does this constraint mean?



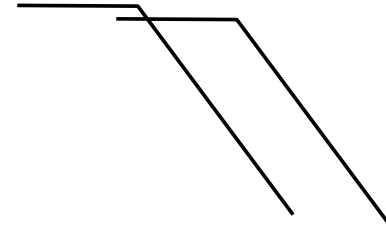
- The component of the flow in the gradient direction is determined
- The component of the flow parallel to an edge is unknown

This explains the Barber Pole illusion

[http://www.sandlotscience.com/Ambiguous/Barberpole\\_Illusion.htm](http://www.sandlotscience.com/Ambiguous/Barberpole_Illusion.htm)

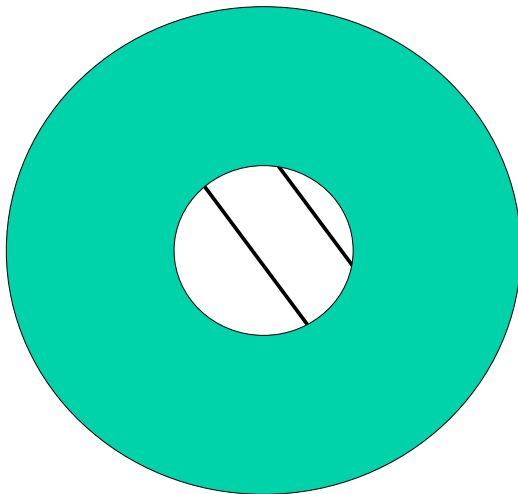
## Aperture problem

---



## Aperture problem

---



## Solving the aperture problem

---

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!

$$0 = I_t(\mathbf{p}_i) + \nabla I(\mathbf{p}_i) \cdot [u \ v]$$

$$\begin{bmatrix} I_x(\mathbf{p}_1) & I_y(\mathbf{p}_1) \\ I_x(\mathbf{p}_2) & I_y(\mathbf{p}_2) \\ \vdots & \vdots \\ I_x(\mathbf{p}_{25}) & I_y(\mathbf{p}_{25}) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(\mathbf{p}_1) \\ I_t(\mathbf{p}_2) \\ \vdots \\ I_t(\mathbf{p}_{25}) \end{bmatrix}$$

$\underset{25 \times 2}{A} \quad \underset{2 \times 1}{d} \quad \underset{25 \times 1}{b}$

## Lucas-Kanade flow

---

Prob: we have more equations than unknowns

$$\begin{matrix} A & d = b \\ 25 \times 2 & 2 \times 1 & 25 \times 1 \end{matrix} \longrightarrow \text{minimize } \|Ad - b\|^2$$

Solution: solve least squares problem

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

$$\begin{matrix} (A^T A) & d = A^T b \\ 2 \times 2 & 2 \times 1 & 2 \times 1 \end{matrix}$$

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

- The summations are over all pixels in the K x K window
- This technique was first proposed by Lucas & Kanade (1981)
  - described in Trucco & Verri reading

## Conditions for solvability

---

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

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

### When is This Solvable?

- $A^T A$  should be invertible
- $A^T A$  should not be too small due to noise
  - eigenvalues  $\lambda_1$  and  $\lambda_2$  of  $A^T A$  should not be too small
- $A^T A$  should be well-conditioned
  - $\lambda_1 / \lambda_2$  should not be too large ( $\lambda_1 =$  larger eigenvalue)

### Does this look familiar?

- $A^T A$  is the Harris matrix

## 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...

## Errors in Lucas-Kanade

---

What are the potential causes of errors in this procedure?

- Suppose  $A^T A$  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?

## Improving accuracy

---

Recall our small motion assumption

$$0 = I(x + u, y + v) - H(x, y)$$
$$\approx I(x, y) + I_x u + I_y v - H(x, y)$$

This is not exact

- To do better, we need to add higher order terms back in:

$$= I(x, y) + I_x u + I_y v + \text{higher order terms} - H(x, y)$$

This is a polynomial root finding problem

- Can solve using **Newton's method** 1D case  
on board
  - Also known as **Newton-Raphson** method
  - Today's reading (first four pages)
    - » <http://www.library.cornell.edu/nr/bookcpdf/c9-4.pdf>
- Approach so far does one iteration of Newton's method
  - Better results are obtained via more iterations

## Iterative Refinement

---

Iterative Lucas-Kanade Algorithm

1. Estimate velocity at each pixel by solving Lucas-Kanade equations
2. Warp H towards I using the estimated flow field
  - use image warping techniques
3. Repeat until convergence

## Revisiting the small motion assumption

---

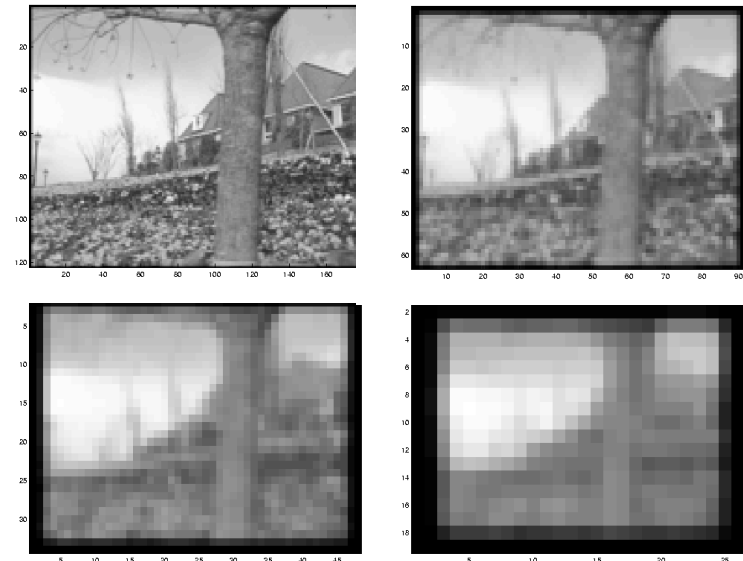


Is this motion small enough?

