EE795: Computer Vision and Intelligent Systems

Spring 2012
TTh 17:30-18:45 FDH 204

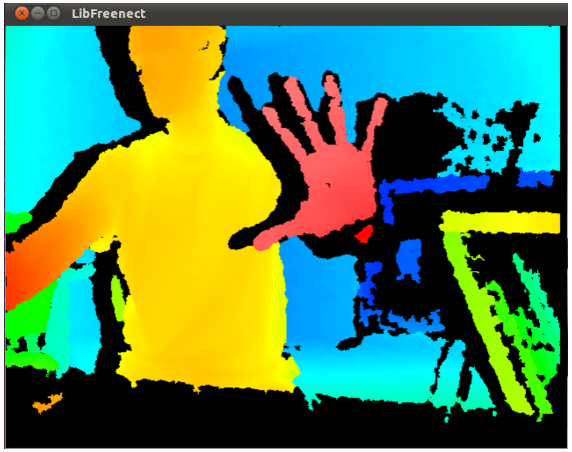
Lecture 14
130307

Outline

- Review
  - Stereo
- Dense Motion Estimation
- Translational Alignment
- Optical Flow

Stereo Matching

- Given two more images of the same scene or object, compute a representation of its shape
- Common application is generating disparity or depth map
  - Popularized for games recently by Kinect
- What are applications?



Stereo Matching

- Given two or more images of the same scene or object, compute a representation of its shape

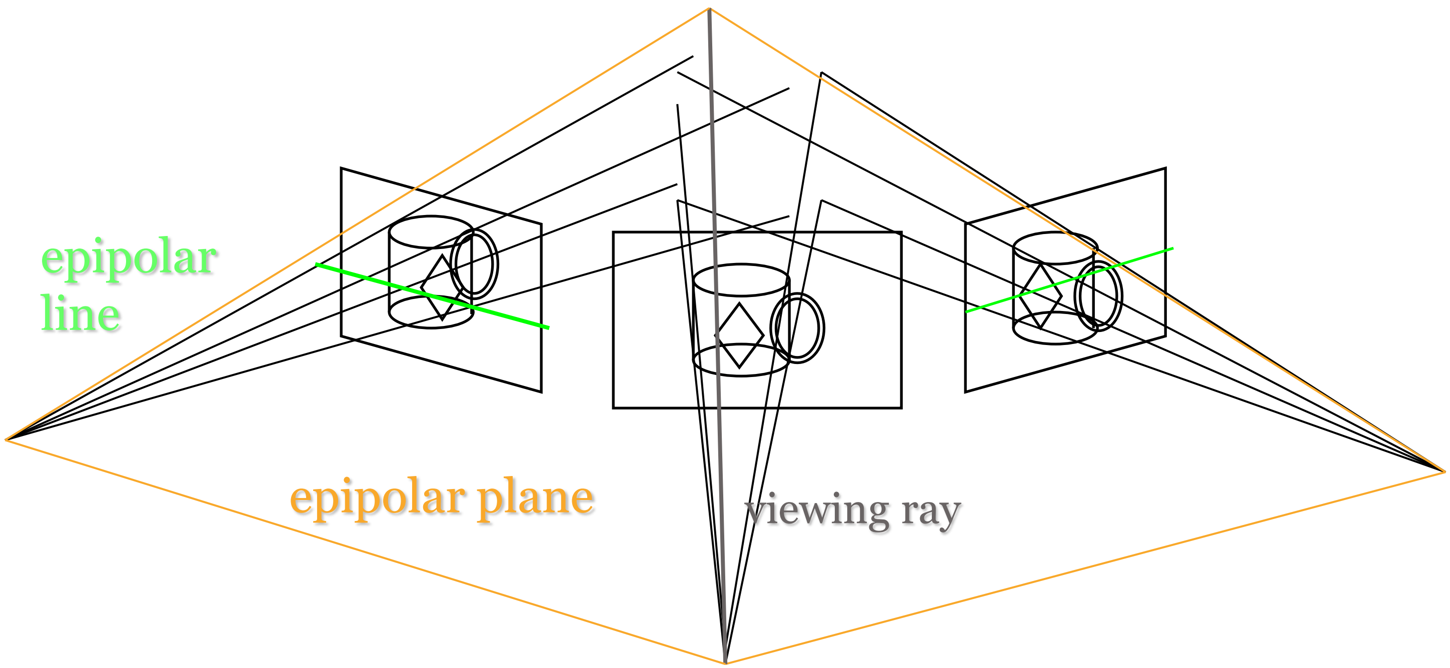
- What are some possible representations?
  - depth maps
  - volumetric models
  - 3D surface models
  - planar (or offset) layers

Stereo Matching

- What are some possible algorithms?
  - match "features" and interpolate
  - match edges and interpolate
  - match all pixels with windows (coarse-fine)
  - use optimization:
    - iterative updating
    - dynamic programming
    - energy minimization (regularization, stochastic)
    - graph algorithms

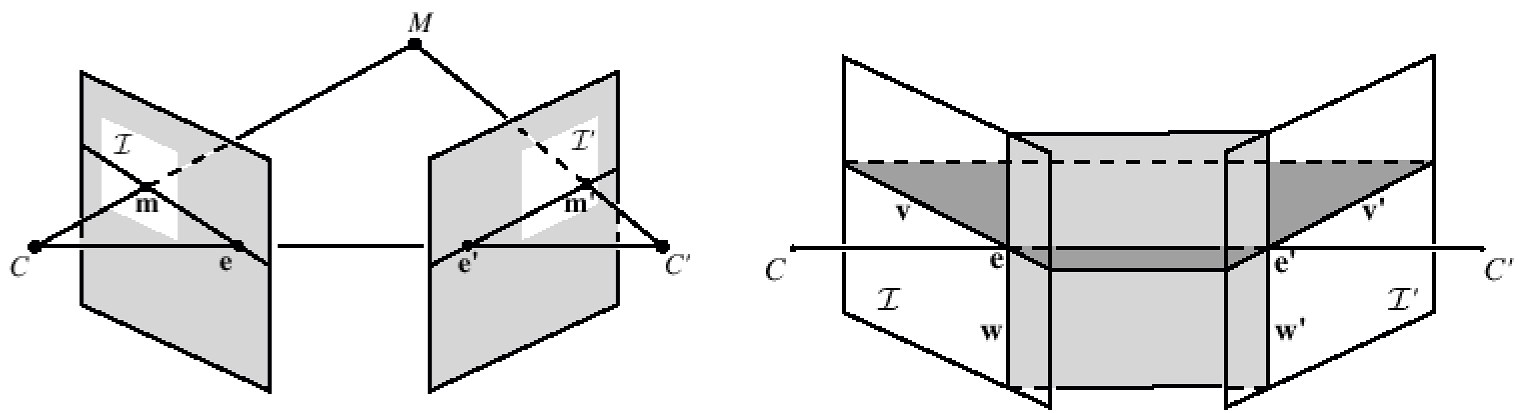
Stereo: epipolar geometry

- Match features along epipolar lines



- **Rectification:** warping the input images (perspective transformation) so that epipolar lines are horizontal

Rectification

- Project each image onto same plane, which is parallel to the epipole
- Resample lines (and shear/stretch) to place lines in correspondence, and minimize distortion
- 

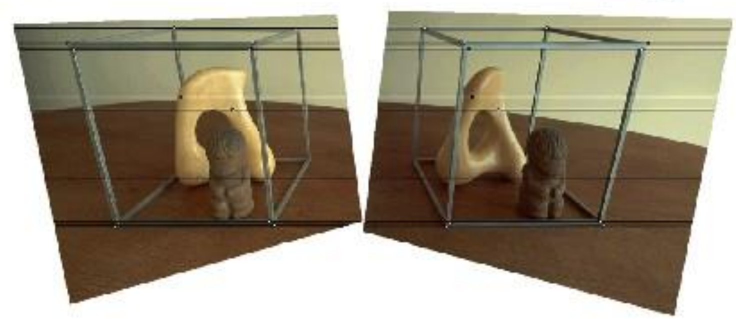
- [Loop and Zhang, CVPR'99]

Rectification

(a) Original image pair overlayed with several epipolar lines.

