Use of Motion In Segmentation

Take the difference between a reference image and a subsequent image to determine the still elements image components.

a b c

FIGURE 10.50 Building a static reference image. (a) and (b) Two frames in a sequence. (c) Eastbound automobile subtracted from (a) and the background restored from the corresponding area in (b). (Jain and Jain.)

Motion detection and estimation

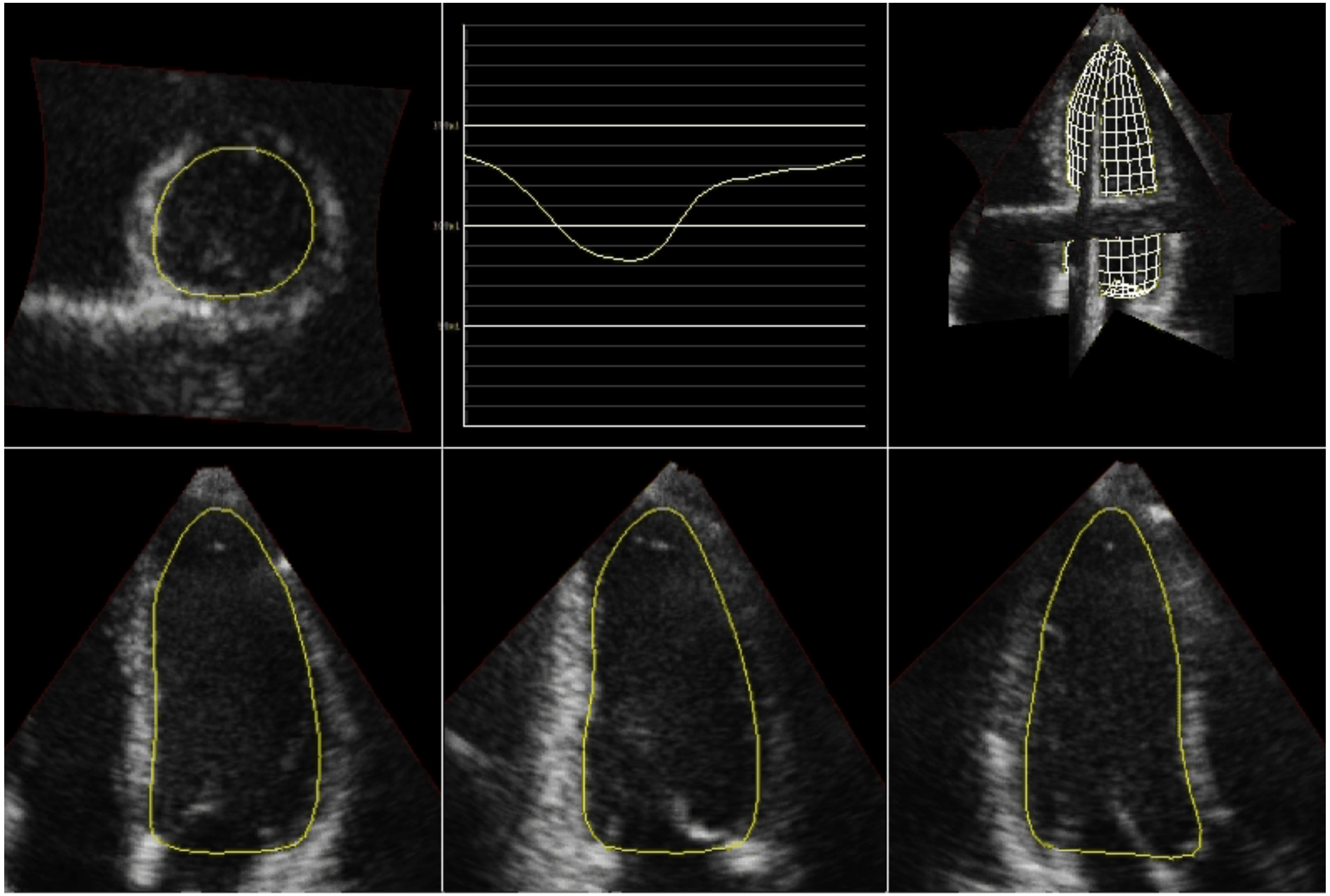
Motion detection and estimation

- A video sequence is a much richer source of visual information than a still image.
  - This is primarily due to the capture of motion; while a single image provides a snapshot of a scene, a sequence of images registers the dynamics in it.
  - The registered motion is a very strong cue for human vision; we can easily recognize objects as soon as they move even if they are inconspicuous when still.
- Main applications
  - Video analysis (through feature extraction)
  - Video compression and coding (MPEG4)
  - Investigation of the dynamics of human organs

Why is it important?