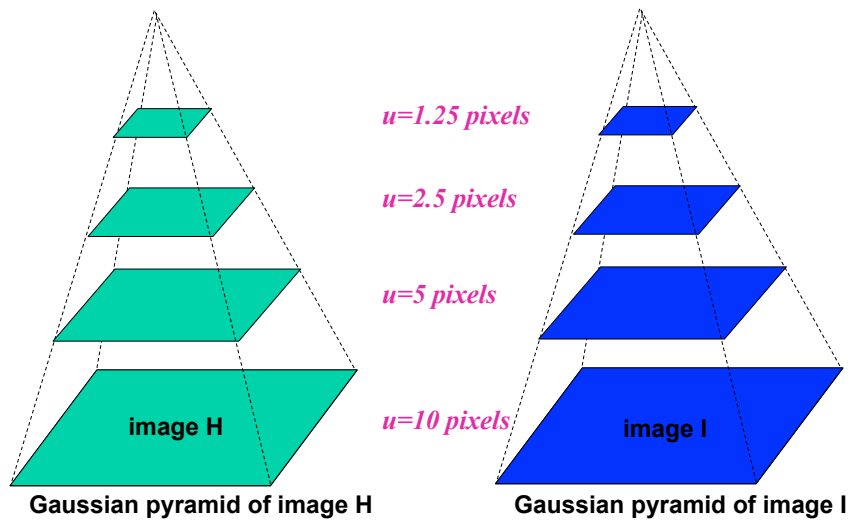
- Probably not—it's much larger than one pixel (2<sup>nd</sup> order terms dominate)
- How might we solve this problem?

## Reduce the resolution!

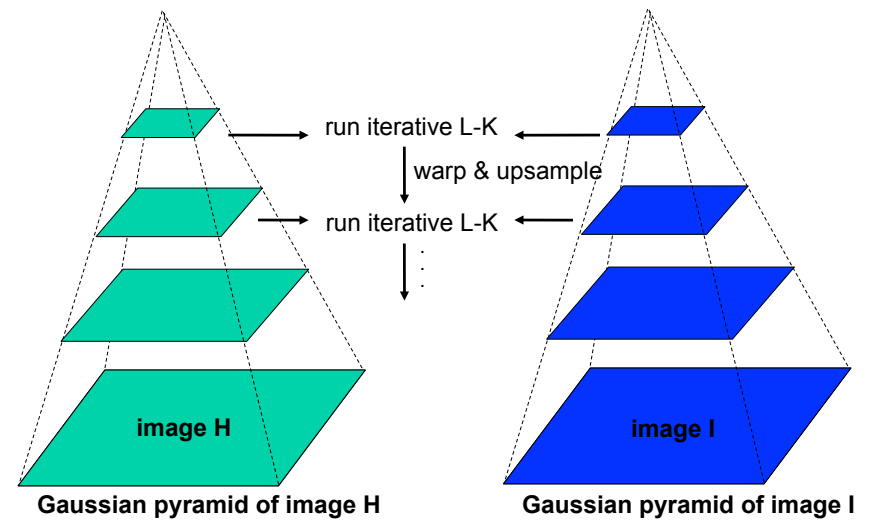
---



## Coarse-to-fine optical flow estimation



## Coarse-to-fine optical flow estimation



## Flow quality evaluation



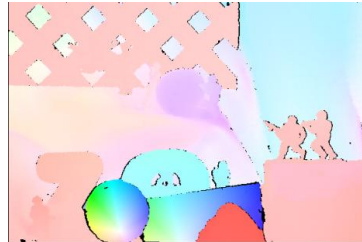
## Flow quality evaluation



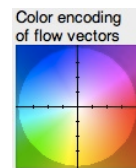
## Flow quality evaluation

Middlebury flow page

- <http://vision.middlebury.edu/flow/>



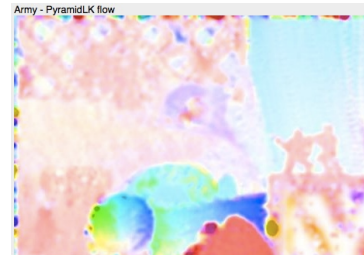
Ground Truth



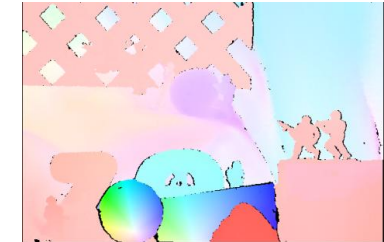
## Flow quality evaluation

Middlebury flow page

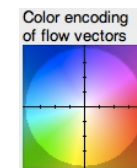
- <http://vision.middlebury.edu/flow/>



Lucas-Kanade flow



Ground Truth



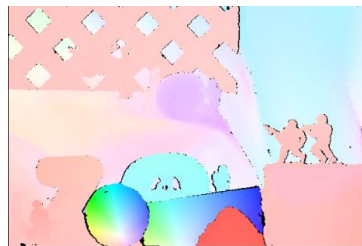
## Flow quality evaluation

Middlebury flow page

- <http://vision.middlebury.edu/flow/>



Best-in-class alg (as of 2/26/12)



Ground Truth

