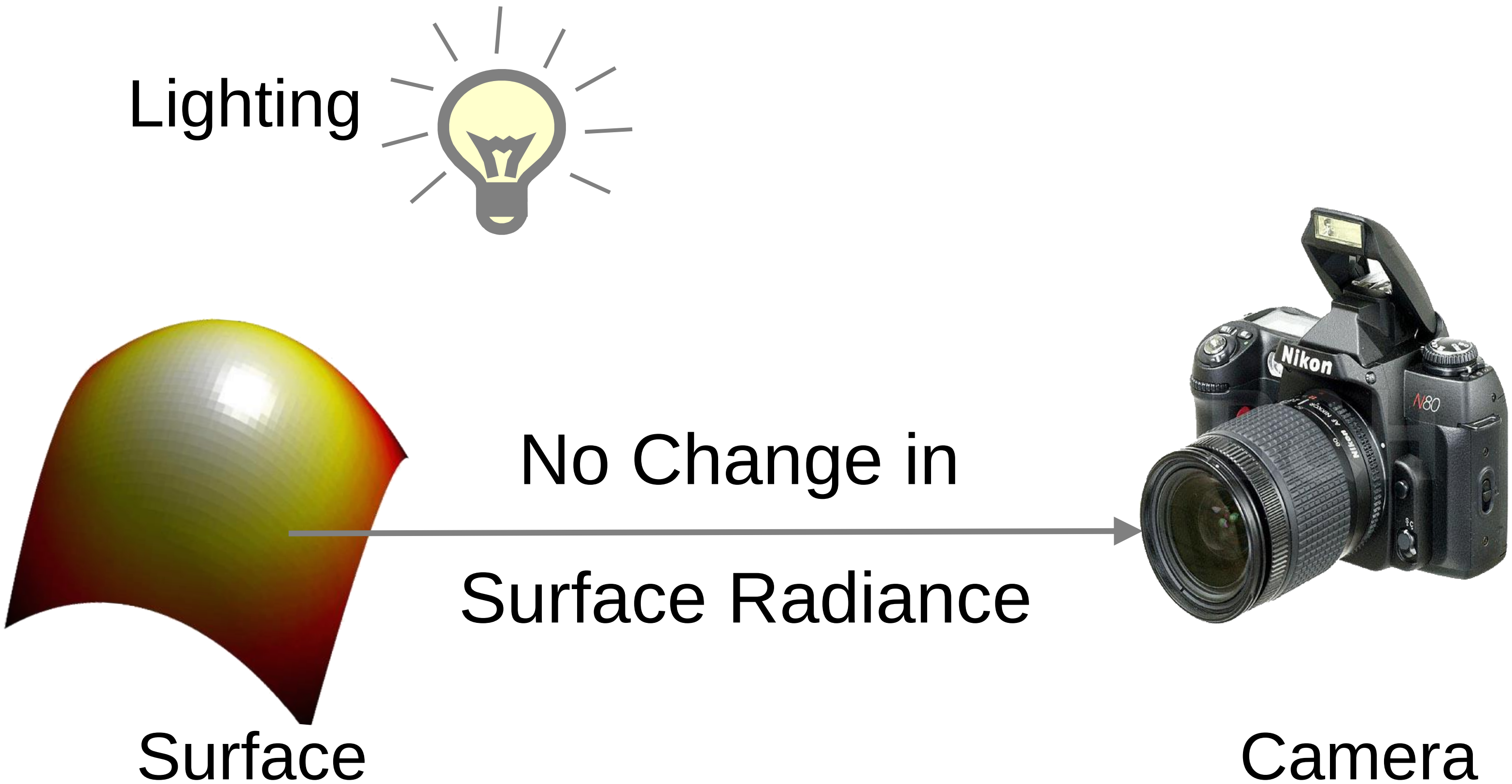
Optical Flow
Reminder lesson

Prof. Hichem Sahli
Abel Díaz Berenguer
VUB-ETRO
2021-2022

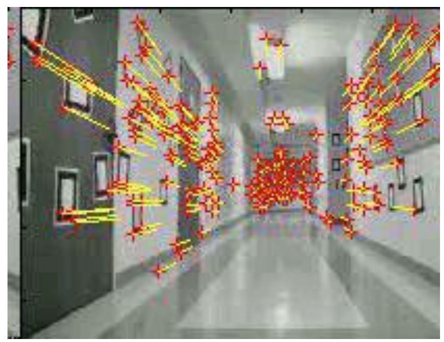
The cause of motion

Three factors in imaging process



Optical Flow

Image tracking



3D computation

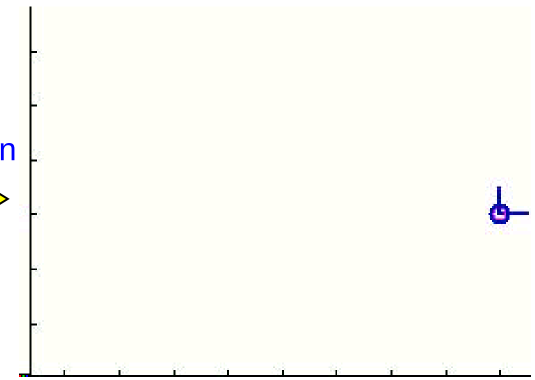
