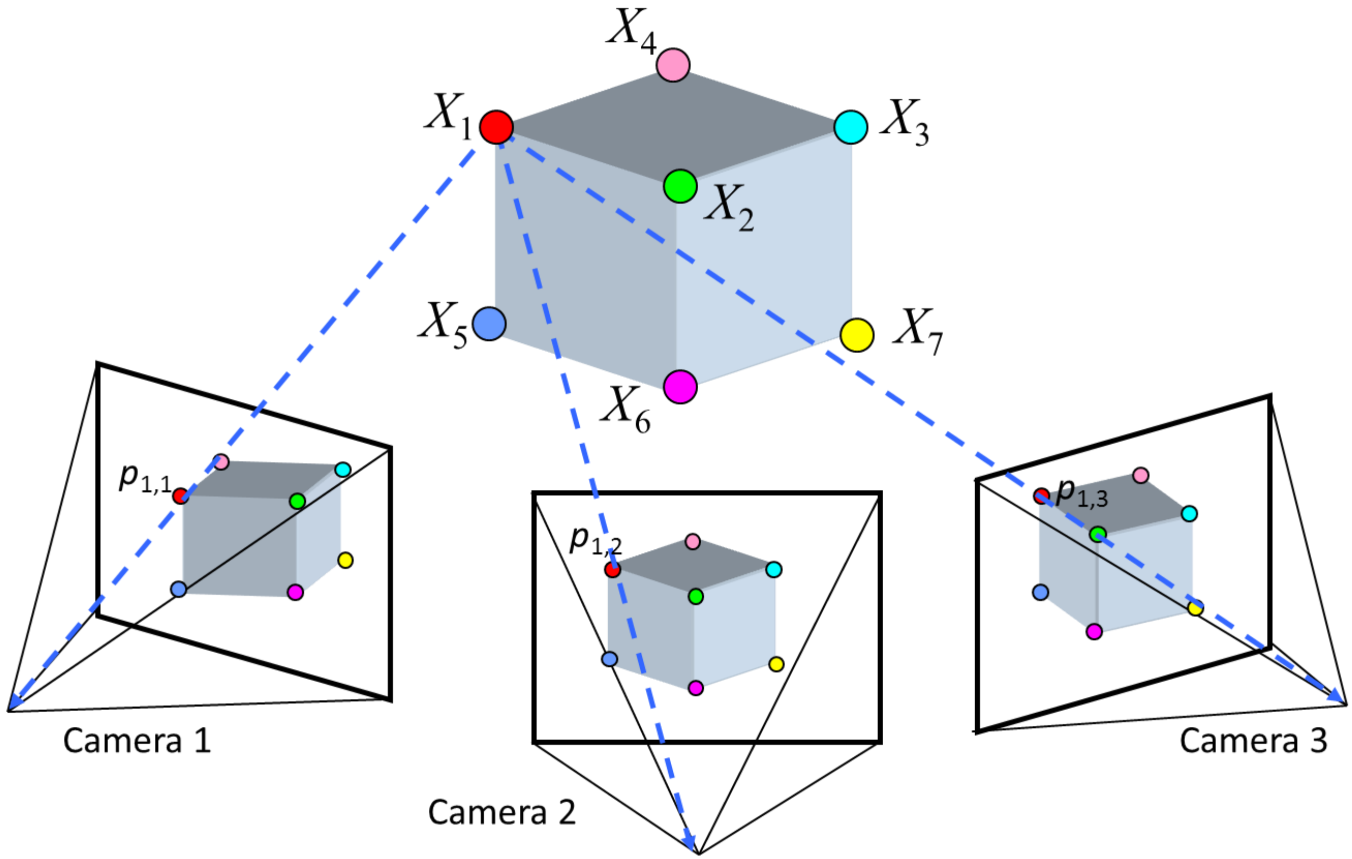
Optical Flow

CS 429
Princeton University

Many slides adapted from K. Grauman, S. Seitz, R. Szeliski, M. Pollefeys, and S. Lazebnik

Last Two Lectures: Images

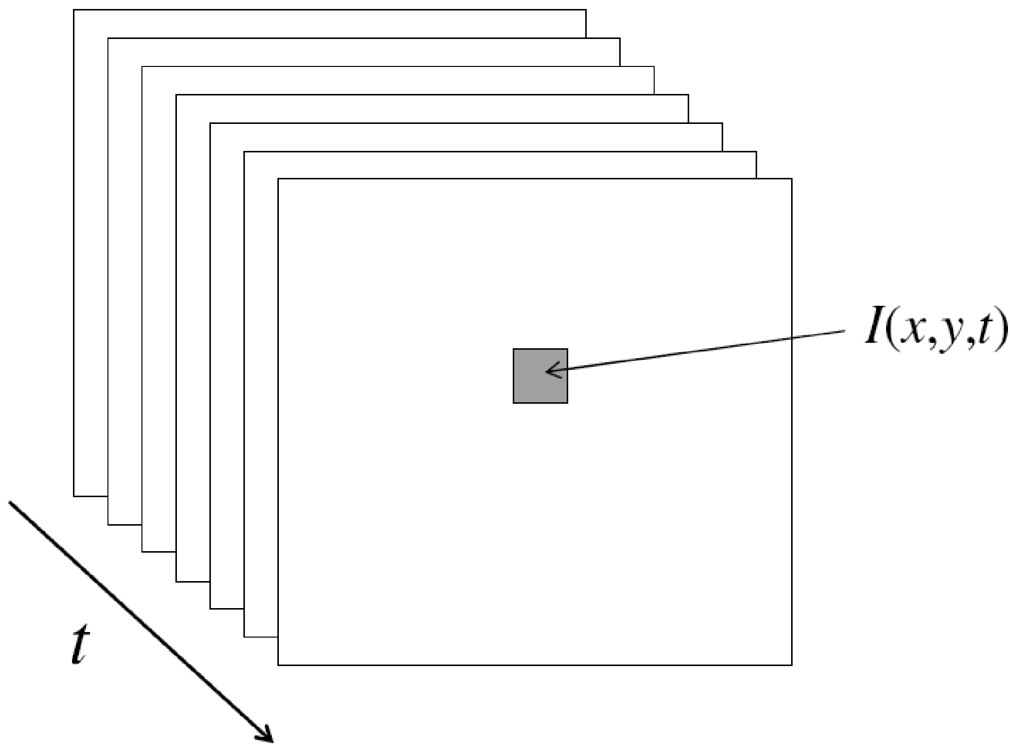
Infer camera and scene geometry from a set of images



Noah Snavely

Next Two Lectures: Video

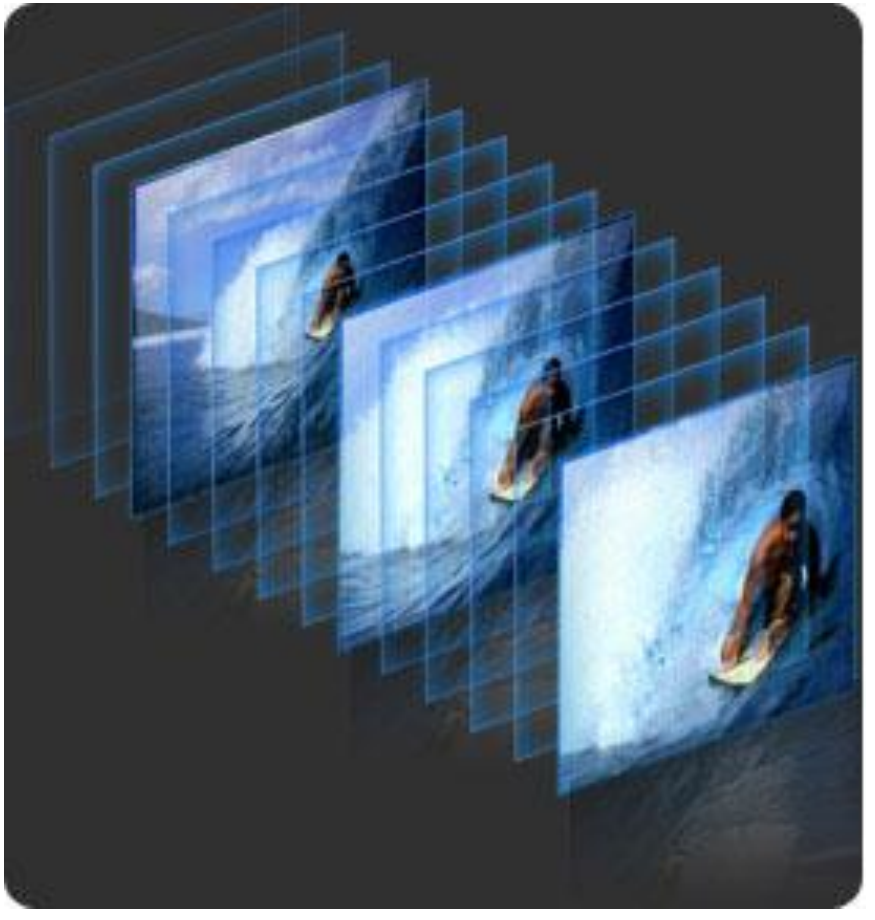
Infer camera and scene geometry from a time-varying sequence of images (video)



Kristen Grauman

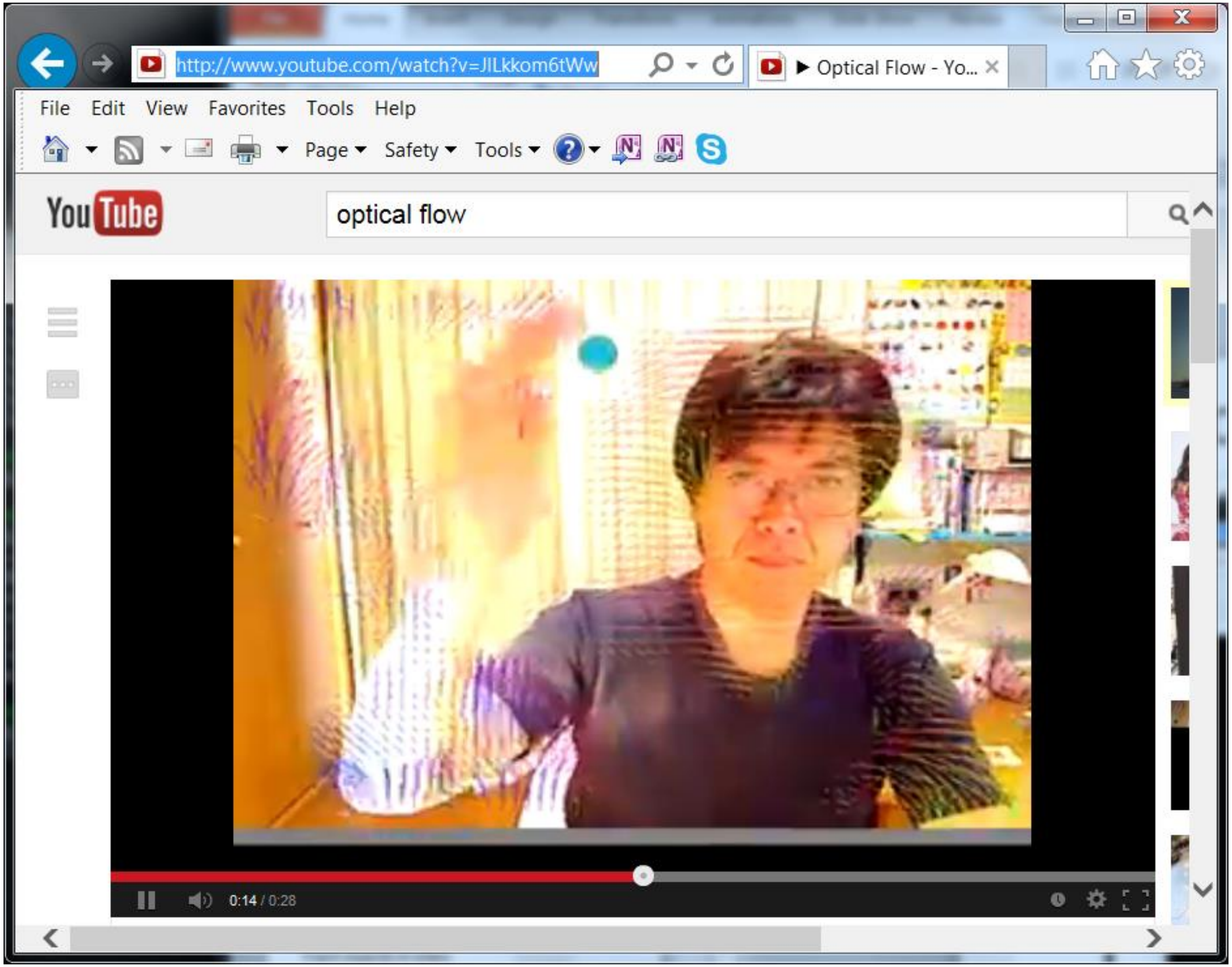
Next Two Lectures: Video

Infer camera and scene geometry from a time-varying sequence of images (video)



Apple Shake

This Lecture: Estimating Motion in Video



http://www.youtube.com/watch?v=JlLkkom6tWw

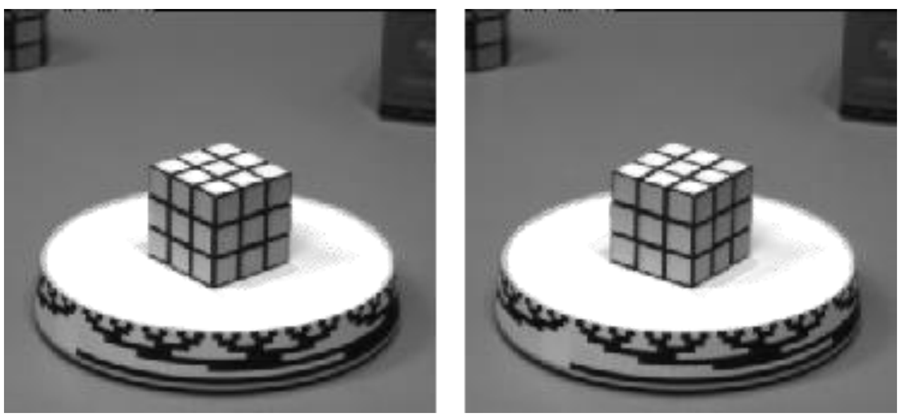
Applications?

