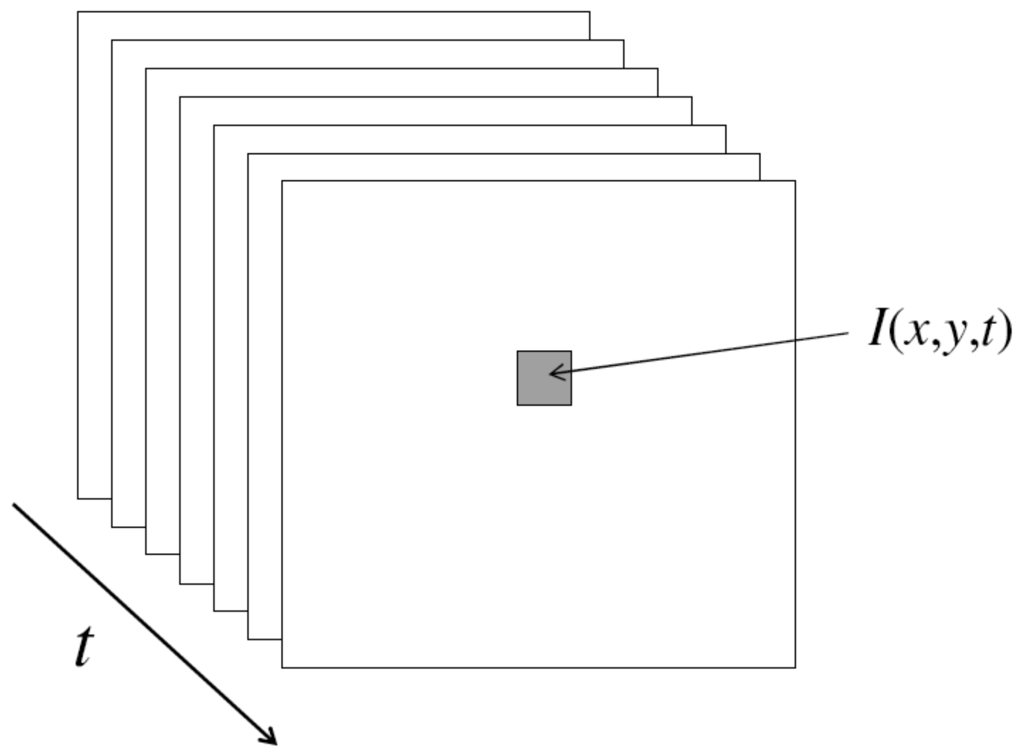
Dense Image-based Motion Estimation Algorithms & Optical Flow

McMaster University

Video

- A video is a sequence of frames captured at different times
- The video data is a function of
  - ❖ time (t)
  - ❖ space (x,y)



Introduction to motion estimation

Given a video sequence of moving objects or camera, what information can we extract?

- How is the camera moving?
- How many moving objects exist?
- What is the direction of each moving object?
- How fast is object moving?

If we could get the answer of these questions we can interpret the scene better.

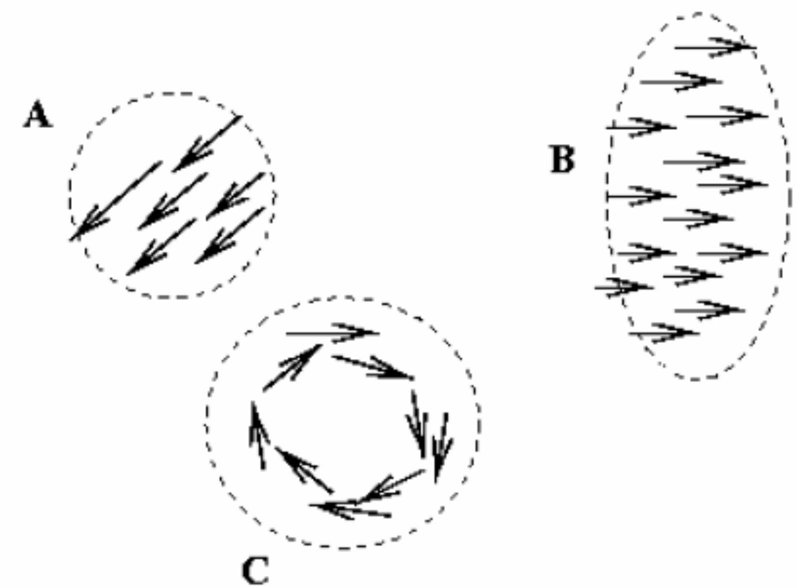
Applications

- Background Subtraction
  - ❖ a stationary camera is observing the scene
  - ❖ Goal: Separate the static background from the moving foreground

Applications

- Motion Segmentation

Segment the video to the moving objects with different motions



Other Applications

- Estimating 3D structure
- Segmenting objects based on motion cues
- Recognizing events and activities
- Improving video quality (motion stabilization)

