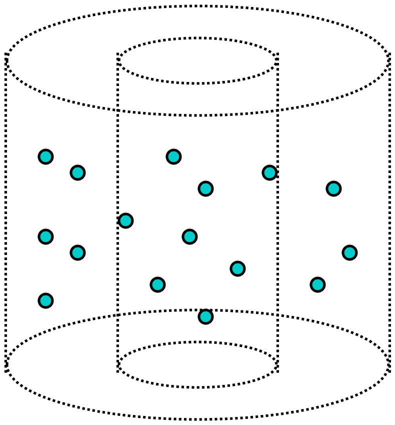
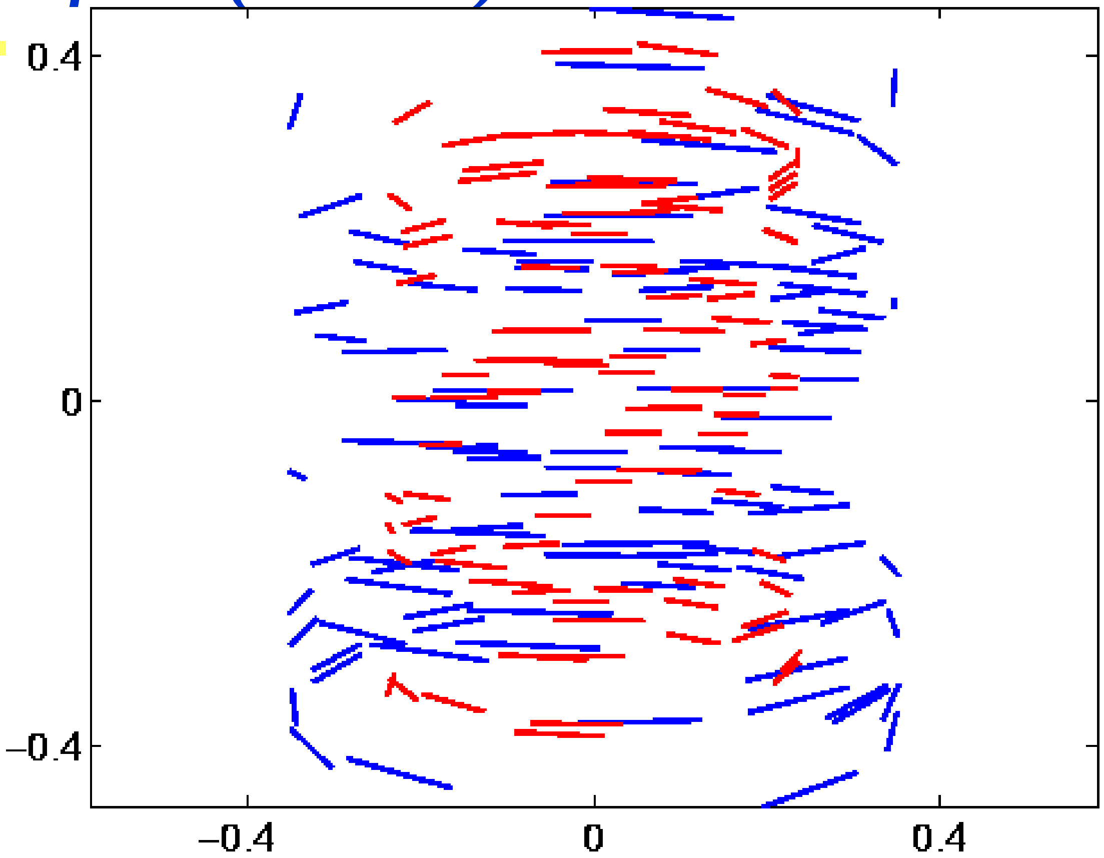
Visual Motion
Analysis and Representation

Example

- Ullman’s concentric counter-rotating cylinder experiment
- Two concentric cylinders of different radii
- W. a random dot pattern on both surfaces (cylinder surfaces and boundaries are not displayed)
- Stationary: not able to tell them apart
- Counter-rotating: structures apparent

Example (cont.)





- Motion helps in
  - segmentation (two structures)
  - identification (two cylinders)

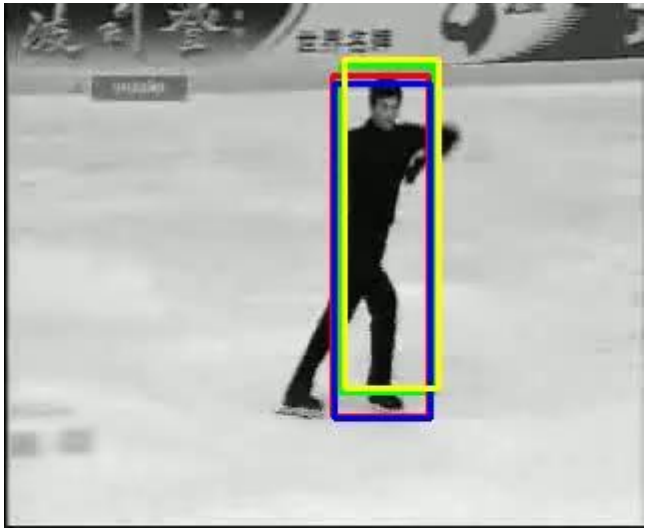
Classes of Techniques

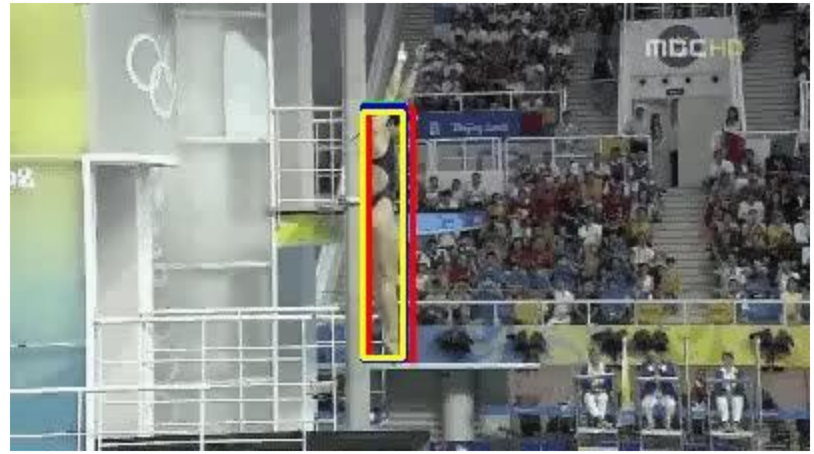
- ***Feature-based methods***
  - Extract visual features (corners, textured areas) and track them
  - Sparse motion fields, but possibly robust tracking
  - Suitable especially when image motion is large (10s of pixels)
- ***Direct-methods (Pixel-based methods)***
  - Directly recover image motion from spatio-temporal image brightness variations
  - Global motion parameters directly recovered without an intermediate feature motion calculation
  - Dense motion fields, but more sensitive to appearance variations
  - Suitable for video and when image motion is small (< 10 pixels)

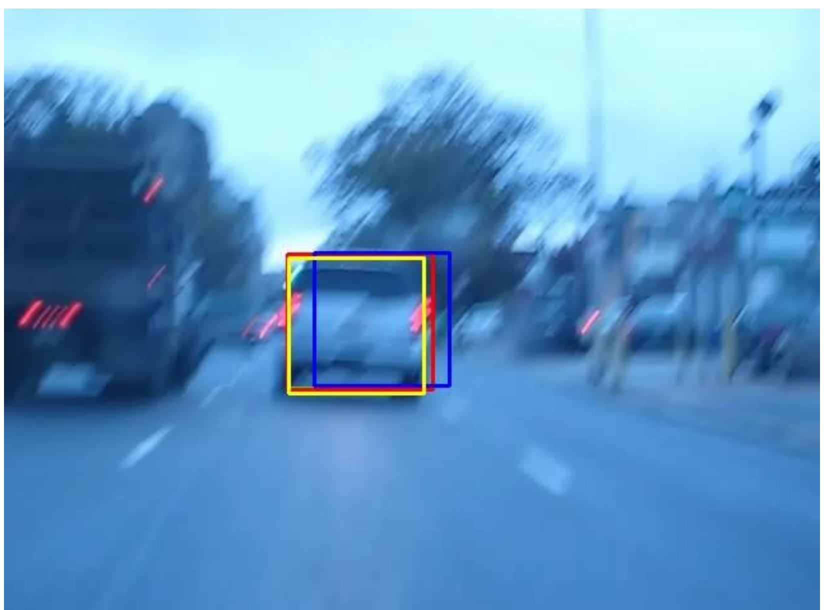
Traditional vs Modern

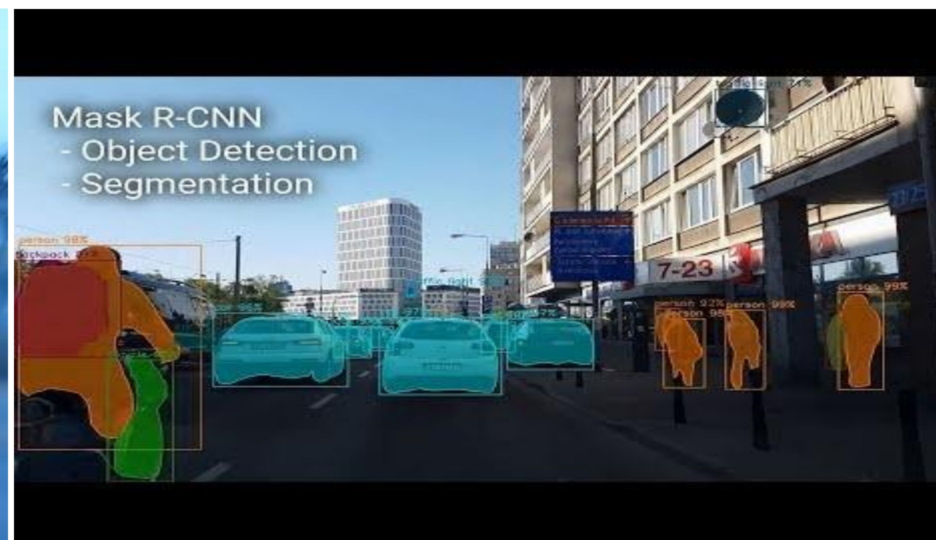
- Syntactic features
- Small deformation
- Precise localization
- Sparse vs dense (vector/pixel) tracking

- Semantic features
- Large deformation
- Global localization
- Object (people, dog, etc.) tracking



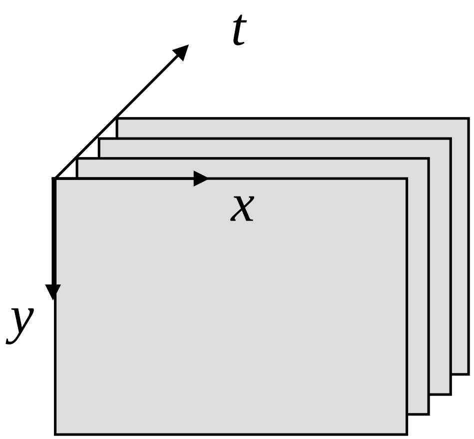





Mask R-CNN
- Object Detection
- Segmentation
7-23

Optical flow and motion analysis

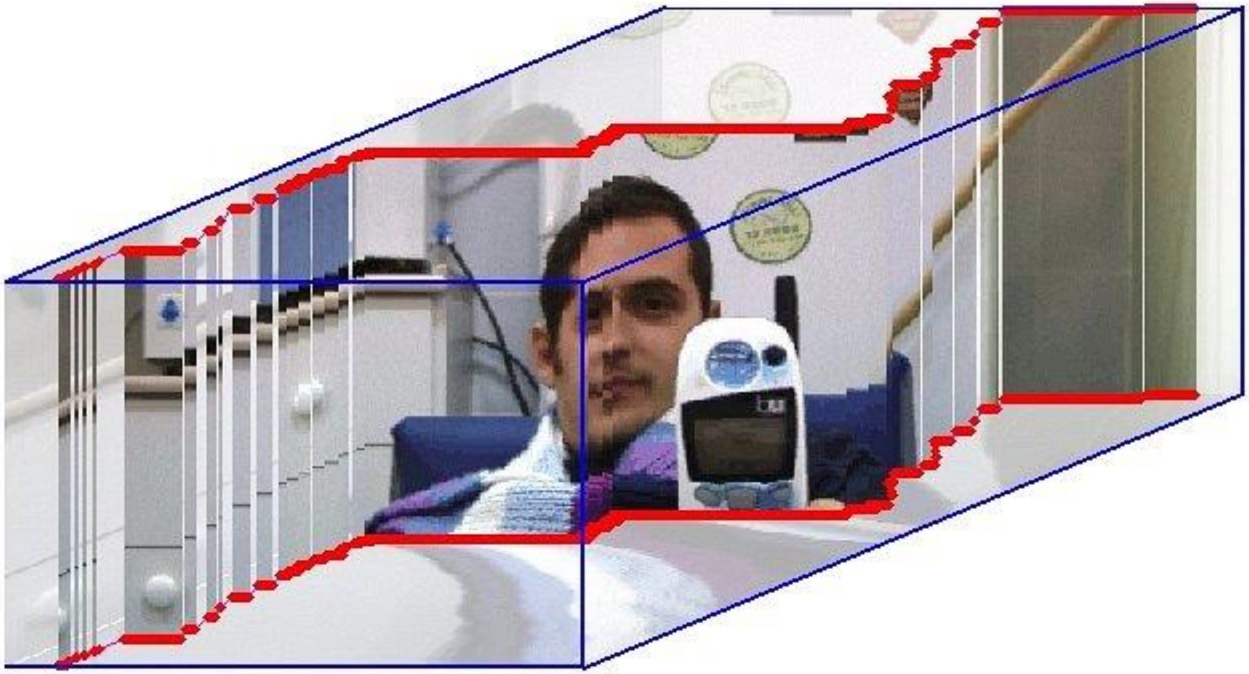
- Now we move to considering images that vary over time – image sequences
  - Typical case is video – images captured at 30 frames/second (or 15, or 60, or ...)
  - $I(x, y, t) \rightarrow I_1(x,y) = I(x, y, t_1)$, $I_2(x,y) = I(x, y, t_2)$, etc.
  - "Spatial-temporal space" describes (x, y, t)

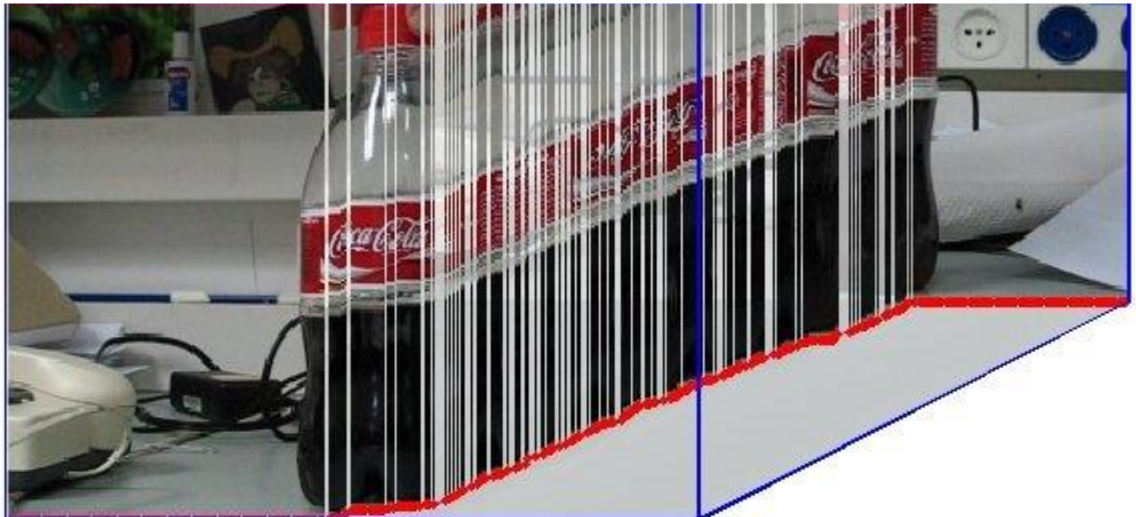


What can change between I_t and I_{t+1}?