Applications

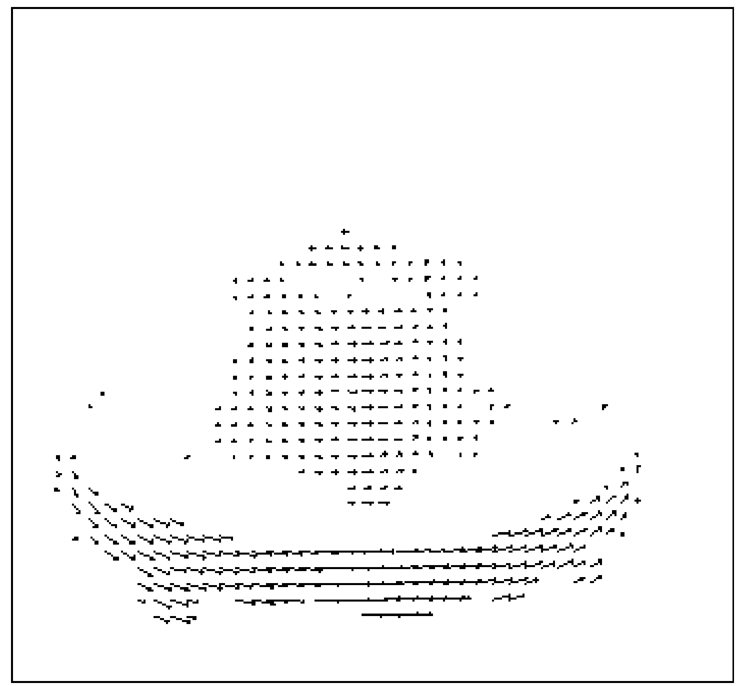
- Estimating depth
- Tracking object motion
- Determining camera motion
- Segmenting objects based on motion cues
- Video compression
- Robot navigation
- Studying dynamical models
- Recognizing events and activities
- Human computer interaction
- Facial animation
- Video filters

Estimating Depth

- The motion field is the projection of the 3D scene motion into the image
- Length of motion vectors is inversely proportional to depth Z of 3D point



Sequence of images in video



Motion field

Estimating Depth

Length of motion vectors is inversely proportional to depth Z of 3D point

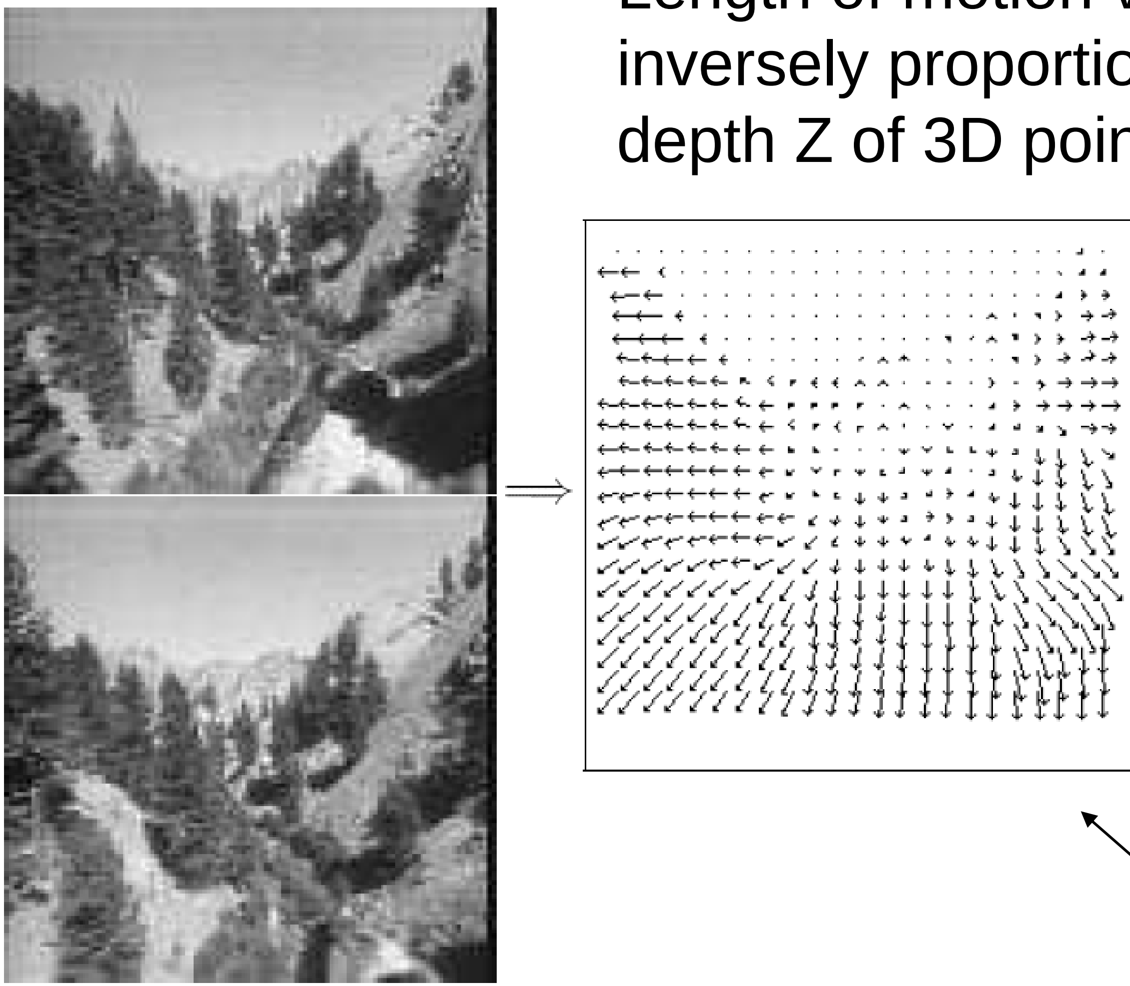


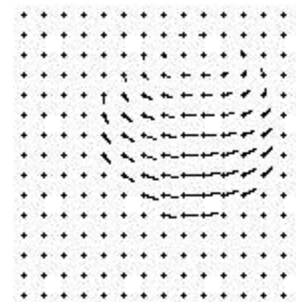
Figure 1.2: Two images taken from a helicopter flying through a canyon and the computed optical flow field.