Image Alignment

- Alignment between two images or image patches

template image

$I_0(\mathbf{x})$

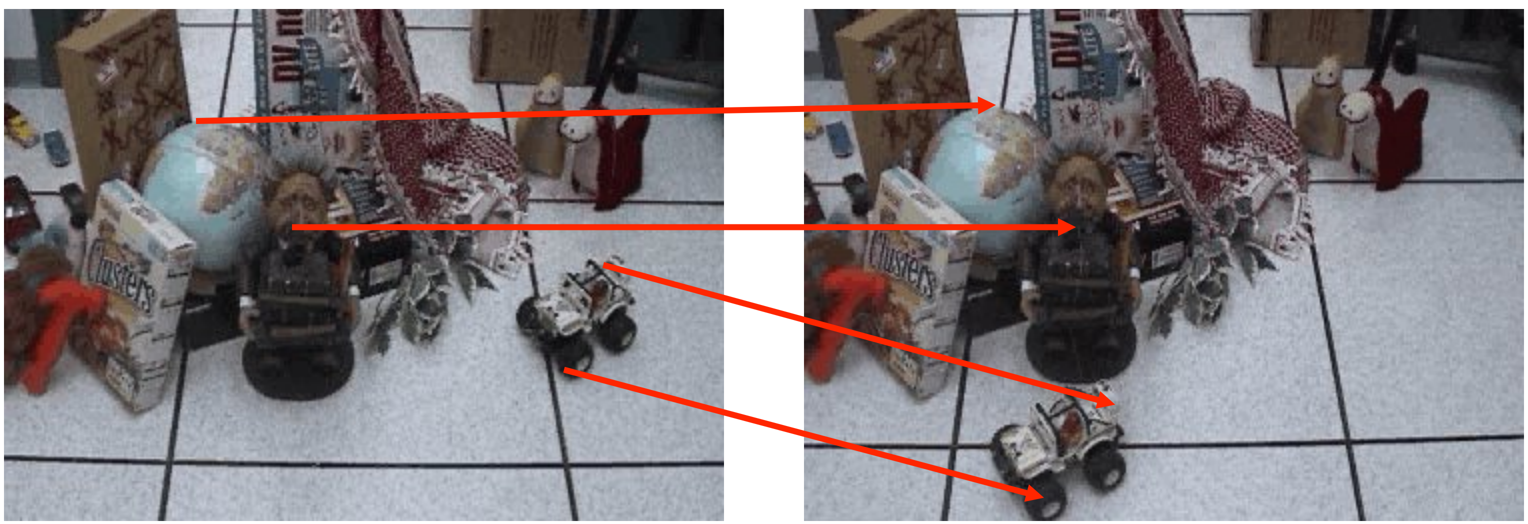
discrete pixel locations

$\{\mathbf{x}_i = (x_i, y_i)\}$

image

$I_1(\mathbf{x})$

$\{\mathbf{x}'_i = (x'_i, y'_i)\}$



Motion Estimation

To estimate motion between two or more images:

- Error Metric

❖ Measuring the similarity/dissimilarity between images

- Search Technique

❖ Full search (Simple but too slow)

❖ Hierarchical coarse-to-fine methods based on image pyramids

- Optical Flow

❖ Multiple independent motions

Translational Alignment

- Sum of Squared Differences

$$E_{SSD}(\mathbf{u}) = \sum_i \left[I_1(\mathbf{x}_i + \mathbf{u}) - I_0(\mathbf{x}_i) \right]^2 = \sum_i e_i^2$$

- ❖ $u=(u,v)$: displacement vector
- ❖ $e_i = I_1(x_i + u) - I_0(x_i)$:the residual error

Displaced Frame Difference (Video Coding)

Translational Alignment

$$E_{SSD}(\mathbf{u}) = \sum_i \left[I_1(\mathbf{x}_i + \mathbf{u}) - I_0(\mathbf{x}_i) \right]^2 = \sum_i e_i^2$$

- ❖ Assumptions:
  - ✓ Ignoring the possibility that for a given alignments some parts of I_0 lie outside of I_1 boundaries and so are not visible
  - ✓ Assuming that corresponding pixel values remain the same in two images.
- ❖ u can be fractional : Interpolation Functions needed
- ❖ Color images :
  - ✓ Do the same for all 3 color channels
  - ✓ Transform image into a different color space

Robust Error Metrics

- Replacing the squared error terms with a robust function $\rho(e_i)$

$$E_{SRD}(\mathbf{u}) = \sum_i \rho(I_1(\mathbf{x}_i + \mathbf{u}) - I_0(\mathbf{x}_i)) = \sum_i \rho(e_i)$$

Grows less quickly than the quadratic penalty associated with least squares

❖ Sum of Absolute differences

$$E_{SAD}(\mathbf{u}) = \sum_i \left| I_1(\mathbf{x}_i + \mathbf{u}) - I_0(\mathbf{x}_i) \right| = \sum_i |e_i|$$

✓ widely used in motion estimation for video coding because of it's speed

✓ESAD is NOT differentiable at the origin, not well suited to gradient descent approaches

Robust Error Metrics

- Smoothly varying function (Black and Rangarajan (1996))
  - Quadratic for small values but
  - grows more slowly away from the origin
- Geman–McClure function

$$\rho_{GM}(x) = \frac{x^2}{1 + x^2 / a^2}$$

a: constant that can be thought of as an outlier threshold

❖ for small values of x :

$$\rho_{GM}(x) \approx x^2$$

❖ as x becomes larger:

$$\rho_{GM}(x) \approx a^2$$

Spatially Varying Weights

- Pixels that may lie outside of the boundaries
- Partially or completely downweight the contribution of certain pixels
  - ❖ Background Stabilization or Background Alignment
    - ✓ downweight the middle part of the image, containing independently moving objects

$$E_{WSSD}(\mathbf{u}) = \sum_i \omega_0(\mathbf{x}_i)\omega_1(\mathbf{x}_i + \mathbf{u})\left[I_1(\mathbf{x}_i + \mathbf{u}) - I_0(\mathbf{x}_i)\right]^2$$

Weighted (or Windowed) SSD function

ω_0 and ω_1 are zero outside of image boundaries

Windowed SSD

- In case of a large range of motion:
  - ❖ The above metric has bias toward smaller overlapping solutions

$$A = \sum_i \omega_0(\mathbf{x}_i)\omega_1(\mathbf{x}_i + \mathbf{u})$$ Overlapping Area

  - ❖ Ton counteract this bias:

$RMS = \sqrt{E_{WSSD} / A} : per - pixel$ squared pixel error

Bias and Gain (Exposure Differences)

- Often the two images being aligned were not taken with the same exposure.
- Simple model of intensity variations:

$$I_1(\mathbf{x}+\mathbf{u}) = (1+\alpha)I_0(\mathbf{x}) + \beta$$

- α is the gain
- β is the bias

Bias and Gain

- Least squares with bias and gain

$$E_{BG}(\mathbf{u}) = \sum_i \left[I_1(\mathbf{x}_i + \mathbf{u}) - (1+\alpha) I_0(\mathbf{x}_i) - \beta \right]^2 = \sum_i \left[\alpha I_0(\mathbf{x}_i) + \beta - e_i \right]^2$$

- Performing a linear regression
- Color image
  - Estimate bias and gain for each color channel
  - Bias and gain compensation is also used in video codecs, known as Weighted Prediction.