What do images close in time have in common?

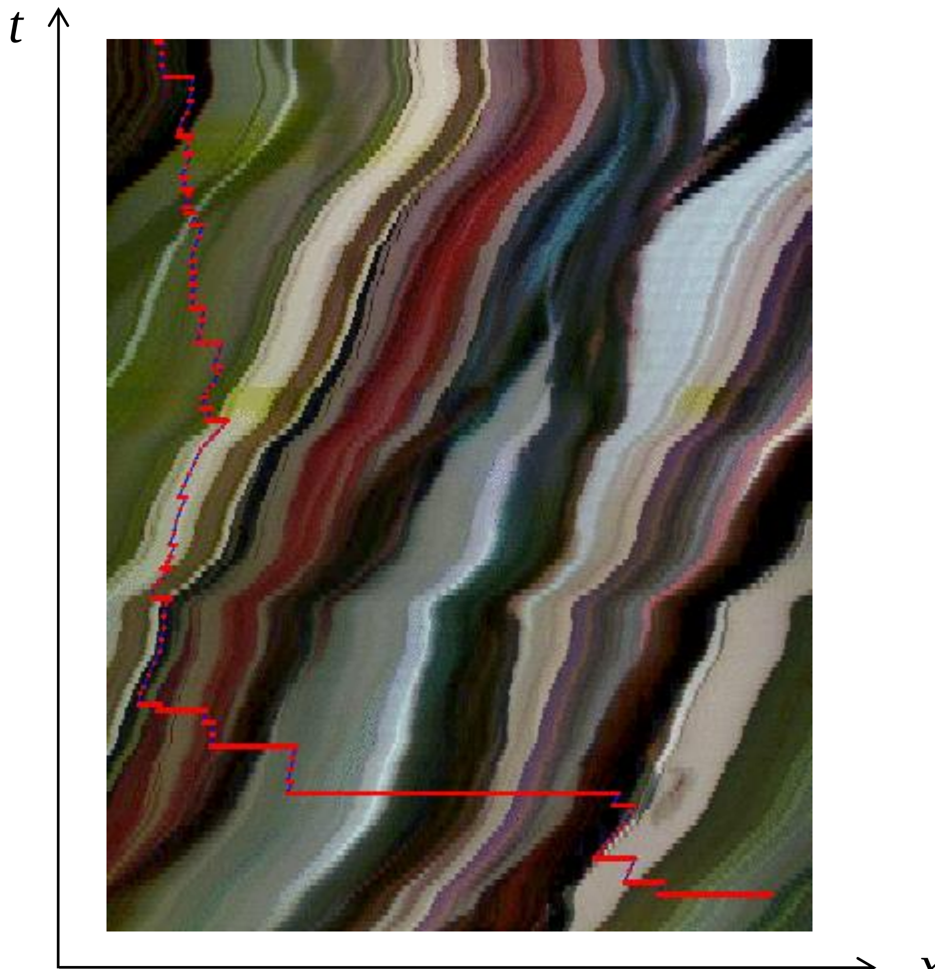
Spatio-temporal image data (examples)





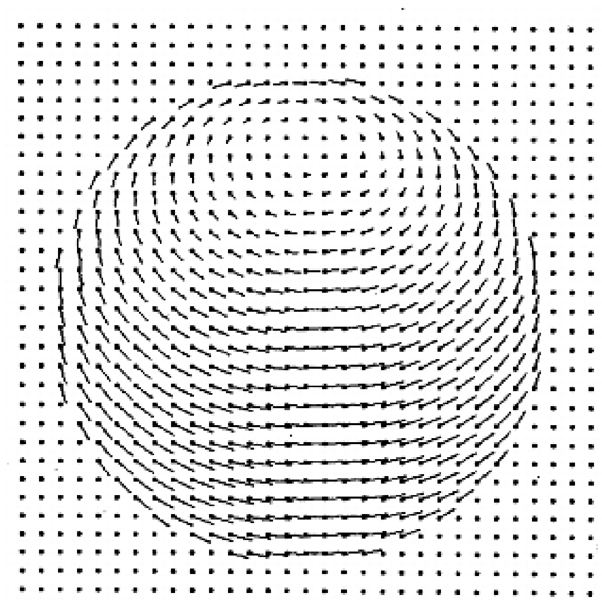
Frames

x-*t* slice



Optical flow and motion analysis

- Optical flow is the *apparent motion* of brightness patterns in the image sequence
  - A 2D vector at each point – a vector field
- The motion field is the *true motion* (3D) at each point, mapped onto the 2D image
  - A vector field
- They are not always the same
  - E.g., white, featureless ball?



In general, we estimate the *motion field* by computing the *optical flow*

The motion field is not *directly* observed

Example

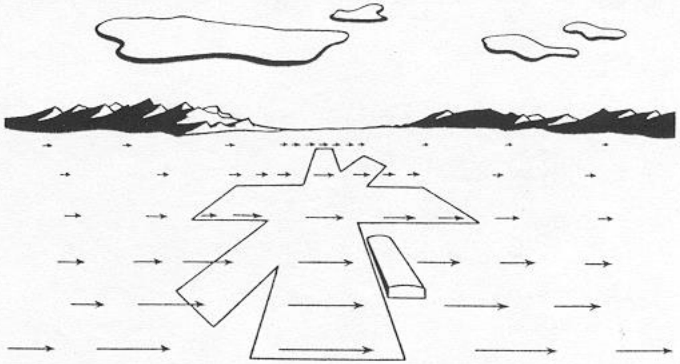
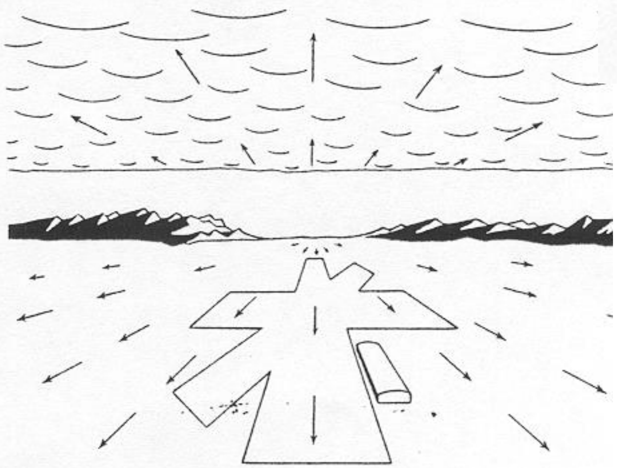
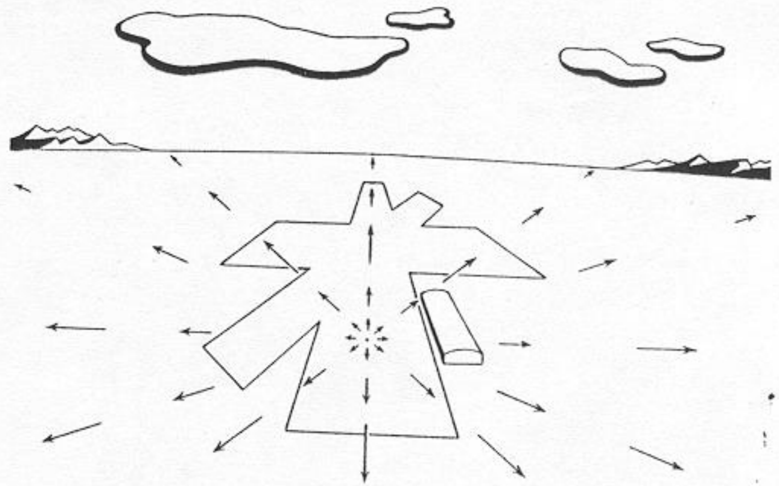
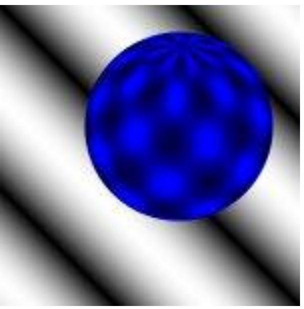


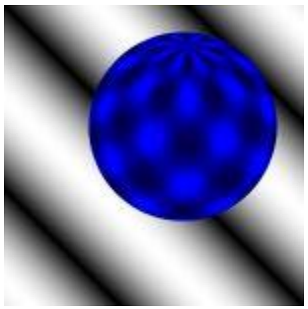
Figure 8.6 The motion field of a pilot looking to the right in level flight. The f
of expansion here is off at infinity to the left of the figure; equivalently, the focu
contraction is off at infinity to the right of the figure. (From [Gibson 1950]
permission. Copyright © 1977, 1950 by Houghton Mifflin Company.)

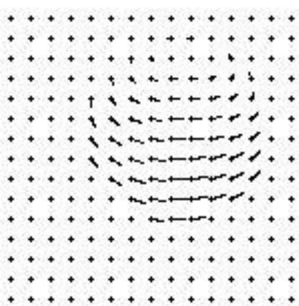




Example







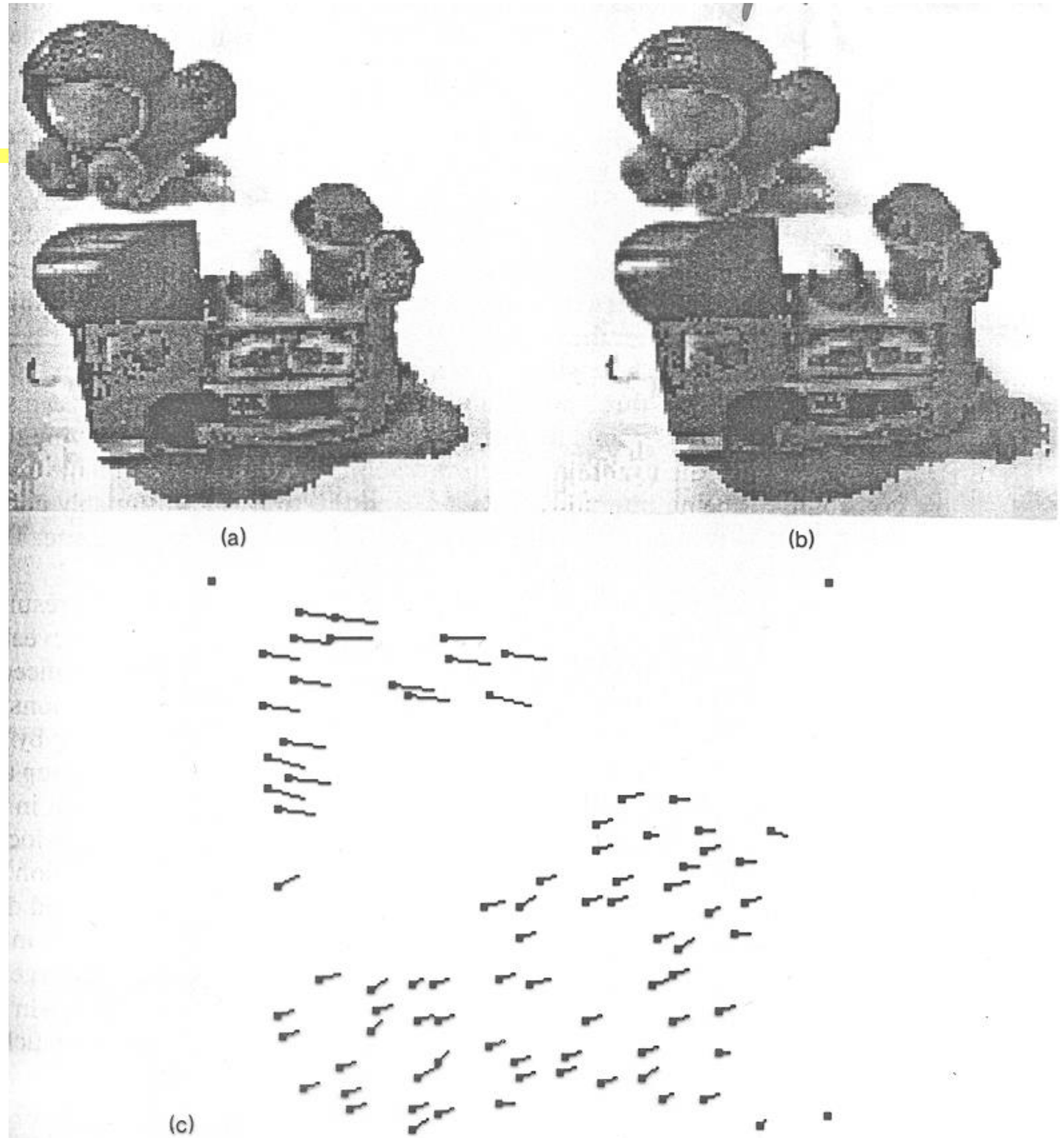
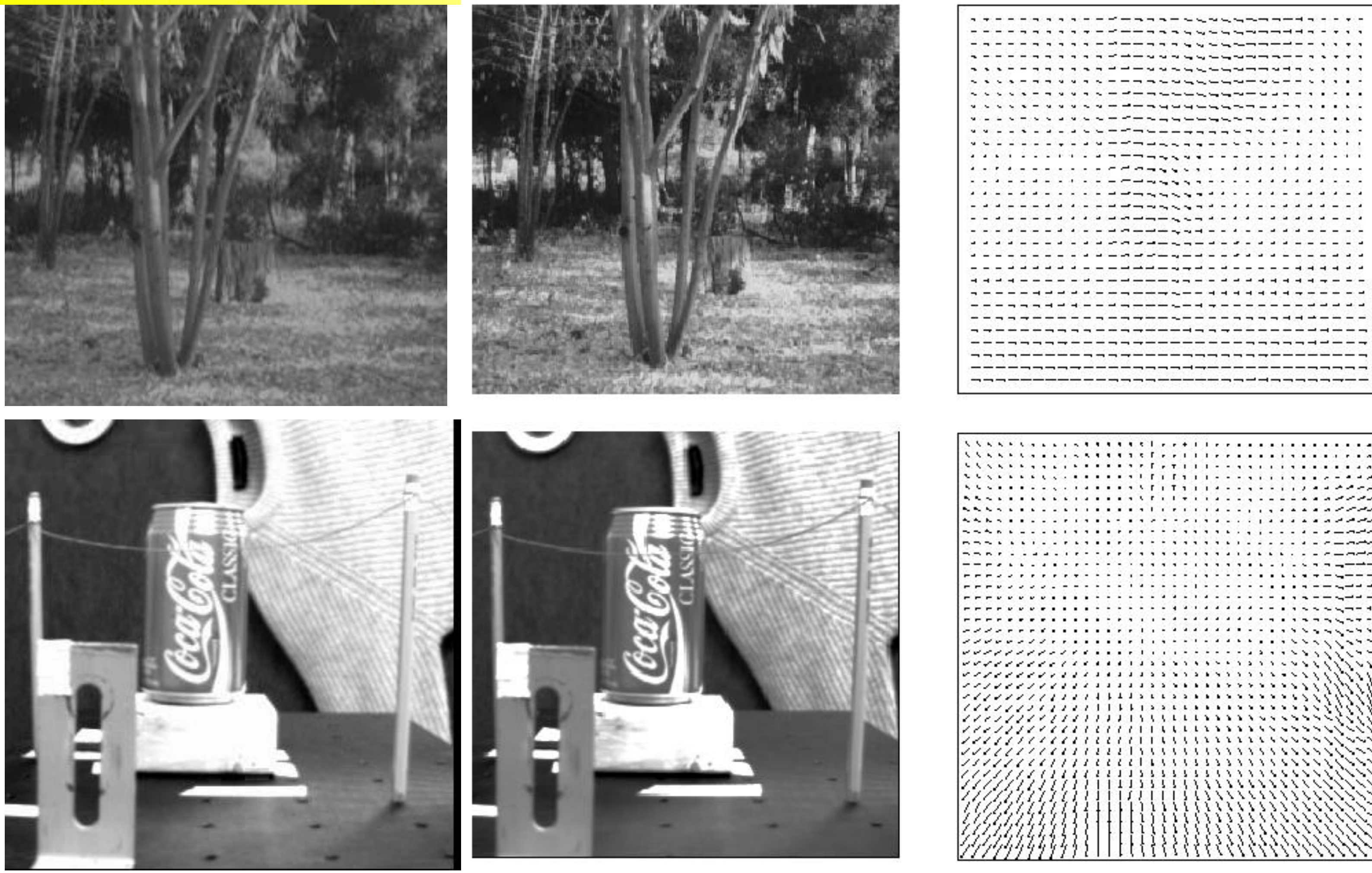
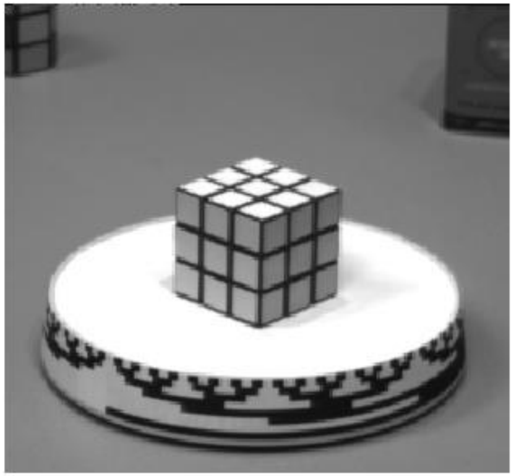
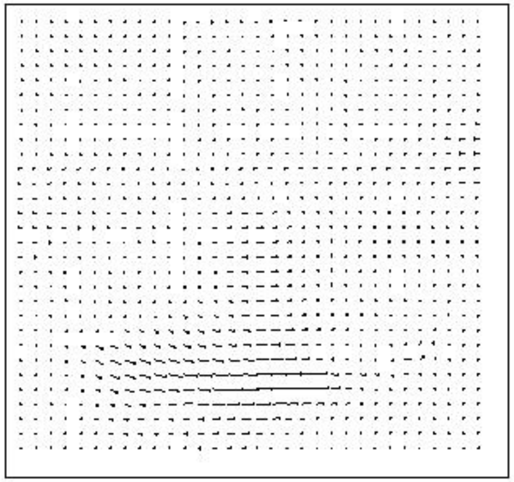
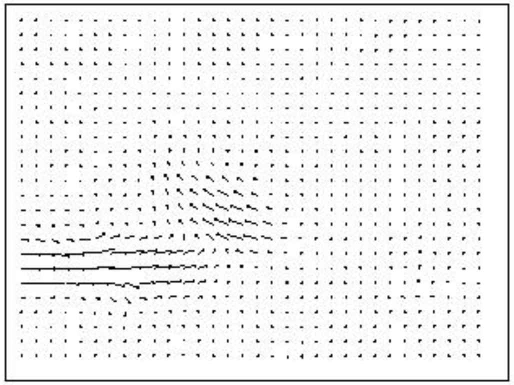