(b) Image pair transformed by the specialized projective mapping $\mathbf{H}_p$ and $\mathbf{H}'_p$. Note that the epipolar lines are now parallel to each other in each image.

(c) Image pair transformed by the similarity $\mathbf{H}_r$ and $\mathbf{H}'_r$. Note that the image pair is now rectified (the epipolar lines are horizontally aligned).



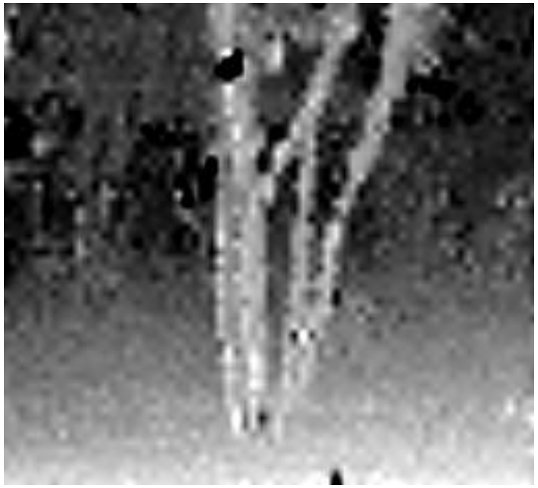
(d) Final image rectification after shearing transform $\mathbf{H}_s$ and $\mathbf{H}'_s$. Note that the image pair remains rectified, but the horizontal distortion is reduced.

BAD!

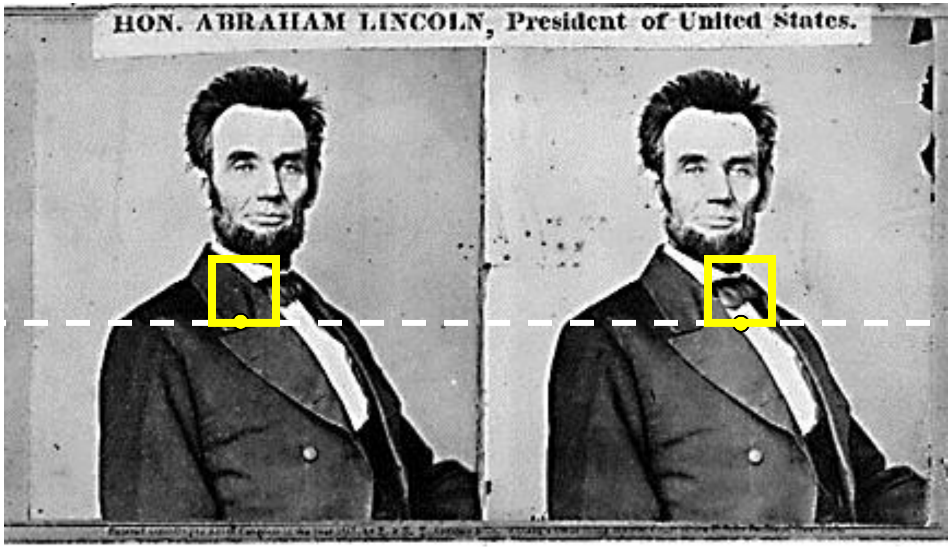
GOOD!

Finding correspondences

- apply feature matching criterion (e.g., correlation or Lucas-Kanade) at *all* pixels simultaneously
- search only over epipolar lines (many fewer candidate positions)



Your basic stereo algorithm



For each epipolar line

For each pixel in the left image

- compare with every pixel on same epipolar line in right image
- pick pixel with minimum match cost

Improvement: match ***windows***

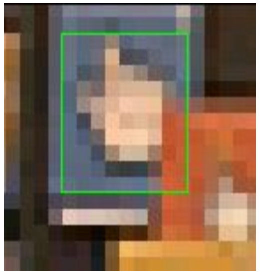
- This should look familar...

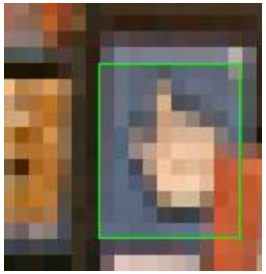
Image registration (revisited)

- How do we determine correspondences?
  - *block matching* or *SSD* (sum squared differences)

$$E(x,y;d) = \sum_{(x',y') \in N(x,y)} [I_L(x'+d,y') - I_R(x',y')]^2$$

d is the *disparity* (horizontal motion)

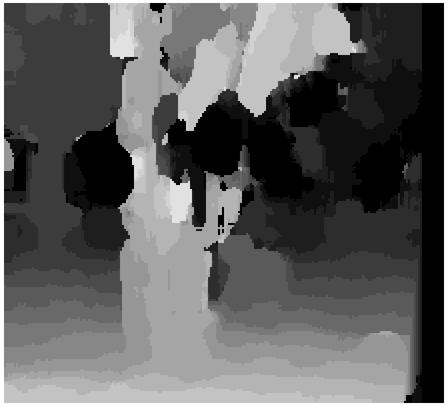


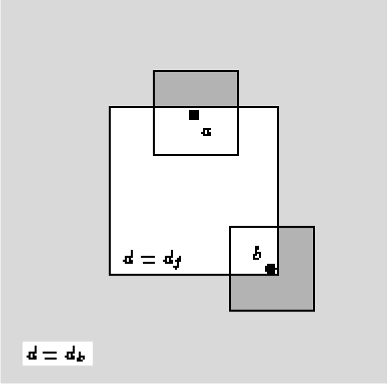


- How big should the neighborhood be?

Neighborhood size

- Smaller neighborhood: more details
- Larger neighborhood: fewer isolated mistakes