Correlation

- Cross-Correlation
  - Alternative to taking intensity difference
  - Maximize the product of two aligned images

$$E_{CC}(\mathbf{u}) = \sum_i I_0(\mathbf{x}_i) I_1(\mathbf{x}_i + \mathbf{u})$$

Is Bias and Gain modeling unnecessary?

Bright patch exists in images

Normalized Cross-Correlation

$$E_{NCC}(\mathbf{u}) = \frac{\sum_i \left[I_0(\mathbf{x}_i) - \overline{I}_0 \right] \left[I_1(\mathbf{x}_i + \mathbf{u}) - \overline{I}_1 \right]}{\sqrt{\sum_i \left[I_0(\mathbf{x}_i) - \overline{I}_0 \right]^2} \sqrt{\sum_i \left[I_1(\mathbf{x}_i + \mathbf{u}) - \overline{I}_1 \right]^2}}$$

where

$$\overline{I}_0 = \frac{1}{N} \sum_i I_0(\mathbf{x}_i) \qquad \textit{and}$$

$$\overline{I}_1 = \frac{1}{N} \sum_i I_1(\mathbf{x}_i + \mathbf{u})$$

- NCC in [-1,1]
- Works well when matching images taken with different exposure
- Degrades for noisy low-contrast regions (Zero variance)

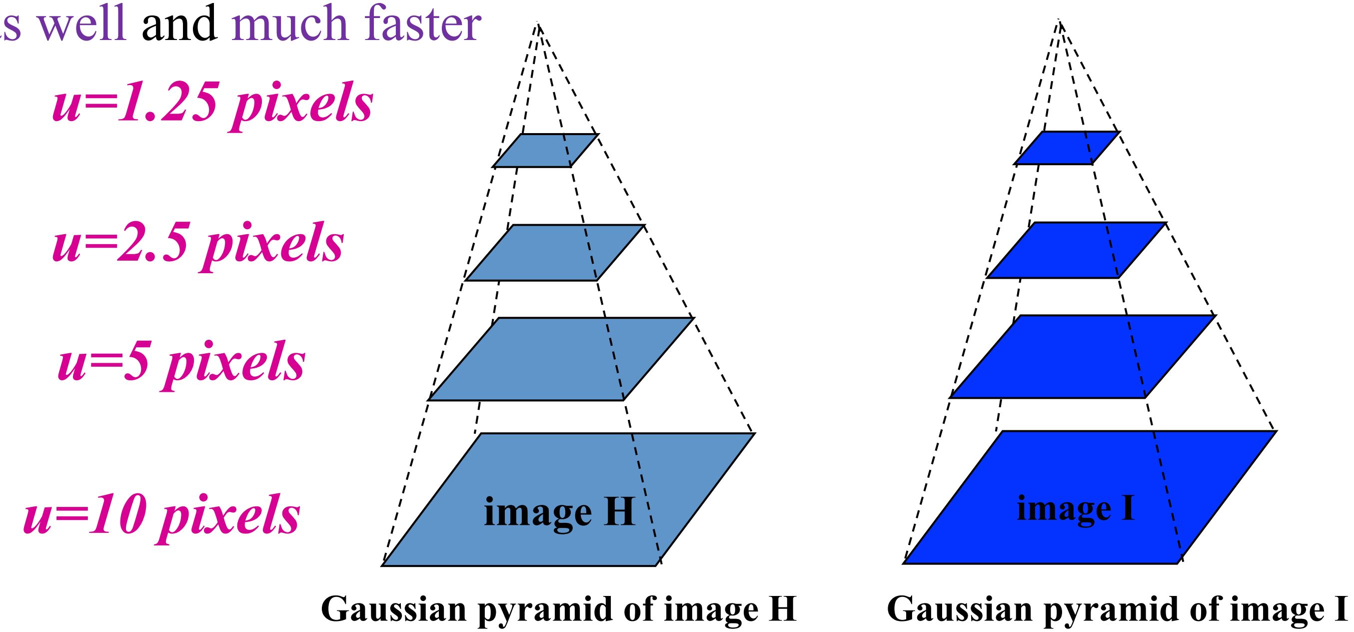
Hierarchical Motion Estimation

- How can we find its minimum?
- Full search over some range of shifts
  - Often used for block matching in motion compensated video compression
  - Simple to implement but slow