Fig. 7.7 Optical flow from feature point analyses. (a) An image. (b) Later image. (c) Optical flow found by relaxation.


Coca-Cola
CLASSIC

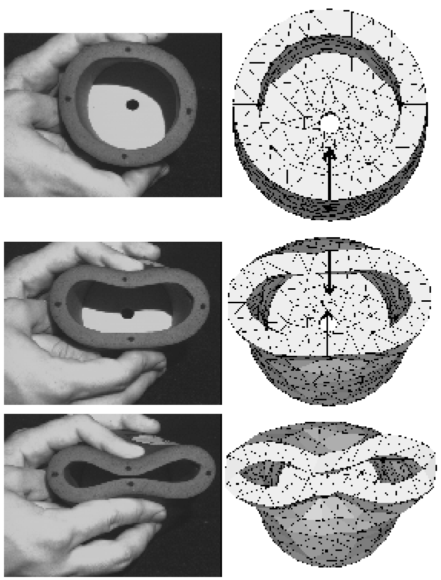



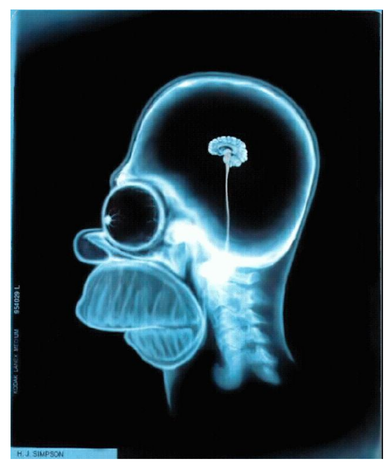
Caveats

- Motion analysis a very important and popular area in computer vision
- A large body of literature exits with maybe hundreds of different formulations (At CVPR, you will find at least 2 or 3 sessions on motion)
- Many of them can be very mathematical
- Apparent motion != True motion

Rigid vs. nonrigid motion

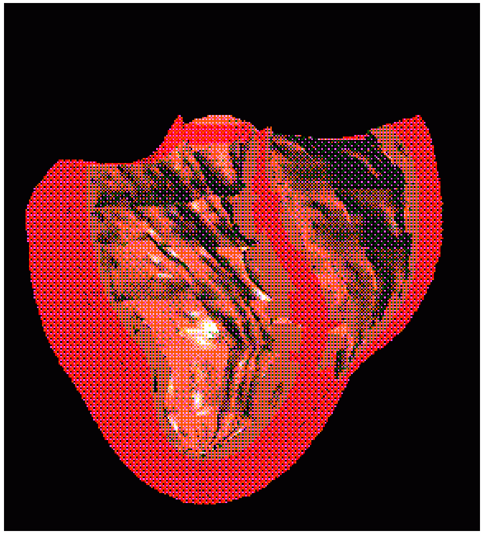
- Camera motion is 6 DOF rigid motion
- Object motion may be rigid or nonrigid
  - Rigid: coffee mugs, silverware, baseballs, jets, ...
  - Nonrigid: humans, face, medical imagery, beach balls, scissors, grass, ...
    - Includes ***articulated*** motion





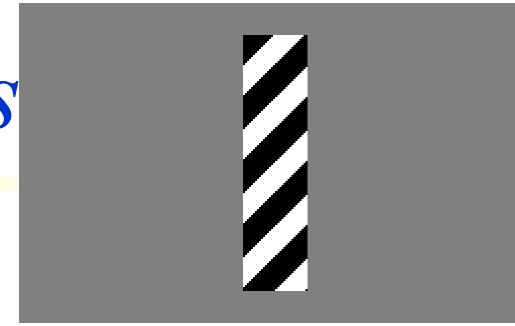
Nonrigid motion

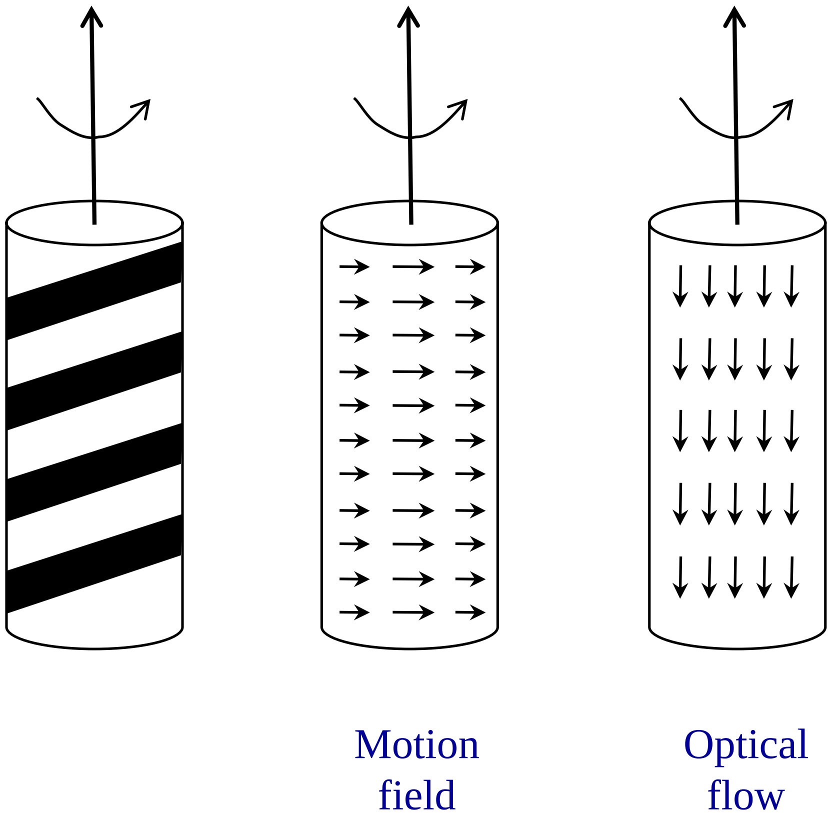
- Nonrigid motion is complicated and difficult, especially with little prior knowledge on what is being viewed
  - Typical problem: What are the parameters of the known nonrigid model of the object being viewed?



We'll just focus on rigid motion

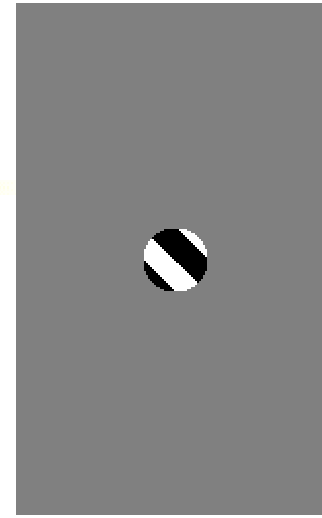
The barber's pole illus



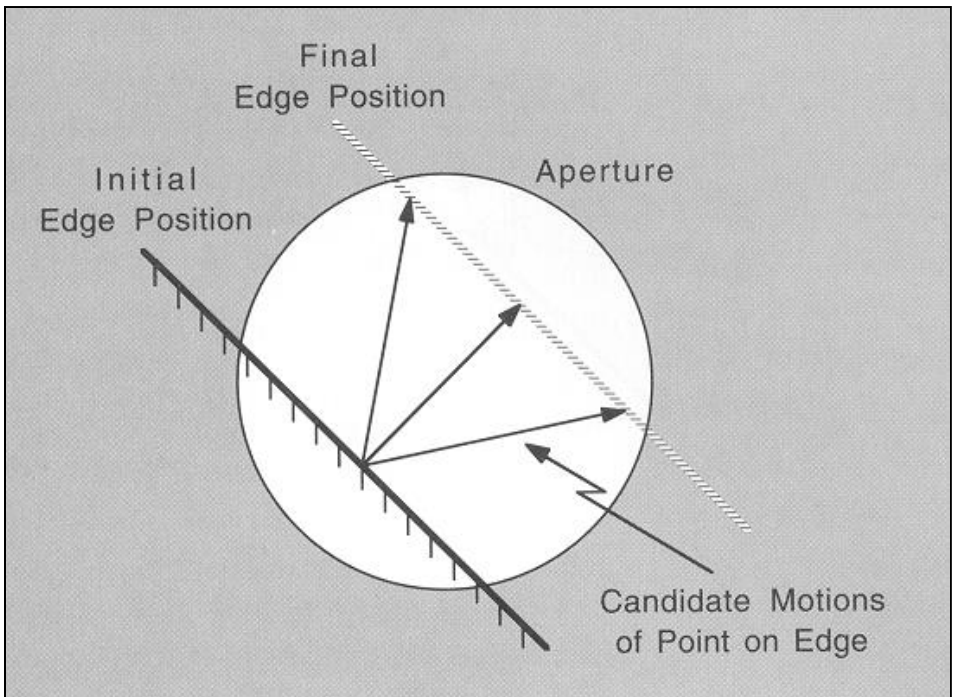


Web example

The aperture problem



- In local processing, we can only measure motion perpendicular to the image gradient



First steps

- Motion processing starts with estimating optical flow from frame to frame, either *densely* or *sparsely*
- The typical approaches are:
  - Dense correspondence:
    - Differential methods, local area/correlation based
    - This could be hierarchical (coarse-to-fine approach)
  - Sparse correspondence
    - Matching methods, feature based
- Asumption: Points/features can be matched in nearby images

Brightness constancy equation

Total derivative

$$\frac{dI}{dt} = \frac{dI(x(t), y(t), t)}{dt} = 0$$

For a given scene point

$$\frac{dI(x(t), y(t), t)}{dt} = \frac{\partial I}{\partial x}\frac{dx}{dt} + \frac{\partial I}{\partial y}\frac{dy}{dt} + \frac{\partial I}{\partial t}\frac{dt}{dt} = 0$$

$$\left(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}\right)^T \left(\frac{dx}{dt}, \frac{dy}{dt}\right) + \frac{\partial I}{\partial t} = 0$$

$$\boxed{\nabla I \cdot \mathbf{v} + I_t = 0}$$

∇I Image gradient

v Optical flow

I_t Time difference

Brightness constancy equation (method #2)

$$I(x,y,t) = I(x+\delta x, y+\delta y, t+\delta t)$$ For a given scene point

$$\boxed{I(x+\delta x, y+\delta y, t+\delta t)} - I(x,y,t) = 0$$

$$\Big\downarrow \approx$$

$$I(x,y,z) + \frac{\partial I(x,y,t)}{\partial x}dx + \frac{\partial I(x,y,t)}{\partial y}dy + \frac{\partial I(x,y,t)}{\partial t}dt$$ by Taylor expansion

$$\frac{\partial I(x,y,t)}{\partial x}dx + \frac{\partial I(x,y,t)}{\partial y}dy + \frac{\partial I(x,y,t)}{\partial t}dt = 0$$

$$\frac{\partial I}{\partial x}\frac{dx}{dt} + \frac{\partial I}{\partial y}\frac{dy}{dt} + \frac{\partial I}{\partial t} = 0$$

$$\boxed{\nabla I \cdot v + I_t = 0}$$

Brightness constancy equation (method #2)

$$I(x,y,t) \quad = \quad I(x+\delta x, y+\delta y, t+\delta t)$$

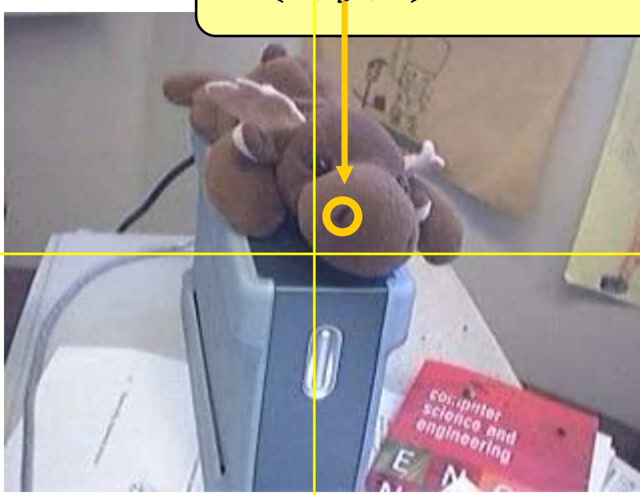


Image at time t

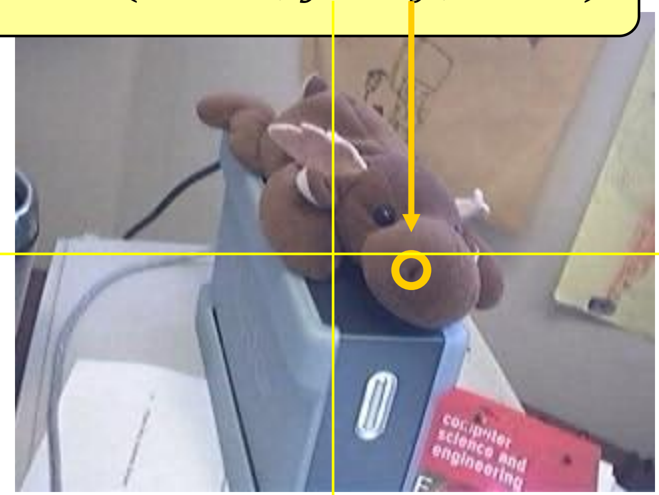
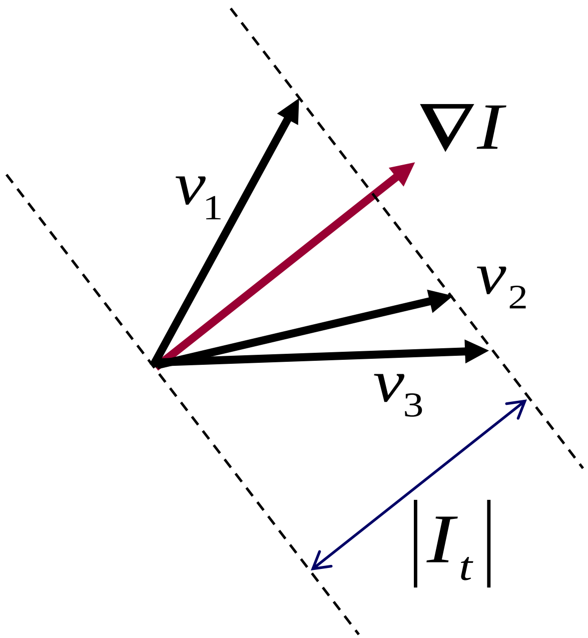