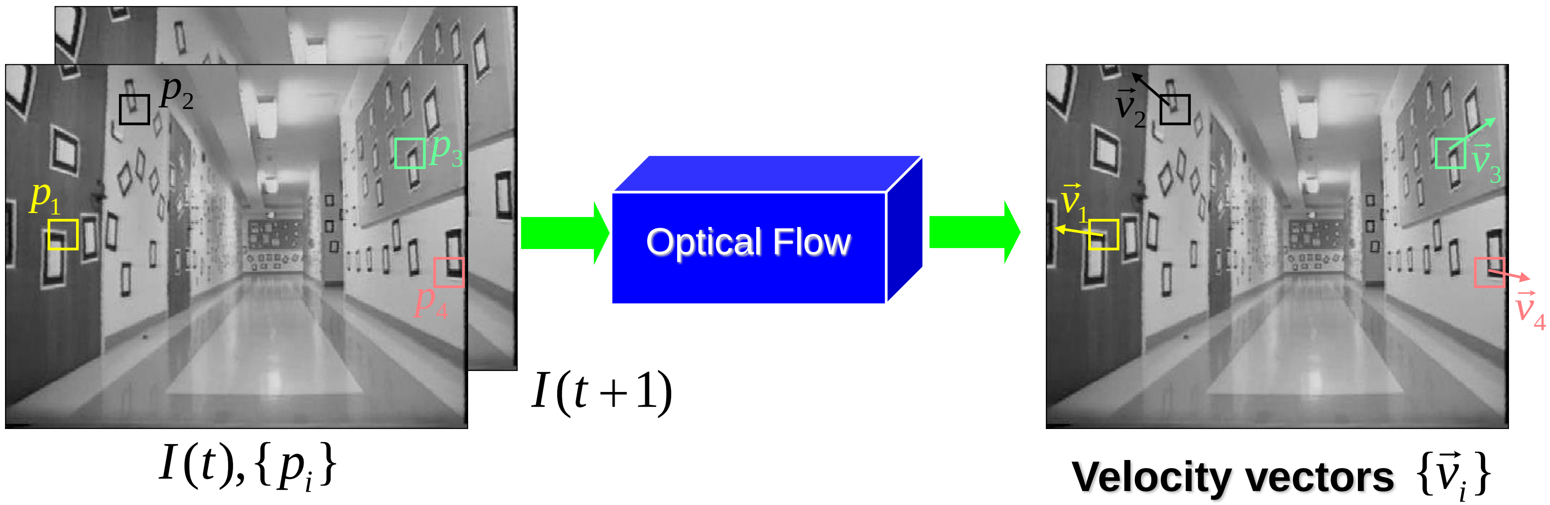


Image sequence
(single camera)

Tracked sequence

3D structure
+
3D trajectory

What is Optical Flow?



Optical flow is the relation of the motion field

- *the 2D projection of the physical movement of points relative to the observer*

to 2D displacement of pixel patches on the image plane.