- To accelerate the search process
  - Hierarchical motion estimation

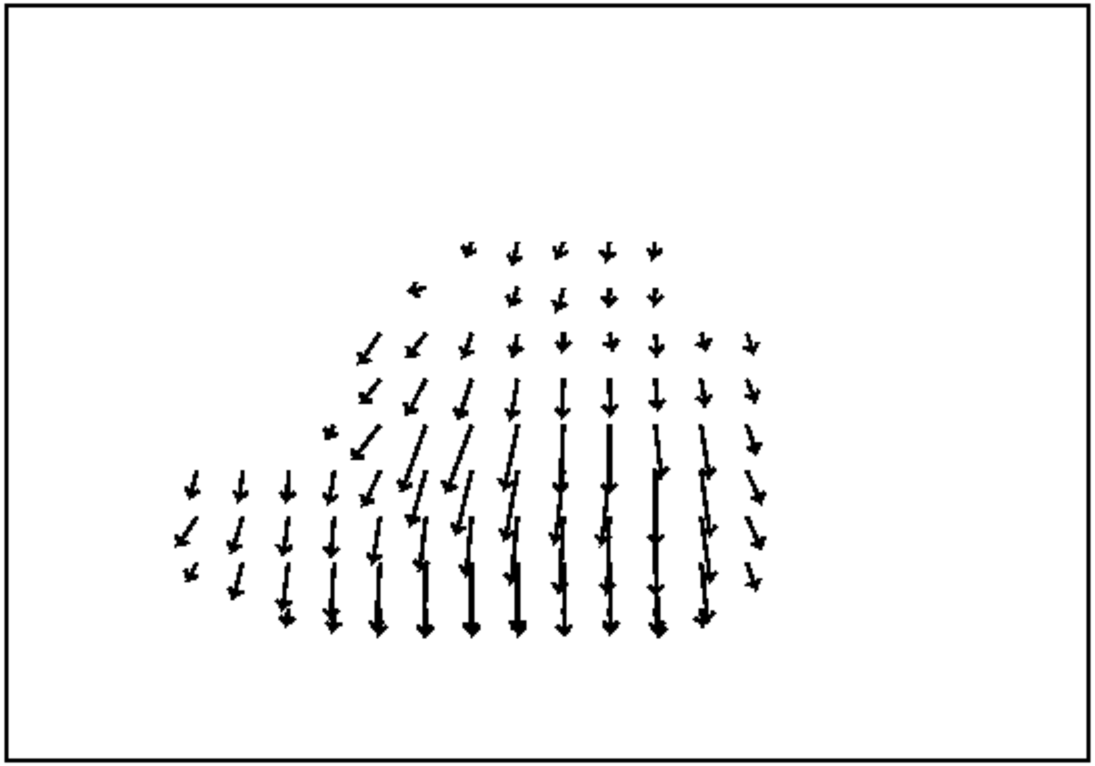
Hierarchical Motion Estimation

- Steps
  - Construct image pyramid
  - Full search over the range $2^{-l}[-S, S]^2$
  - At coarser levels, search over a smaller number of discrete pixels
  - The motion estimation from one level is used to initialize a smaller local search at next finer level
  - Not guaranteed to produce the same results as a full search, but works almost as well and much faster

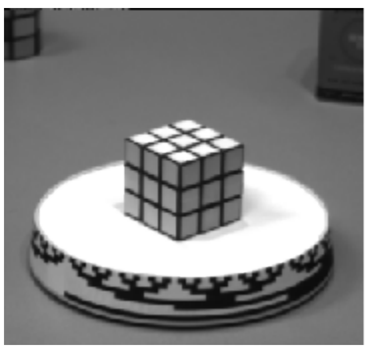
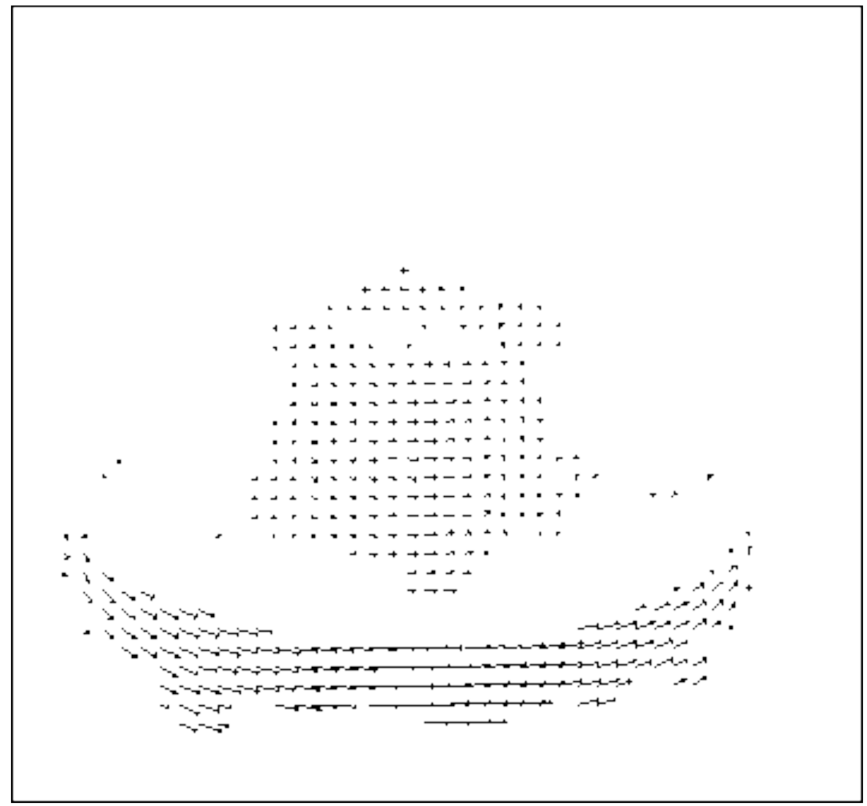


Optical Flow

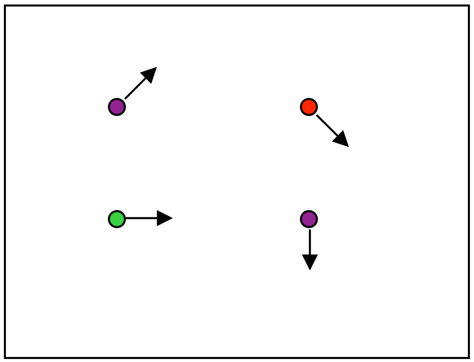
- The most general and challenging version of motion estimation
- Computing an independent estimate of motion at each pixel of the image