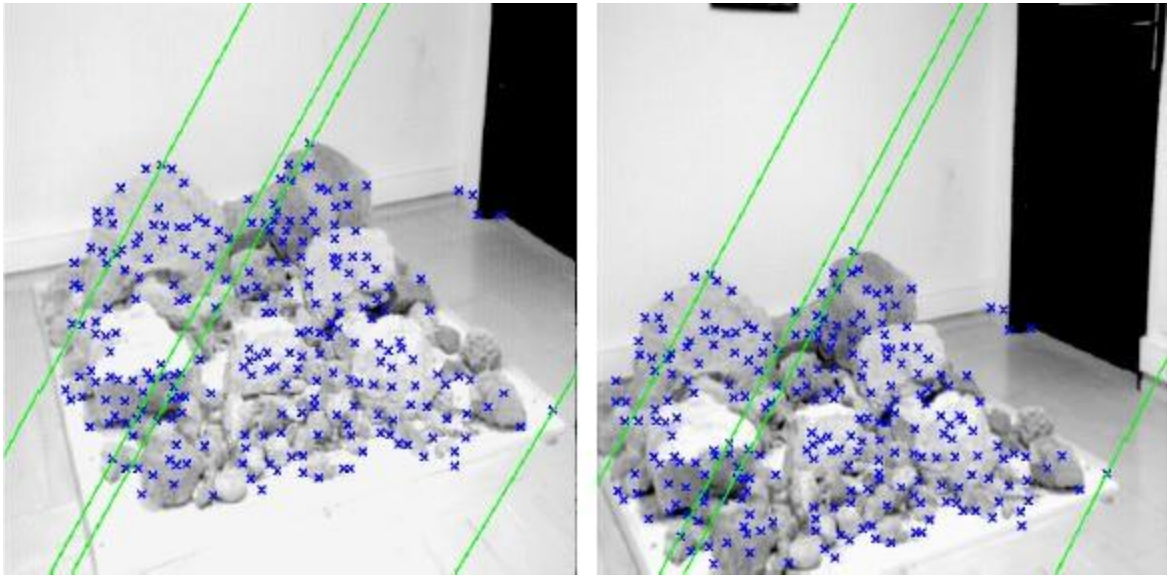
- w = 3 w = 20

Traditional Stereo Matching

- Advantages:
  - gives detailed surface estimates
  - fast algorithms based on moving averages
  - sub-pixel disparity estimates and confidence
- Limitations:
  - narrow baseline $\Rightarrow$ noisy estimates
  - fails in textureless areas
  - gets confused near occlusion boundaries

Feature-based stereo

- Match “corner” (interest) points



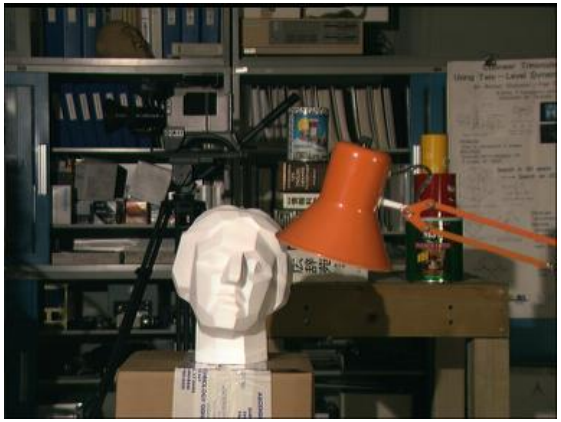
- Interpolate complete solution

Data interpolation

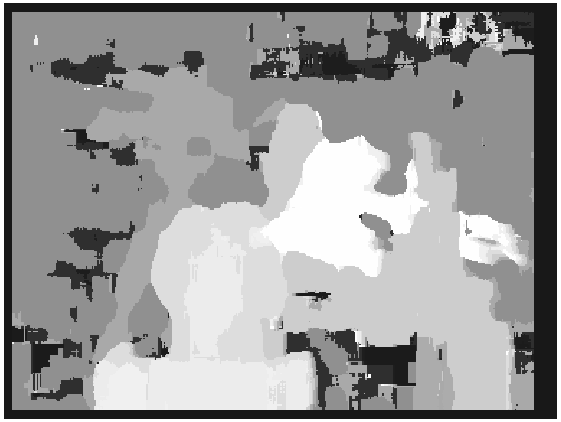
- Given a sparse set of 3D points, how do we *interpolate* to a full 3D surface?
- Scattered data interpolation [Nielson93]
- triangulate
- put onto a grid and fill (use pyramid?)
- place a *kernel function* over each data point
- minimize an energy function