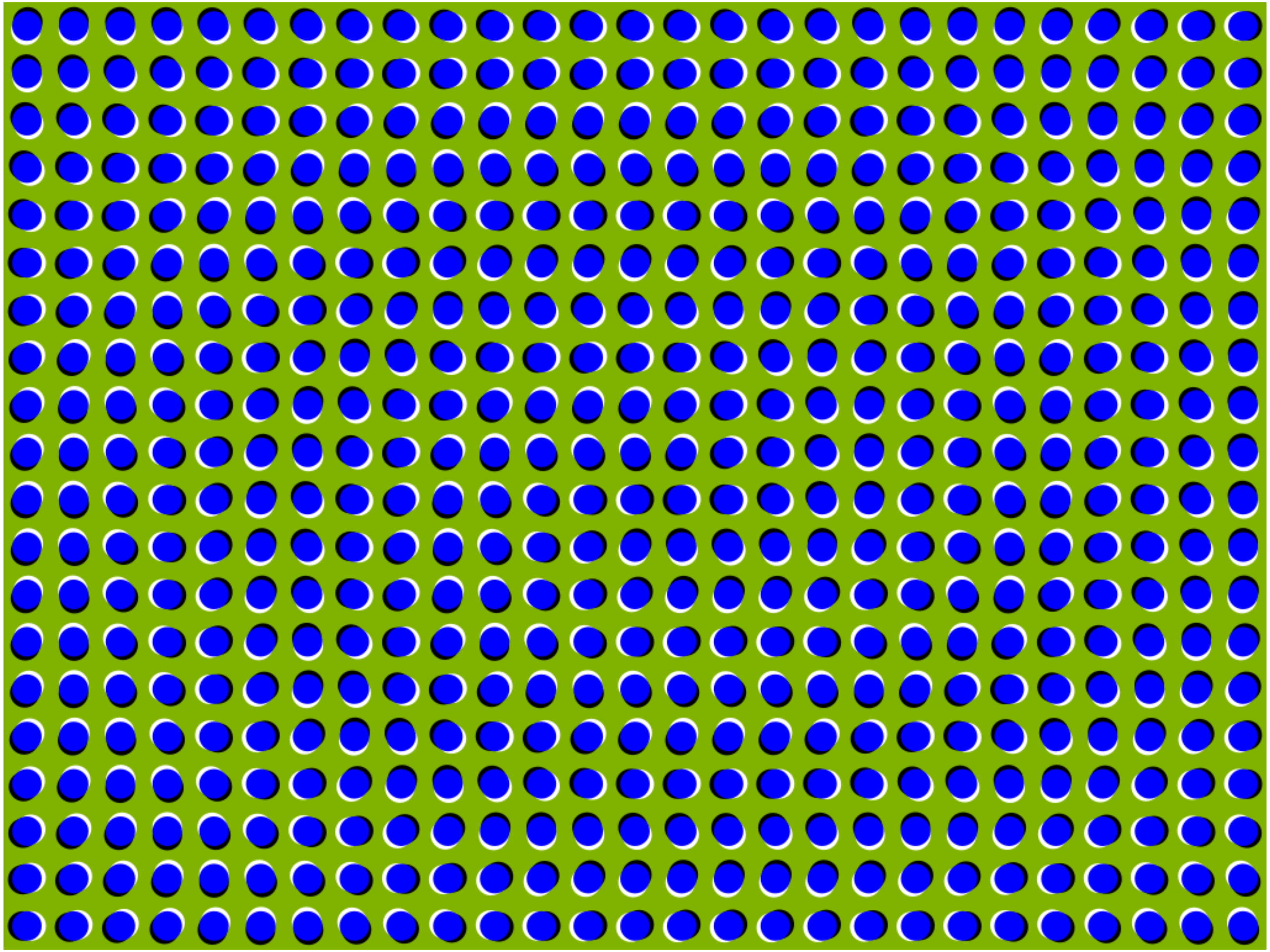
- First, motion carries a lot of information about *spatiotemporal relationships* between image objects. This information can be used in such applications as traffic monitoring or security surveillance, for example to identify objects entering or leaving the scene or objects that just moved.
- Secondly, image properties, such as intensity or color, have a very high correlation in the direction of motion, i.e., they do not change significantly when tracked in the image (the color of a car does not change as the car moves across the image). This can be used for the removal of temporal video redundancy;

Example: Left ventricle dynamics from US



Basic operations

- Motion detection: do the points (or objects) move?
- Motion estimation: how do they move?
- Special case: apparent motion
  - One object displayed at different positions in different time instants is perceived as a moving object



Motion based segmentation

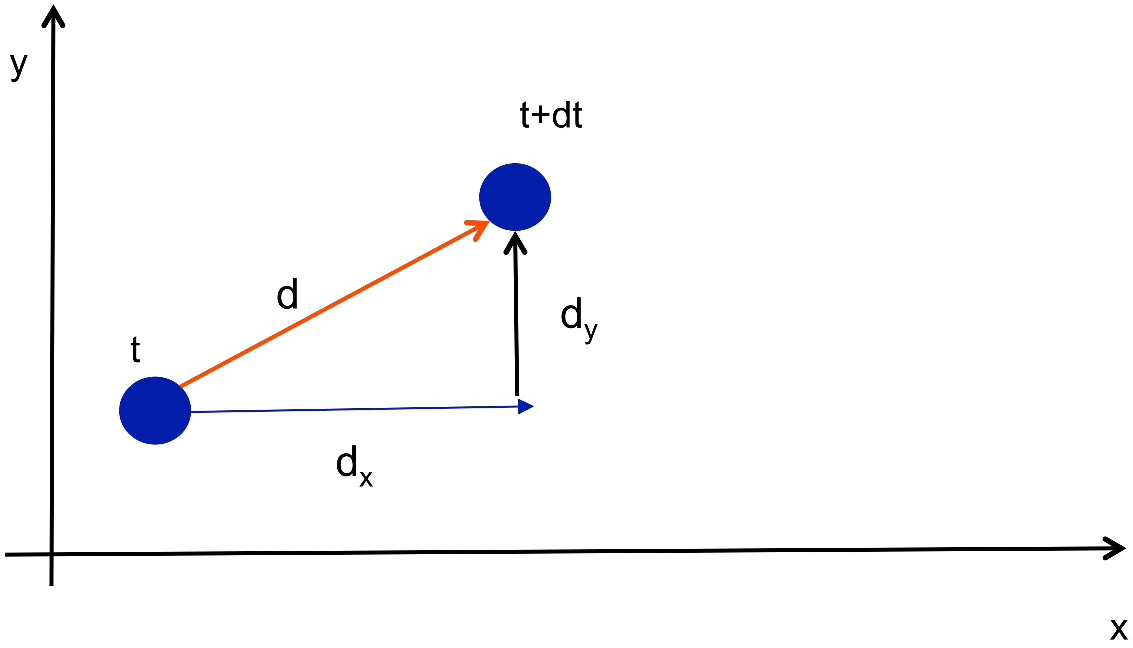
- Motion segmentation, i.e., the identification of groups of image points moving similarly
- Concept: detect the changes from one image to the next
- Possible approaches
  - Taking image differences
  - Block matching
  - Optical flow

Notions and preliminaries

- Let **x** = (x**,** y)^T be a spatial position of a pixel in continuous coordinates, i.e., x is in R^2 within image limits, and let I_t denote image intensity at time t
- After sampling, x is discrete and lives in Z^2
- Let v(x,y) be the velocity at the spatial position (x,y). Then, v_t will denote a velocity field or motion field, i.e., the set of all velocity vectors within the image, at time t
- For discrete images, *the notion of velocity is replaced by displacement d*, but the meaning is unchanged since d represents the displacement of pixel (x,y) between two time instants t_1 and t_2 thus it is representative of its "velocity"