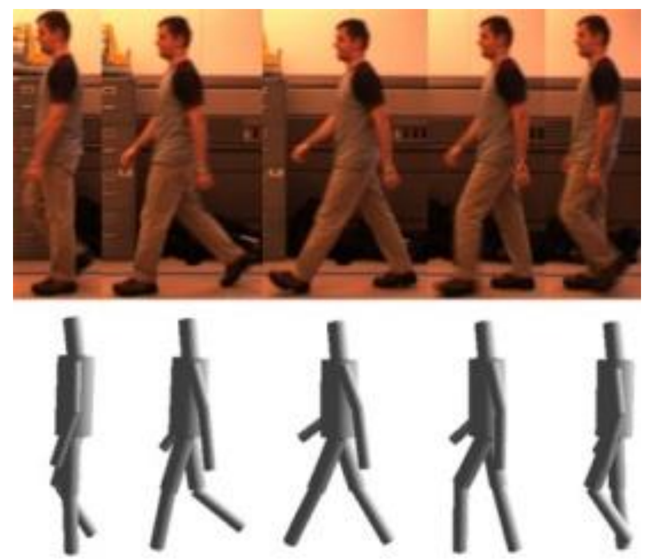
Figure from Michael Black, Ph.D. Thesis

points closer to the camera move more quickly across the image plane

Tracking objects

Motion field reveals movement of objects

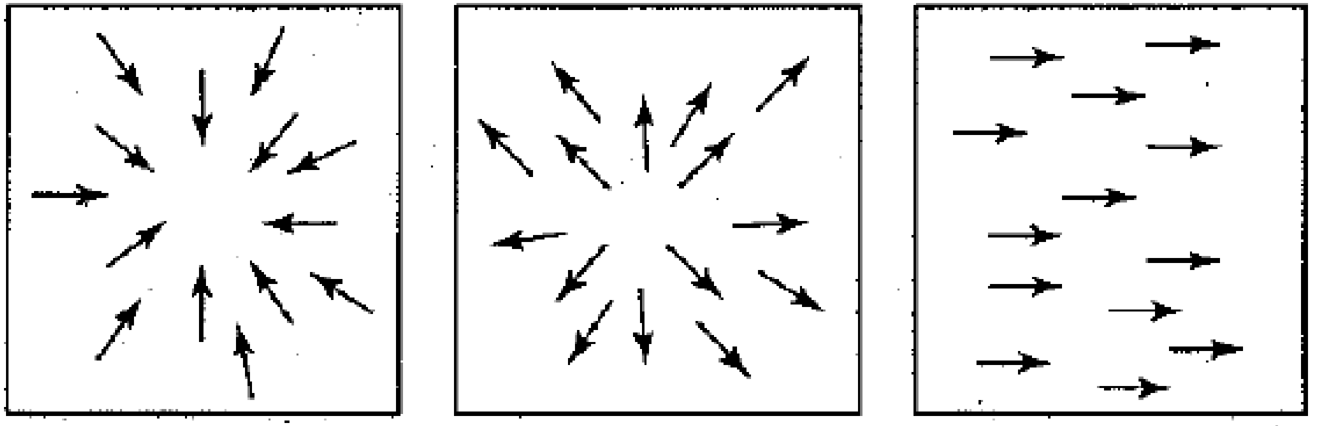




Tomas Izo

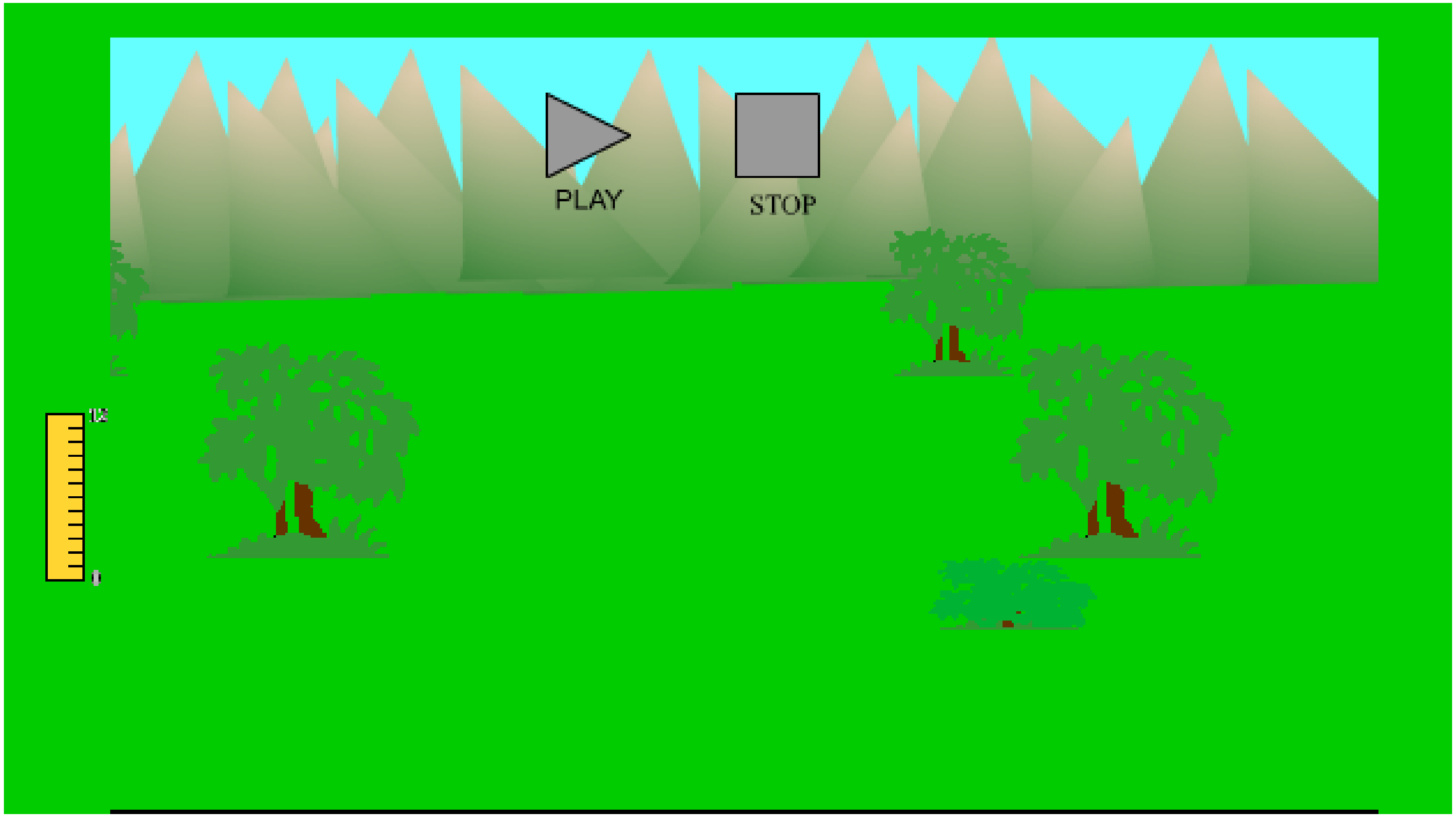
Estimating camera motion

Motion field reveals movement of camera



Zoom out **Zoom in** **Pan right to left**

Segmenting objects based on parallax



http://psych.hanover.edu/KRANTZ/MotionParallax/MotionParallax.html

Outline

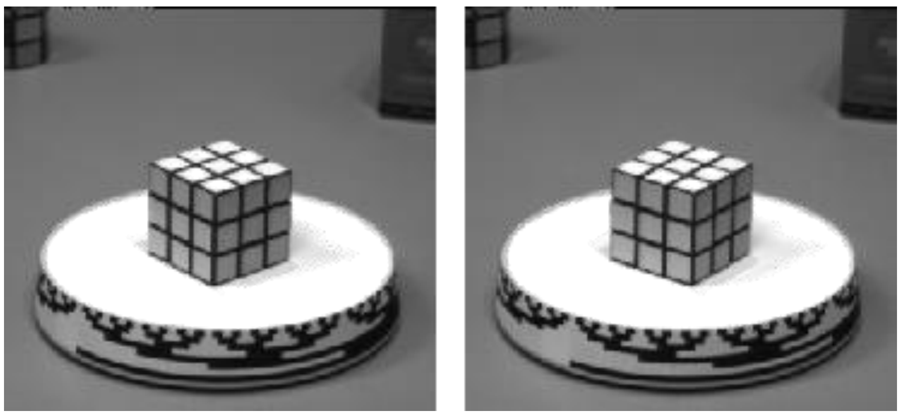
Motivation

Algorithms

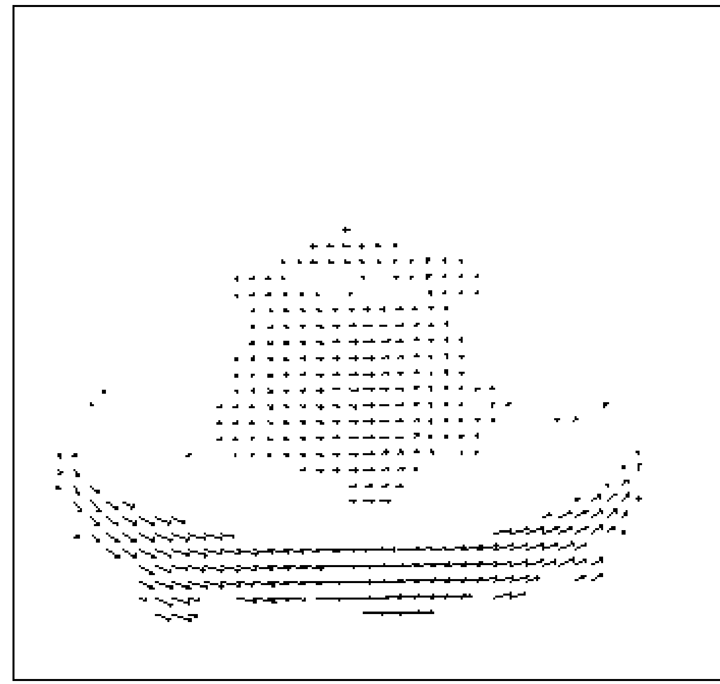
Evaluation

Applications

Motion estimation algorithms?



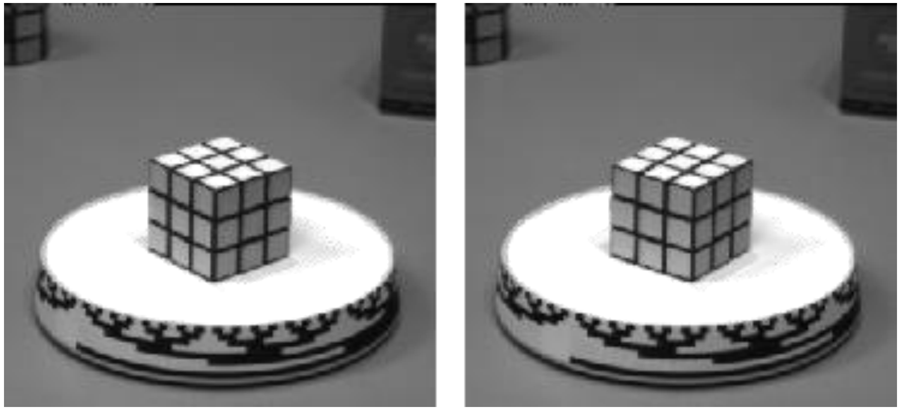
Sequence of images in video



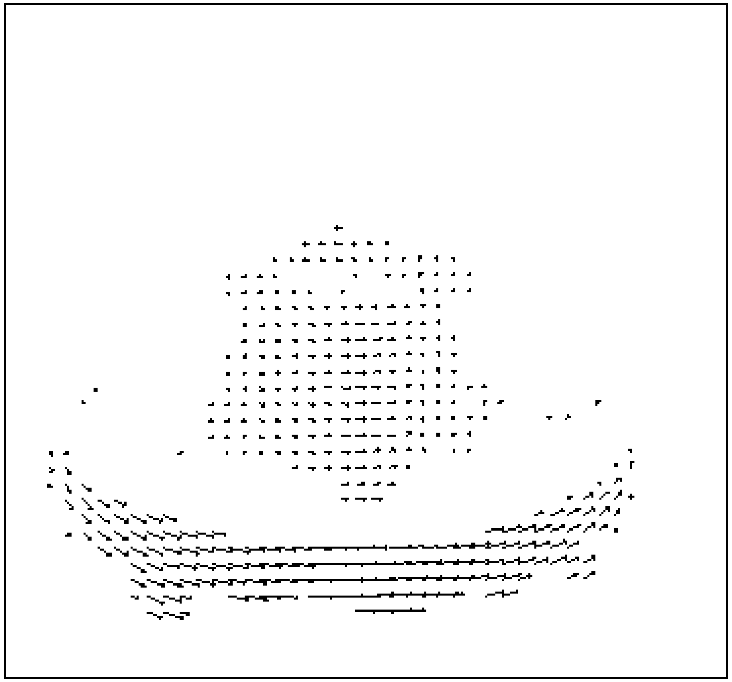
Motion field

Motion estimation algorithms

- Feature-based methods
- Pixel-based methods



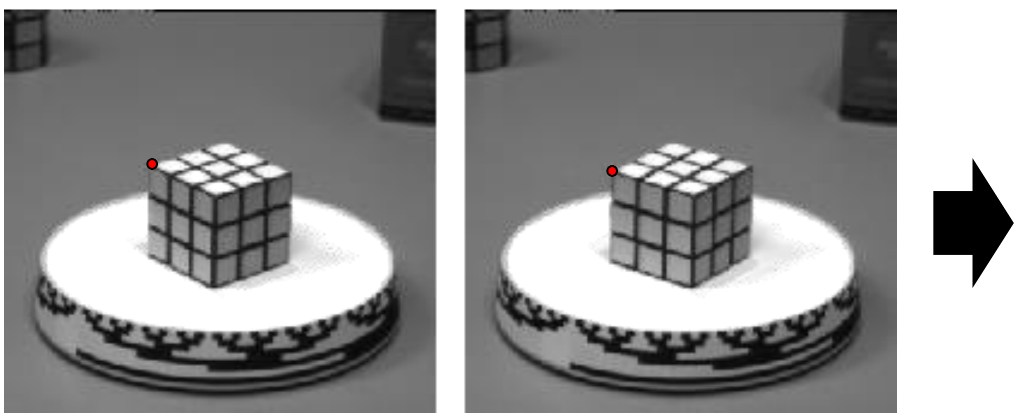
Sequence of images in video



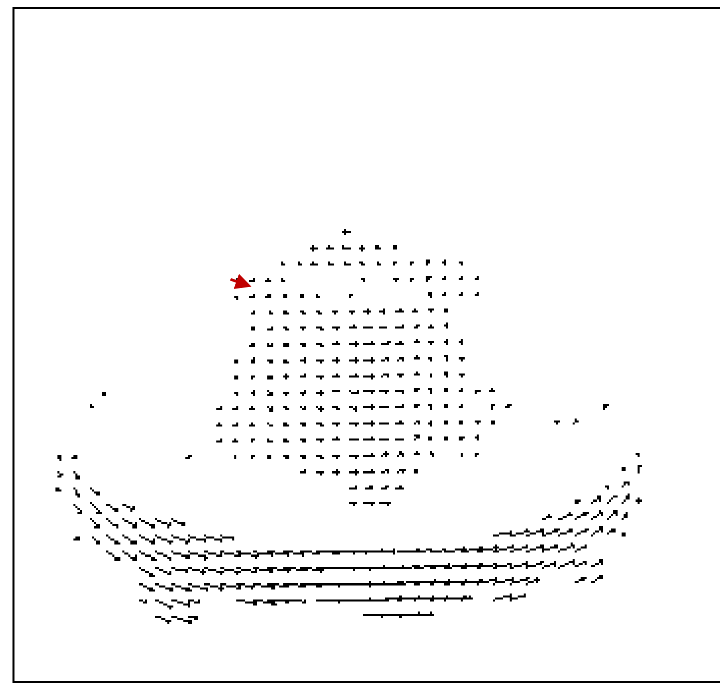
Motion field

Feature-based Motion Estimation

- Detect features in images
- Find correspondences between frames
  - Similar to mosaicing, but can track features based on continuous motion (more on this next time)