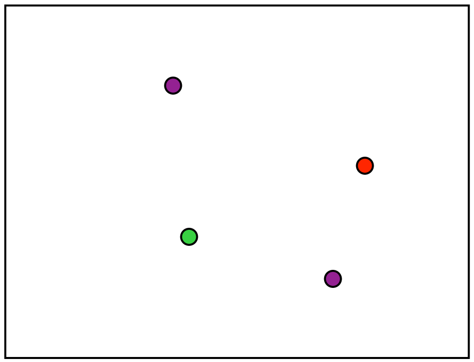
Optical Flow Field

Problem Definition : Optical Flow



$H(x, y)$



$I(x, y)$

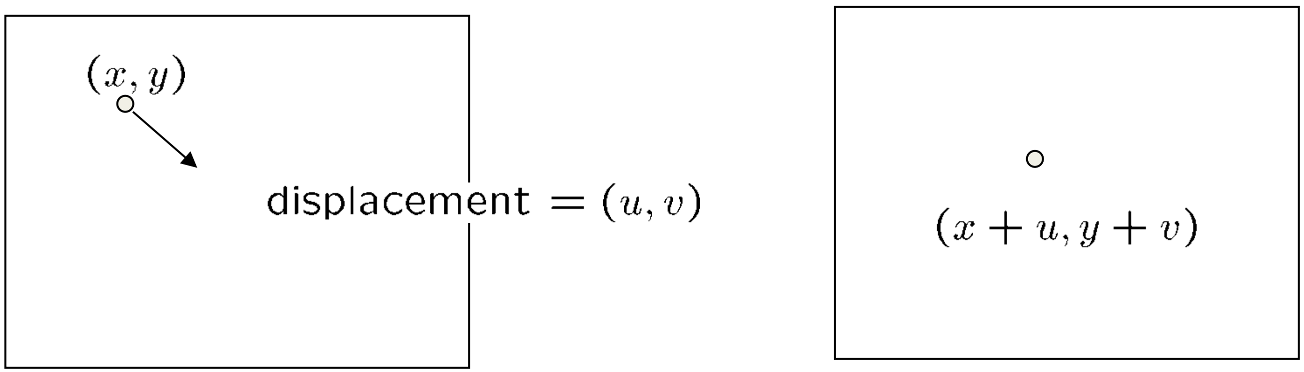
- How to estimate pixel motion from image H to image I?
  - Solve pixel correspondence problem
    - given a pixel in H, look for nearby pixels of the same color in I

Problem Definition

- 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 (gray scale images)



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

- 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:

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

Combining Equations

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

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

The x-component of the gradient vector.

What is I_t? The time derivative of the image at (x,y)

$\boxed{\text{How do we calculate it?}}$

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

Optical Flow Equation

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

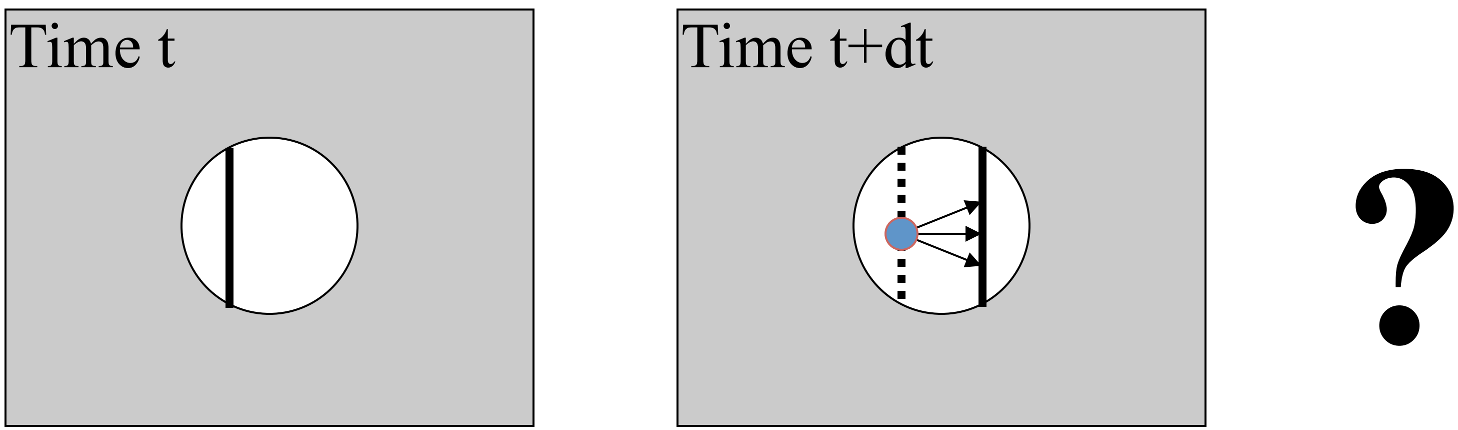
Problem 1:

- Q: how many unknowns and equations per pixel?

1 equation, but 2 unknowns (u and v)