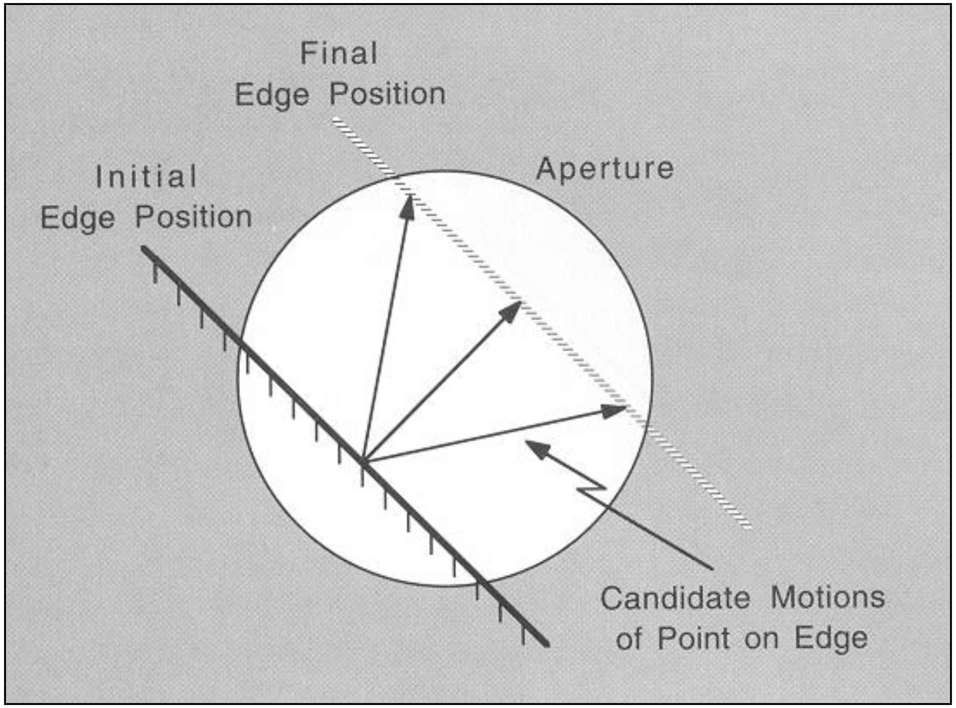
Image at time $t+\delta t$

Back to the aperture problem

$$\nabla I \cdot v + I_t = 0$$

$$\nabla I \cdot v = -I_t$$

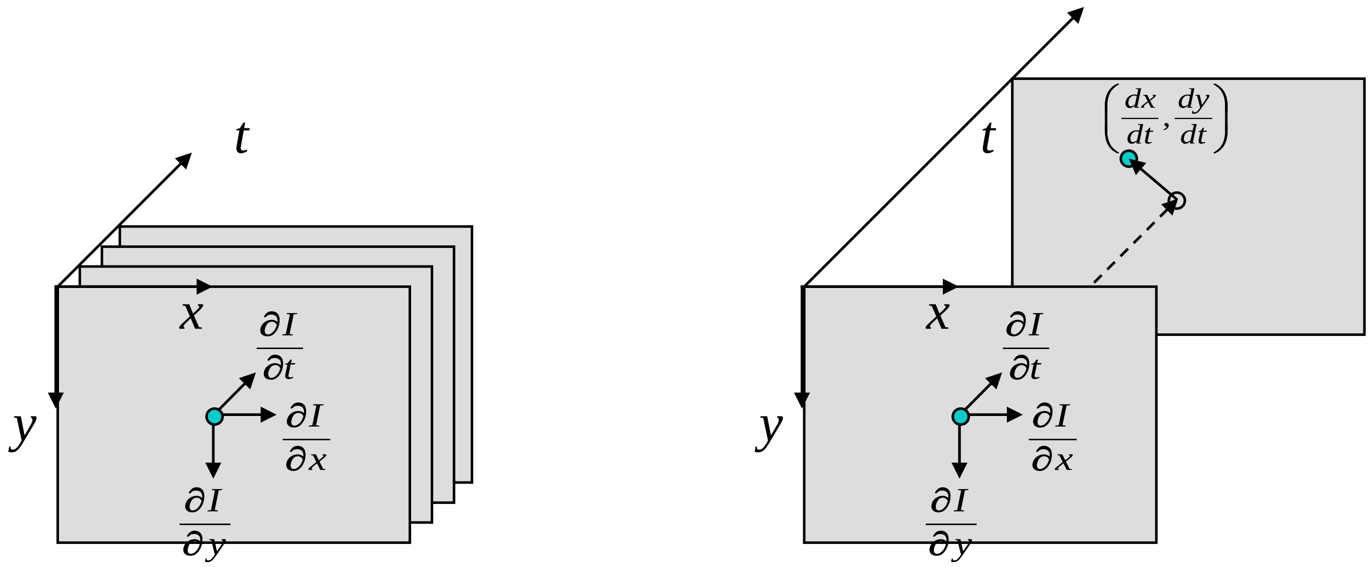




Many vectors ***v*** satisfy this

Only the normal direction is constrained

On images…

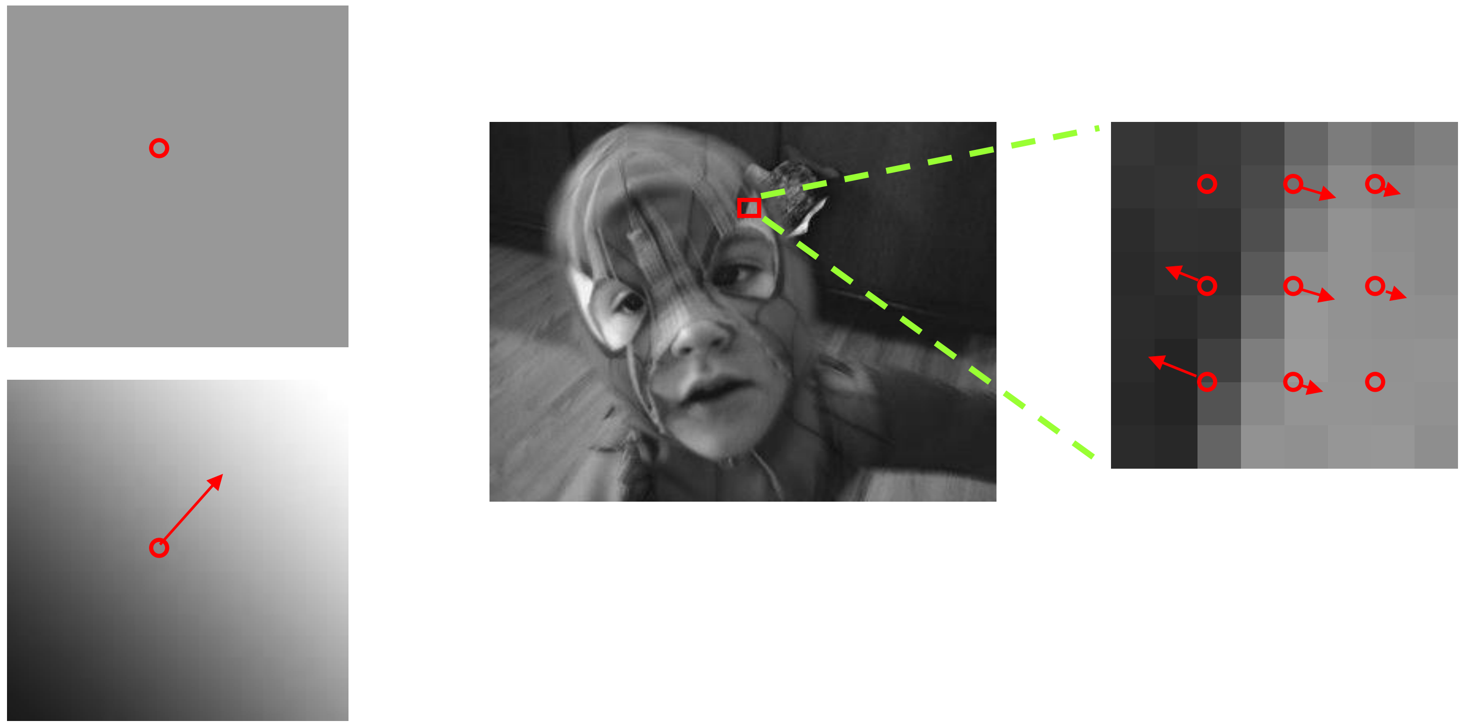


$$\nabla I \cdot v + I_t = 0$$

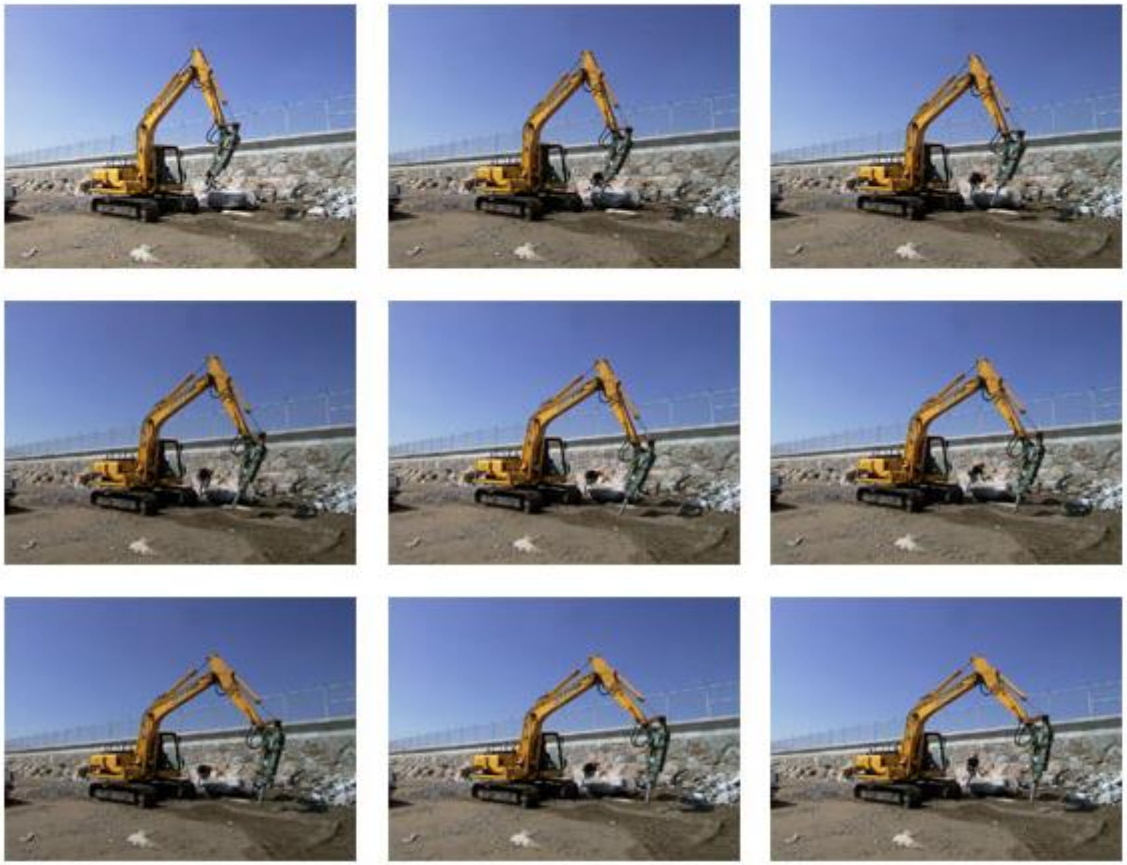
This equation defines and constrains the optical flow ***v*(*x*, *y*)**

What is the image gradient?

Image gradient – the first derivative (slope) of the intensity variation in (x, y)



What is the temporal gradient?



t_1

t_2

$$\frac{\partial I}{\partial t}$$

Brightness constancy of a point

Scene

Image sequence

I_1

I_2

I_3

$I(x(t_1),y(t_1),t_1)$

$=$

$I(x(t_2),y(t_2),t_2)$

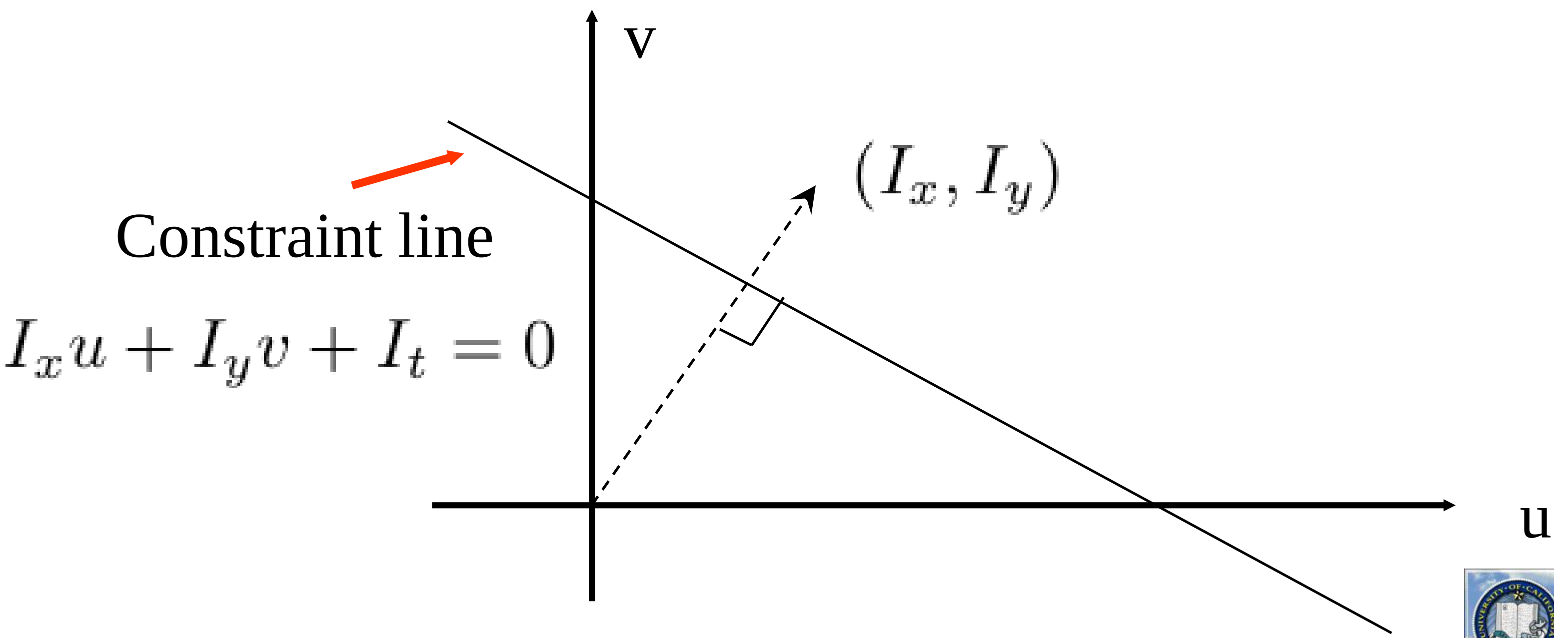
$I(x(t_2),y(t_2),t_2)$

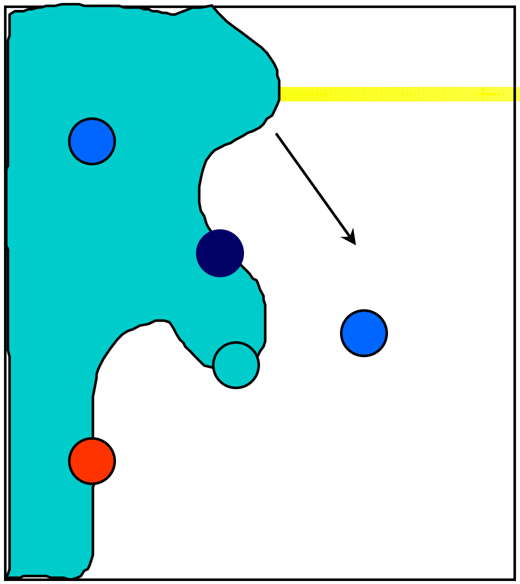
$=$

$I(x(t_3),y(t_3),t_3)$

Difficulty

- One equation with two unknowns
- Aperture problem
  - spatial derivatives use only a few adjacent pixels (limited aperture and visibility)
  - many combinations of (u,v) will satisfy the equation