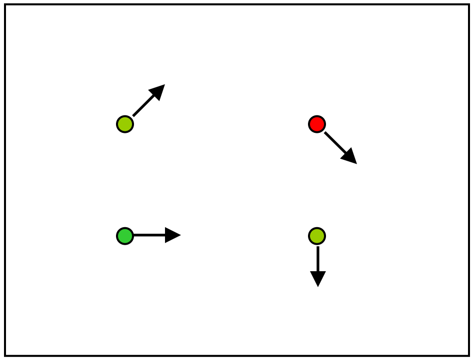
Common assumption:

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

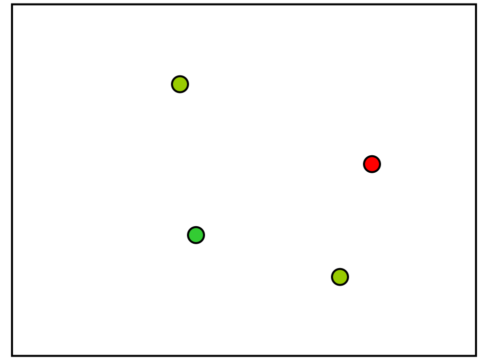
$$I(p_i, t) = I(p_i + \vec{v}_i, t+1)$$

Note: more elaborate tracking models can be adopted if more frames are process all at once

Optical Flow - Problem Definition



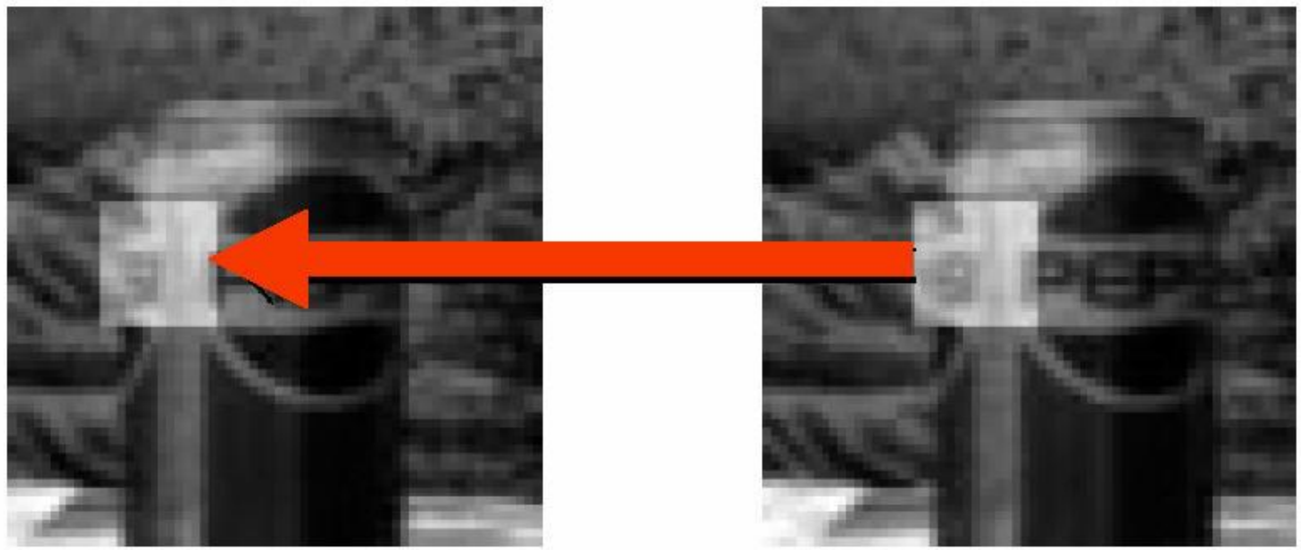
$I(x, y)$



$J(x, y)$

- How to estimate pixel motion from image I to image J?
  - Find pixel correspondences
    - Given a pixel in I, look for nearby pixels of the same color in J
- Key assumptions
  - **color constancy**: a point in I looks "the same" in image J
    - For grayscale images, this is **brightness constancy**
  - **small motion**: points do not move very far