Problem 2: The Aperture Problem

- For points on a line of fixed intensity we can only recover the normal flow

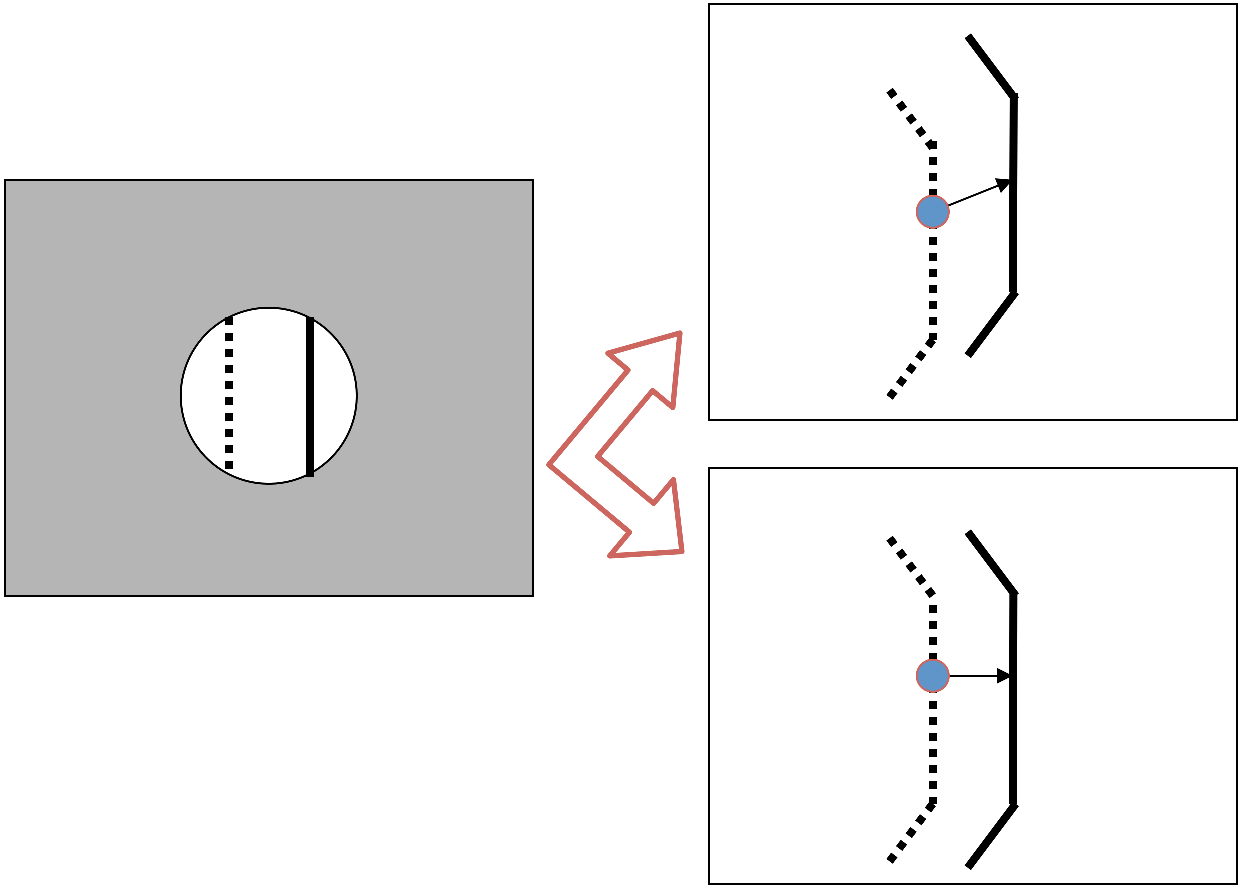


Where did the blue point move to?

We need additional constraints

Use Local Information

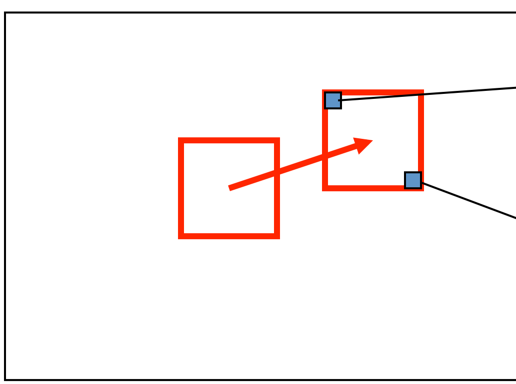
Sometimes enlarging the aperture can help



Local smoothness
Lucas Kanade (1984)

$$I_x u + I_y v = -I_t \quad \Rightarrow \quad \begin{bmatrix} I_x & I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = -I_t$$

- assume locally constant motion
  - pretend the pixel's neighbors have the same (u,v)
    - If we use a 5x5 window, that gives us 25 equations per pixel!



$$\begin{bmatrix} I_{x1} & I_{y1} \\ I_{x2} & I_{y2} \\ \vdots & \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_{t1} \\ I_{t2} \\ \vdots \end{bmatrix}$$

$$\boxed{A\vec{\mathrm{u}} = b}$$

Lucas Kanade (1984)

Goal: Minimize $\left\| A\vec{\mathrm{u}} - b \right\|^2$

Method: Least-Squares

$$A\vec{\mathrm{u}} = b$$

$$\Downarrow$$

$$\underbrace{A^T A}_{2\mathrm{x}2}\, \underbrace{\vec{\mathrm{u}}}_{2\mathrm{x}1} = \underbrace{A^T b}_{2\mathrm{x}1}$$

$$\Downarrow$$

$$\boxed{\vec{\mathrm{u}} = \left(A^T A\right)^{-1} A^T b}$$

How does Lucas-Kanade behave?

$$\boxed{\vec{\mathrm{u}} = \left(A^T A\right)^{-1} A^T b}$$

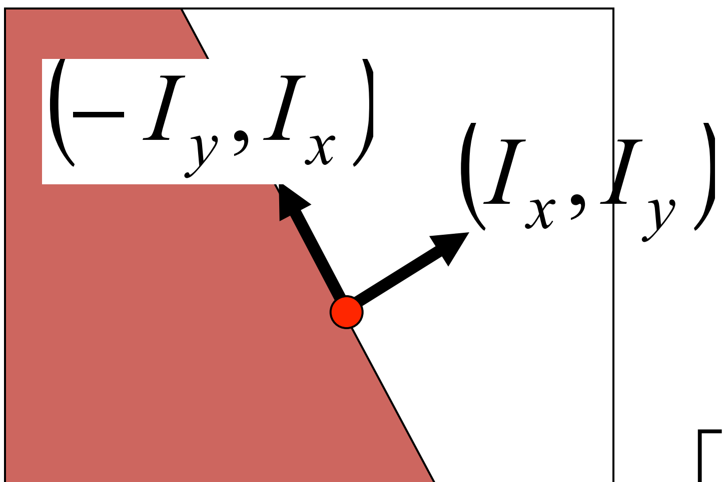
$$A^T A = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix}$$

We want this matrix to be invertible.

i.e., no zero eigenvalues

How does Lucas-Kanade behave?

- Edge ➔ $A^T A$ becomes singular



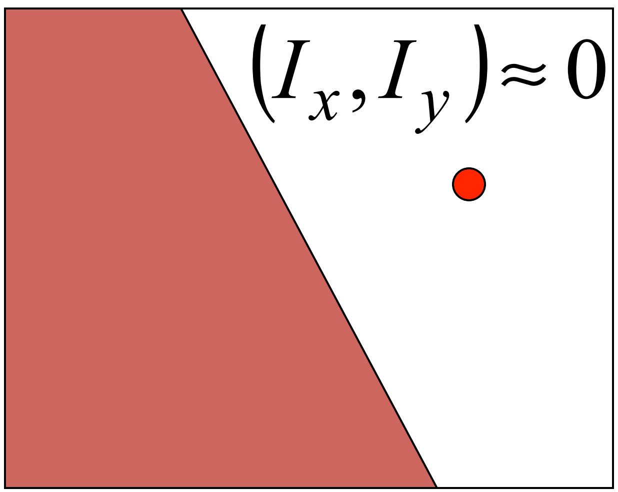
$$\begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} \begin{bmatrix} -I_y \\ I_x \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$\Downarrow$

$\begin{bmatrix} -I_y \\ I_x \end{bmatrix}$ is eigenvector with eigenvalue 0

How does Lucas-Kanade behave?