- Lots of more advanced stereo matching options and algorithms exist

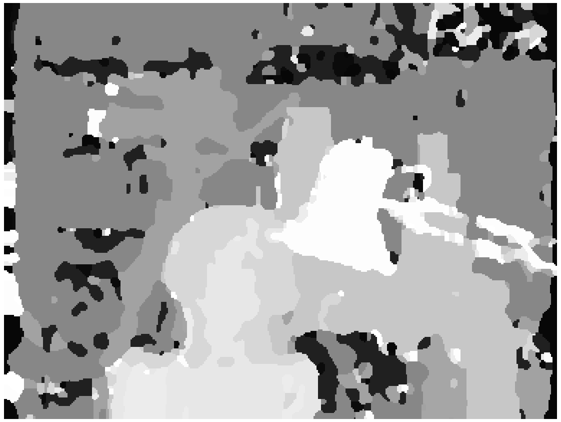
Depth Map Results



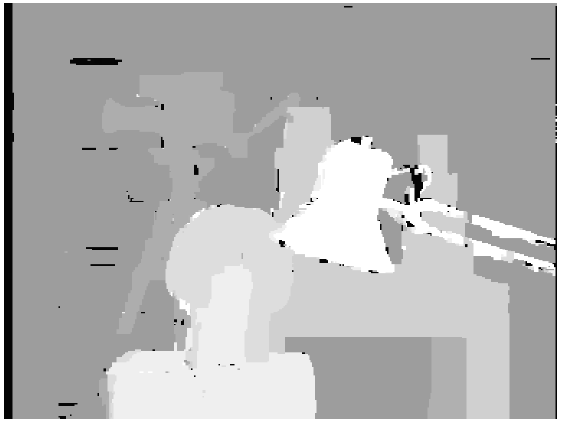
- Input image



Sum Abs Diff



- Mean field



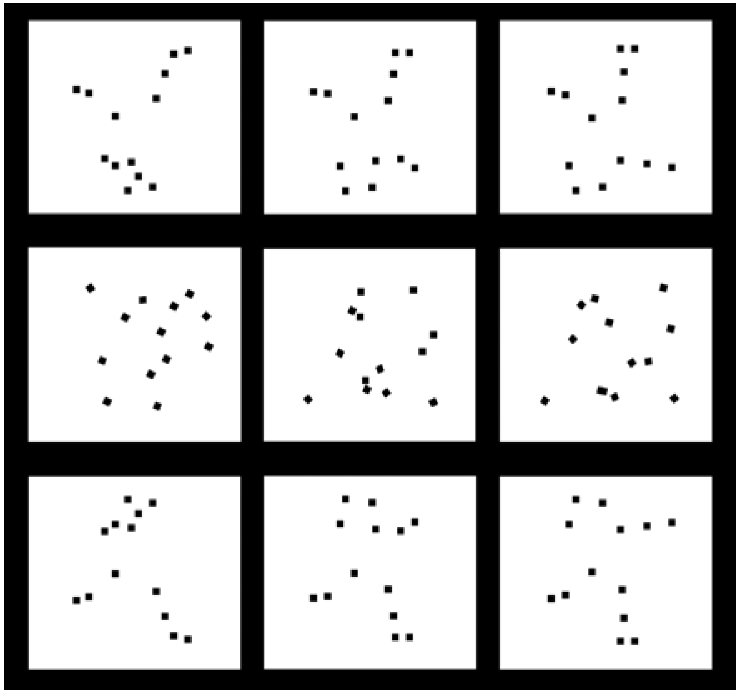
Graph cuts

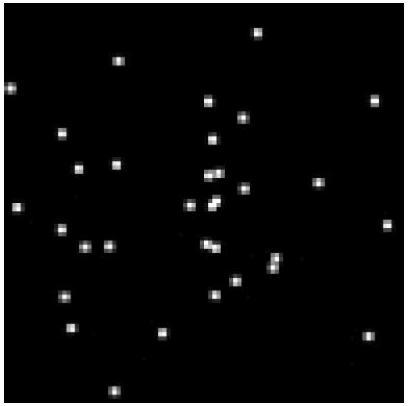
Dense Motion Estimation

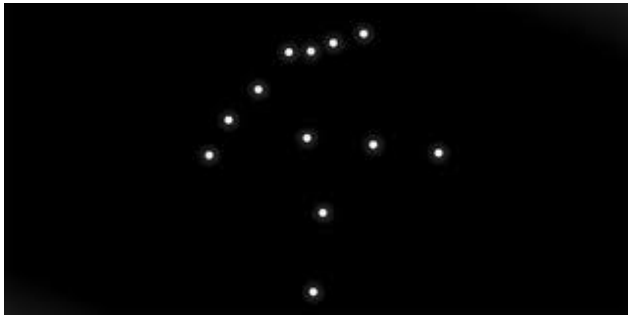
- Motion is extremely important in vision
- Biologically: motion indicates what is food and when to run away
  - We have evolved to be very sensitive to motion cues (peripheral vision)
- Alignment of images and motion estimation is widely used in computer vision
  - Optical flow
  - Motion compensation for video compression
  - Image stabilization
  - Video summarization

Biological Motion

- Even limited motion information is perceptually meaningful







- http://www.biomotionlab.ca/Demos/BMLwalker.html

Translational Alignment

- Motion estimation between images requires a error metric for comparison
- Sum of squared differences (SSD)
  - $E_{SSD}(u) = \sum_i [I_1(x_i + u) - I_0(x_i)]^2 = \sum_i e_i^2$
    - $u = (u, v)$ – is a displacement vector (can be subpixel)
    - e_i - residual error
- Brightness constancy constraint
  - Assumption that that corresponding pixels will retain the same value in two images
  - Objects tend to maintain the perceived brightness under varying illumination conditions [Horn 1974]
- Color images processed by channels and summed or converted to colorspace that considers only luminance

SSD Improvements

- As we have seen many times in class, SSD is the simplest approach and can be improved
- Robust error metrics
  - L_1 norm (sum absolute differences)
    - Better outlier resilience
- Spatially varying weights
  - Weighted SSD to weight contribution of each pixel during matching
    - Ignore certain parts of the image (e.g. foreground), down-weight objects during images stabilization
- Bias and gain
  - Normalize exposure between images
    - Address brightness constancy

Correlation

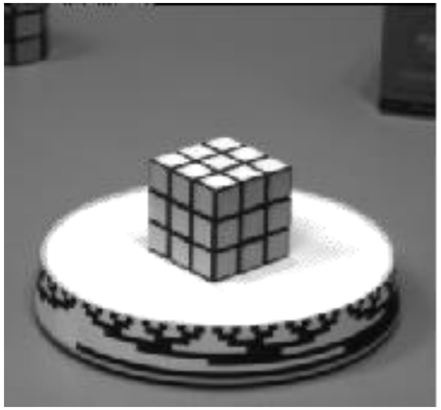
- Instead of minimizing pixel differences, maximize correlation
- Normalized cross-correlation

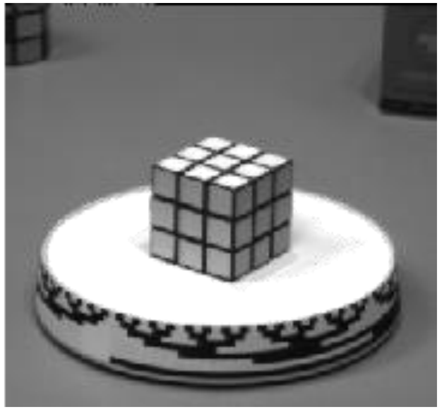
$$E_{\mathrm{NCC}}(\boldsymbol{u}) = \frac{\sum_i [I_0(\boldsymbol{x}_i) - \overline{I_0}]\ [I_1(\boldsymbol{x}_i + \boldsymbol{u}) - \overline{I_1}]}{\sqrt{\sum_i [I_0(\boldsymbol{x}_i) - \overline{I_0}]^2}\sqrt{\sum_i [I_1(\boldsymbol{x}_i + \boldsymbol{u}) - \overline{I_1}]^2}},$$

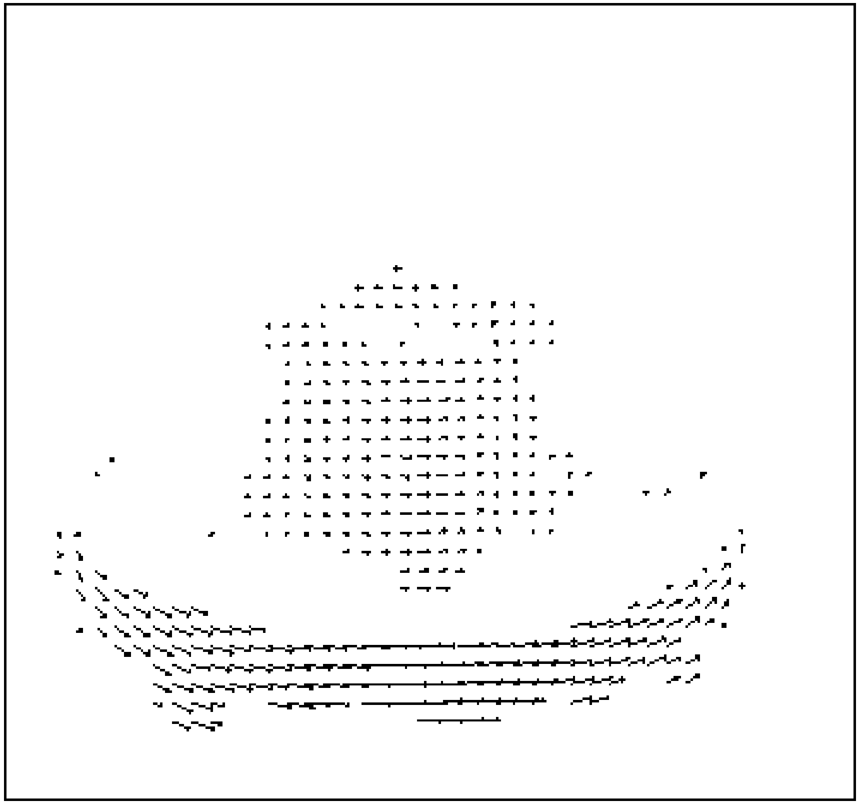
$$\begin{aligned} \overline{I_0} &= \frac{1}{N}\sum_i I_0(\boldsymbol{x}_i) \quad \text{and} \\ \overline{I_1} &= \frac{1}{N}\sum_i I_1(\boldsymbol{x}_i + \boldsymbol{u}) \end{aligned}$$

  - Normalize by the patch intensities
  - Value is between [-1, 1] which makes it easy to use results (e.g. threshold to find matching pixels)