- intensity gradient is zero
  no constraints on (u,v) $\quad (0,0)\cdot(u,v)=0$
  interpolated from other places

- intensity gradient is nonzero
  but is *constant*
  one constraints on (u,v) $\quad (\frac{\partial I}{\partial x},\frac{\partial I}{\partial y})\cdot(u,v)=-\frac{\partial I}{\partial t}$
  only the component along the gradient are recoverable

- intensity gradient is nonzero
  and *changing*
  multiple constraints on (u,v)
  motion recoverable

$$(\frac{\partial I}{\partial x_1},\frac{\partial I}{\partial y_1})\cdot(u,v)=-\frac{\partial I}{\partial t}_{(x_1,y_1)}$$

$$(\frac{\partial I}{\partial x_2},\frac{\partial I}{\partial y_2})\cdot(u,v)=-\frac{\partial I}{\partial t}_{(x_2,y_2)}$$

Patch Translation [Lucas-Kanade]

Assume a single velocity for all pixels within an image patch

$$E(u,v) = \sum_{x,y\in\Omega} \left(I_x(x,y)u + I_y(x,y)v + I_t\right)^2$$

Minimizing

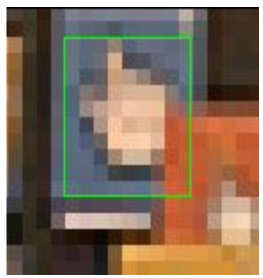
$$\begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = -\begin{pmatrix} \sum I_x I_t \\ \sum I_y I_t \end{pmatrix}$$

$$\left(\sum \nabla I \nabla I^T\right)\vec{U} = -\sum \nabla I I_t$$

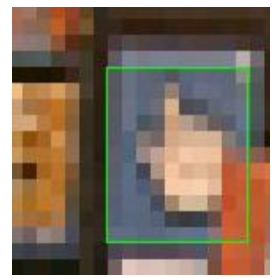
LHS: sum of the 2x2 outer product of the gradient vector

Image motion

How do we determine correspondences?



I



J

Assume all change between frames is due to **motion:**

$$J(x, y) \approx I(x+u(x, y), y+v(x, y))$$

The Aperture Problem

Let $M = \sum (\nabla I)(\nabla I)^T$ and $b = \begin{bmatrix} -\sum I_x I_t \\ -\sum I_y I_t \end{bmatrix}$

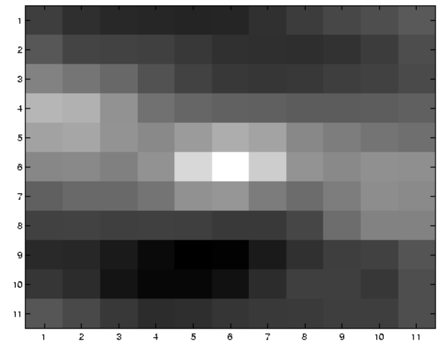
- Algorithm: At each pixel compute U by solving $MU = b$

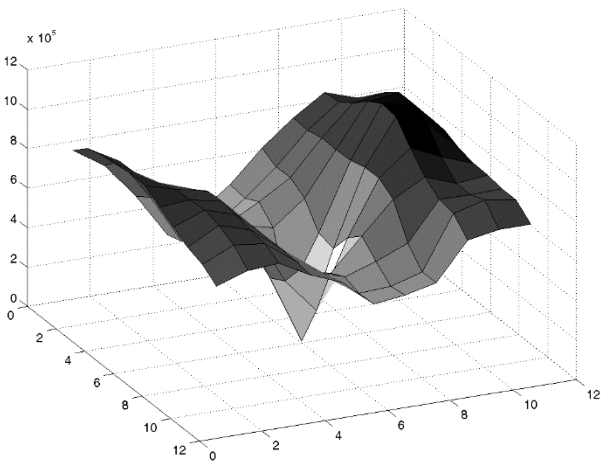
- M is singular if all gradient vectors point in the same direction
  - e.g., along an edge
  - of course, trivially singular if the summation is over a single pixel or there is no texture
  - i.e., only *normal flow* is available (aperture problem)

- Corners and textured areas are OK

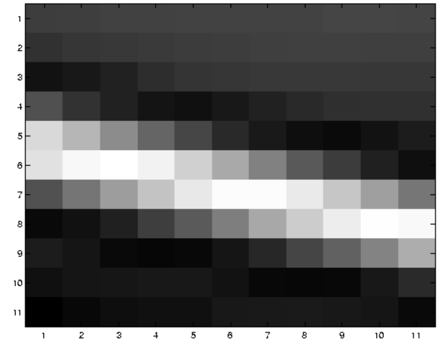
Szeliski

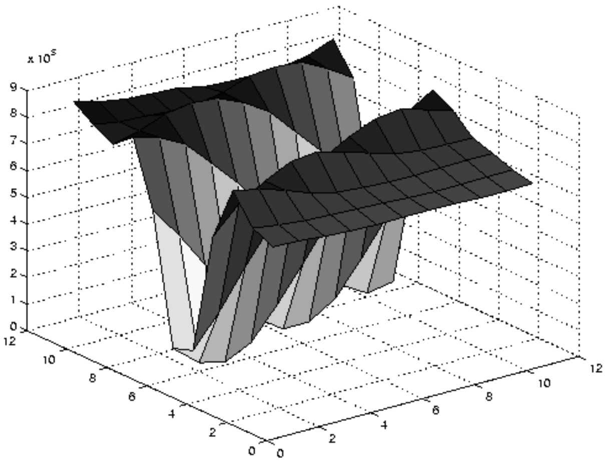
SSD Surface – Textured area



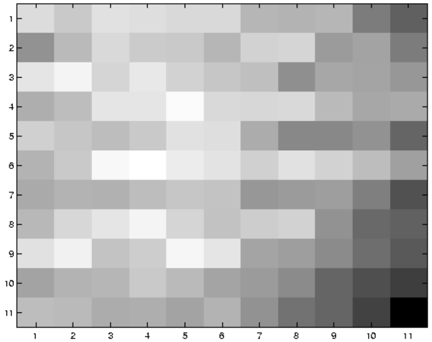


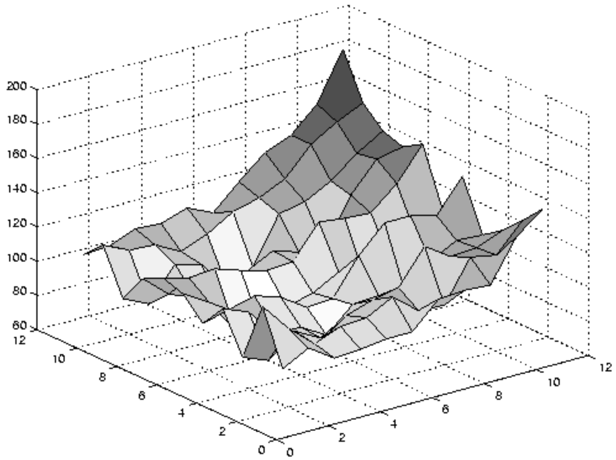
SSD Surface -- Edge





SSD – homogeneous area





Limits of the gradient method

Fails when intensity structure in window is poor

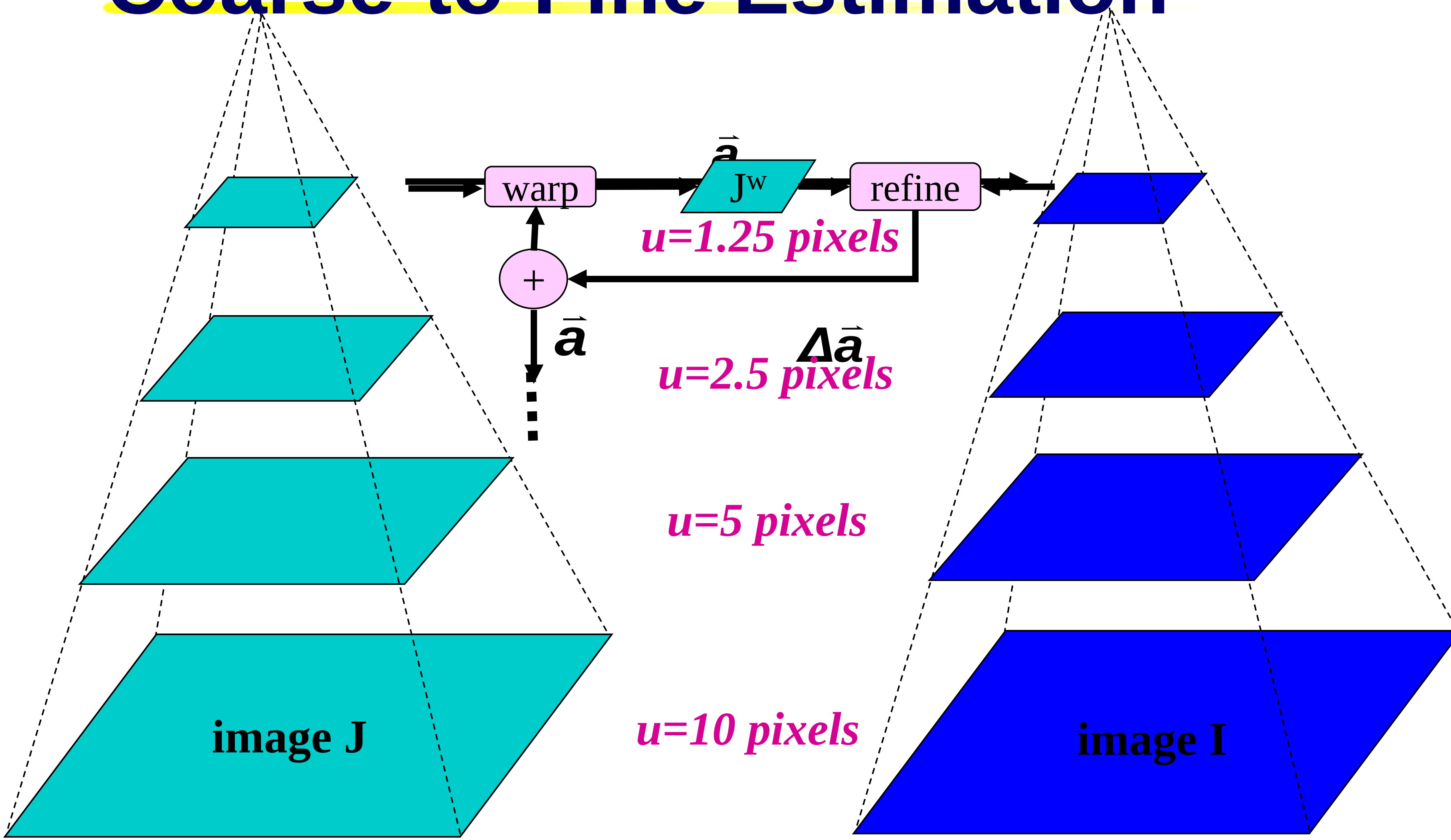
Fails when the displacement is large (typical operating range is motion of 1 pixel)

Linearization of brightness is suitable only for small displacements

- Also, brightness is not strictly constant in images

actually less problematic than it appears, since we can pre-filter images to make them look similar

Coarse-to-Fine Estimation



Pyramid of image J

Pyramid of image I

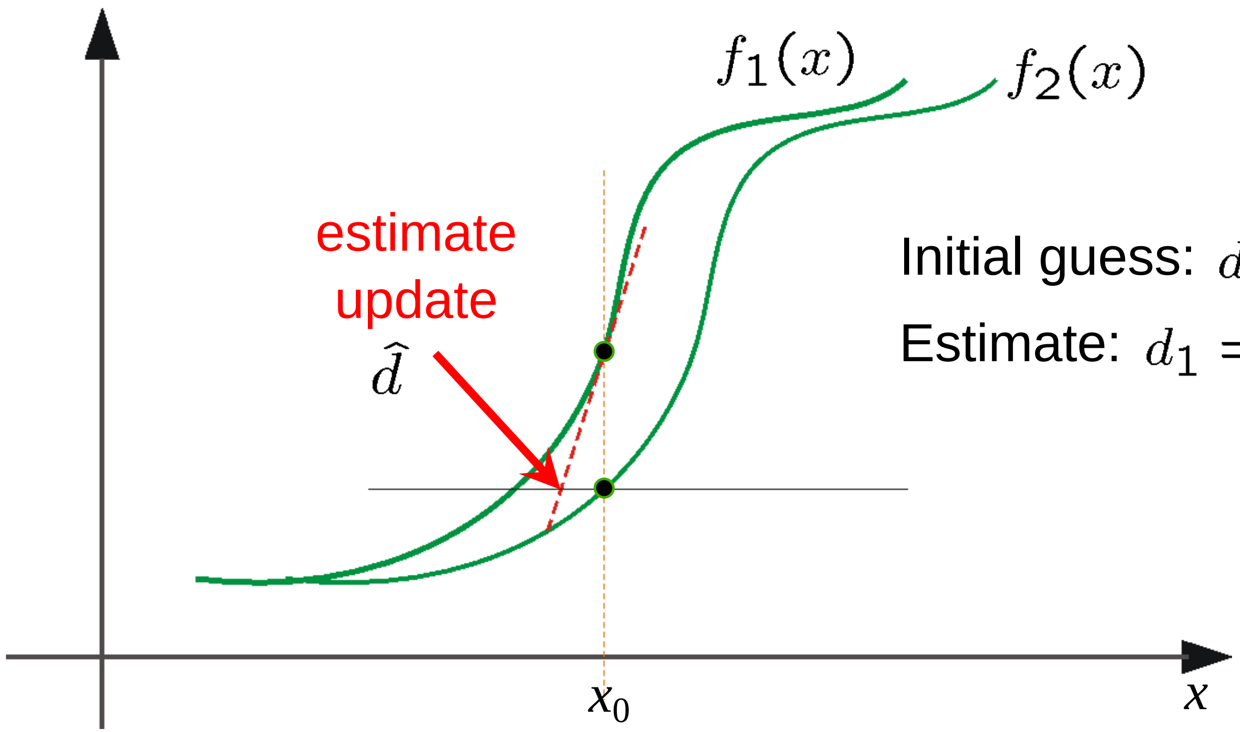
Szeliski

Iterative Refinement

- Estimate velocity at each pixel using one iteration of Lucas and Kanade estimation
- Warp one image toward the other using the estimated flow field