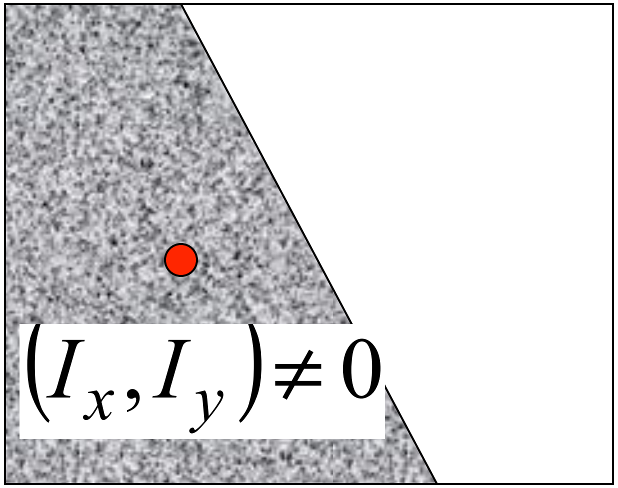
- Homogeneous ➔ $A^T A \approx 0$ ➔ 0 eigenvalues



$$A^T A = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix}$$

How does Lucas-Kanade behave?

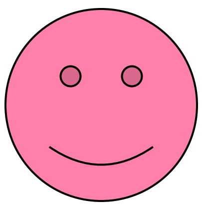
- Textured regions ➔ two high eigenvalues



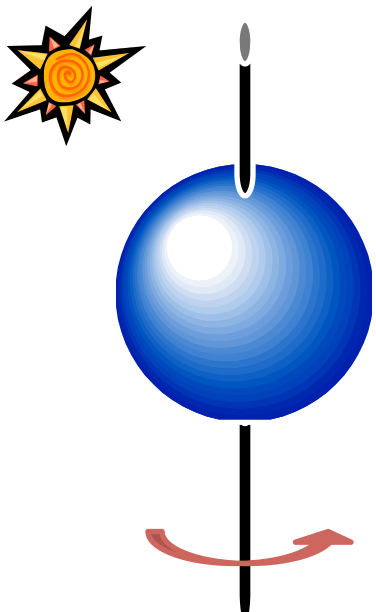
How does Lucas-Kanade behave?

- Edge ➔ $A^T A$ becomes singular 

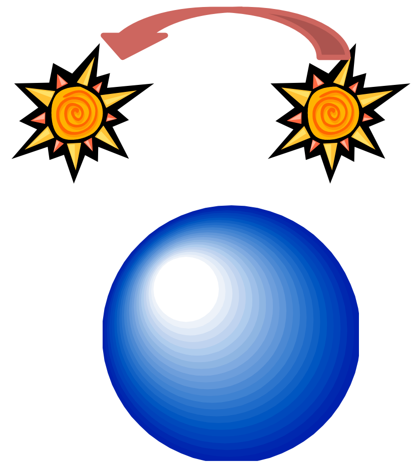
- Homogeneous regions ➔ low gradients
  $A^T A \approx 0$ 

- High texture ➔ 

When does it break?



Homogeneous objects generate zero optical flow.



Fixed sphere. Changing light source.

Non-rigid texture motion

Other break-downs

- Brightness constancy is **not** satisfied

Correlation based methods

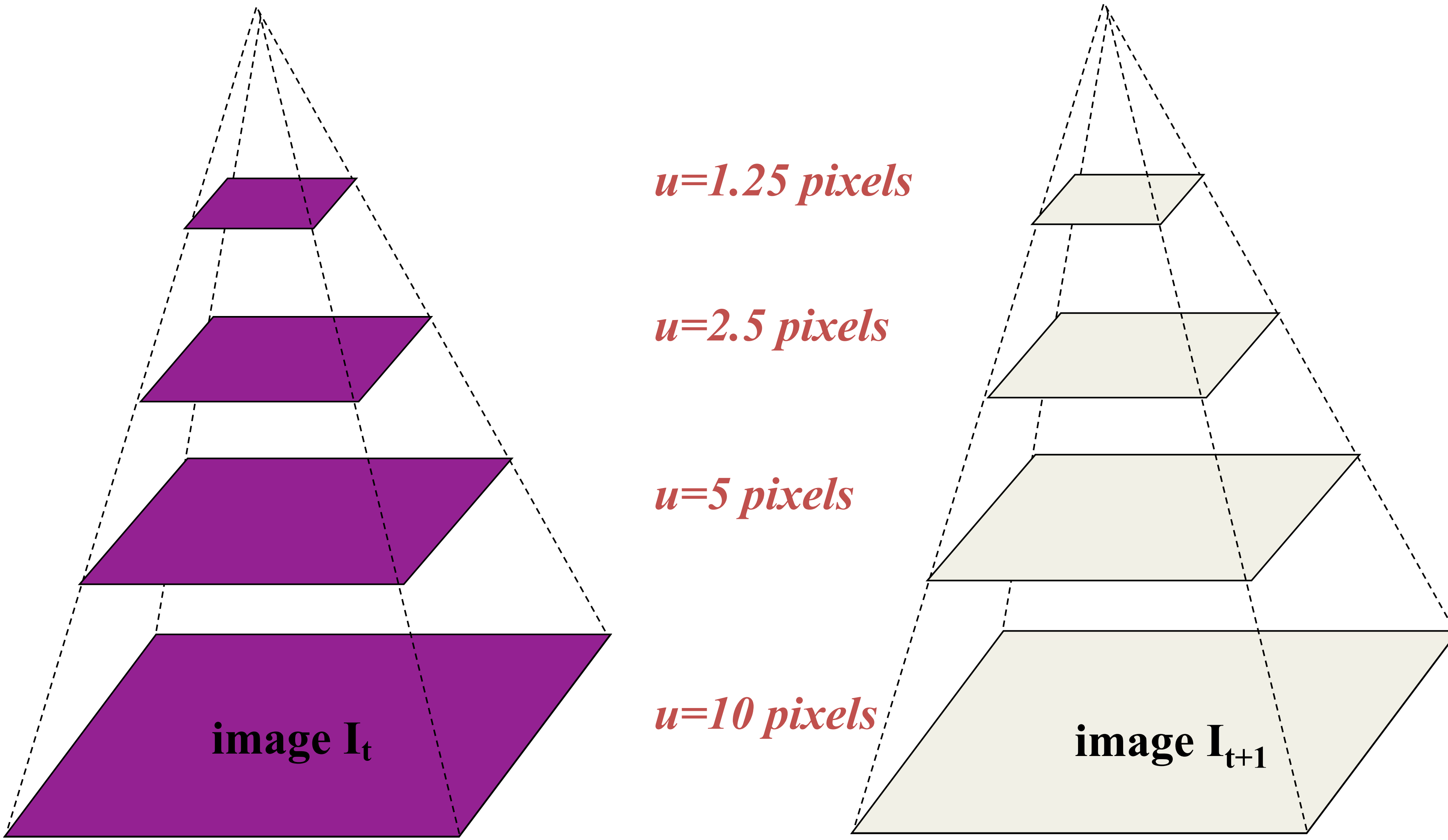
- A point does **not** move like its neighbors
  - what is the ideal window size?