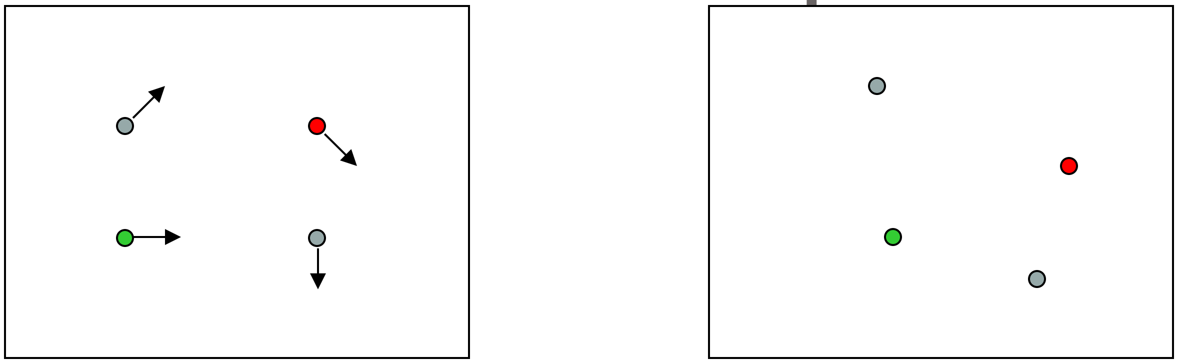
Optical flow







Problem definition: optical flow



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

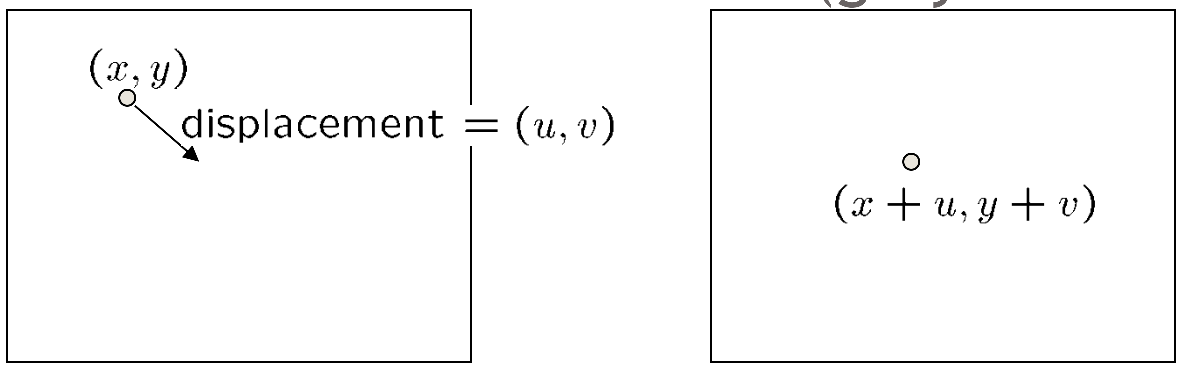
- How to estimate pixel motion from image H to image I?
  - Solve pixel correspondence problem
    - given a pixel in H, look for nearby pixels of the same color in I

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

Optical flow constraints (grayscale images)



$H(x, y)$

• brightness constancy: Q: what's the equation?
• $H(x,y) = I(x+u,y+v)$

- 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:

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

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

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

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

Optical flow equation

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

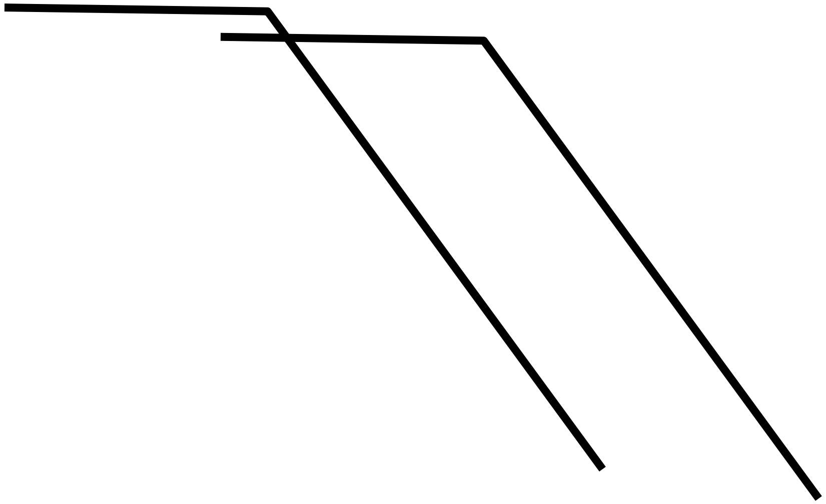
- Q: how many unknowns and equations per pixel?
  - u and v are unknown, 1 equation

Intuitively, what does this constraint mean?

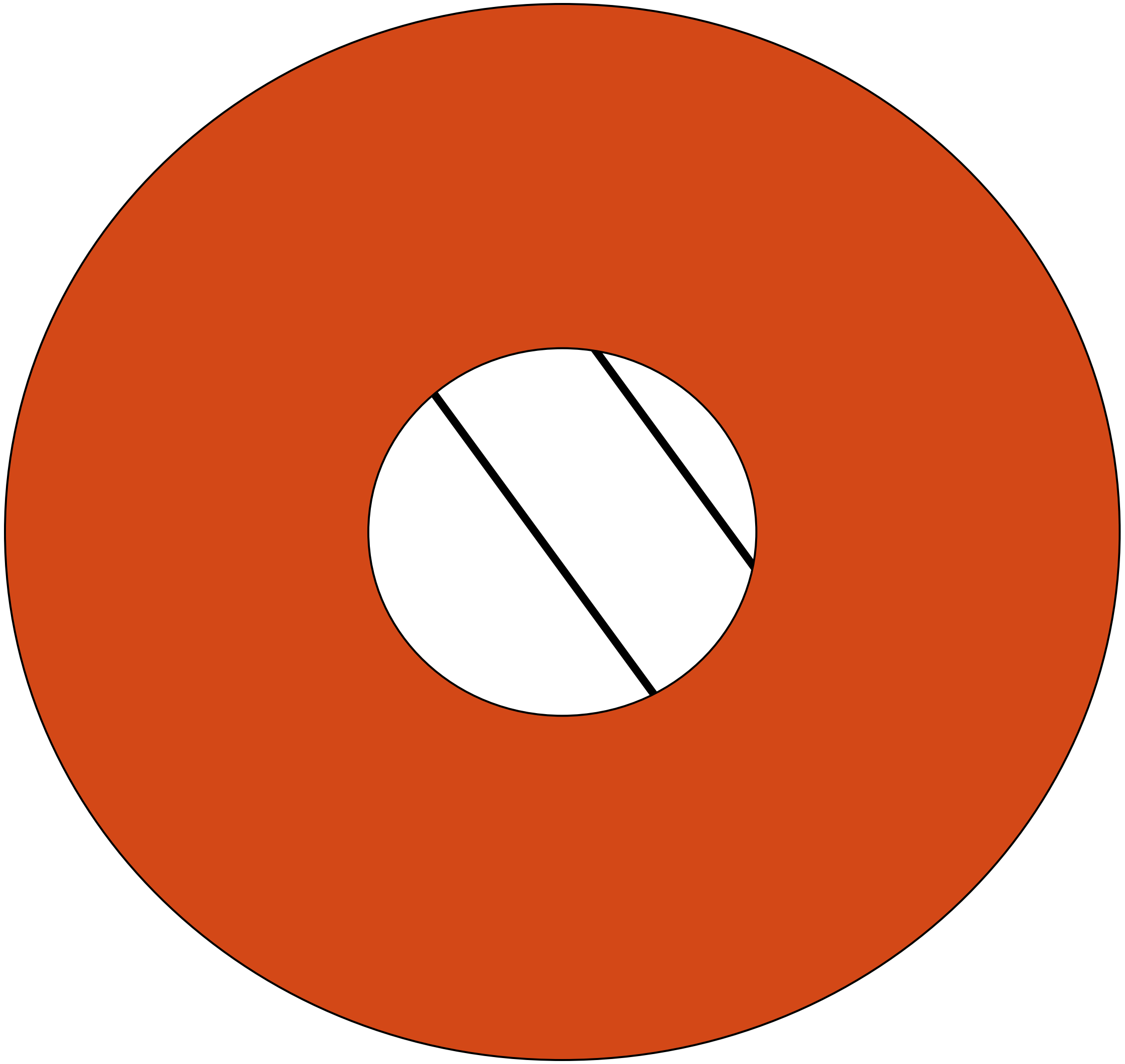
- The component of the flow in the gradient direction is determined
- The component of the flow parallel to an edge is unknown