  (easier said than done)
- Refine estimate by repeating the process

Optical Flow: Iterative Estimation

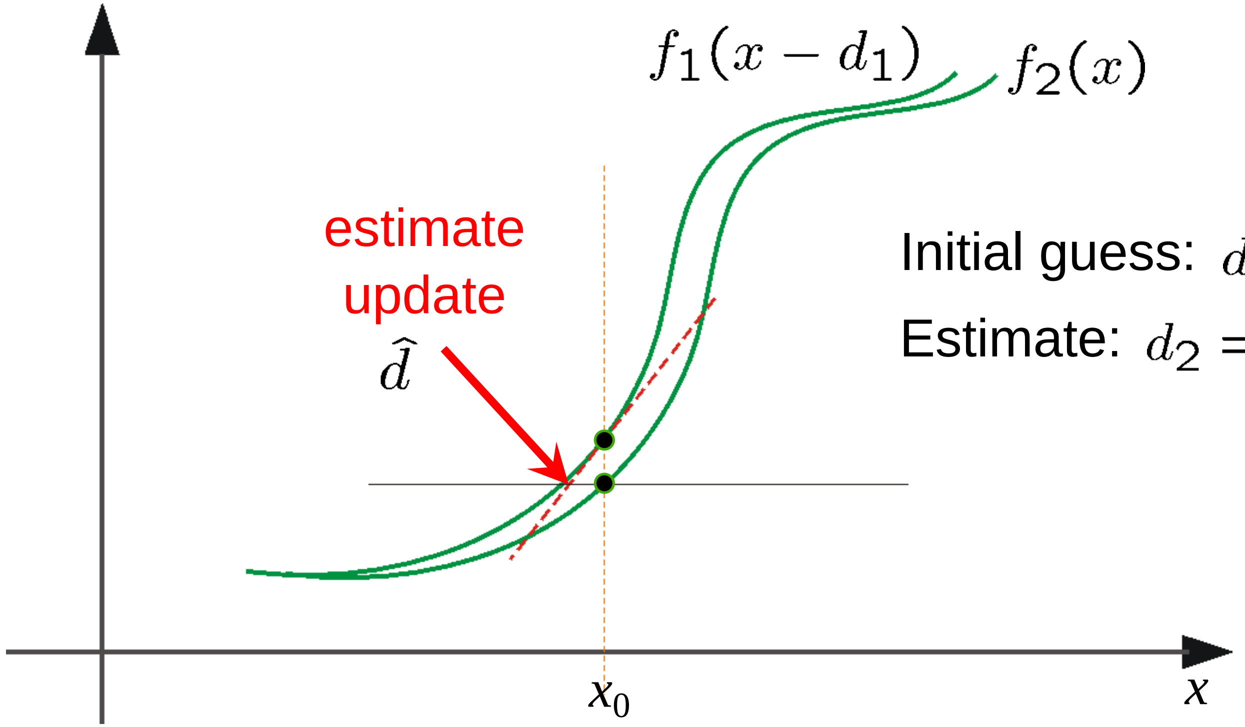


Initial guess: $d_0 = 0$

Estimate: $d_1 = d_0 + \hat{d}$

(using d for *displacement* here instead of u)

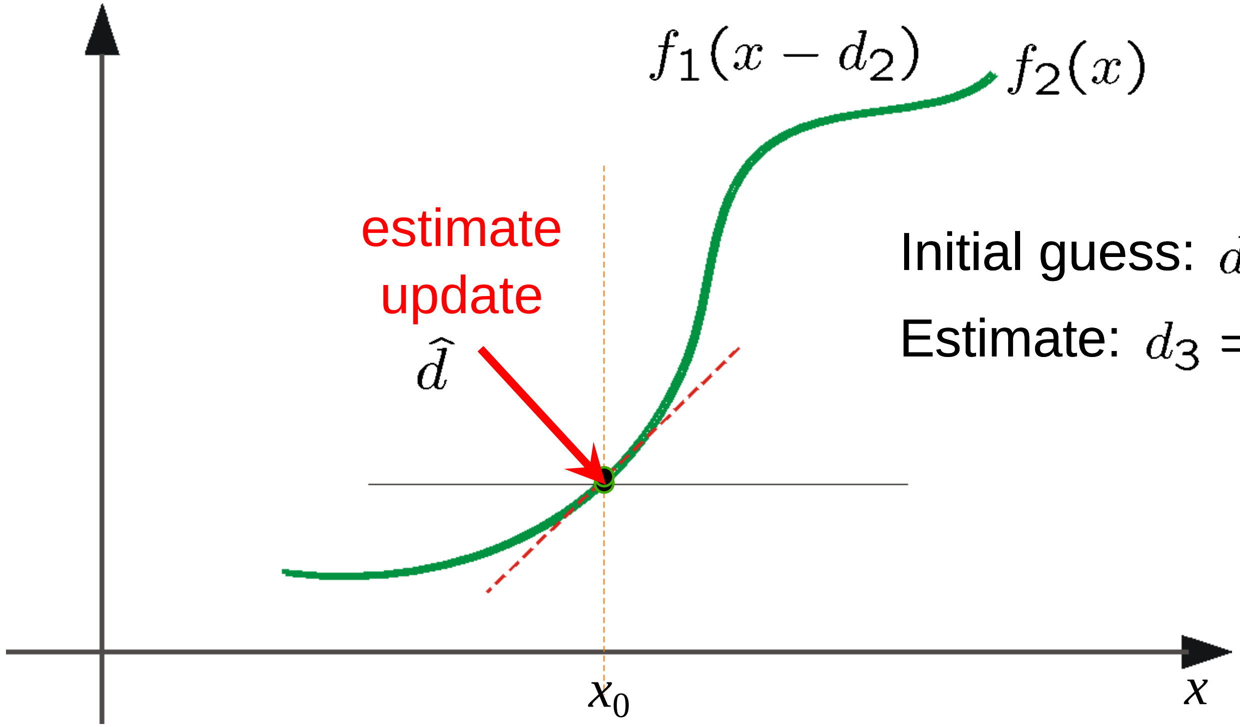
Optical Flow: Iterative Estimation



Initial guess: d_1

Estimate: $d_2 = d_1 + \hat{d}$

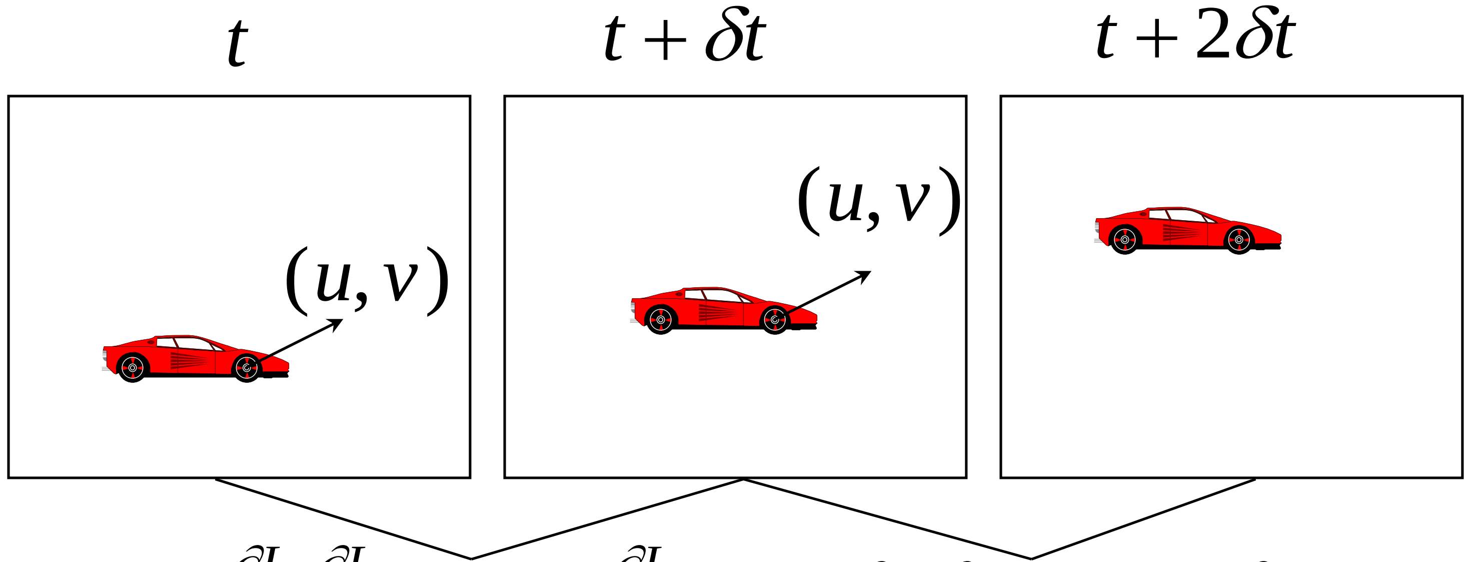
Optical Flow: Iterative Estimation



Initial guess: d_2

Estimate: $d_3 = d_2 + \hat{d}$

Temporal coherency



$$\left(\frac{\partial I}{\partial x},\frac{\partial I}{\partial y}\right)\cdot(u,v)=-\frac{\partial I}{\partial t}$$

$$\left(\frac{\partial I}{\partial x},\frac{\partial I}{\partial y}\right)\cdot(u,v)=-\frac{\partial I}{\partial t}$$

- Caveat:
  - (*u*,*v*) must stay the same across several frames
  - scenes highly textured
  - *(u,v)* at the same location actually refers to different object points

Spatial coherency

- neighboring pixels should have "similar" flow vector
- Q: What do you mean by "similar"
- A1: identical
- A2: change slowly

$$\frac{\partial u}{\partial x} = \frac{\partial u}{\partial y} = \frac{\partial v}{\partial x} = \frac{\partial v}{\partial y} = 0$$

$$\frac{\partial u}{\partial x}, \frac{\partial u}{\partial y}, \frac{\partial v}{\partial x}, \frac{\partial v}{\partial y} \cong 0$$

$$(\frac{\partial u}{\partial x})^2 + (\frac{\partial u}{\partial y})^2 + (\frac{\partial v}{\partial x})^2 + (\frac{\partial v}{\partial y})^2 \text{ small}$$

Mathematical formulation

- Based on Lagrange Multiplier
- Incorporate smoothness as an additional constraint
- Can be thought of as a weighting of two terms:
  - optical flow constraint
  - smoothness constraint

$$(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}) \cdot (u, v) = -\frac{\partial I}{\partial t}$$

$$(\frac{\partial u}{\partial x})^2 + (\frac{\partial u}{\partial y})^2 + (\frac{\partial v}{\partial x})^2 + (\frac{\partial v}{\partial y})^2$$

- Optimize over all image plane:

$$E = \int\int \left(\frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \frac{\partial I}{\partial t}\right)^2 + \lambda\left[\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2\right]dxdy$$

- Discretize the governing equation, at (*i*,*j*):

$$\frac{\partial u}{\partial x} = u_{i+1,j} - u_{i,j} \qquad \frac{\partial u}{\partial y} = u_{i,j+1} - u_{i,j}$$

$$\frac{\partial v}{\partial x} = v_{i+1,j} - v_{i,j} \qquad \frac{\partial v}{\partial y} = v_{i,j+1} - v_{i,j}$$

- Discretized expression:

$$E = \sum_i \sum_j \left(\frac{\partial I}{\partial x}_{i,j} u_{i,j} + \frac{\partial I}{\partial y}_{i,j} v_{i,j} + \frac{\partial I}{\partial t}_{i,j}\right)^2$$
$$+ \lambda[(u_{i+1,j} - u_{i,j})^2 + (u_{i,j+1} - u_{i,j})^2 + (v_{i+1,j} - v_{i,j})^2 + (v_{i,j+1} - v_{i,j})^2]$$

- At a pixel location (k,l):

$$\frac{\partial E}{\partial u_{k,l}} = 2\left(\frac{\partial I}{\partial x}_{k,l} u_{k,l} + \frac{\partial I}{\partial y}_{k,l} v_{k,l} + \frac{\partial I}{\partial t}_{k,l}\right)\frac{\partial I}{\partial x}_{k,l}$$
$$-2\lambda[(u_{k-1,l} - u_{k,l}) + (u_{k,l-1} - u_{k,l}) + (u_{k+1,l} - u_{k,l}) + (u_{k,l+1} - u_{k,l})] = 0$$

$$\frac{\partial E}{\partial v_{k,l}} = 2\left(\frac{\partial I}{\partial x}_{k,l} u_{k,l} + \frac{\partial I}{\partial y}_{k,l} v_{k,l} + \frac{\partial I}{\partial t}_{k,l}\right)\frac{\partial I}{\partial y}_{k,l}$$
$$-2\lambda[(v_{k-1,l} - v_{k,l}) + (v_{k,l-1} - v_{k,l}) + (v_{k+1,l} - v_{k,l}) + (v_{k,l+1} - v_{k,l})] = 0$$

$$\frac{\partial E}{\partial \lambda} = \cdots$$

- Putting it all together:

$$\left(\frac{\partial I}{\partial x}_{k,l}\right)^2 u_{k,l} + \frac{\partial I}{\partial x}_{k,l} \frac{\partial I}{\partial y}_{k,l} v_{k,l} + \frac{\partial I}{\partial x}_{k,l} \frac{\partial I}{\partial t}_{k,l} - 4\lambda(\bar{u} - u_{k,l}) = 0$$

$$\frac{\partial I}{\partial x}_{k,l} \frac{\partial I}{\partial y}_{k,l} u_{k,l} + \left(\frac{\partial I}{\partial y}_{k,l}\right)^2 v_{k,l} + \frac{\partial I}{\partial y}_{k,l} \frac{\partial I}{\partial t}_{k,l} - 4\lambda(\bar{v} - v_{k,l}) = 0$$

$$\bar{u} = (u_{k-1,l} + u_{k,l-1} + u_{k+1,l} + u_{k,l+1})/4$$

$$\bar{v} = (v_{k-1,l} + v_{k,l-1} + v_{k+1,l} + v_{k,l+1})/4$$

- Or:

$$[4\lambda + (\frac{\partial I}{\partial x}_{k,l})^2]u_{k,l} + \frac{\partial I}{\partial x}_{k,l}\frac{\partial I}{\partial y}_{k,l} v_{k,l} = 4\lambda\bar{u} - \frac{\partial I}{\partial x}_{k,l}\frac{\partial I}{\partial t}_{k,l}$$

$$\frac{\partial I}{\partial x}_{k,l}\frac{\partial I}{\partial y}_{k,l} u_{k,l} + [4\lambda + (\frac{\partial I}{\partial y}_{k,l})^2]v_{k,l} = 4\lambda\bar{v} - \frac{\partial I}{\partial y}_{k,l}\frac{\partial I}{\partial t}_{k,l}$$

$$u_{k,l} = \bar{u} - \frac{\frac{\partial I}{\partial x}_{k,l}\bar{u} + \frac{\partial I}{\partial y}_{k,l}\bar{v} + \frac{\partial I}{\partial t}_{k,l}}{4\lambda + (\frac{\partial I}{\partial x}_{k,l})^2 + (\frac{\partial I}{\partial y}_{k,l})^2}\frac{\partial I}{\partial x}_{k,l}$$

$$v_{k,l} = \bar{v} - \frac{\frac{\partial I}{\partial x}_{k,l}\bar{u} + \frac{\partial I}{\partial y}_{k,l}\bar{v} + \frac{\partial I}{\partial t}_{k,l}}{4\lambda + (\frac{\partial I}{\partial x}_{k,l})^2 + (\frac{\partial I}{\partial y}_{k,l})^2}\frac{\partial I}{\partial y}_{k,l}$$

$$u_{k,l} = \bar{u} - \frac{\left.\frac{\partial I}{\partial x}\right|_{k,l}\bar{u} + \left.\frac{\partial I}{\partial y}\right|_{k,l}\bar{v} + \left.\frac{\partial I}{\partial t}\right|_{k,l}}{4\lambda + \left(\left.\frac{\partial I}{\partial x}\right|_{k,l}\right)^2 + \left(\left.\frac{\partial I}{\partial y}\right|_{k,l}\right)^2} \left.\frac{\partial I}{\partial x}\right|_{k,l}$$

$$v_{k,l} = \bar{v} - \frac{\left.\frac{\partial I}{\partial x}\right|_{k,l}\bar{u} + \left.\frac{\partial I}{\partial y}\right|_{k,l}\bar{v} + \left.\frac{\partial I}{\partial t}\right|_{k,l}}{4\lambda + \left(\left.\frac{\partial I}{\partial x}\right|_{k,l}\right)^2 + \left(\left.\frac{\partial I}{\partial y}\right|_{k,l}\right)^2} \left.\frac{\partial I}{\partial y}\right|_{k,l}$$

- estimate based on smoothness
- how much does the smooth estimate violate optical flow constraint
- how much does the optical flow constraint matters
- direction for correction