Optical Flow Assumptions: Brightness Constancy



Assumption

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

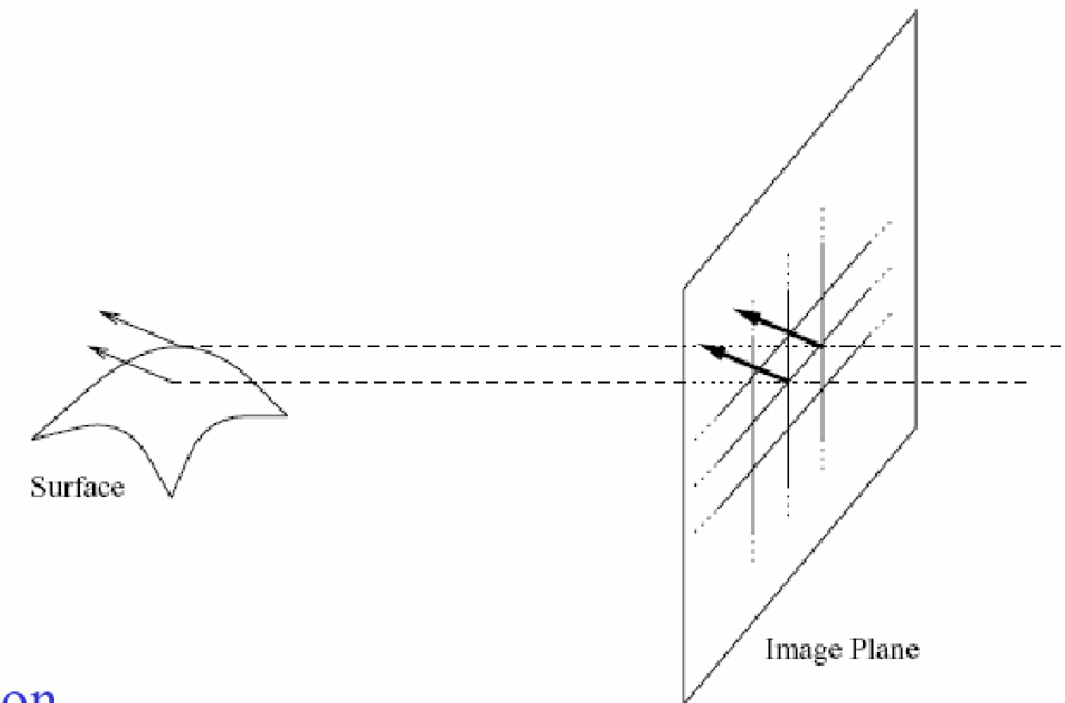
$$I(x+u, y+v, t+1) = I(x, y, t)$$

(assumption)

* Slide from Michael Black, CS143 2003

Optical Flow Assumptions:

Spatial Coherence



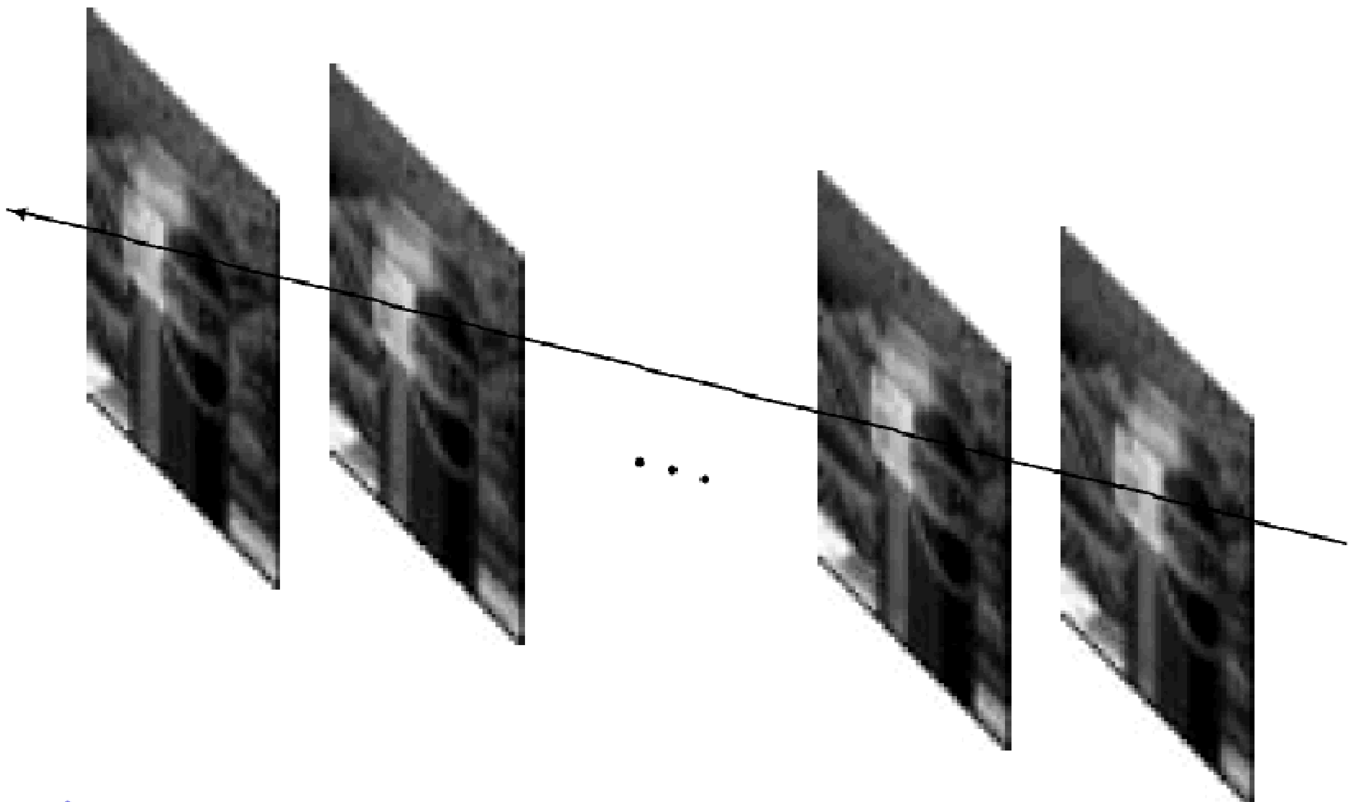
Assumption

* 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.

* Slide from Michael Black, CS143 2003

Optical Flow Assumptions:

Temporal Persistence

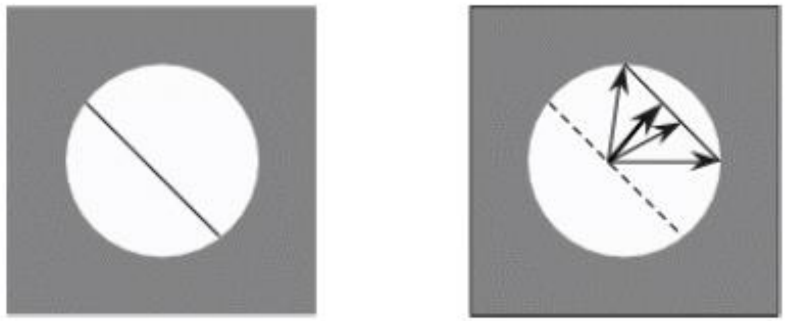


Assumption:

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

* Slide from Michael Black, CS143 2003

Aperture Problem



(a) (b)

(a) Line feature observed through a small aperture at time t.

(b) At time $t + \delta t$ the line has moved to a new position.

- It is not possible to determine exactly where each point has moved.
- From local image measurements only the flow component perpendicular to the line feature can be computed.

Normal flow: Component of flow perpendicular to line feature.

Optical Flow Constraint Equation

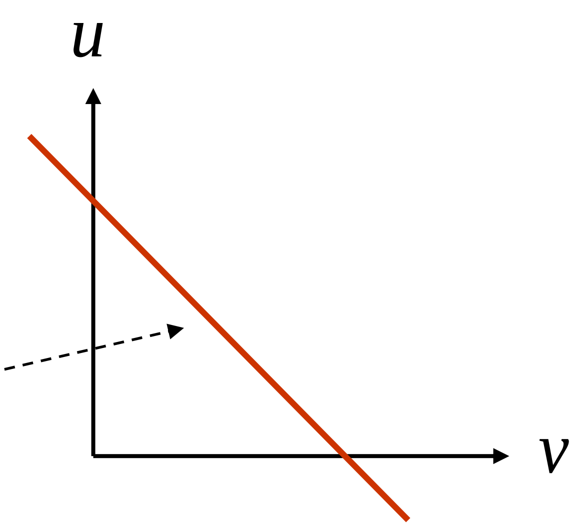
$$\delta x \frac{\partial E}{\partial x} + \delta y \frac{\partial E}{\partial y} + \delta t \frac{\partial E}{\partial t} = 0$$

Divide by δt and take the limit $\delta t \to 0$

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

Constraint Equation

$$E_x\, u + E_y\, v + E_t = 0$$



NOTE: $\boldsymbol{u} = (\boldsymbol{u}, \boldsymbol{v})$ must lie on a straight line

We can compute E_x, E_y, E_t using gradient operators!

But, (u,v) cannot be found uniquely with this constraint!

Computing Optical Flow (HS)

- Formulate Error in Optical Flow Constraint:

$$e_c = \iint_{image} (E_x u + E_y v + E_t)^2 \, dx \, dy$$

- We need additional constraints!