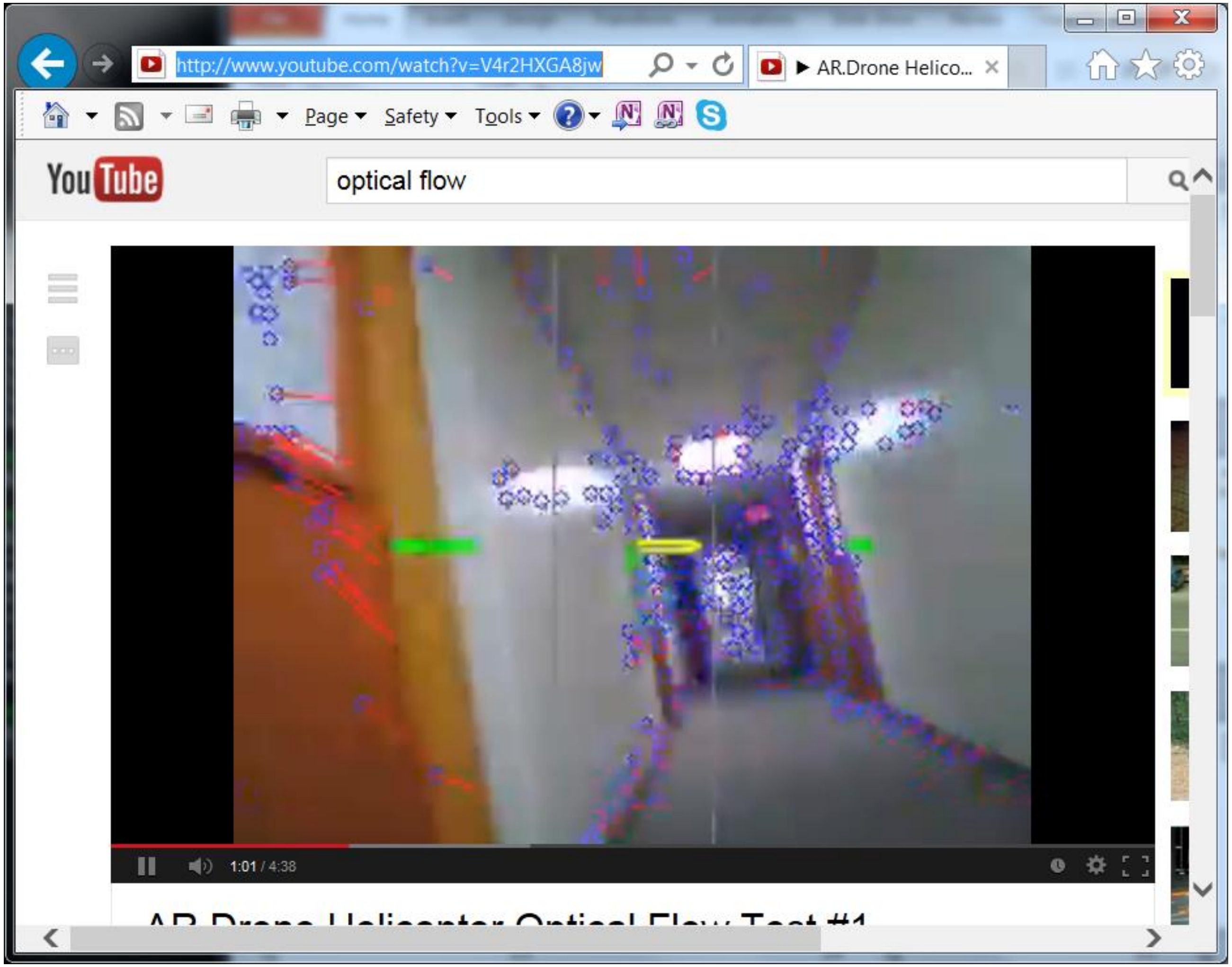


Sequence of images in video



Motion field

Feature-Based Motion Estimation



http://www.youtube.com/watch?v=V4r2HXGA8jw

Feature-based Motion Estimation

- Pros:
  - Provides robust tracking of some points
  - Suitable for large motions
- Cons:
  - Sparse motion field

Pixel-based Motion Estimation

Directly recover image motion at each pixel from spatio-temporal image brightness variations

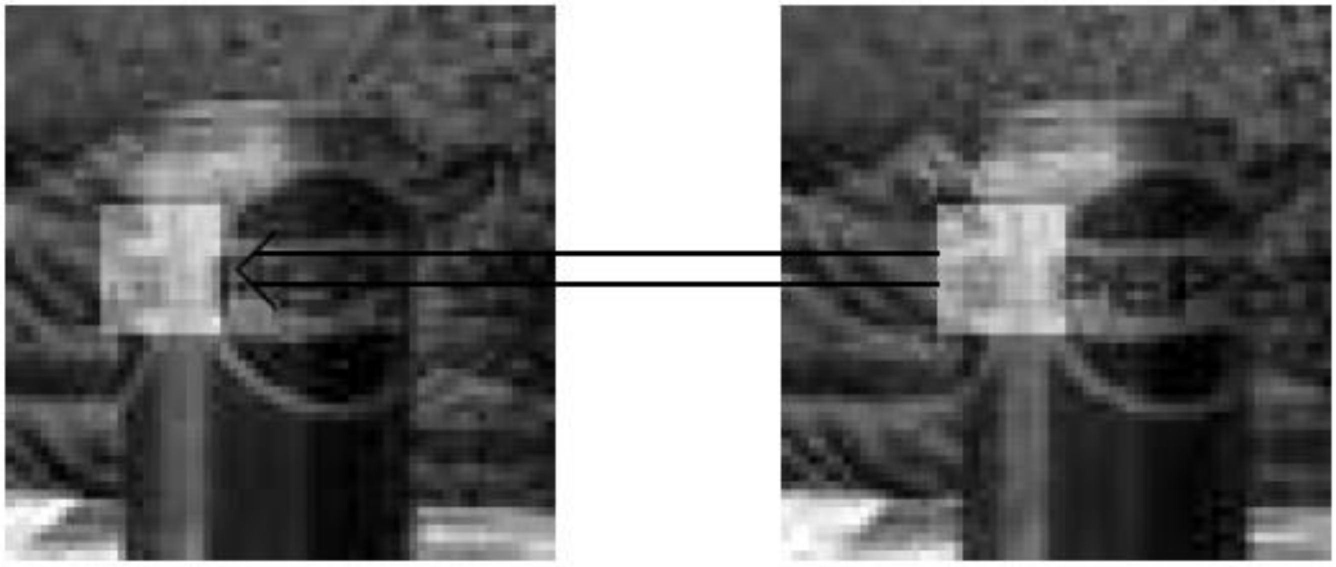


Figure by Michael Black

Pixel-based Motion Estimation

- Note: motion of pixels (optical flow) may not match motion in camera or scene
- Optical flow can be caused by scene motion, camera motion, lighting changes, etc.
- Or, may have no optical flow even when scene is changing

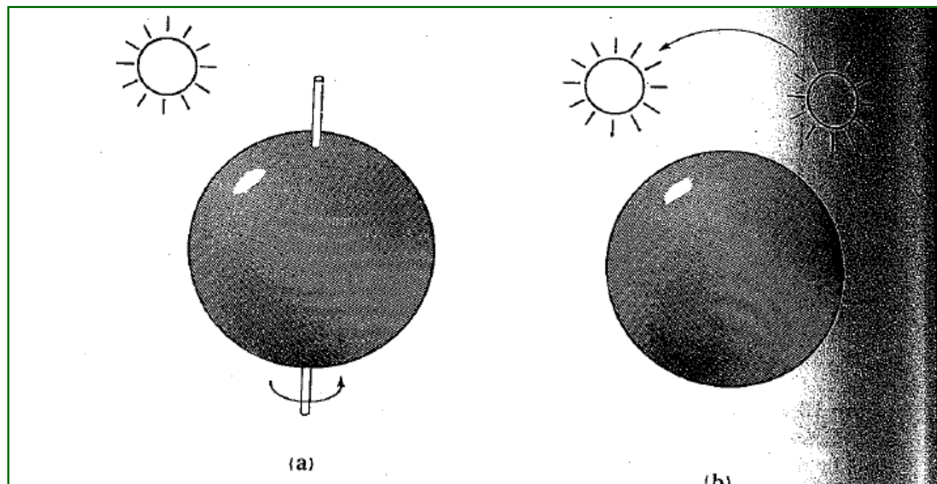
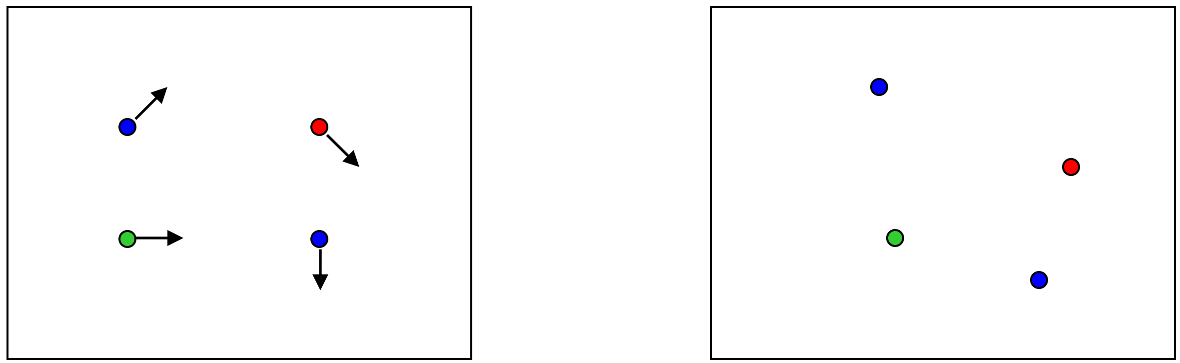


Figure 12-2. The optical flow is not always equal to the motion field. In (a) a smooth sphere is rotating under constant illumination—the image does not change, yet the motion field is nonzero. In (b) a fixed sphere is illuminated by a moving source—the shading in the image changes, yet the motion field is zero.

Figure from Horn book

Problem definition: optical flow

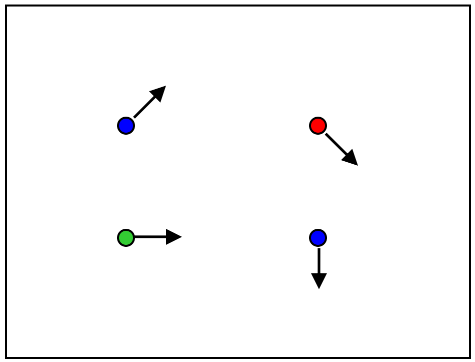


$H(x, y)$ $\qquad$ $I(x, y)$

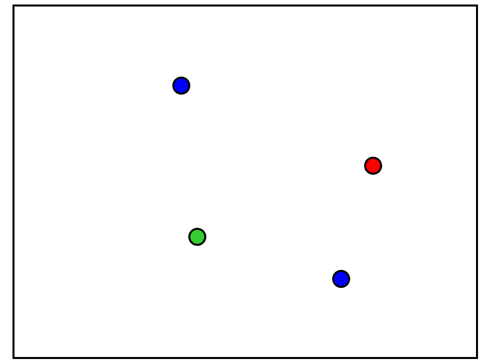
Goal: estimate pixel motion from image H to image I

Steve Seitz

Problem definition: optical flow



$H(x, y)$



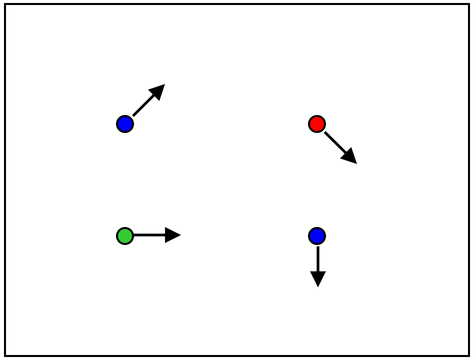
$I(x, y)$

Goal: estimate pixel motion from image H to image I

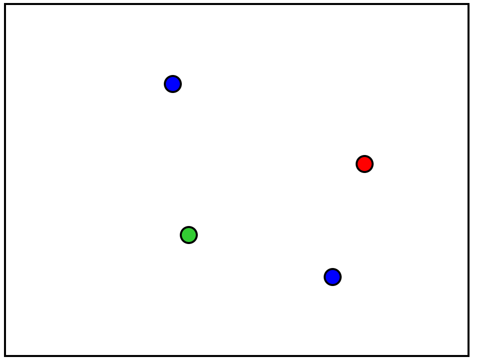
General strategy: for blocks of pixels in H, look for pixels in I that are both nearby and similar-looking

Steve Seitz

Problem definition: Optical flow



$H(x, y)$



$I(x, y)$

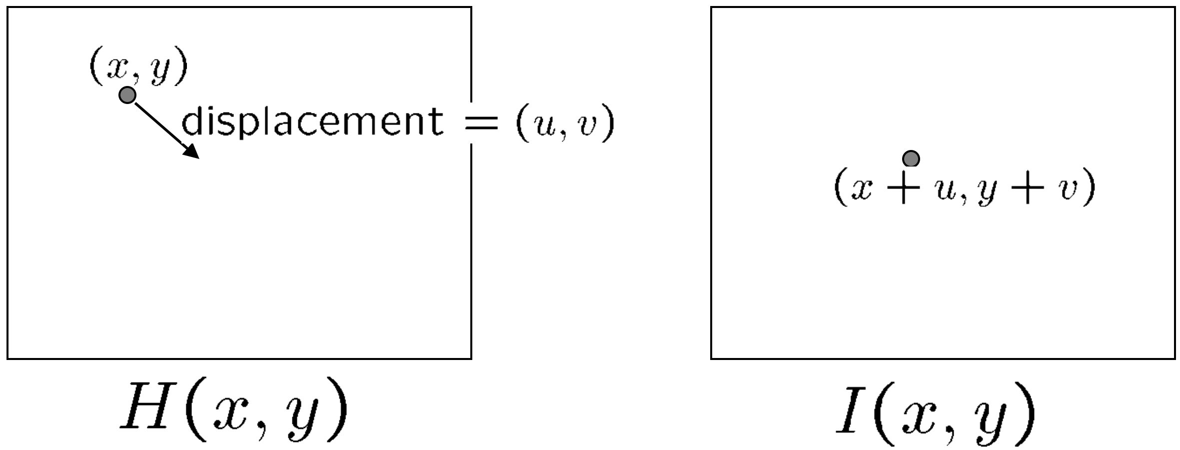
Goal: estimate pixel motion from image H to image I