Displacement / velocity

$$P\left(x+d_x, y+d_y\right)_{t+\Delta t} = P\left(x, y\right)_t$$



Motion estimation

- Pixel differences (difference image)
- Optical flow
- Block matching

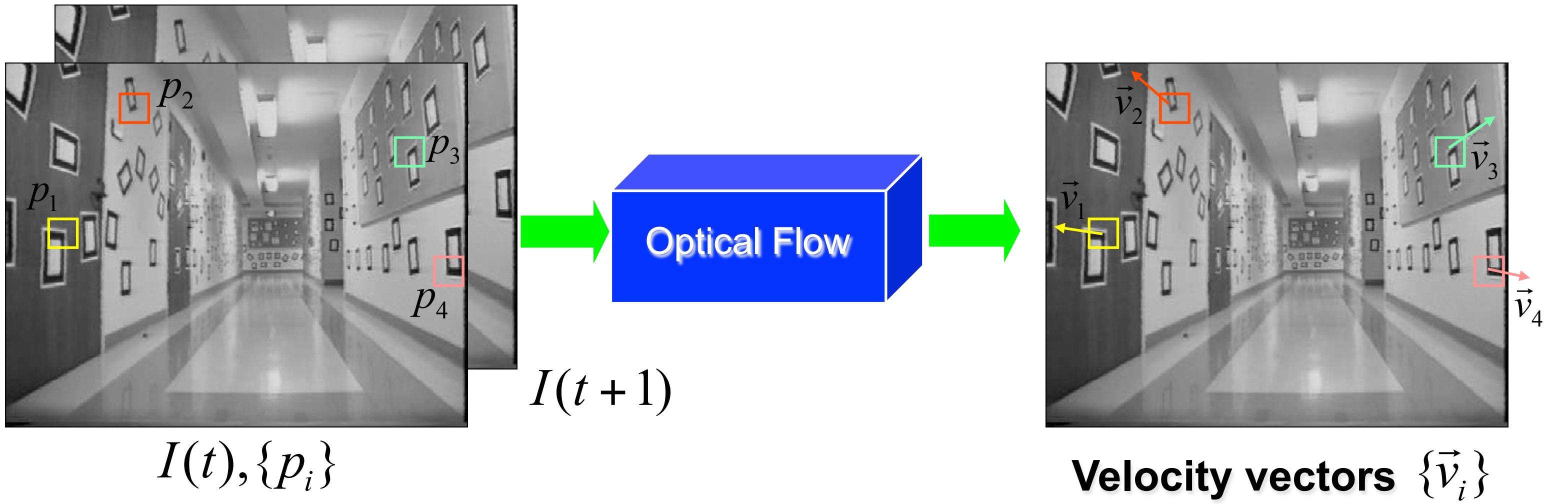
Difference images

- Difference image between two images taken at time points ti and tj

$$d_{ij}(x,y) = \begin{cases} 1 & \text{if } |f(x,y,t_i) - f(x,y,t_j)| > T \\ 0 & \text{otherwise} \end{cases}$$

  - d_{ij}=1 only if the difference between the pixel values in the two images are above a given threshold T
  - d_{ij} has the same size as the two images
- Drawbacks
  - Sensitivity to noise
    - *Accumulation* strategies can be devised
  - Only allows to detect motion but not to characterize it
    - This would require establishing correspondences among pixels to calculate *motion vectors*

What is Optical Flow?



Optical flow: *2D projection of the physical movement of points relative to the observer* to 2D displacement of pixel patches on the image plane.

Common assumption:

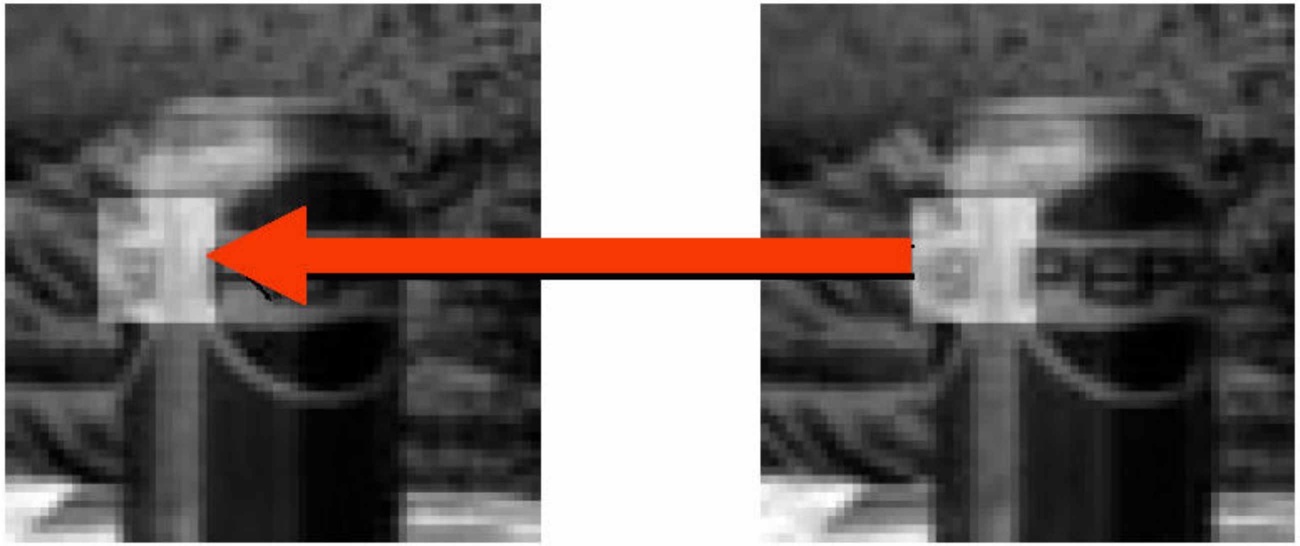
The appearance of the image patches do not change (brightness constancy)

$$I(p_i, t) = I(p_i + \vec{v}_i, t+1)$$

When does it fail?

- Illusory motion: the set is stationary yet things seem to move
- A uniform rotating sphere: nothing seems to move, yet it is rotating
- Changing directions or intensities of lighting can make things seem to move
  - for example, if the specular highlight on a rotating sphere moves.
- And infinitely more break downs of optical flow.

OF Assumptions: Brightness Constancy

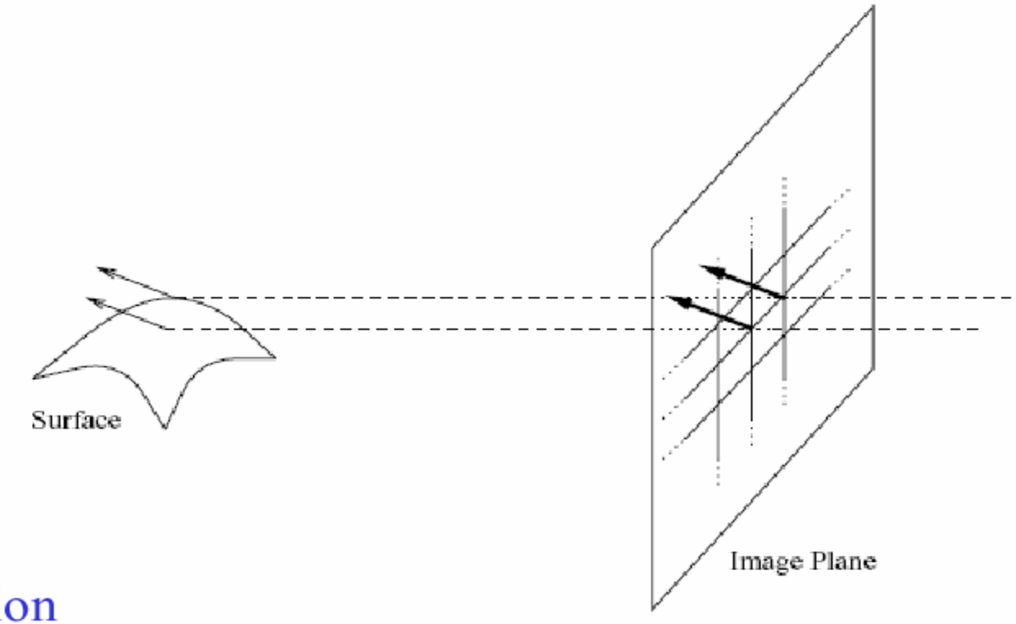


Assumption

Image measurements (e.g. brightness) in a small region remain the same although their location may change.