Algorithms

1. Compute $\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}, \frac{\partial I}{\partial t}$ from a pair of input images
2. Choose a weighting factor λ
3. Compute $(\bar{u}, \bar{v})$
4. At each pixel location (k,l), do

$$u_{k,l}^{(n+1)} = \bar{u}^{(n)} - \frac{\frac{\partial I}{\partial x}_{k,l} \bar{u}^{(n)} + \frac{\partial I}{\partial y}_{k,l} \bar{v}^{(n)} + \frac{\partial I}{\partial t}_{k,l}}{4\lambda + (\frac{\partial I}{\partial x}_{k,l})^2 + (\frac{\partial I}{\partial y}_{k,l})^2} \frac{\partial I}{\partial x}_{k,l}$$

$$v_{k,l}^{(n+1)} = \bar{v}^{(n)} - \frac{\frac{\partial I}{\partial x}_{k,l} \bar{u}^{(n)} + \frac{\partial I}{\partial y}_{k,l} \bar{v}^{(n)} + \frac{\partial I}{\partial t}_{k,l}}{4\lambda + (\frac{\partial I}{\partial x}_{k,l})^2 + (\frac{\partial I}{\partial y}_{k,l})^2} \frac{\partial I}{\partial y}_{k,l}$$

5. Iterate steps 3 and 4 until no change or count exceeds

Results

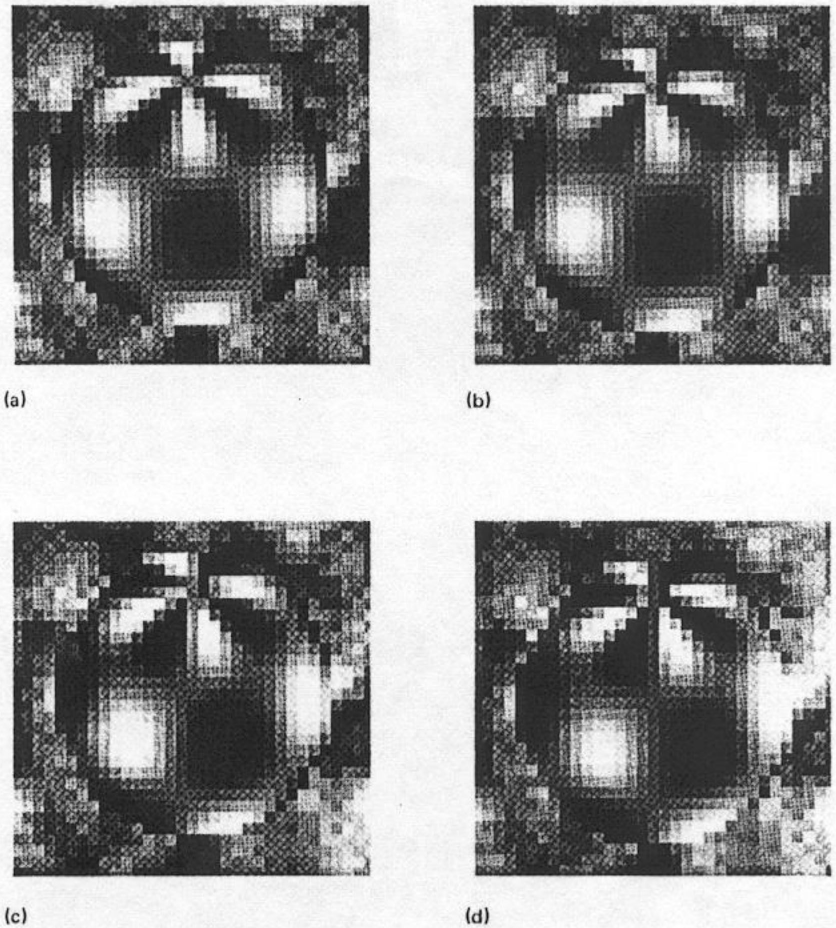
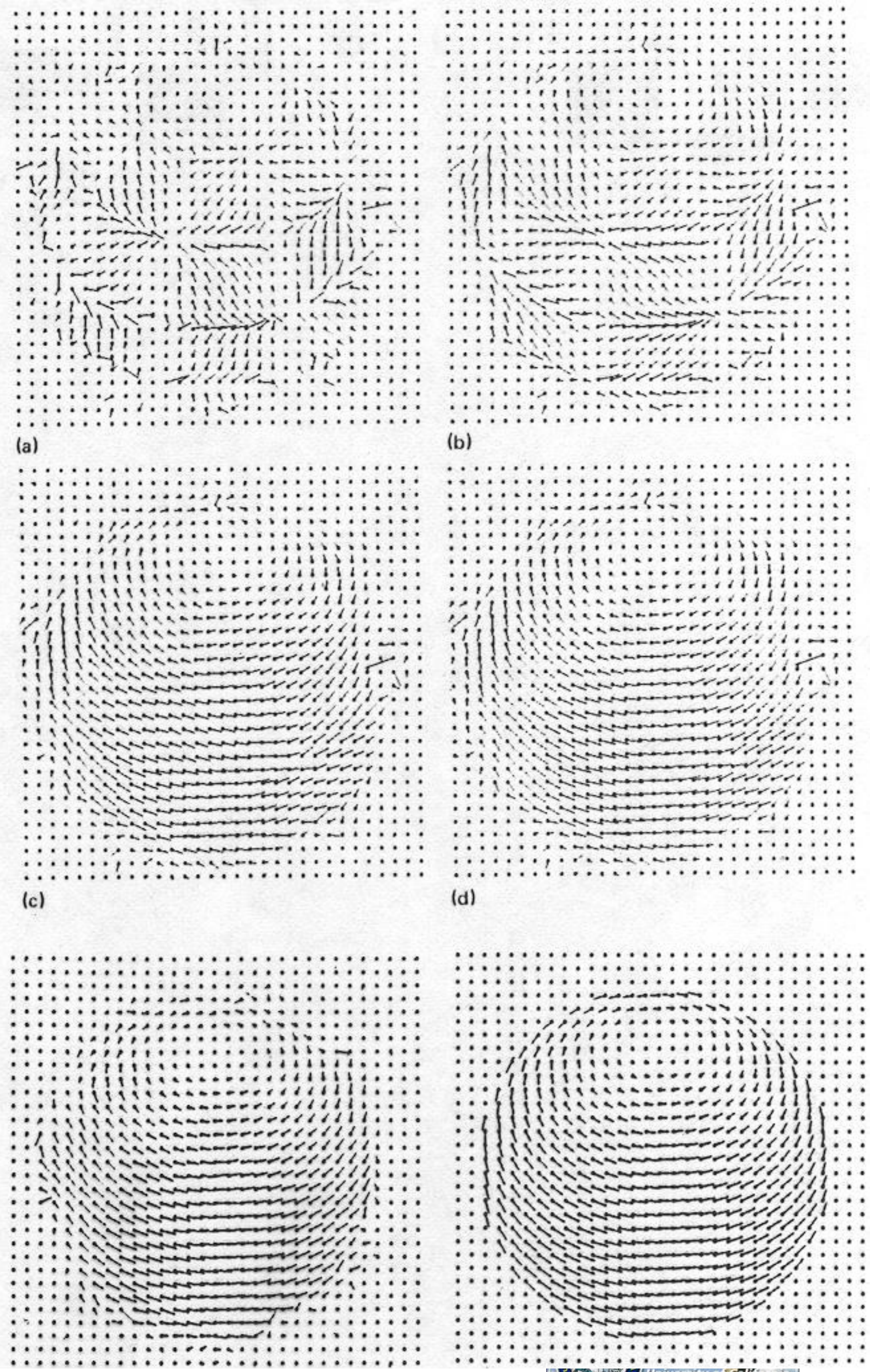


Figure 12-8. Four frames of a synthetic image sequence showing a sphere slowly rotating in front of a randomly patterned background.

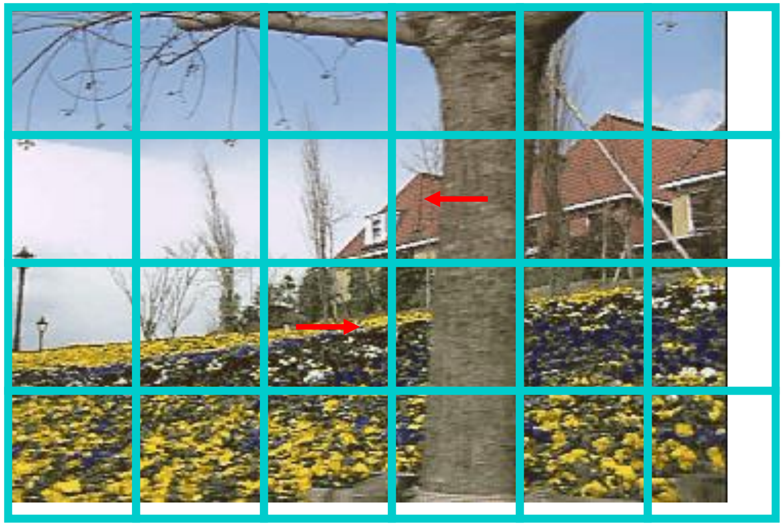


Motion representations

- How can we describe this scene?

Block-based motion prediction

- Break image up into square blocks
- Estimate translation for each block
- Use this to predict next frame, code difference (MPEG-2)



Layered motion

- Break image sequence up into "layers":
-  ·  =

- Describe ea

Layered motion

- Advantages:
  - can represent occlusions / disocclusions
  - each layer's motion can be smooth
  - video segmentation for semantic processing
- Difficulties:
  - how do we determine the correct number?
  - how do we assign pixels?
  - how do we model the motion?

Layers for video summarization

Frame 0

Frame 50

Frame 80

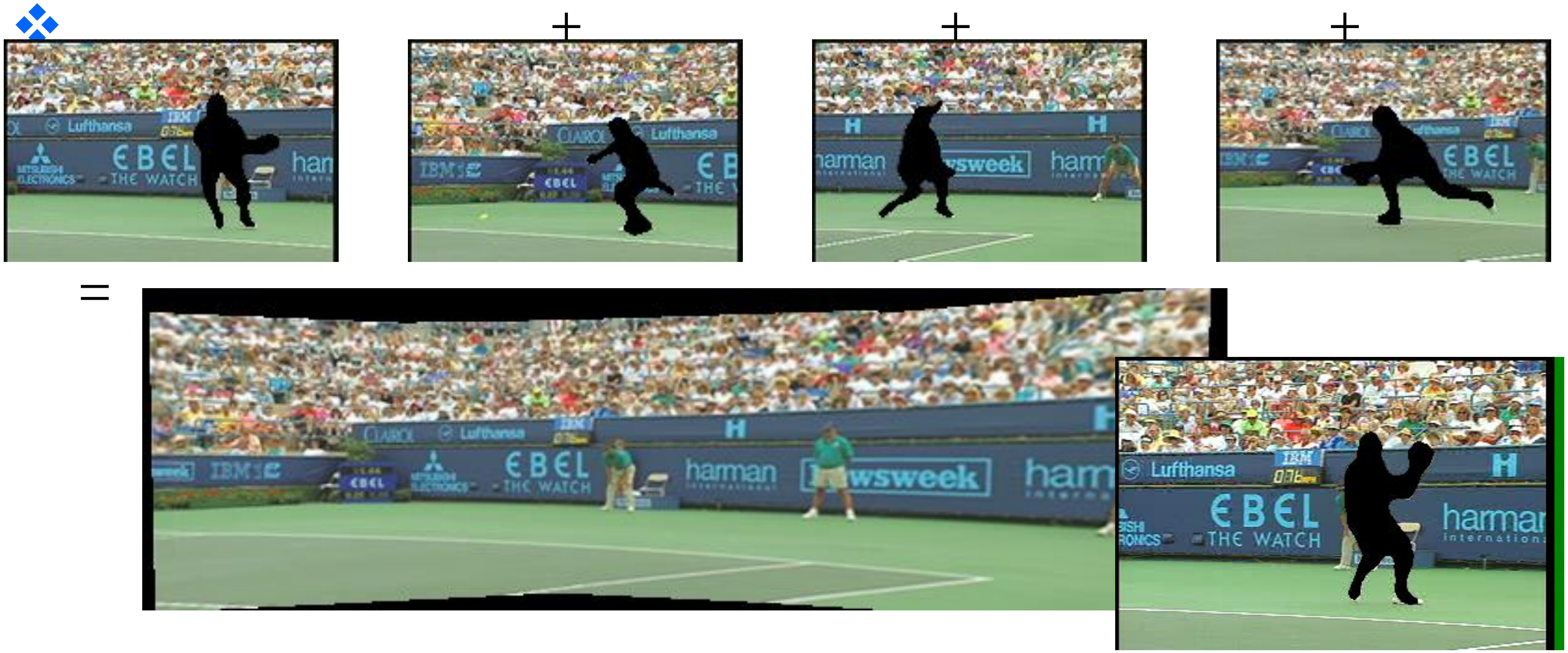
Background scene (players removed)