General strategy: for blocks of pixels in H, look for pixels in I that are both nearby and similar-looking

- Key assumptions
  - **small motion**: points do not move very far
  - **color constancy**: a point in H looks the same in I
  - coherent motion: nearby points move together

Steve Seitz

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:

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

Steve Seitz

Optical flow equation

Combining these two equations

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

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

Steve Seitz

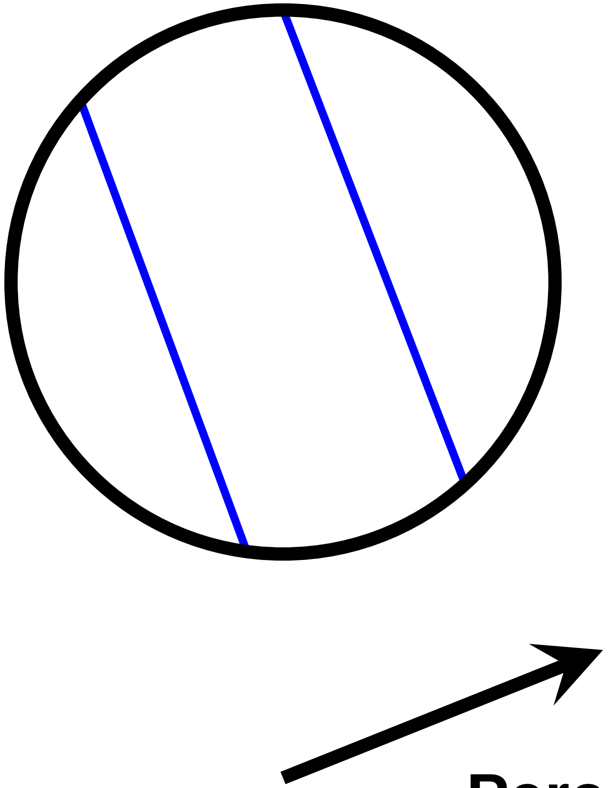
Optical flow equation

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

Q: how many unknowns and equations per pixel?

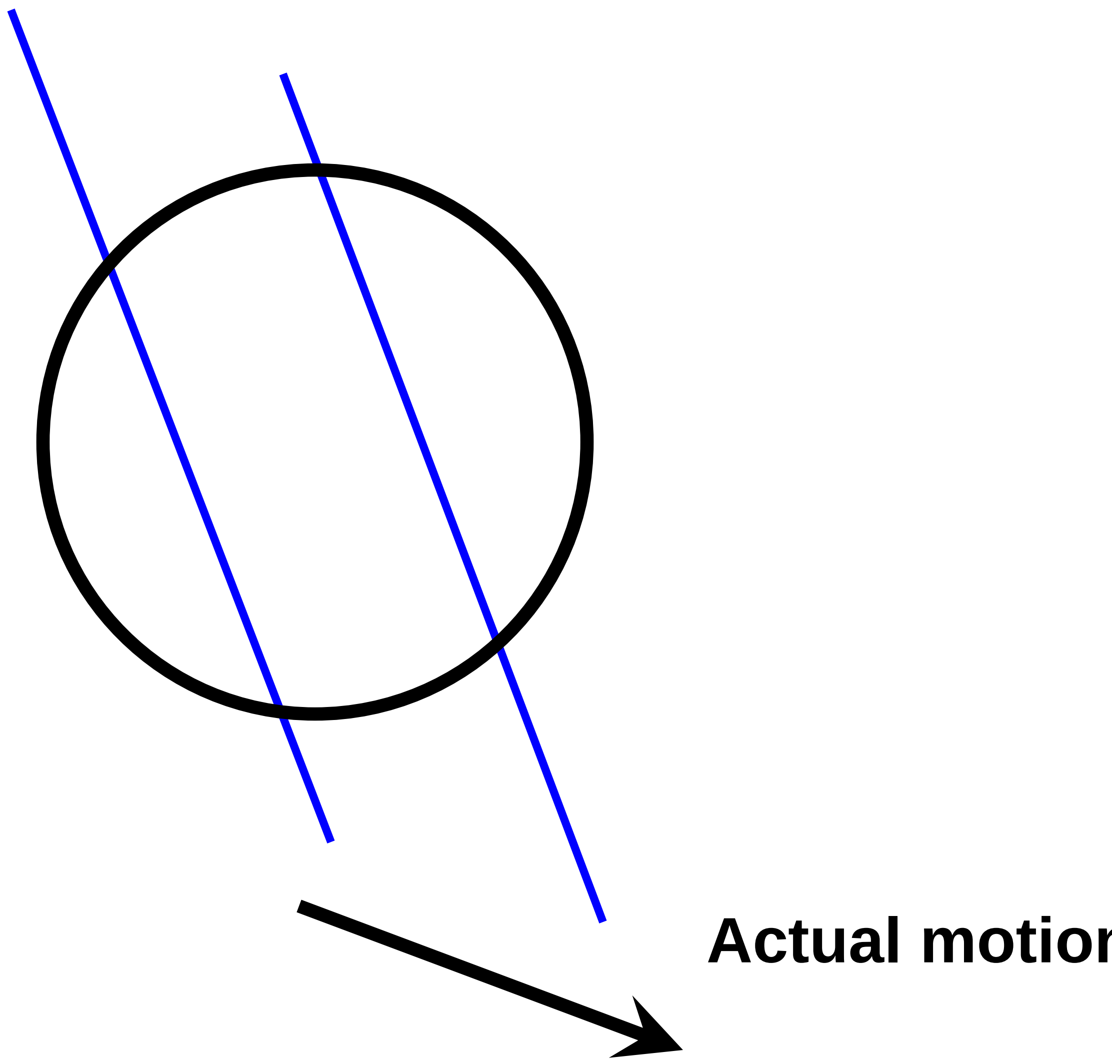
Intuitively, what does this ambiguity mean?

The aperture problem

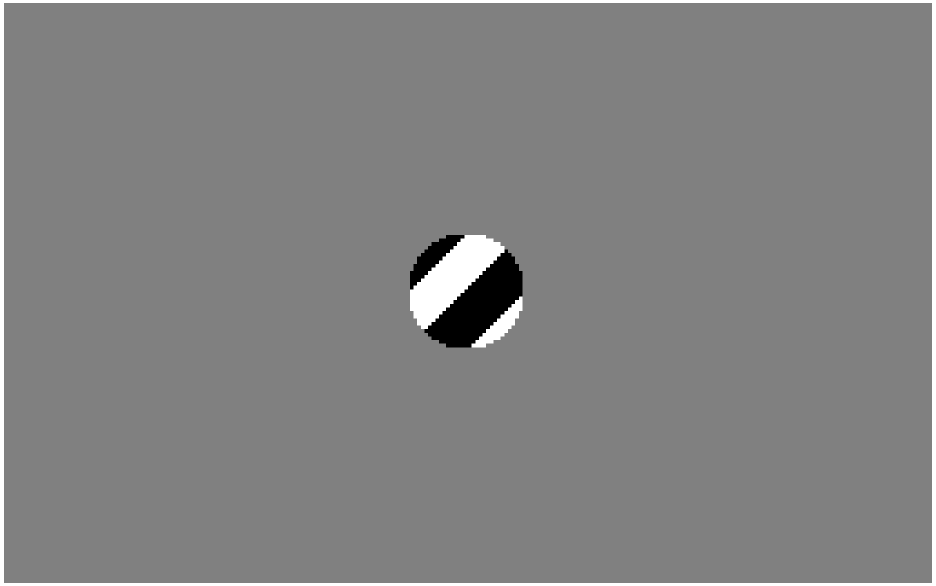


Perceived motion

The aperture problem

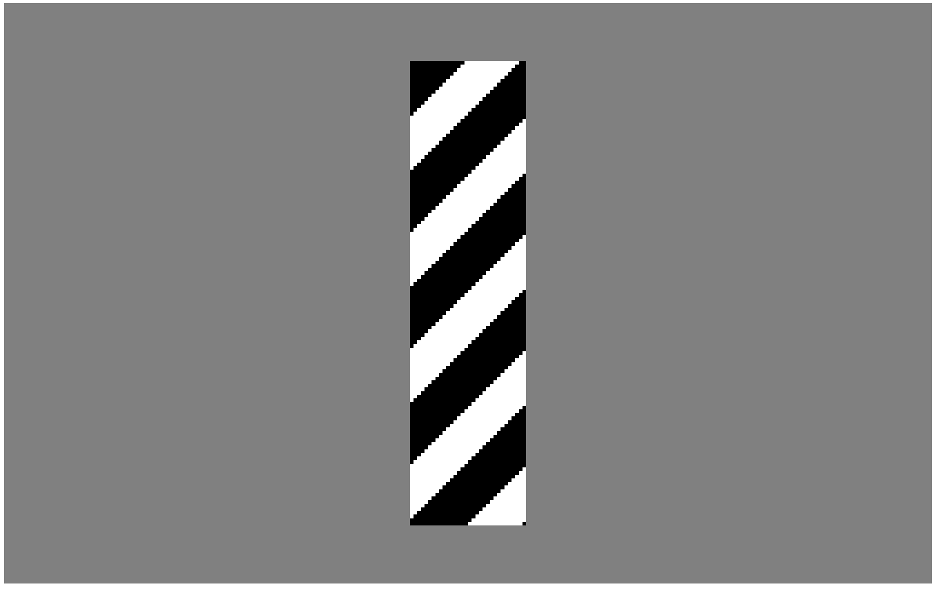


The barber pole illusion



http://en.wikipedia.org/wiki/Barberpole_illusion

The barber pole illusion



http://www.sandlotscience.com/Ambiguous/Barberpole_Illusion.htm

Computing Optical Flow

- How to get more equations for a pixel?
- **Spatial coherence constraint:** pretend the pixel's neighbors have the same (u,v)

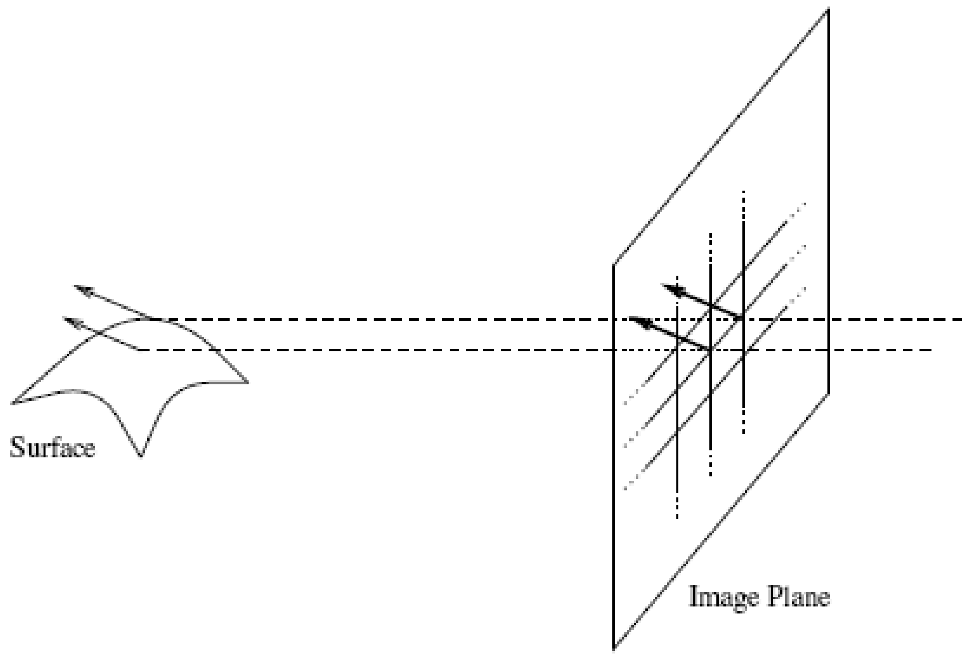


Figure 1.7: Spatial coherence assumption. Neighboring points in the image are assumed to belong to the same surface in the scene.