OF Assumptions

Spatial Coherence

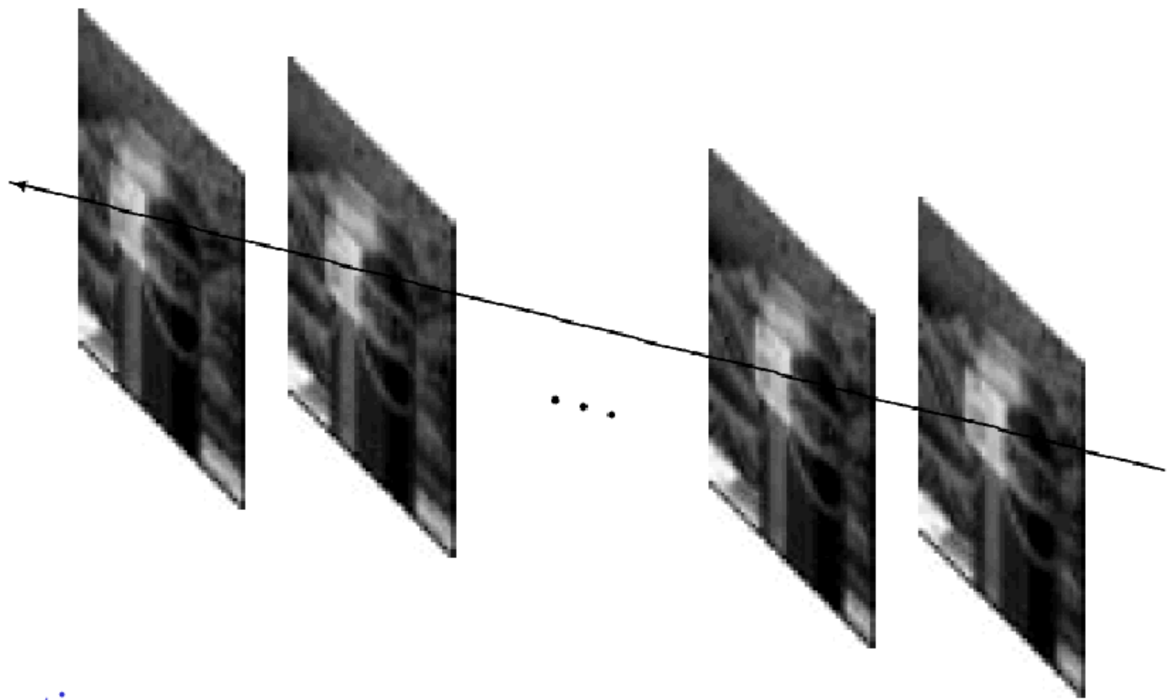


Assumption

* Neighboring points in the scene typically belong to the same surface and hence typically have similar motions.
* Since they also project to nearby points in the image, we expect spatial coherence in image flow.

OF Assumptions

Temporal Persistence



Assumption:

The image motion of a surface patch changes gradually over time.

1D case

Brightness Constancy Assumption:

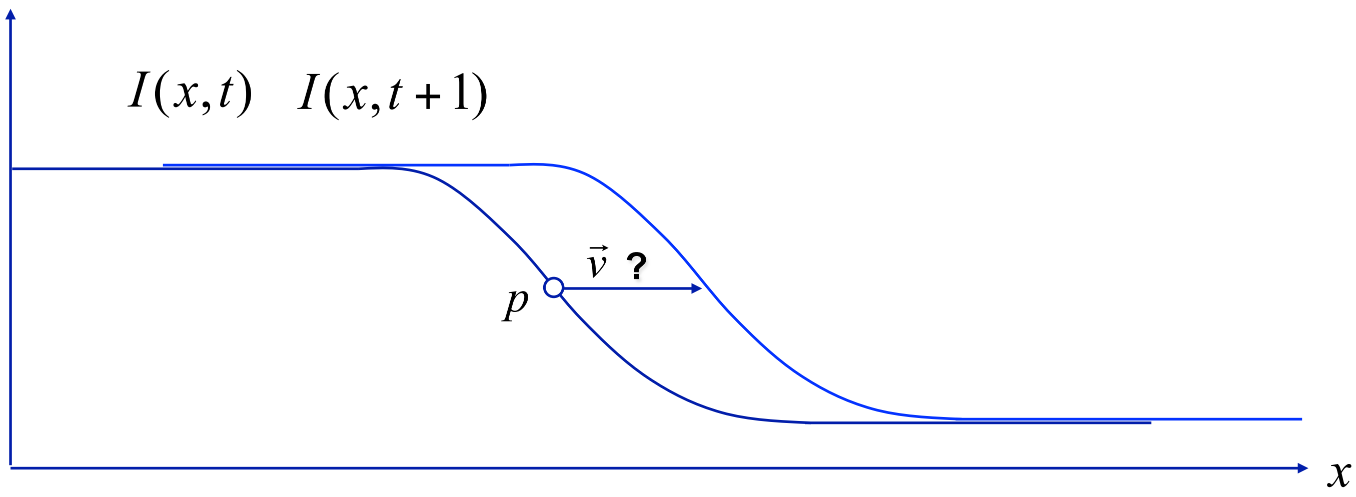
$$f(x(t)) \equiv I(x(t),t) = I(x(t+dt),t+dt)$$

$$\frac{\partial f(x)}{\partial t} = 0 \quad \Longleftrightarrow$$ No changes in brightness in time