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

Aperture problem



Aperture problem



Solving the aperture problem

- Basic idea: assume motion field is smooth

- Horn & Schunk: add smoothness term

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

- Lucas & Kanade: assume locally constant motion
  - pretend the pixel's neighbors have the same (u,v)

- Many other methods exist. Here's an overview:
  - S. Baker, M. Black, J. P. Lewis, S. Roth, D. Scharstein, and R. Szeliski. *A database and evaluation methodology for optical flow*. In Proc. ICCV, 2007
  - http://vision.middlebury.edu/flow/

Lucas-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!

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

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

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!

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

$$\underset{75\text{x}2}{\underbrace{\begin{bmatrix} I_x(\mathbf{p}_1)[0] & I_y(\mathbf{p}_1)[0] \\ I_x(\mathbf{p}_1)[1] & I_y(\mathbf{p}_1)[1] \\ I_x(\mathbf{p}_1)[2] & I_y(\mathbf{p}_1)[2] \\ \vdots & \vdots \\ I_x(\mathbf{p}_{25})[0] & I_y(\mathbf{p}_{25})[0] \\ I_x(\mathbf{p}_{25})[1] & I_y(\mathbf{p}_{25})[1] \\ I_x(\mathbf{p}_{25})[2] & I_y(\mathbf{p}_{25})[2] \end{bmatrix}}_{A}} \underset{2\times1}{\underbrace{\begin{bmatrix} u \\ v \end{bmatrix}}_{d}} = - \underset{75\text{x}1}{\underbrace{\begin{bmatrix} I_t(\mathbf{p}_1)[0] \\ I_t(\mathbf{p}_1)[1] \\ I_t(\mathbf{p}_1)[2] \\ \vdots \\ I_t(\mathbf{p}_{25})[0] \\ I_t(\mathbf{p}_{25})[1] \\ I_t(\mathbf{p}_{25})[2] \end{bmatrix}}_{b}}$$

Lucas-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 Lucas & Kanade (1981)

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 $\mathbf{A^T A}$ should not be too small
  - $\mathbf{A^T A}$ should be well-conditioned
    - λ_1 / λ_2 should not be too large (λ_1 = larger eigenvalue)

- Does this look familiar?
  - $\mathbf{A^T A}$ 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 for feature tracking...

Errors in Lucas-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?

Improving accuracy

- Recall our small motion assumption

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

- Not exact, need higher order terms to do better

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

- Results in polynomial root finding problem
  - Can be solved using Newton's method
    - Also known as Newton-Raphson

1D case on board

- Lucas-Kanade method does a single iteration of Newton's method
  - Better results are obtained with more iterations

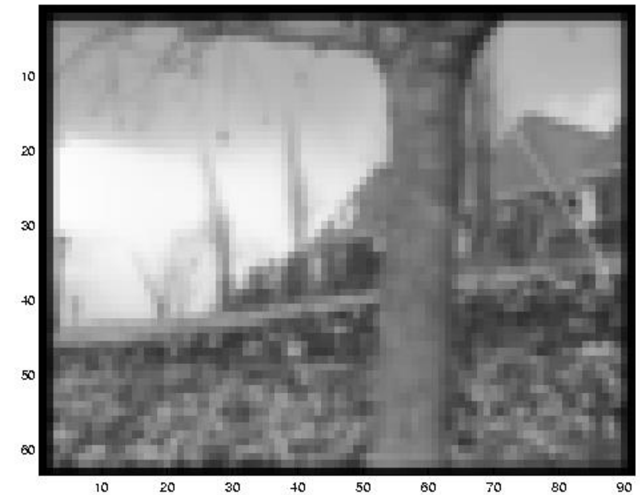
Iterative Refinement

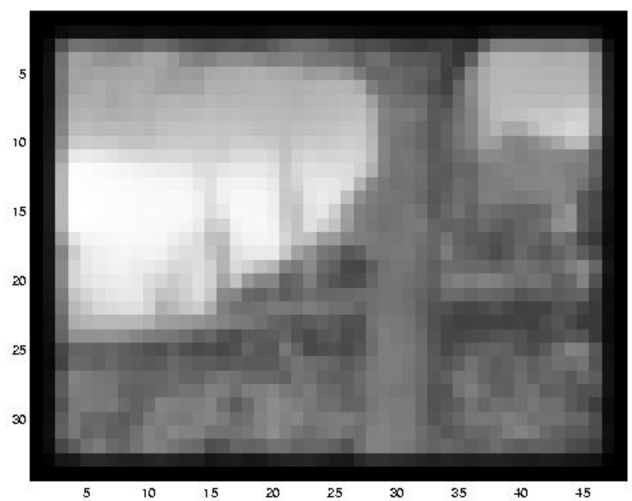
- Iterative Lucas-Kanade Algorithm
  1. Estimate velocity at each pixel by solving Lucas-Kanade equations
  2. Warp H towards I using the estimated flow field
     - *- use image warping techniques*
  3. Repeat until convergence