Regularization based methods

- The motion is **not** small (Taylor expansion doesn’t hold)

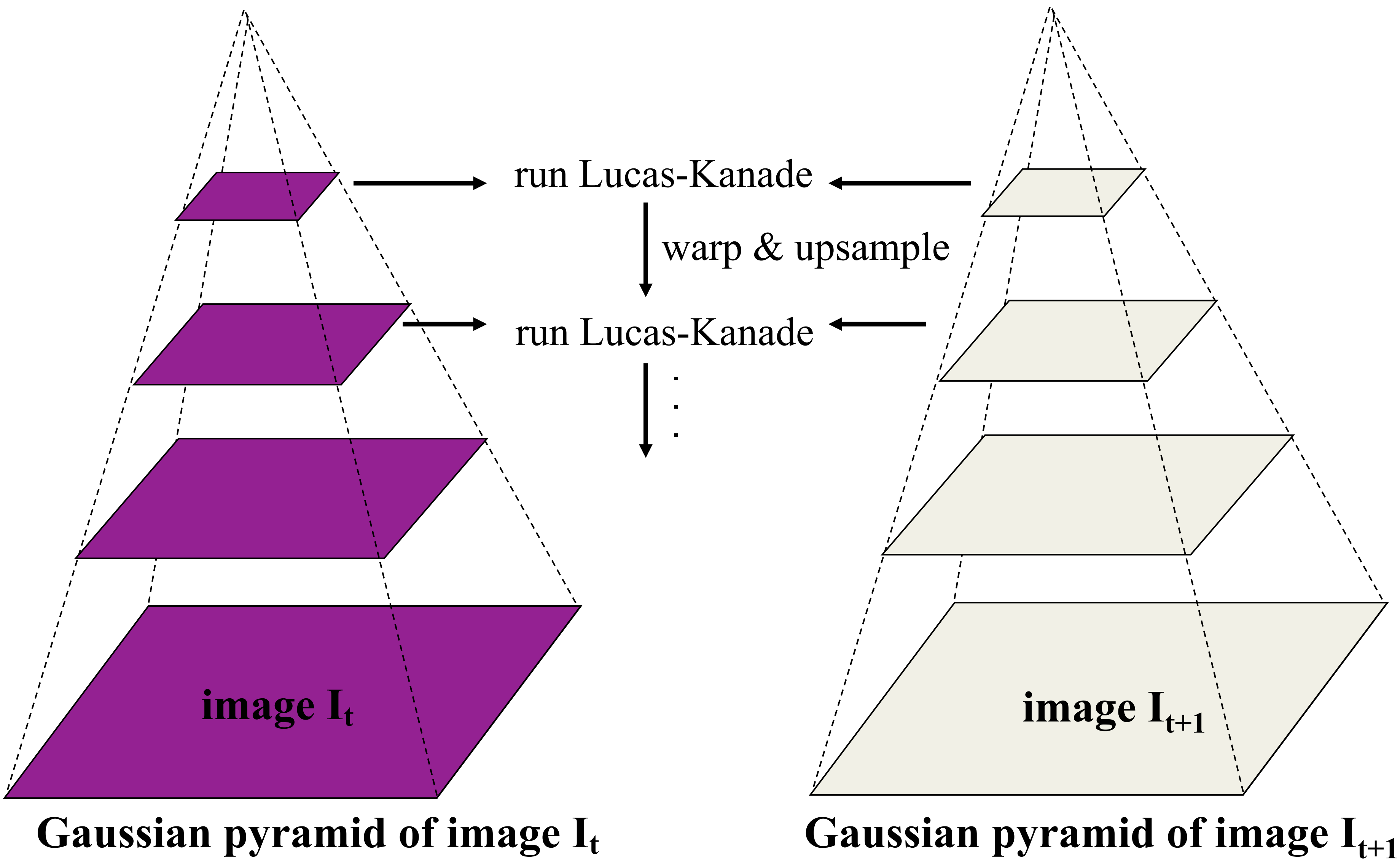
Use multi-scale estimation

Multi-Scale Flow Estimation



Gaussian pyramid of image I_t **Gaussian pyramid of image I_{t+1}**

Multi-Scale Flow Estimation



Examples: Motion Based Segmentation

Input

Segmentation result

Examples: Motion Based Segmentation

Input

Segmentation result

Other break-downs

- Brightness constancy is **not** satisfied

  ➡ Correlation based methods

- A point does **not** move like its neighbors
  - what is the ideal window size?

  ➡ Regularization based methods

- The motion is **not** small (Taylor expansion doesn't hold)

  ➡ Use multi-scale estimation

Regularization
Horn and Schunk (1981)

Add global smoothness term

Smoothness error: $$E_s = \iint_D \left(u_x^2 + u_y^2\right) + \left(v_x^2 + v_y^2\right) dx\, dy$$

Error in brightness constancy equation $$E_c = \iint_D \left(I_x u + I_y v + I_t\right)^2 dx\, dy$$

Minimize: $E_c + \lambda E_s$

Solve by calculus of variations