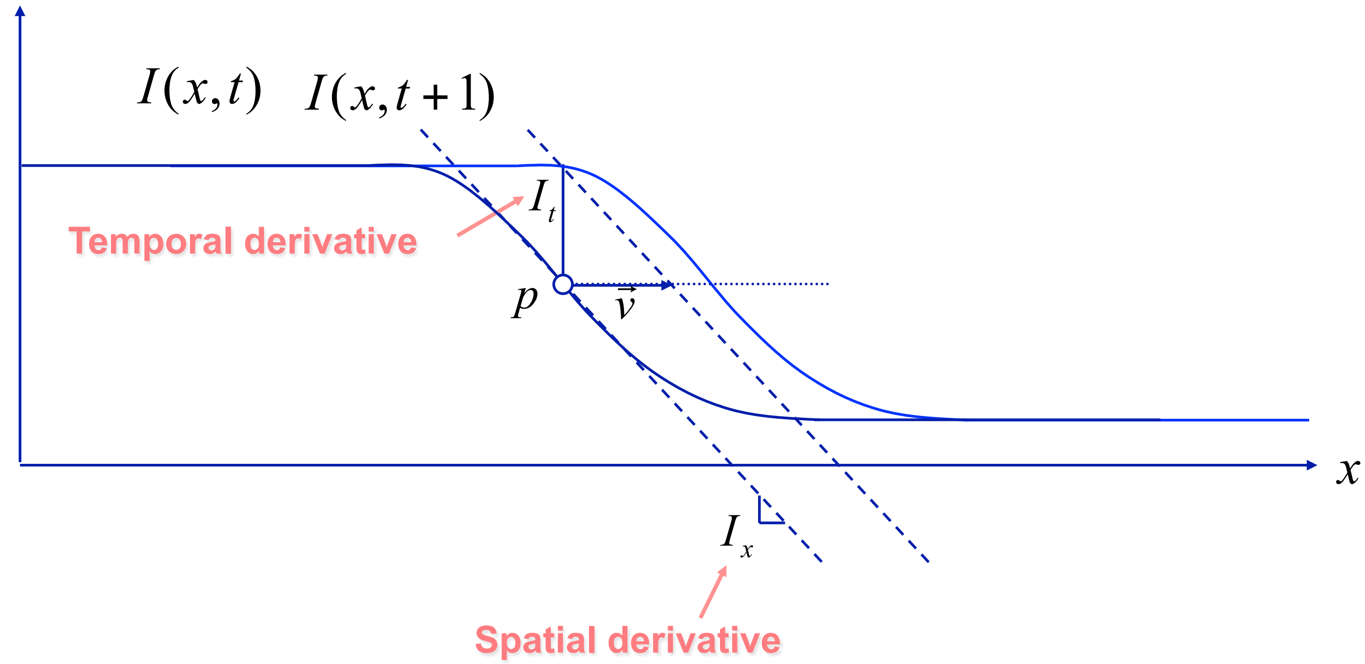
$$\underbrace{\left.\frac{\partial I}{\partial x}\right|_{t}}_{I_x} \underbrace{\left(\frac{\partial x}{\partial t}\right)}_{v} + \underbrace{\left.\frac{\partial I}{\partial t}\right|_{x(t)}}_{I_t} = 0$$

$$\Longrightarrow v = -\frac{I_t}{I_x}$$

Tracking in the 1D case



Tracking in the 1D case



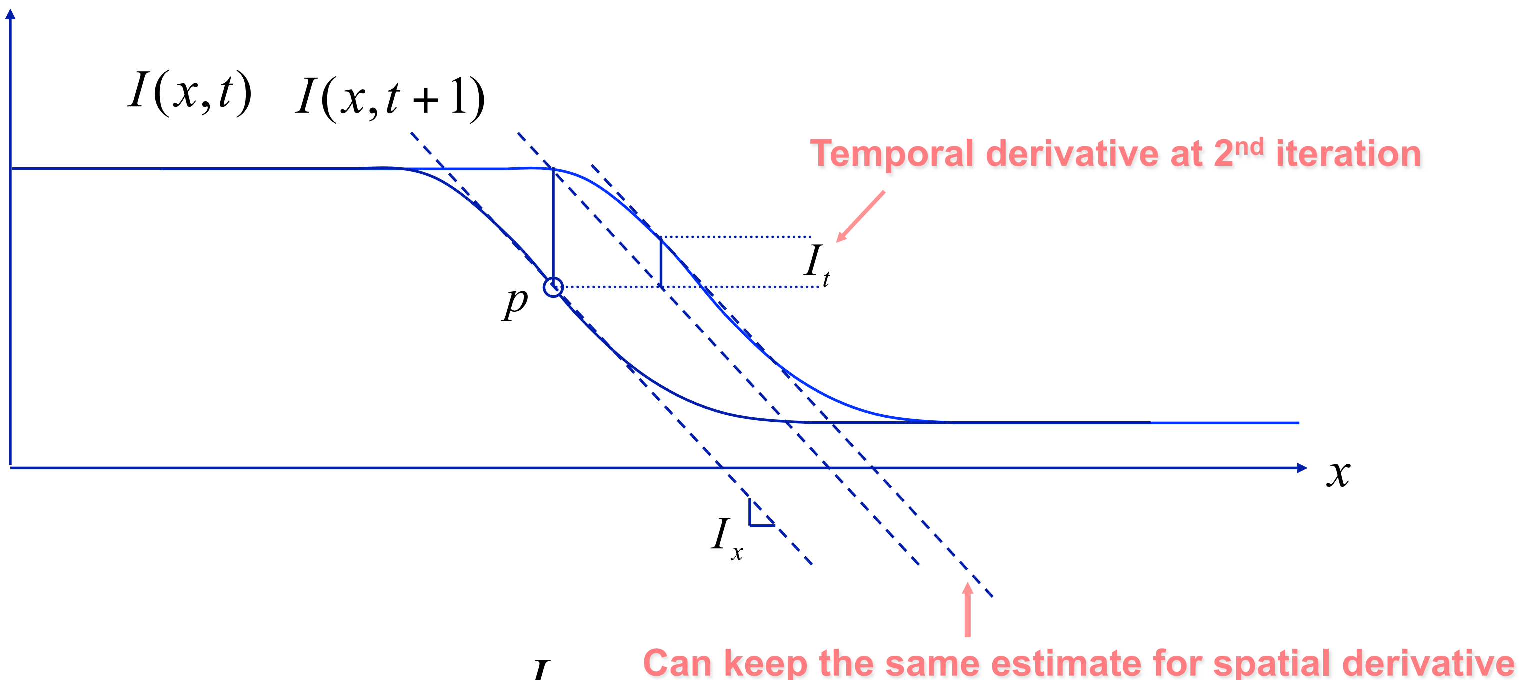
$$I_x = \left.\frac{\partial I}{\partial x}\right|_t \qquad I_t = \left.\frac{\partial I}{\partial t}\right|_{x=p} \qquad \Rightarrow \qquad \vec{v} \approx -\frac{I_t}{I_x}$$

Assumptions:

- **Brightness constancy**
- **Small motion**

Tracking in the 1D case

Iterating helps refining the velocity vector



$$\vec{v} \leftarrow \vec{v}_{previous} - \frac{I_t}{I_x}$$

Converges in about 5 iterations

Algorithm

For all pixel of interest *p*:

- **Compute local image derivative at *p*:** I_x
- **Initialize velocity vector:** $\vec{v} \leftarrow 0$
- **Repeat until convergence:**
  - **Compensate for current velocity vector:** $I'(x,t+1) = I(x+\vec{v},t+1)$
  - **Compute temporal derivative:** $I_t = I'(p,t+1) - I(p,t)$
  - **Update velocity vector:** $\vec{v} \leftarrow \vec{v} - {I_t}/{I_x}$

Requirements:

- Need access to neighborhood pixels around p to compute I_x
- Need access to the second image patch, for velocity compensation:
  - The pixel data to be accessed in next image depends on current velocity estimate (bad?)
  - Compensation stage requires a bilinear interpolation (because *v* is not integer)
- The image derivative I_x needs to be kept in memory throughout the iteration process

From 1D to 2D tracking

1D: $$\left.\frac{\partial I}{\partial x}\right|_t \left(\frac{\partial x}{\partial t}\right) + \left.\frac{\partial I}{\partial t}\right|_{x(t)} = 0$$

2D: $$\left.\frac{\partial I}{\partial x}\right|_t \left(\frac{\partial x}{\partial t}\right) + \left.\frac{\partial I}{\partial y}\right|_t \left(\frac{\partial y}{\partial t}\right) + \left.\frac{\partial I}{\partial t}\right|_{x(t)} = 0$$

$$\left.\frac{\partial I}{\partial x}\right|_t u + \left.\frac{\partial I}{\partial y}\right|_t v + \left.\frac{\partial I}{\partial t}\right|_{x(t)} = 0$$

One equation, two velocity components unknown *(u,v)*

From 1D to 2D tracking

Notation

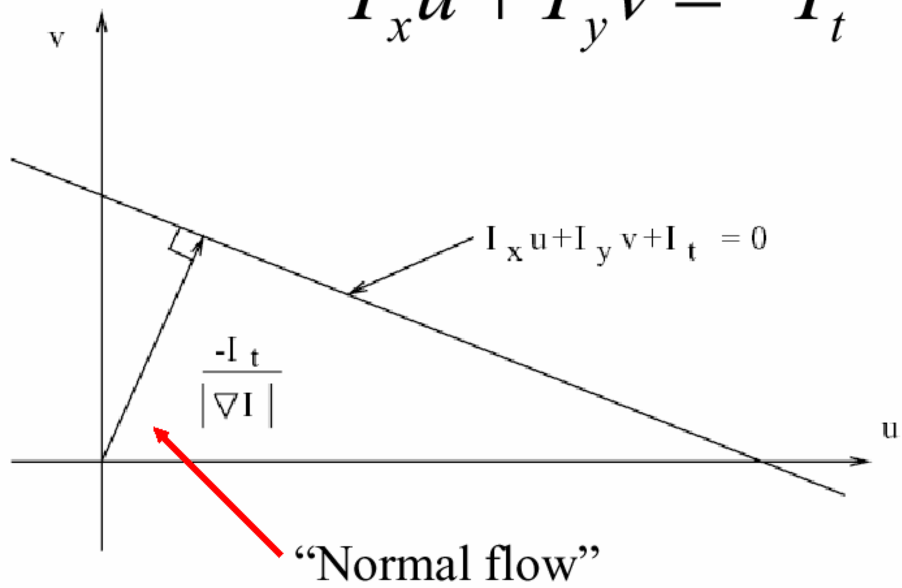
$$I_x u + I_y v + I_t = 0$$

$$\nabla I^T \mathbf{u} = -I_t$$

$$\mathbf{u} = \begin{bmatrix} u \\ v \end{bmatrix} \quad \nabla I = \begin{bmatrix} I_x \\ I_y \end{bmatrix}$$

At a single image pixel, we get a line:

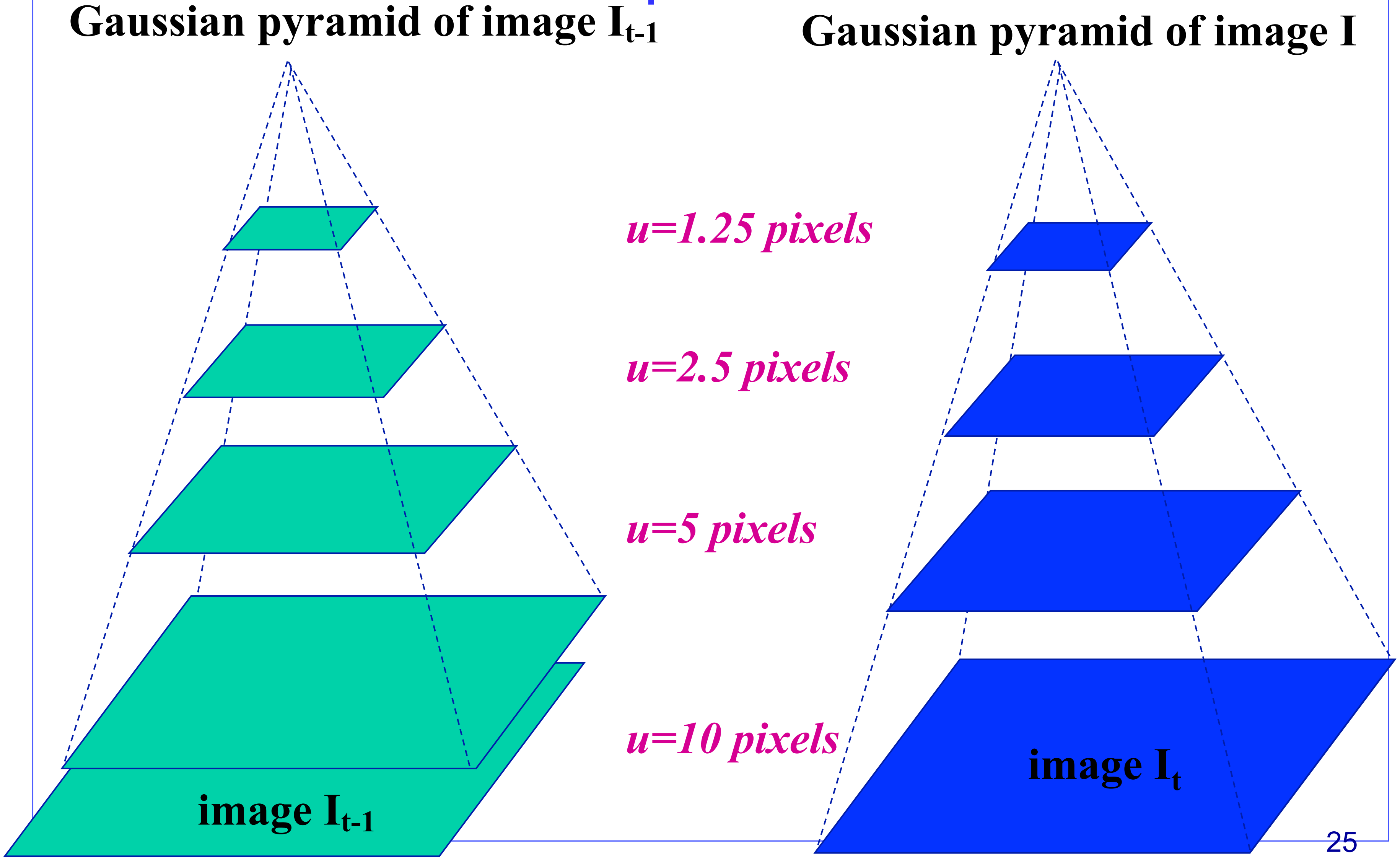
$$I_x u + I_y v = -I_t$$



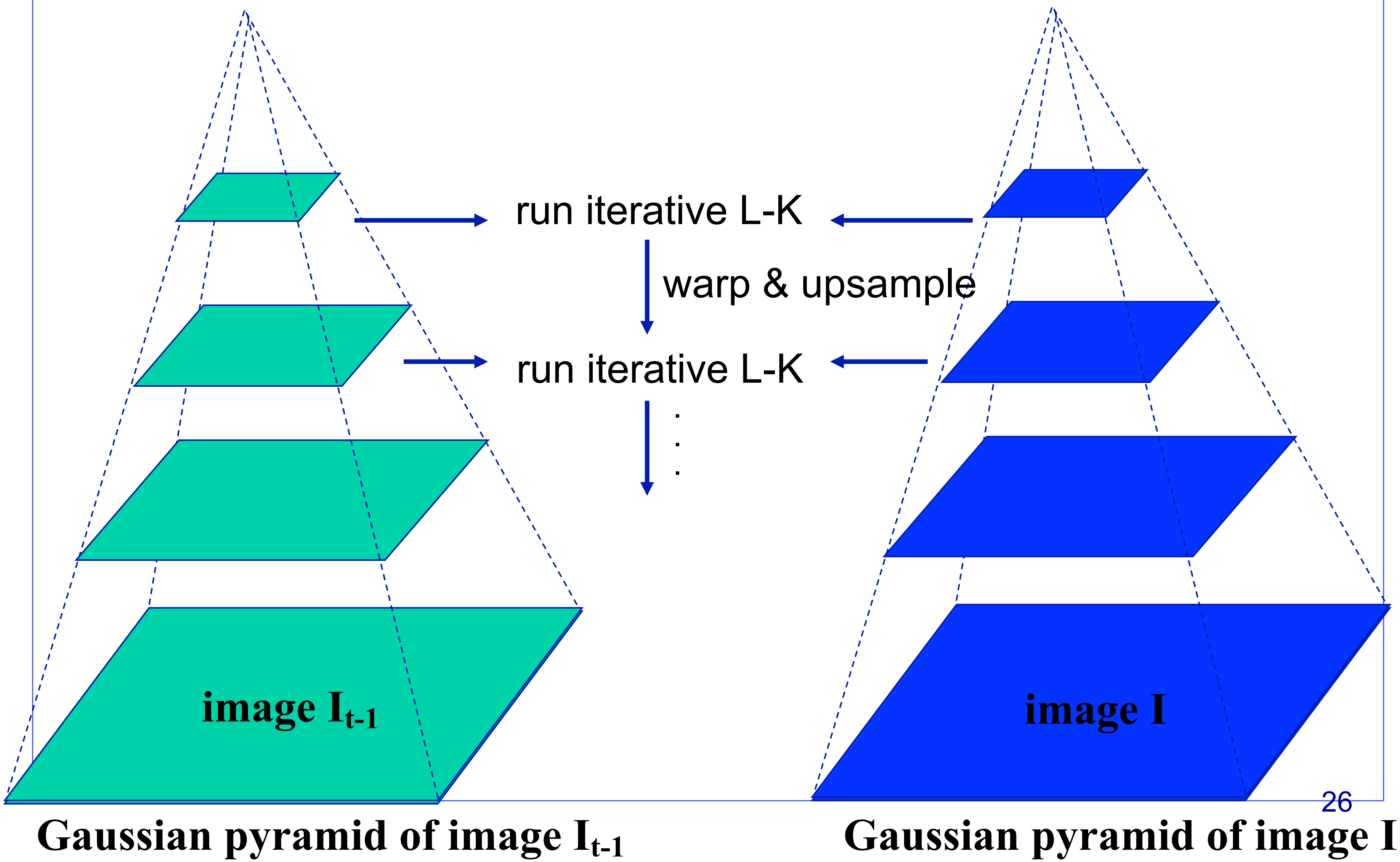
We get at most "Normal Flow" – with one point we can only detect movement perpendicular to the brightness gradient. Solution is to take a patch of pixels Around the pixel of interest.