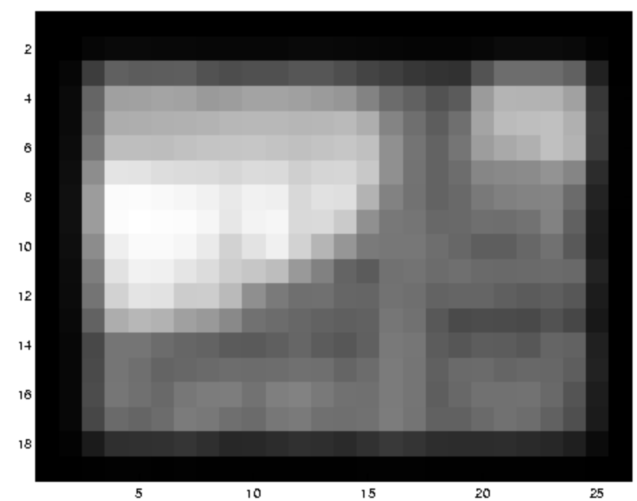
Revisiting the small motion assumption

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

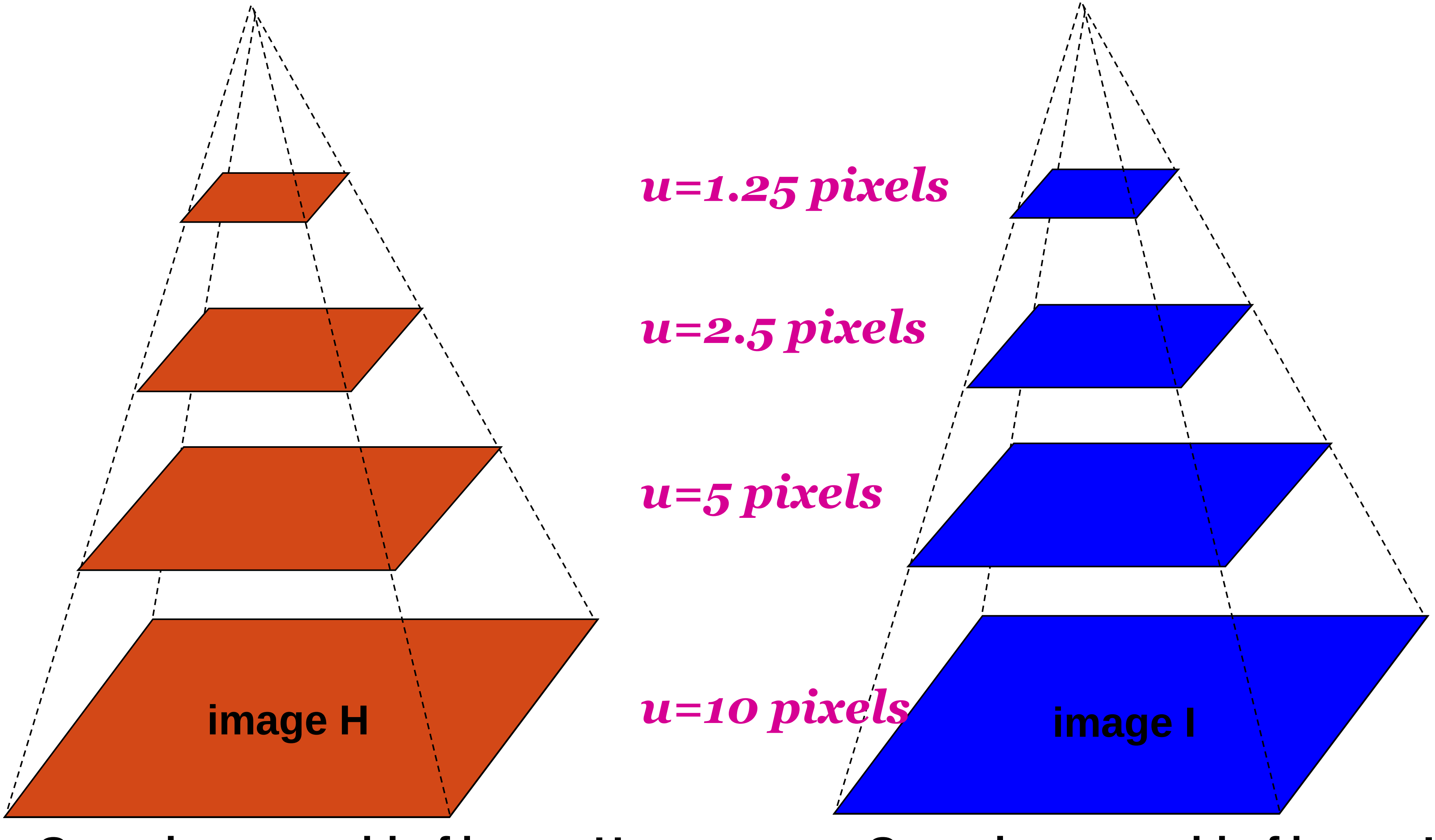
Reduce the resolution!