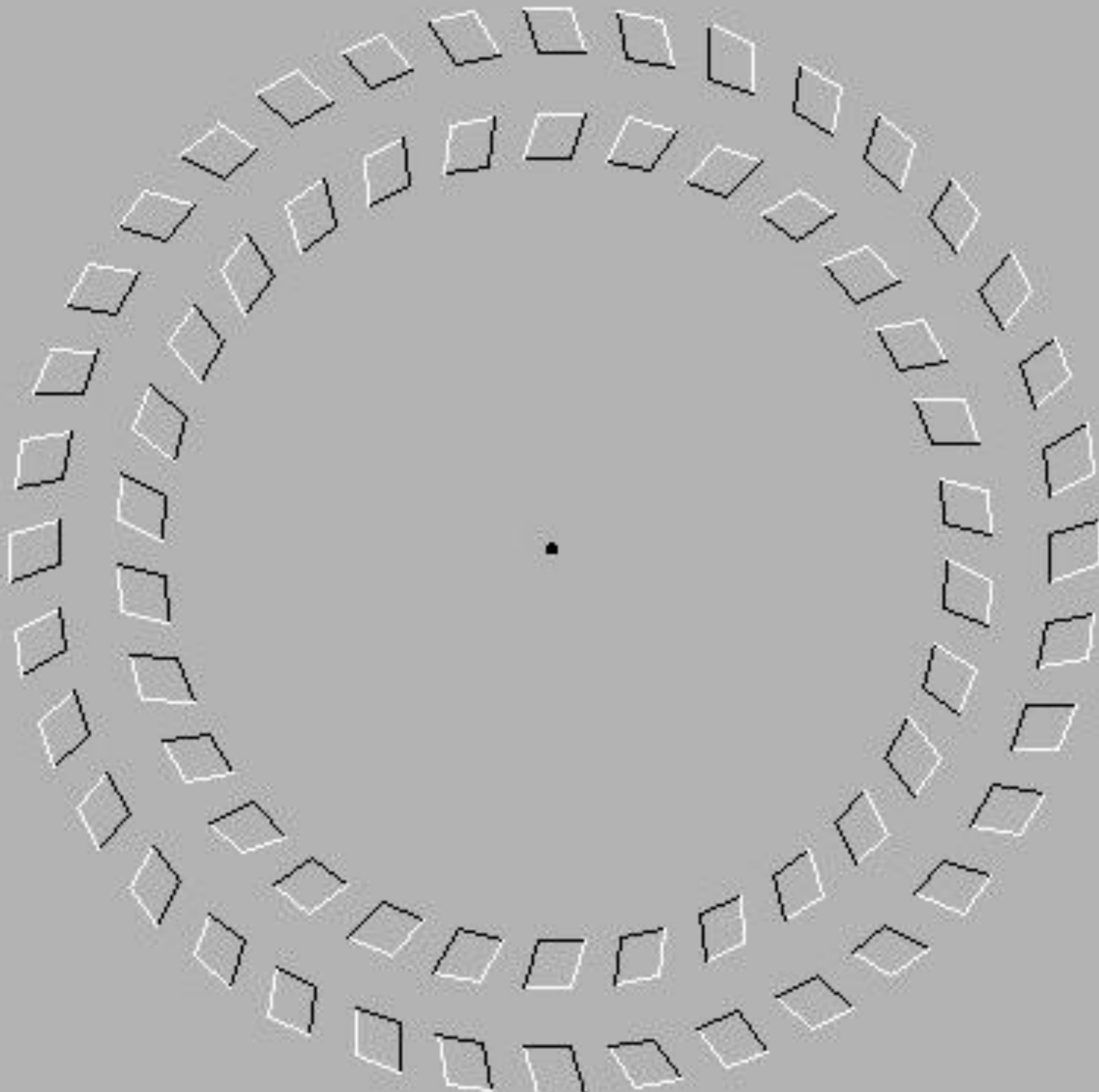
Complete synopsis of the video

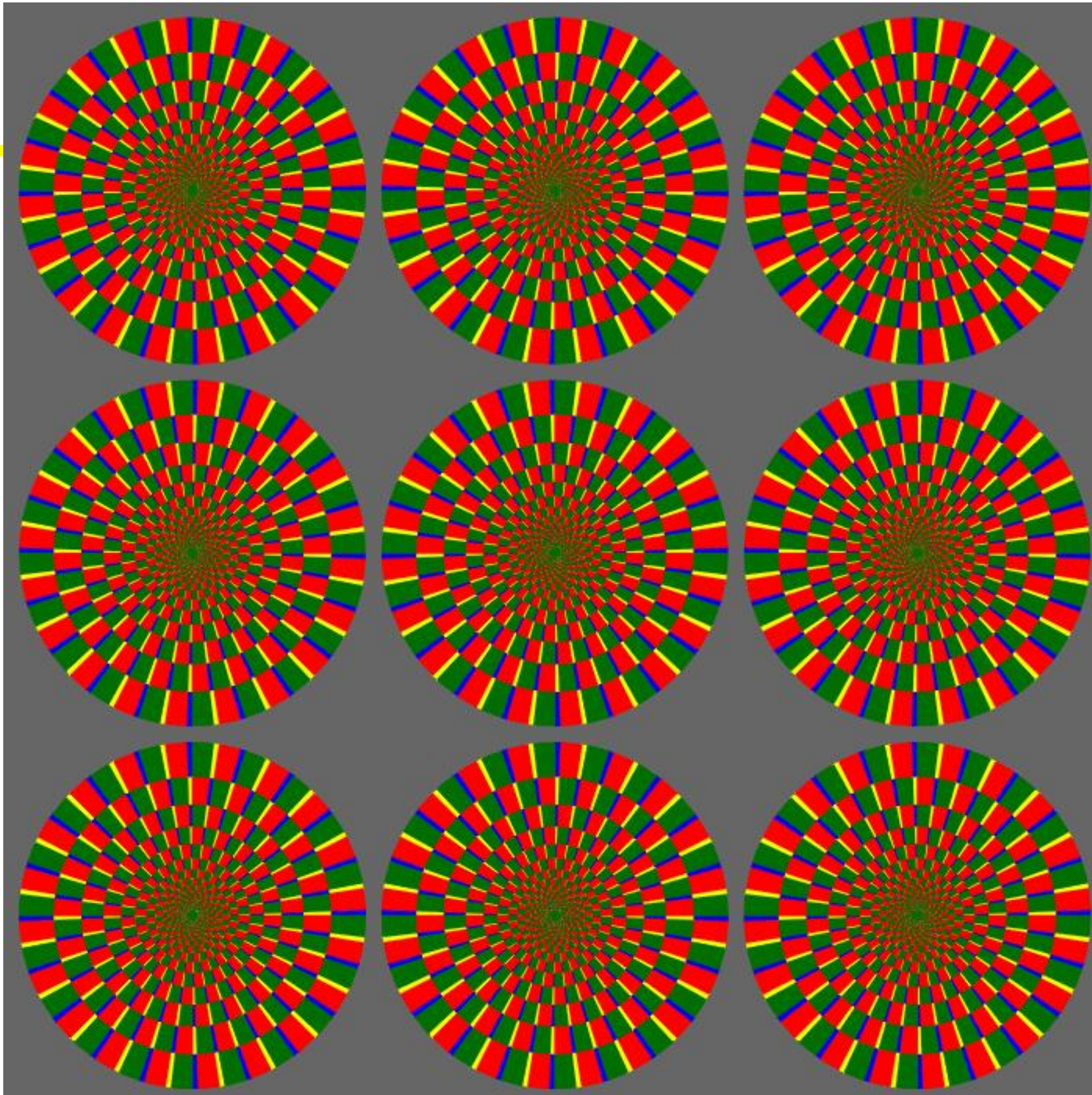
Szelisk

Background modeling (MPEG-4)

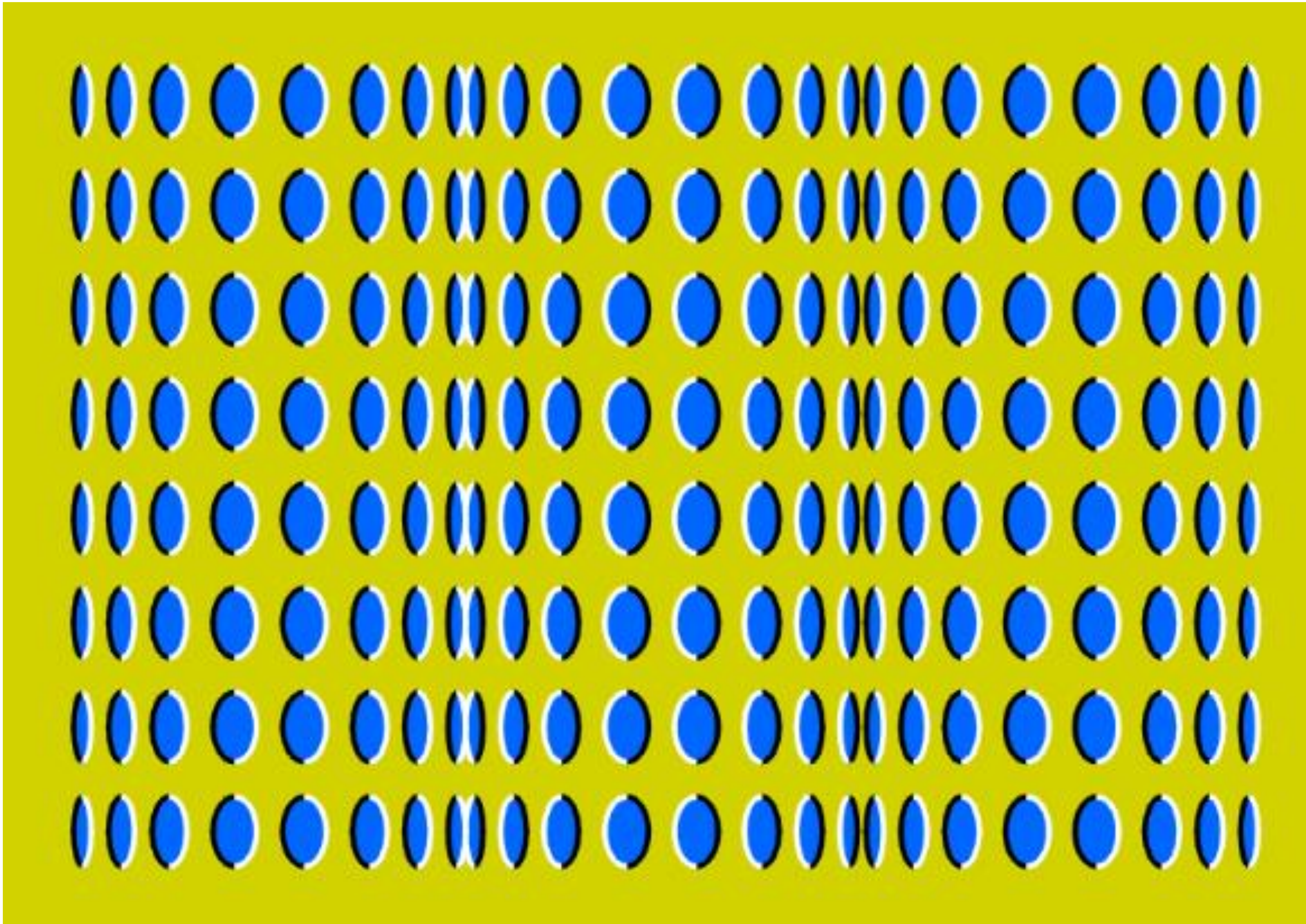
- Convert masked images into a background sprite for layered video coding

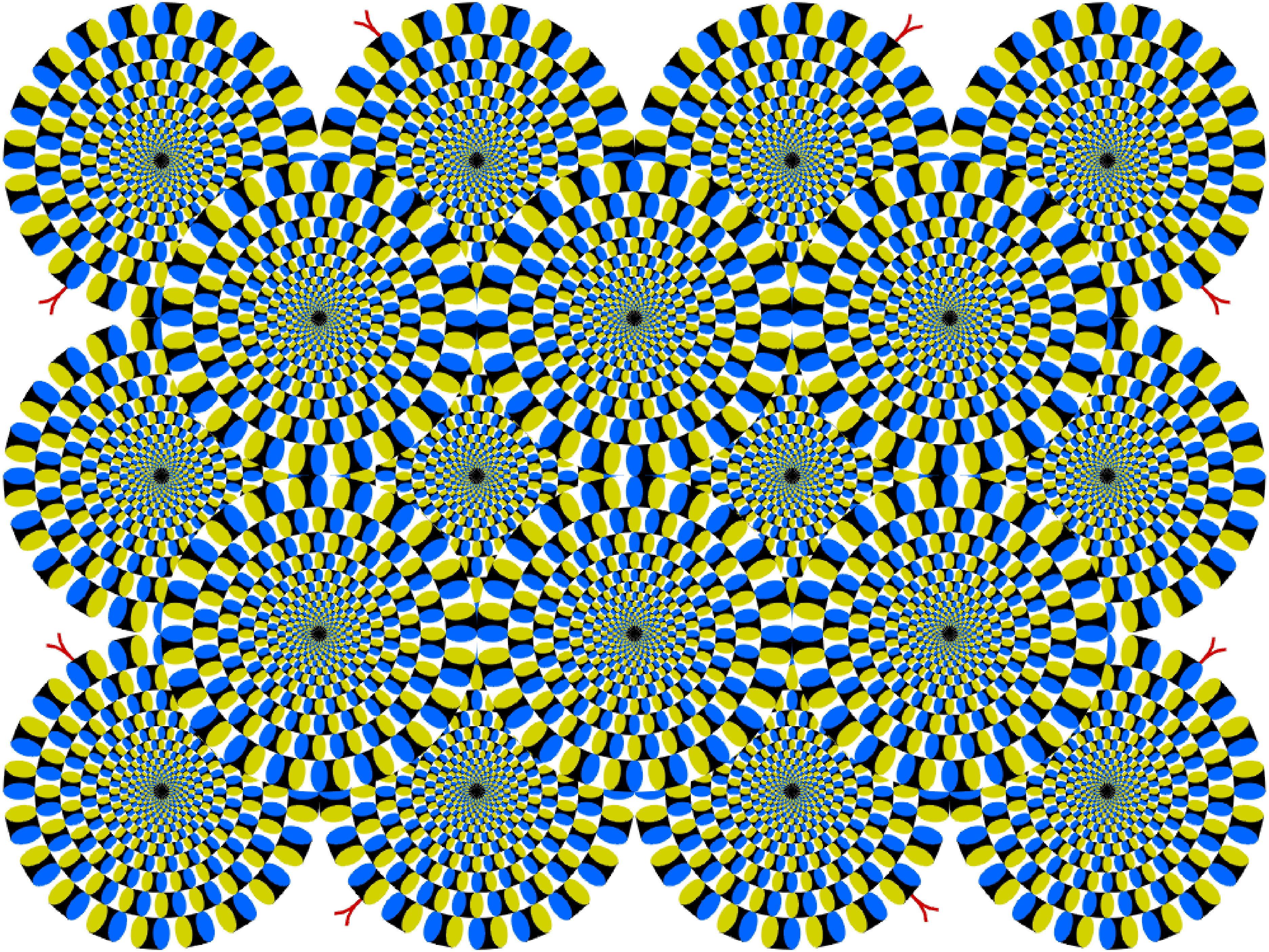
- 





University of California Santa Barbara





Optical flow summary

- Optical flow techniques:
  - Techniques that estimate the motion field from the image brightness constancy equation
- Optical flow:
  - Is best estimated (least noisy) at image points with high spatial image gradients. (Why?)
  - Works best for Lambertian surfaces (Why?)
  - Works best for very high frame rates (Why?)
- From optical flow, we can compute shape/structure/depth, motion parameters, segmentation, etc.
  - But if you primarily want to <u>track</u> an object, other methods may be preferred

Tracking

- Tracking is the process of updating an object’s position (and orientation, and articulation?) over time through a video sequence
  - Estimate the object ***pose*** at each time point
    - “Pose” – position and orientation

- Applications
  - Surveillance
  - Targeting
  - Motion-based recognition (e.g., gesture recognition, computation of egomotion)
  - Motion analysis (golf swing, gait, character animation)
  - ……..

Tracking vs. optical flow

- In tracking, we are generally acknowledging that some sparse features are the points to track
  - Corners, lines, regions, patterns, contours....

- Rather than computing the full motion field from optical flow, let's keep track of the time-varying position of these sparse features
  - Then compute {egomotion, object pose, etc.} from this

- This typically involves a loop of prediction, measurement, and correction
  - Often with presumed models of motion dynamics and measurement noise

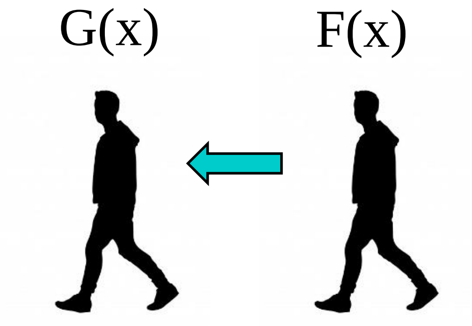
Tracking vs. Matching

- Tracking requires videos
- Small displacement is assumed
- Simple features
- Use image constraint (similar to optical flow constraint)

- Matching can be done on discrete frames
- Displacement can be large (>10 pixels)
- Often more elaborate features
- Independent detection in each frame and then match

Examples LKT tracker

$F(x)$ and $G(x) = F(x+h)$



$$F'(x) \approx \frac{F(x+h) - F(x)}{h} = \frac{G(x) - F(x)}{h}$$

I_t

$$h \approx \frac{G(x) - F(x)}{F'(x)}$$

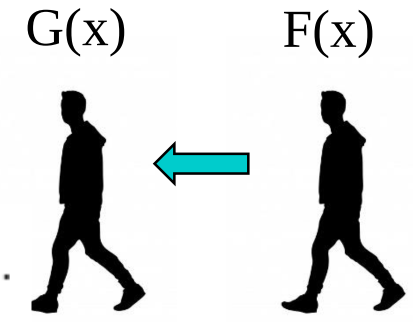
u

VI

Examples LKT tracker

$$h \approx \frac{\sum_x \frac{G(x) - F(x)}{F'(x)}}{\sum_x 1}.$$

$$w(x) = \frac{1}{|G'(x) - F'(x)|}.$$



$$h = \frac{\sum_x \frac{w(x)\,[G(x) - F(x)]}{F'(x)}}{\sum_x w(x)}.$$

$$\begin{cases} h_0 = 0 \\ h_{k+1} = h_k + \dfrac{\sum_x \frac{w(x)\,[G(x) - F(x + h_k)]}{F'(x + h_k)}}{\sum_x w(x)} \end{cases}$$

KLT tracker (better formulation)

$$F(x+h) \approx F(x) + hF'(x)$$

$$E = \sum_x \left[F(x+h) - G(x)\right]^2 .$$

$$\begin{aligned}
0 &= \frac{\partial E}{\partial h} \\
&\approx \frac{\partial}{\partial h} \sum_x \left[F(x) + hF'(x) - G(x)\right]^2 , \\
&= \sum_x 2F'(x)\left[F(x) + hF'(x) - G(x)\right]
\end{aligned}$$

$$\Rightarrow h \approx \frac{\sum_x F'(x)[G(x) - F(x)]}{\sum_x F'(x)^2}$$

$$\begin{cases} h_0 = 0 \\ h_{k+1} = h_k + \dfrac{\sum_x w(x) F'(x+h_k)\left[G(x) - F(x+h_k)\right]}{\sum_x w(x) F'(x+h_k)^2} \end{cases}$$