* Slide from Michael Black, CS143 2003

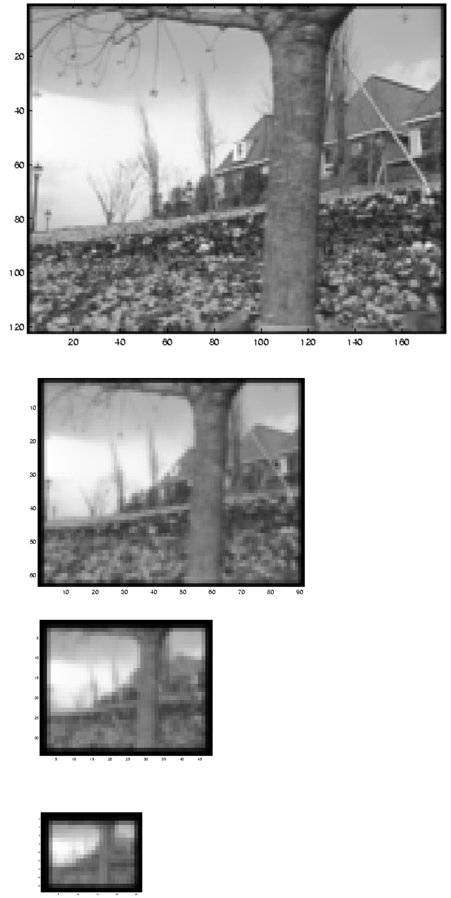
Coarse-to-fine optical flow estimation



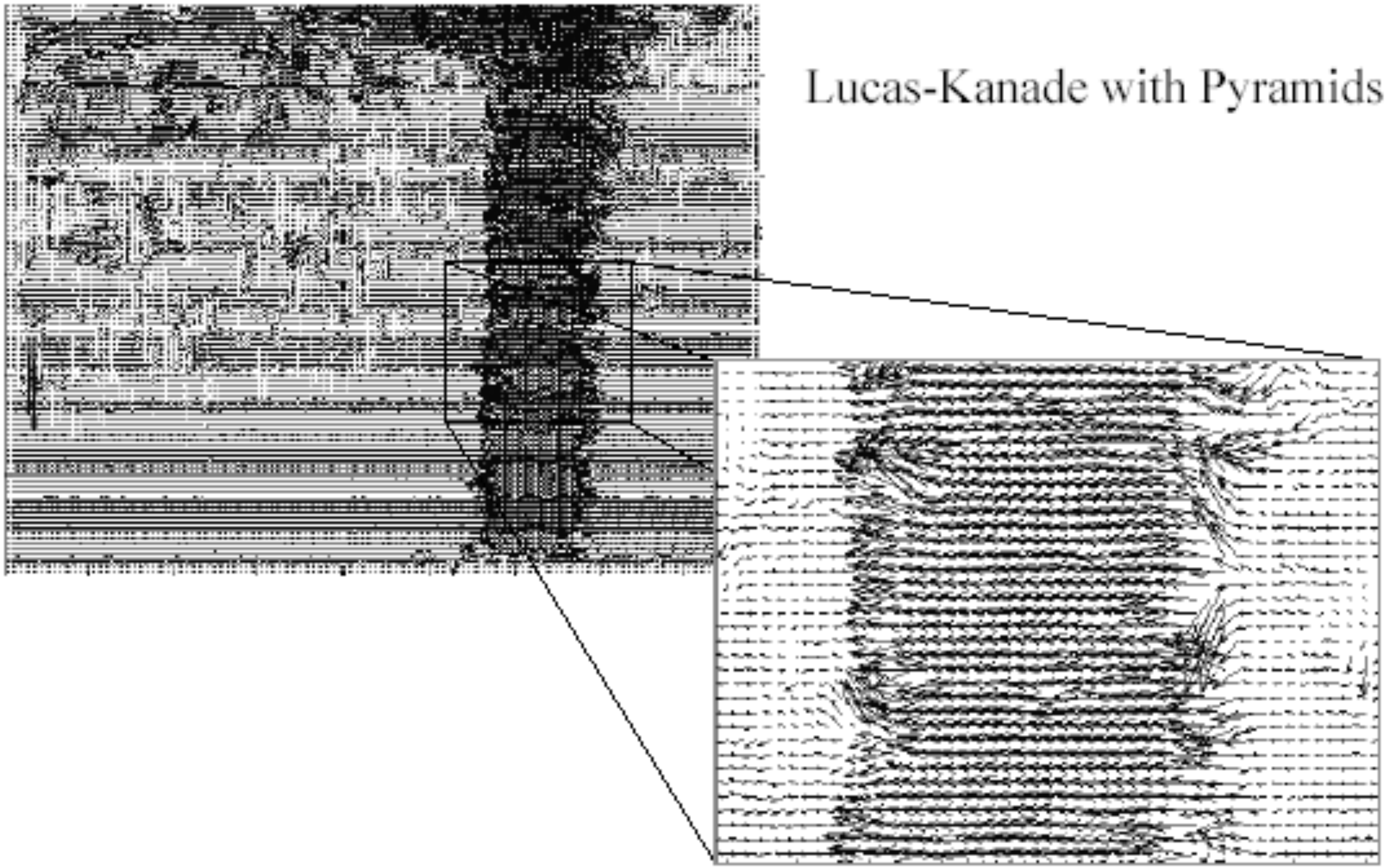
Coarse-to-fine optical flow estimation



Example


20
40
60
80
100
120
20 40 60 80 100 120 140 160
10
20
30
40
50
60
10 20 30 40 50 60 70 80 90

Example: Results



Block matching

Block-matching

16x16 pixels/block

Search window: ±16 pixels from the original position

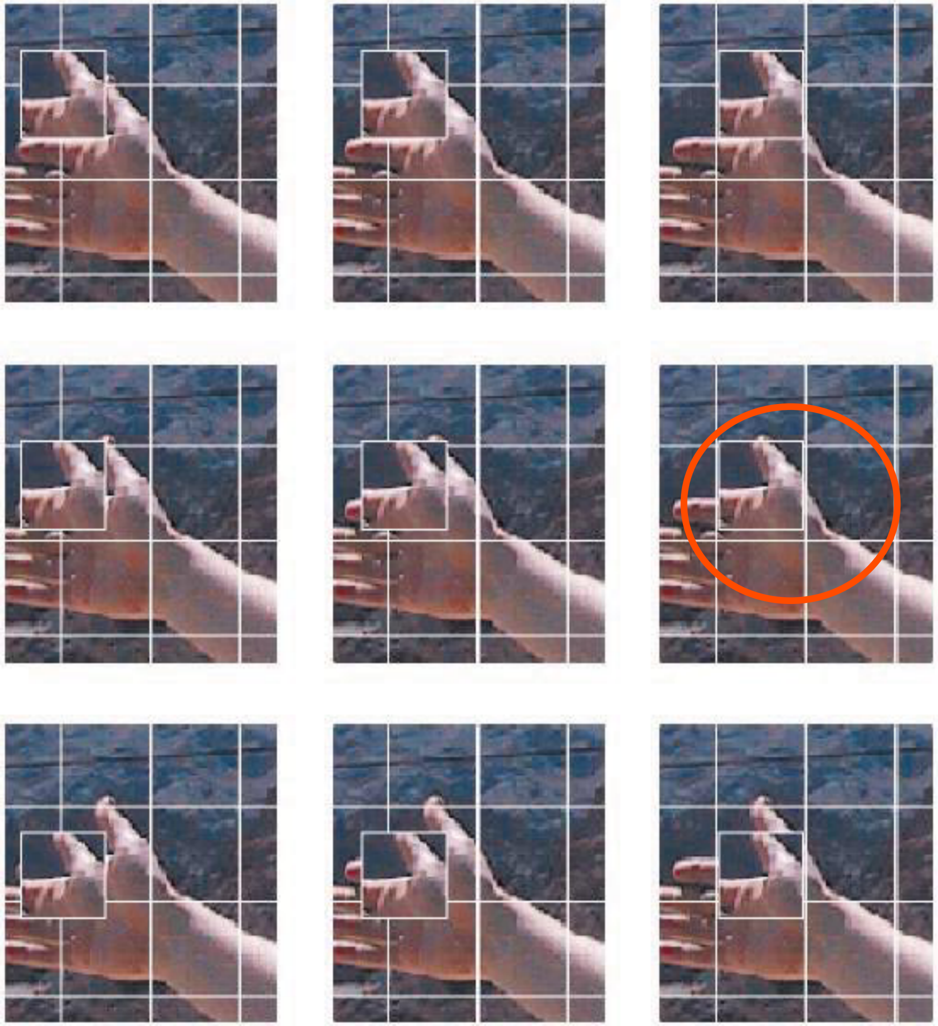
Computationally heavy!

To reduce the complexity

Sub-optimal algorithms

Hardware assisted

Block matching



Motion estimation & Motion compensation