Figure by Michael Black

Computing Optical Flow

- How to get more equations for a pixel?
- **Spatial coherence constraint:** pretend the pixel's neighbors have the same (u,v)
  - If we use a 5x5 window, that gives us 25 equations per pixel

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

$$\begin{bmatrix} I_x(\mathbf{p_1}) & I_y(\mathbf{p_1}) \\ I_x(\mathbf{p_2}) & I_y(\mathbf{p_2}) \\ \vdots & \vdots \\ I_x(\mathbf{p_{25}}) & I_y(\mathbf{p_{25}}) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(\mathbf{p_1}) \\ I_t(\mathbf{p_2}) \\ \vdots \\ I_t(\mathbf{p_{25}}) \end{bmatrix}$$

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

Steve Seitz

Computing Optical Flow

Now we have more equations than unknowns

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

Solve least squares problem

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

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

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

- The summations are over all pixels in the K x K window

Computing Optical Flow

$$\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$$

$$A^T A \qquad\qquad\qquad A^T b$$

When is this solvable robustly?

- **A^{T}A** should be invertible
- **A^{T}A** should not be too small
  - – 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)

Computing Optical Flow

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

Where have we seen this matrix before?

Computing Optical Flow

Edge:

– gradients very large or very small

– large λ_1, small λ_2

Computing Optical Flow

Low texture region:

– gradients have small magnitude

– small λ_1, small λ_2

Computing Optical Flow

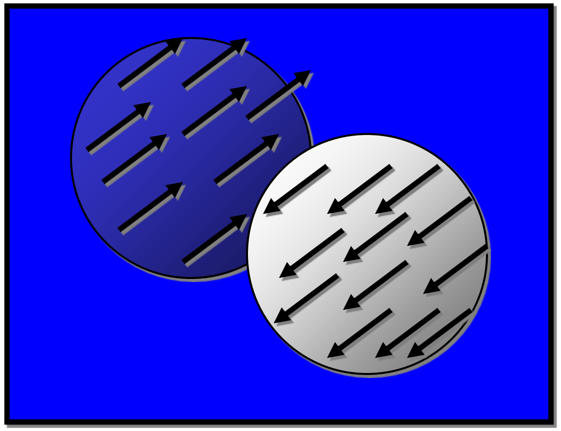
High texture region:

– gradients are different, large magnitudes

– large λ_1, large λ_2

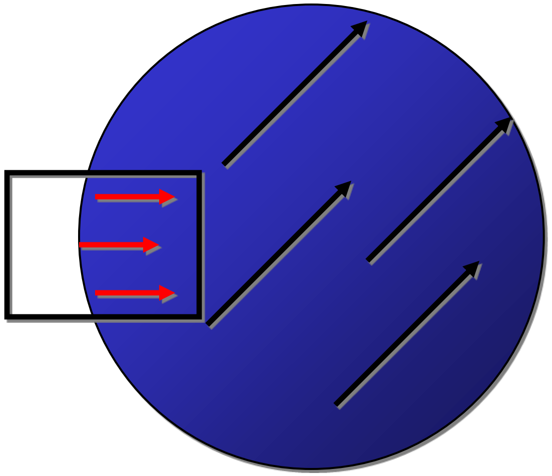
Computing Optical Flow

Still must choose window size:



Too big:
  confused by
  multiple motions

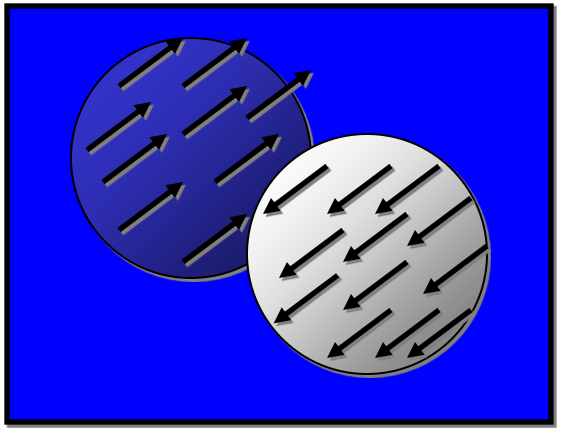
Too small:
  only get motion
  perpendicular
  to edge

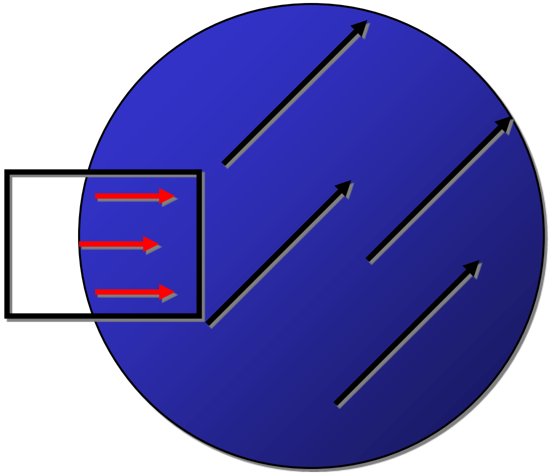


Computing Optical Flow: Improvements

Problem:

- Assumption that optical flow is constant over neighborhood is not always good





Computing Optical Flow: Improvements

Improvement 1:

- Use large neighborhood, but weight pixels higher if closer to center

$$\mathbf{A} \to \mathbf{WA}$$

$$\mathbf{b} \to \mathbf{Wb}$$

$$\mathbf{v} = -(\mathbf{A}^{\mathrm{T}}\mathbf{A})^{-1}\mathbf{A}^{\mathrm{T}}\mathbf{b}$$

$$\Rightarrow \mathbf{v}_w = -(\mathbf{A}^{\mathrm{T}}\mathbf{W}^2\mathbf{A})^{-1}\mathbf{A}^{\mathrm{T}}\mathbf{W}^2\mathbf{b}$$

Computing Optical Flow: Improvements

Improvement 2:

- Use affine model of motion (instead of translation)
- Must solve for 6 unknowns per pixel instead of 2

Translation:

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

Affine:

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

Computing Optical Flow: Improvements

Problem:

- Small motion assumption not always true
- i.e., differential approximation not good for large motions

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

Computing Optical Flow: Improvements

Improvement 1: iteration

Add higher order terms back in and solve with iterative algorithm

$$0 = I(x+u, y+v) - H(x,y)$$

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

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

This is a polynomial root finding problem

- Can solve using **Newton's method**
  - Also known as **Newton-Raphson** method
- Approach so far does one iteration of Newton's method
  - Better results are obtained via more iterations
- Warp image based on estimated flow after each iteration

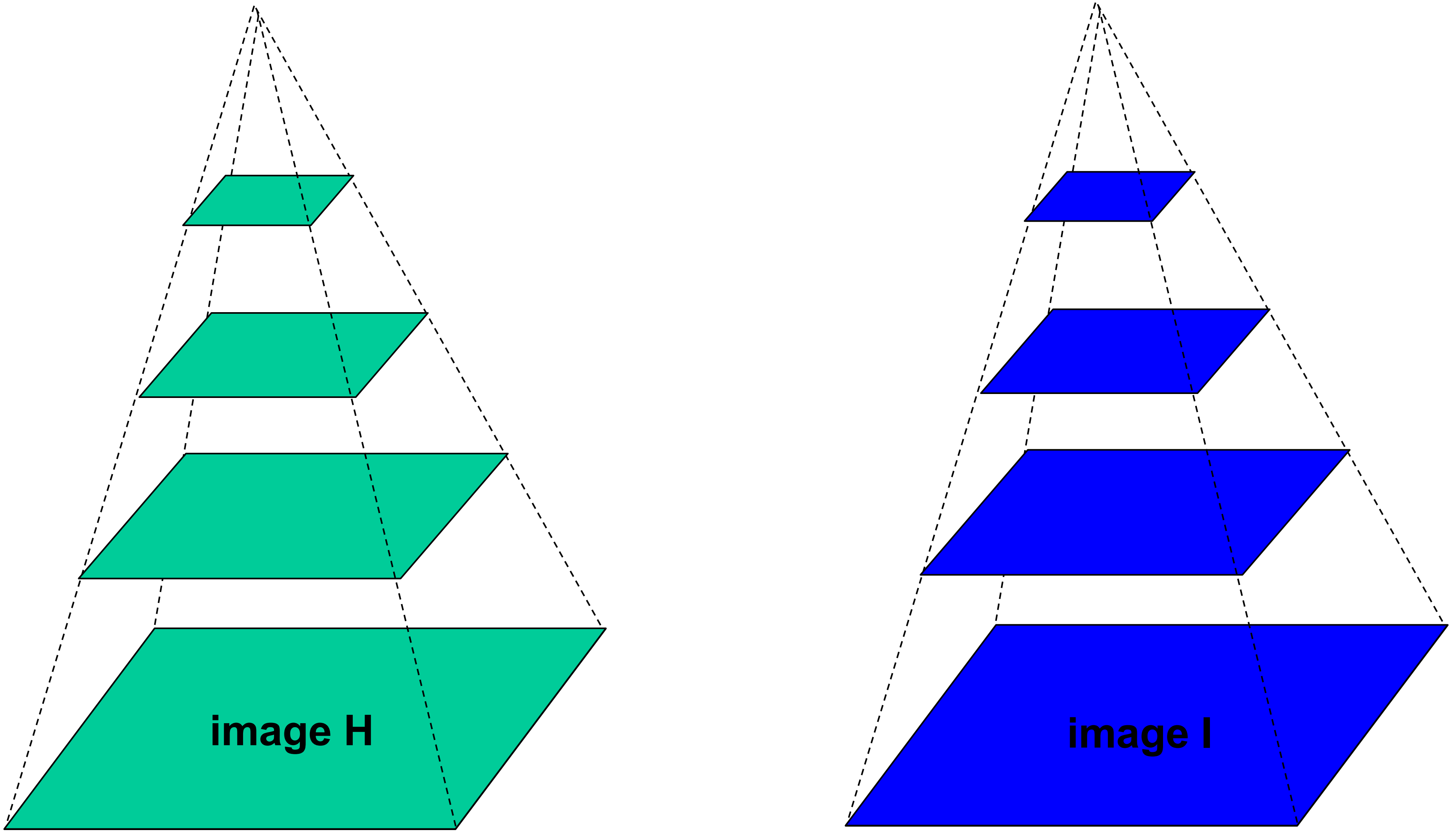
Computing Optical Flow: Improvements

Improvement 2: multiresolution

- Use large-scale gradients in early iterations, smaller-scale in late iterations (coarse-to-fine)

Computing Optical Flow: Improvements

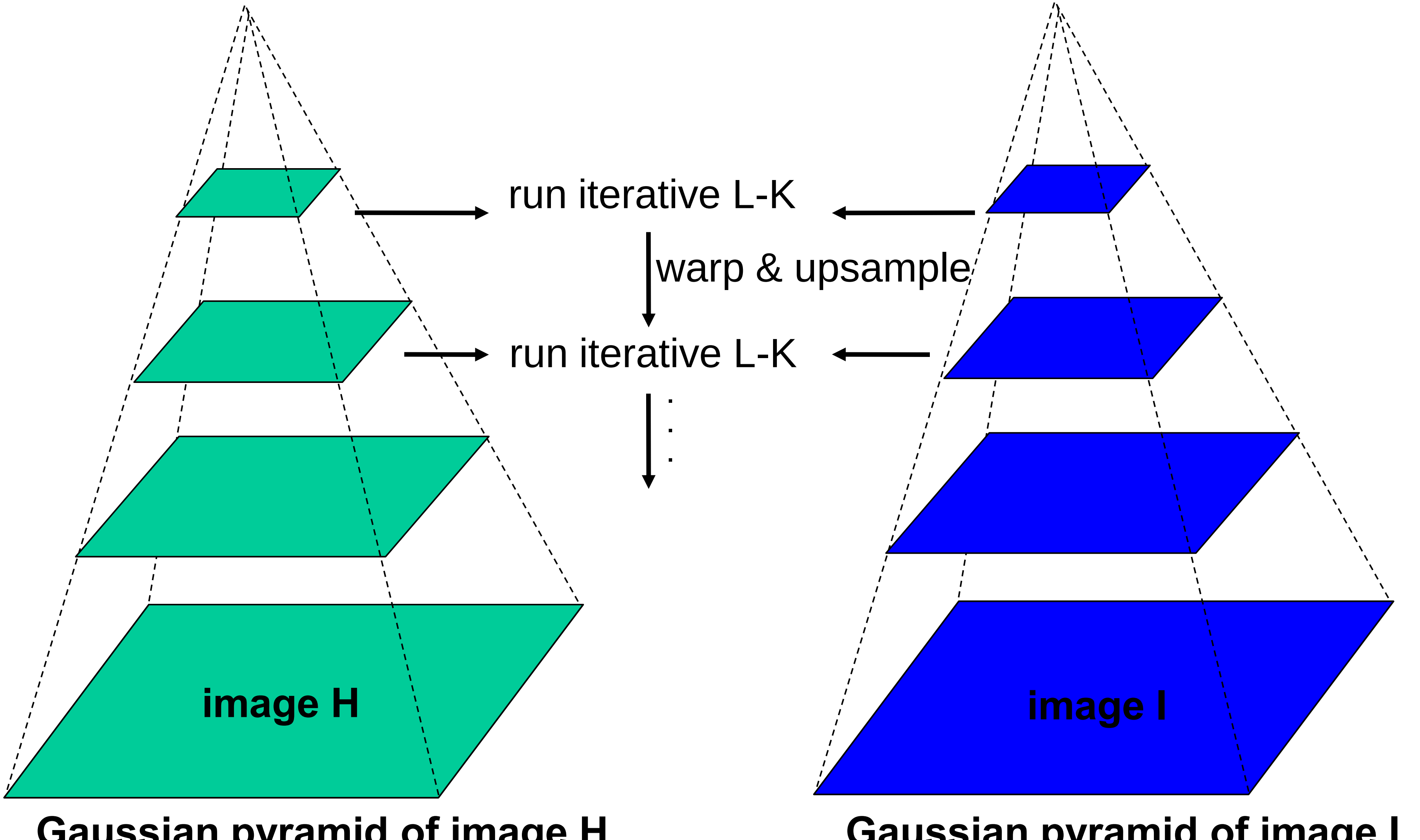
Improvement 2: multiresolution



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

Computing Optical Flow: Improvements

Improvement 2: multiresolution



Computing Optical Flow: Lucas-Kanade

Coarse-to-fine, iterative algorithm:

1. Set σ = large (e.g. 10 pixels)
2. Set I' $\leftarrow$ I_1
3. Set **v** $\leftarrow$ 0
4. Repeat while SSD(I', I_2) > τ
   1. **v** += Optical flow(I' $\rightarrow$ I_2)
   2. I' $\leftarrow$ Warp(I_1, **v**)
5. After *n* iterations,
   set σ = small (e.g. 1 pixels)

Outline

Motivation

Algorithms

Evaluation

Applications

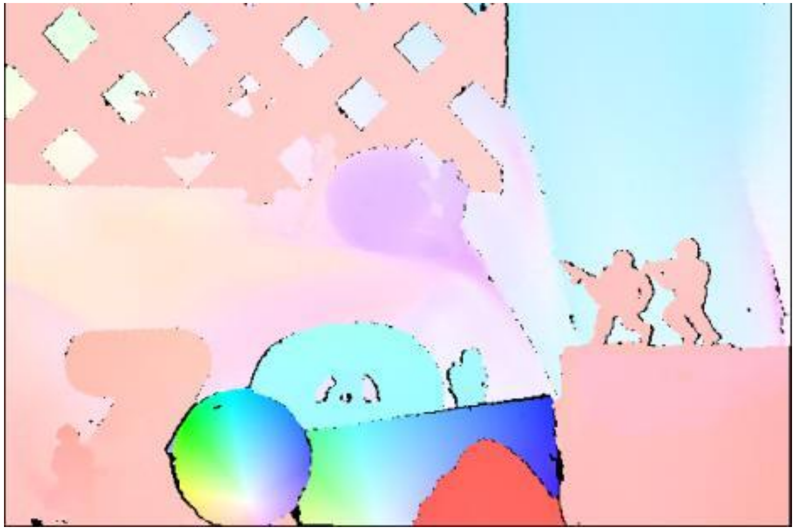
Evaluation of Optical Flow Algorithms

Evaluation of Optical Flow Algorithms

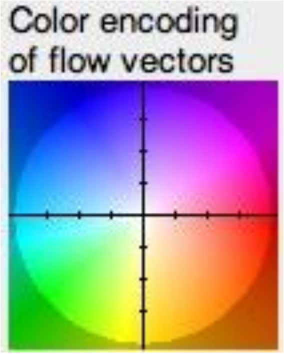
Evaluation of Optical Flow Algorithms

Optical flow benchmarks

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



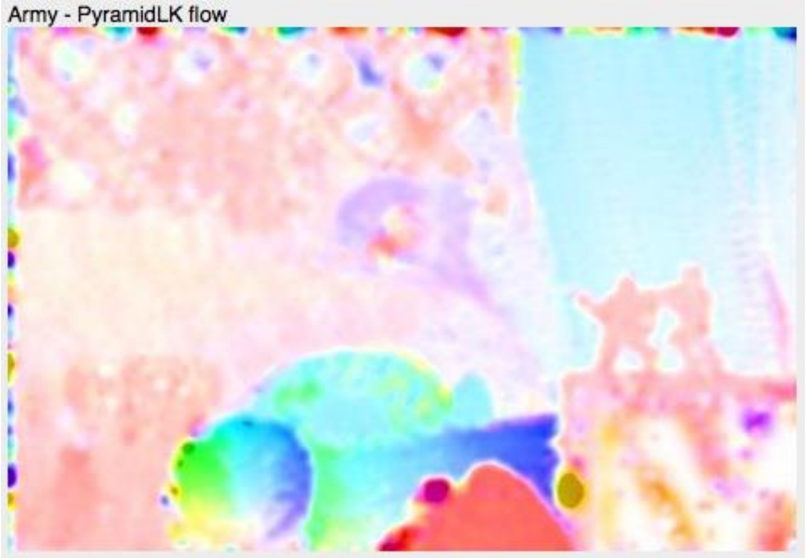
Ground Truth



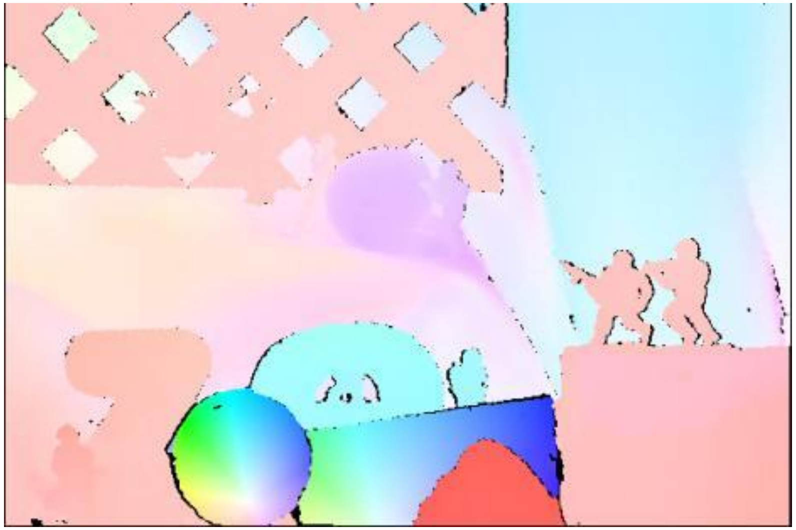
Evaluation of Optical Flow Algorithms

Optical flow benchmarks

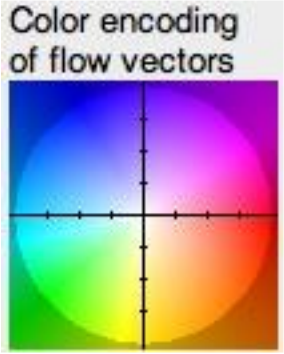
- http://vision.middlebury.edu/flow/



Lucas-Kanade flow



Ground Truth

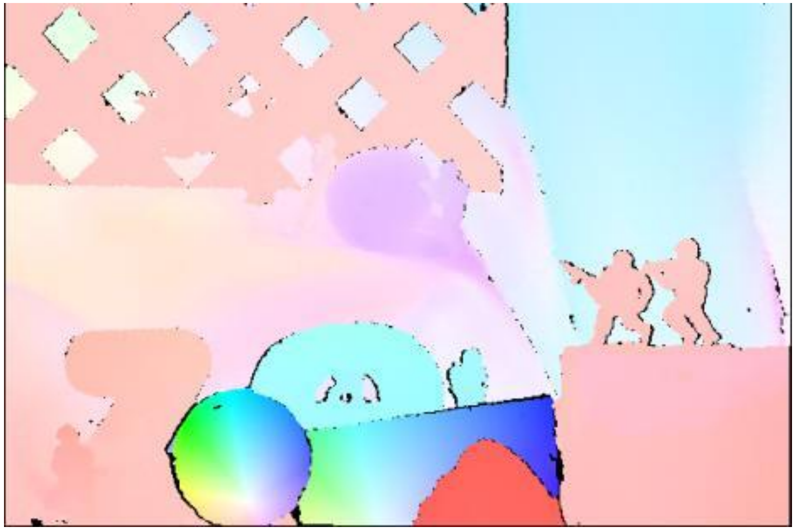


Evaluation of Optical Flow Algorithms

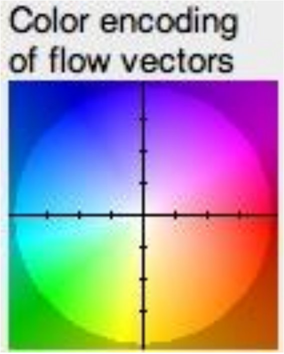
Optical flow benchmarks

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

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



Ground Truth



Outline

Motivation

Algorithms

Evaluation

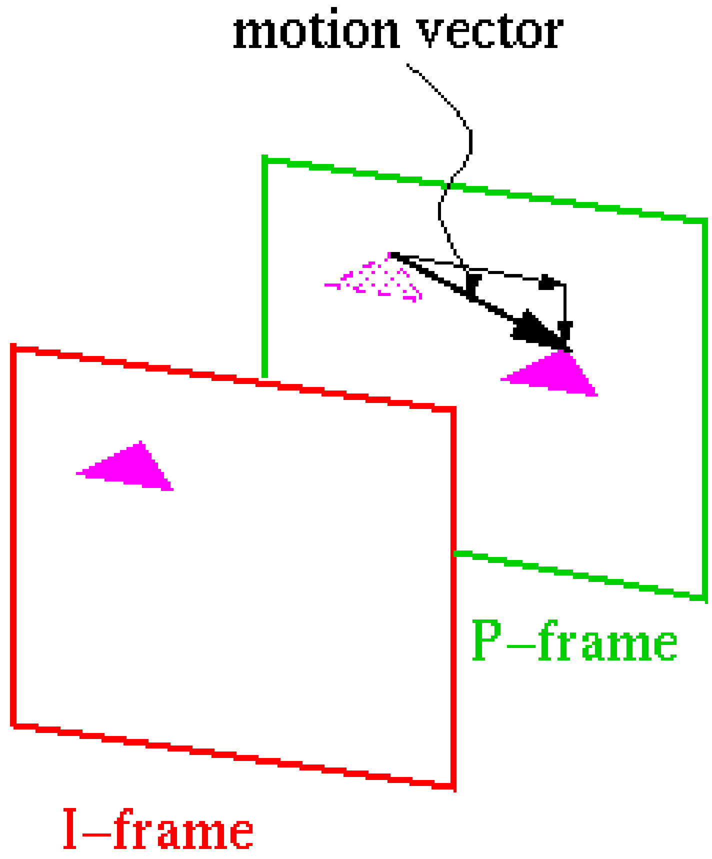
Applications

Applications

- Estimating depth
- Tracking object motion
- Determining camera motion
- Segmenting objects based on motion cues
- Video compression
- Robot navigation
- Studying dynamical models
- Recognizing events and activities
- Human computer interaction
- Facial animation
- Video filters