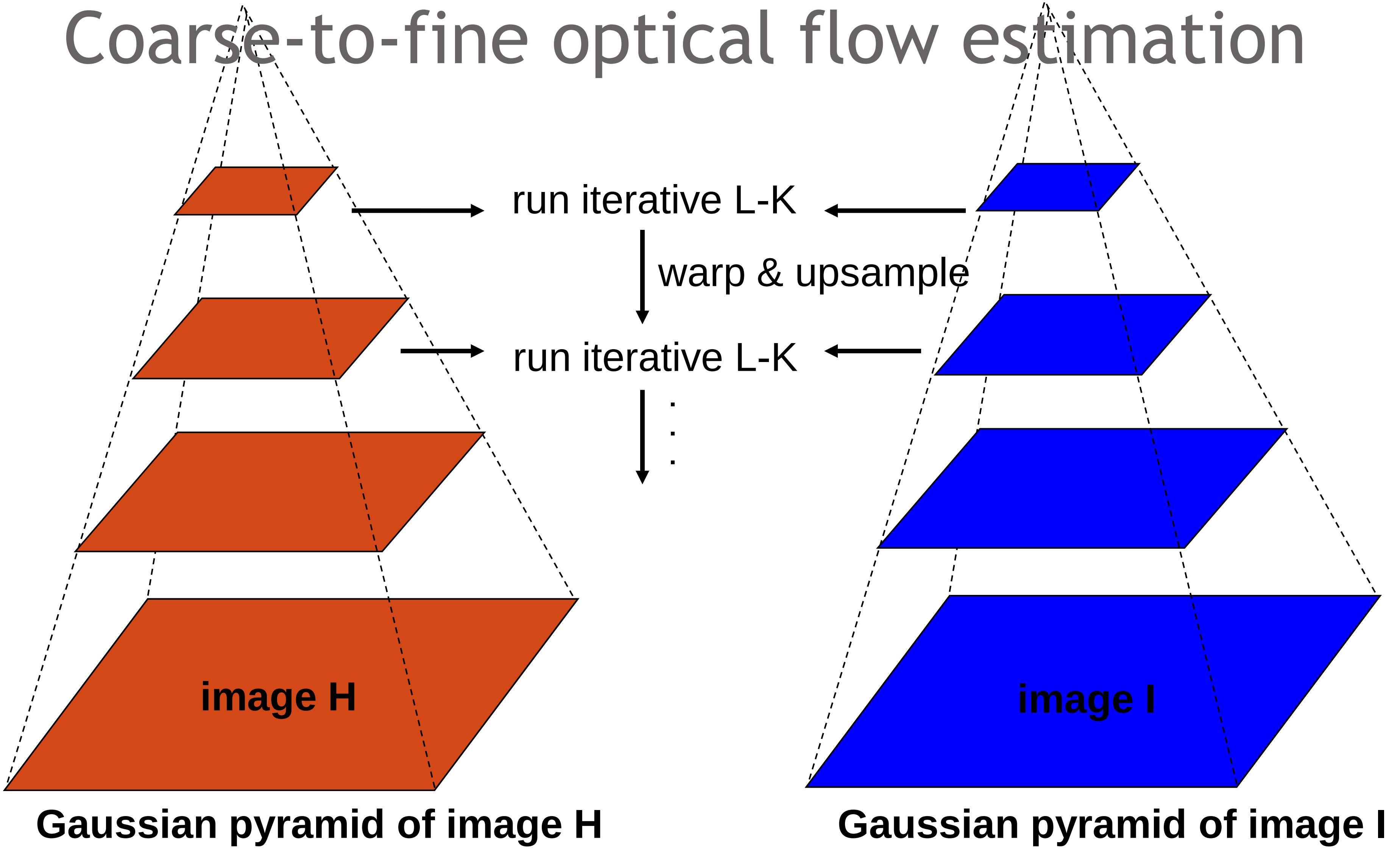




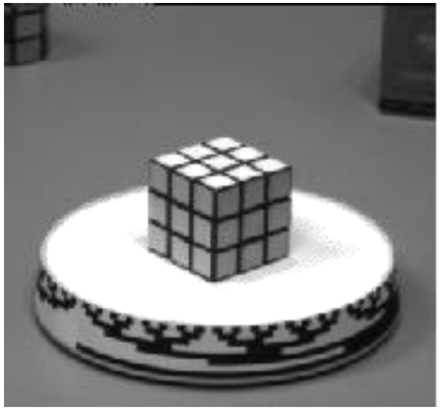
Coarse-to-fine optical flow estimation

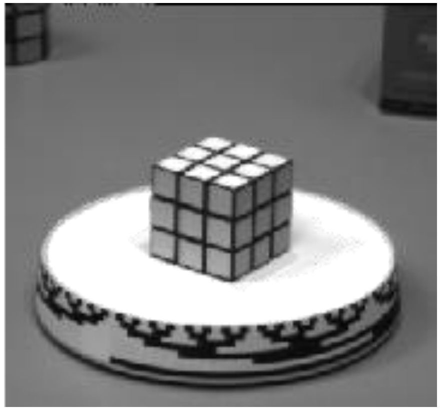


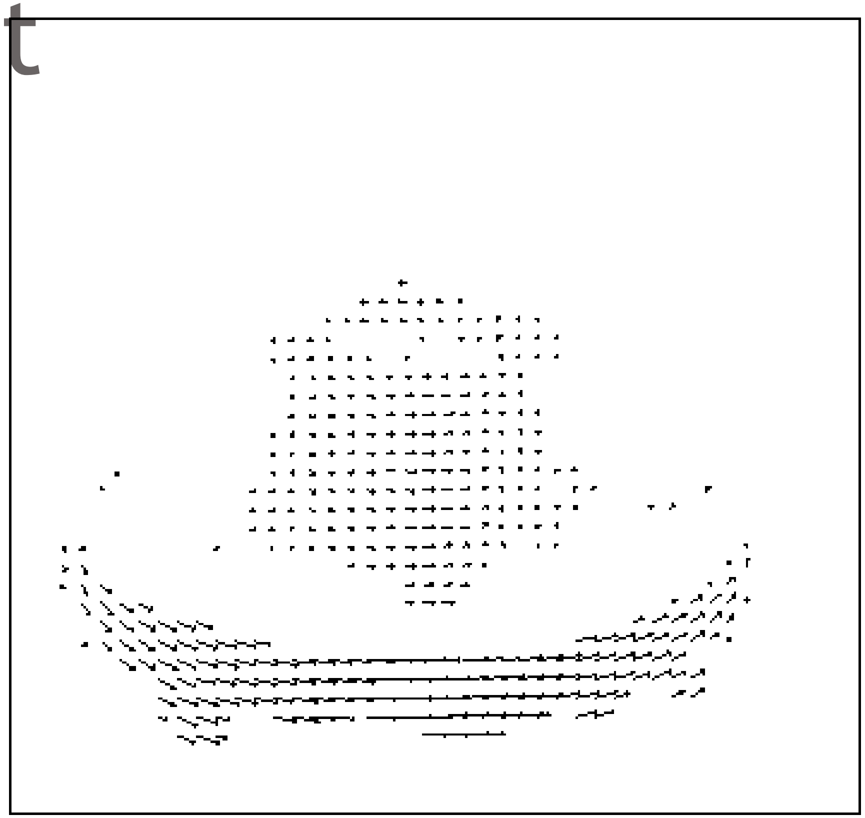
Coarse-to-fine optical flow estimation



Optical flow result



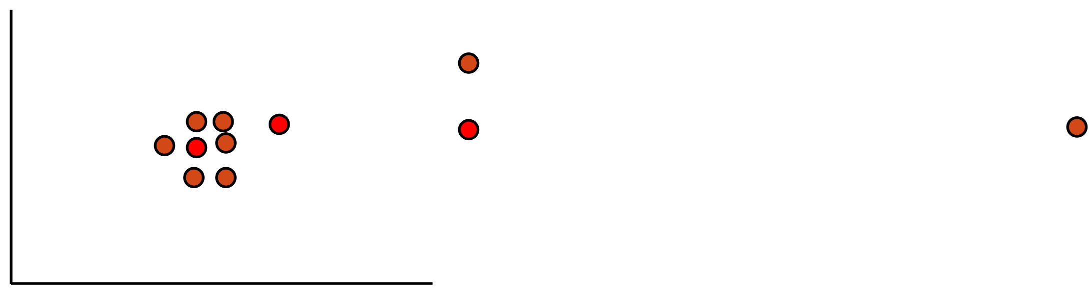




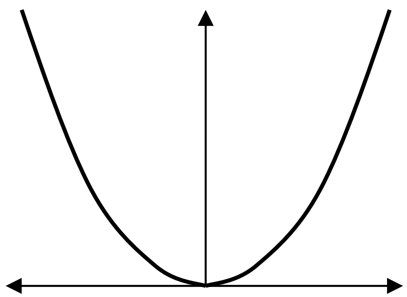
Dewey morph

Robust methods

- L-K minimizes a sum-of-squares error metric
  - least squares techniques overly sensitive to outliers

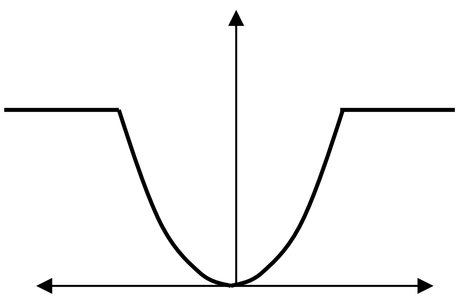


Error metrics



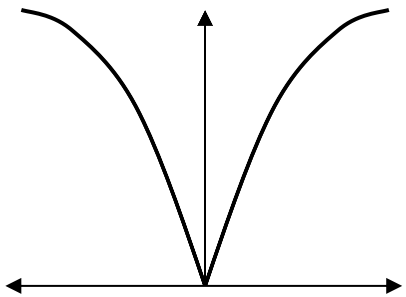
quadratic

$$\rho(x) = x^2$$



truncated quadratic

$$\rho_{\alpha,\lambda}(x) = \begin{cases} \lambda x^2 & \text{if } |x| < \frac{\sqrt{\alpha}}{\sqrt{\lambda}} \\ \alpha & \text{otherwise} \end{cases}$$



lorentzian

$$\rho_\sigma(x) = log\left(1 + \frac{1}{2}(\frac{x}{\sigma})^2\right)$$

Robust optical flow

- Robust Horn & Schunk

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

- Robust Lucas-Kanade

$$\sum_{(x,y)\in W} \rho(I_t + \nabla I \cdot [u\ v])$$

first image | quadratic flow | lorentzian flow | detected outliers

Black, M. J. and Anandan, P., A framework for the robust estimation of optical flow, *Fourth International Conf. on Computer Vision* (ICCV), 1993, pp. 231-236
http://www.cs.washington.edu/education/courses/576/03sp/readings/black93.pdf

Benchmarking optical flow algorithms

- Middlebury flow page
  - http://vision.middlebury.edu/flow/

Discussion: features vs. flow?

- Features are better for:
- Flow is better for:

Advanced topics

- Particles: combining features and flow
  - Peter Sand et al.
  - http://rvsn.csail.mit.edu/pv/
- State-of-the-art feature tracking/SLAM
  - Georg Klein et al.
  - http://www.robots.ox.ac.uk/~gk/