Application: video compression

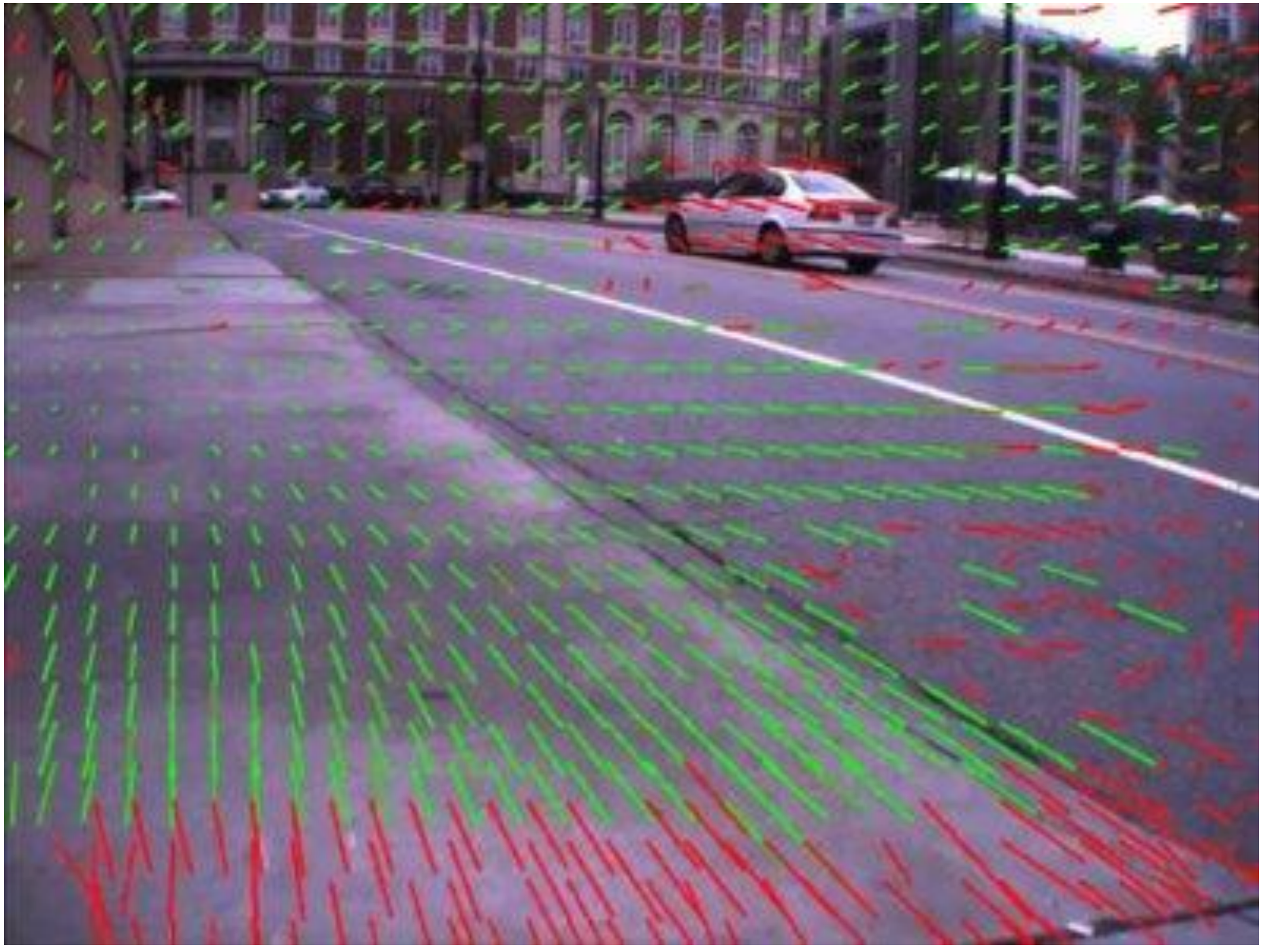
Encode some frames (p-frames) based on motion of blocks in others (i-frames)



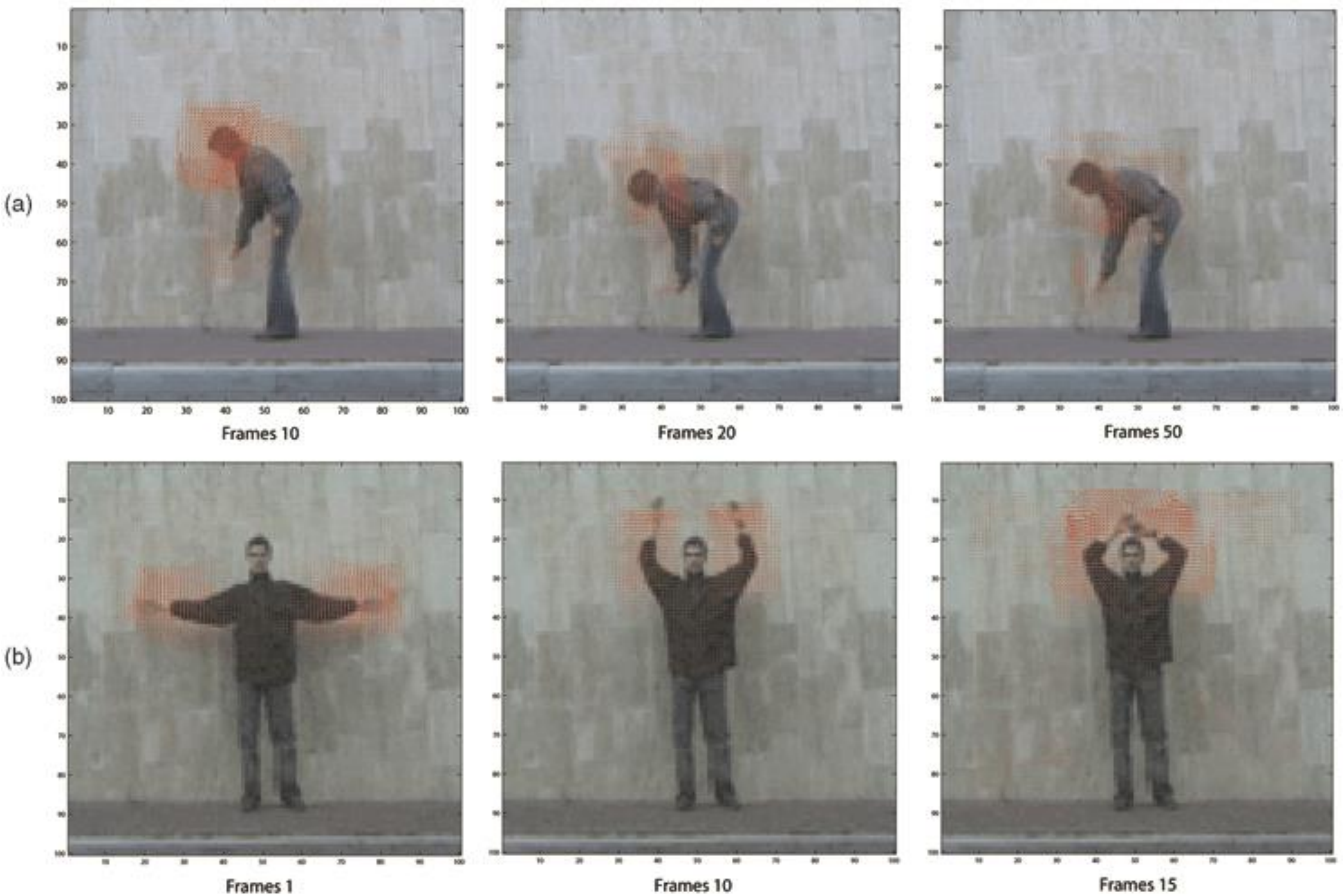
http://vsr.informatik.tu-chemnitz.de/~jan/MPEG/HTML/motion.gif

Application: robot navigation

Scene understanding, obstacle avoidance, etc.



Application: action recognition



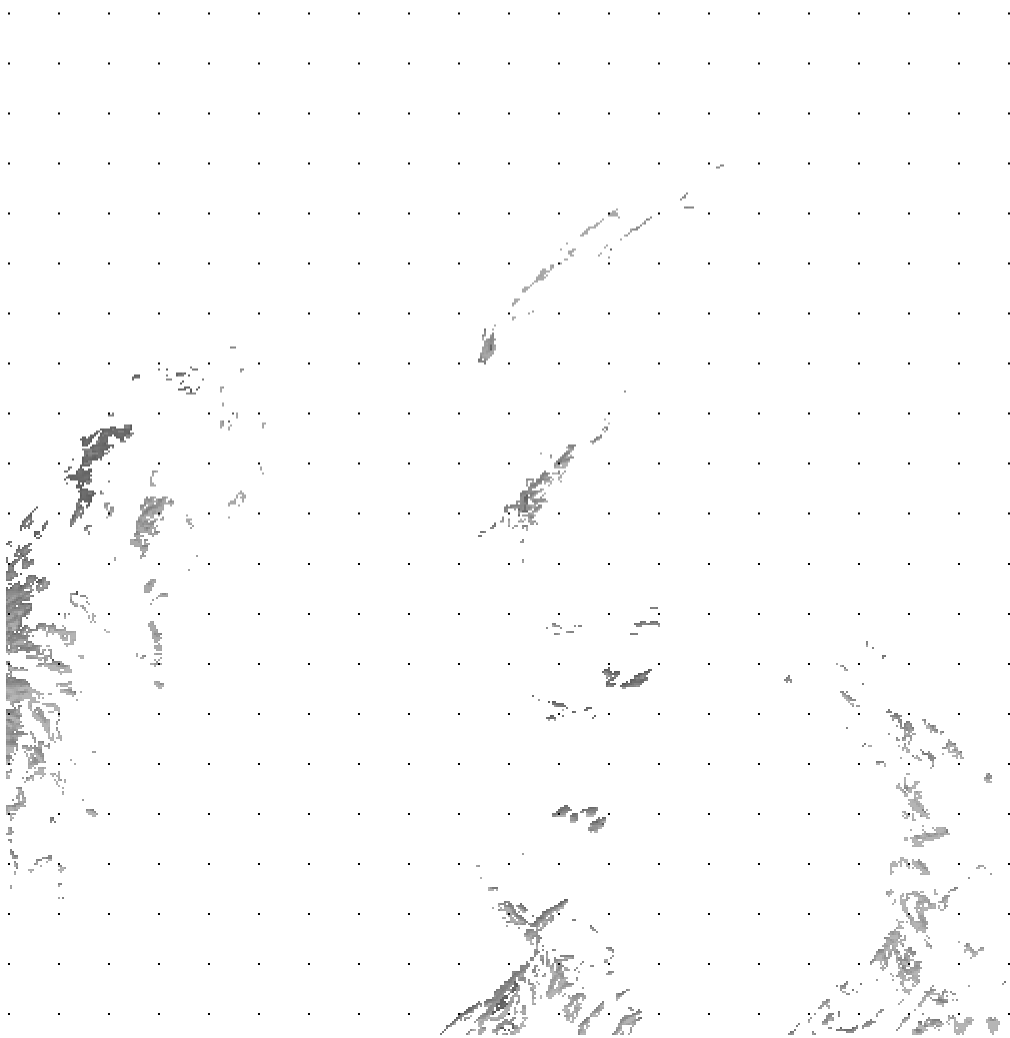
Application: human-computer interaction

Track people (more on this next time)

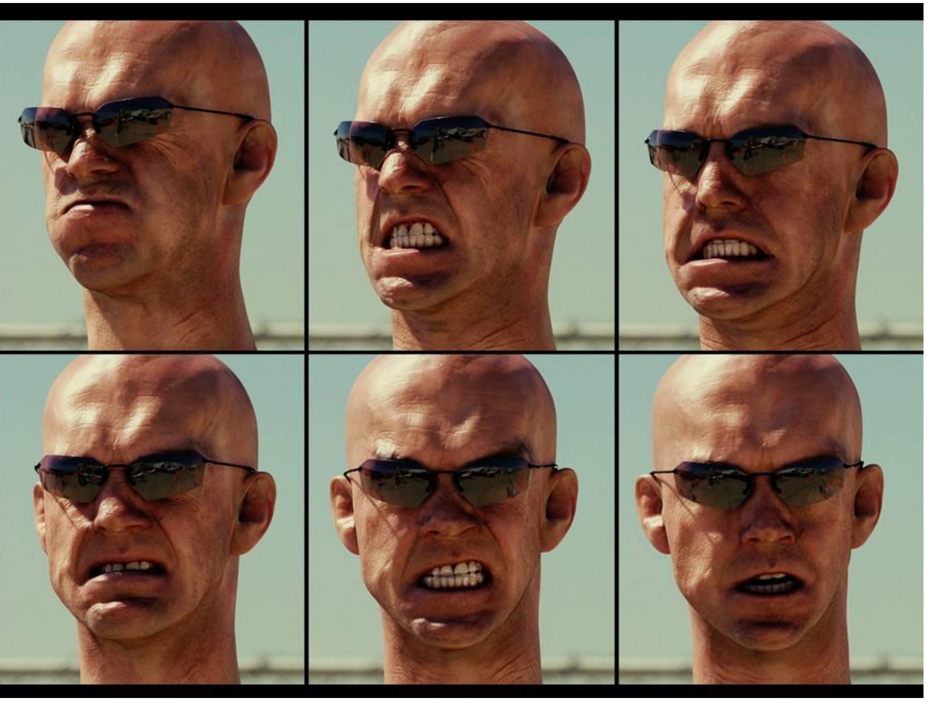
http://www.youtube.com/watch?v=TbJrc6QCeU0&feature=related

Application: studying dynamical systems

Measuring fluid flow



Application: facial animation



http://www.fxguide.com/article333.html

Application: video filters

Track pixels so that can provide coherence in brush strokes when making video appear painted by an artist

http://www.fxguide.com/article333.html

Optical Flow Summary

- Problem:
  - Solve for motion field by minimizing differences in intensity between corresponding pixels
- Techniques:
  - Differential approximation, windows
  - Weighting, iteration, multiresolution
- Pros and cons:

  \+ Dense motion field

  -- Works well only for small motions

  -- Sensitive to appearance variations

Lots of applications