- Smoothness Constraint (as in shape from shading and stereo):

Usually motion field varies smoothly in the image.
So, penalize departure from smoothness:

$$e_s = \iint_{image} (u_x^2 + u_y^2) + (v_x^2 + v_y^2) \;\; dx \, dy$$

- Find (u,v) at each image point that MINIMIZES:

$$e = e_s + \lambda e_c$$

weighting factor

Discrete Optical Flow Algorithm (HS)

Consider image pixel (i, j)

- Departure from Smoothness Constraint:

$$es_{kl} = \frac{1}{4}[(u_{k+1,l} - u_{k,l})^2 + (u_{k,l+1} - u_{k,l})^2 + (v_{k+1,l} - v_{k,l})^2 + (v_{k,l+1} - v_{k,l})^2]$$

- Error in Optical Flow constraint equation:

$$ec_{kl} = (E_x^{kl} u_{kl} + E_y^{kl} v_{kl} + E_t^{kl})^2$$

- We seek the set $\{u_{kl}\}$ & $\{v_{kl}\}$ that minimize:

$$e = \sum_k \sum_l (es_{kl} + \lambda\, ec_{kl})$$

NOTE: $\{u_{kl}\}$ & $\{v_{kl}\}$ show up in more than one term

Discrete Optical Flow Algorithm (HS)

- Differentiating e w.r.t v_{kl} & u_{kl} and setting to zero:

$$\frac{\partial e}{\partial u_{kl}} = 2\,(u_{kl} - \overline{u_{kl}}) + 2\lambda\,(E_x^{kl} u_{kl} + E_y^{kl} v_{kl} + E_t^{kl})E_x^{kl} = 0$$

$$\frac{\partial e}{\partial v_{kl}} = 2\,(v_{kl} - \overline{v_{kl}}) + 2\lambda\,(E_x^{kl} u_{kl} + E_y^{kl} v_{kl} + E_t^{kl})E_y^{kl} = 0$$

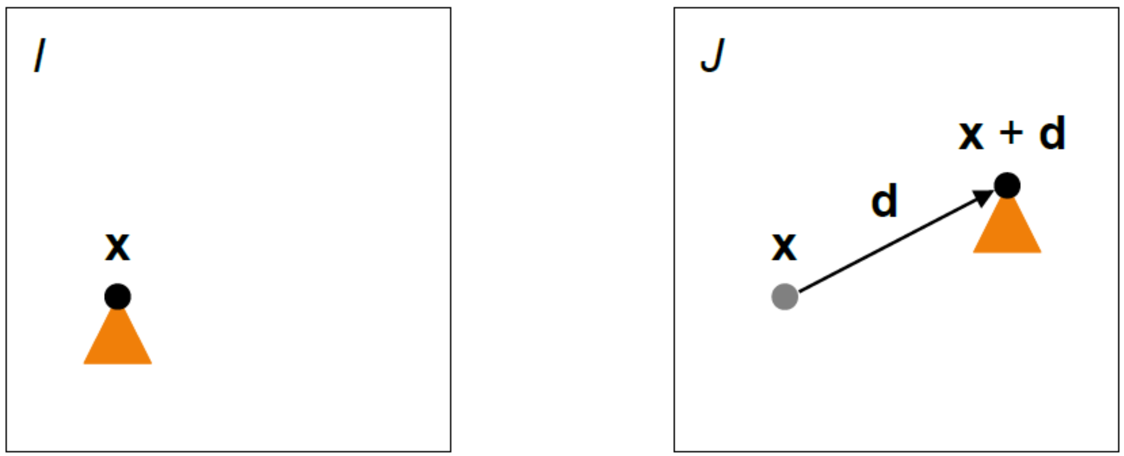
- $\overline{v_{kl}}$ & $\overline{u_{kl}}$ are averages of (u, v) around pixel (k, l)

Update Rule:

$$u_{kl}^{n+1} = \overline{u_{kl}^{n}} - \frac{E_x^{kl}\,\overline{u_{kl}^{n}} + E_y^{kl}\,\overline{v_{kl}^{n}} + E_t^{kl}}{1 + \lambda\,[(E_x^{kl})^2 + (E_y^{kl})^2]} E_x^{kl}$$

$$v_{kl}^{n+1} = \overline{v_{kl}^{n}} - \frac{E_x^{kl}\,\overline{u_{kl}^{n}} + E_y^{kl}\,\overline{v_{kl}^{n}} + E_t^{kl}}{1 + \lambda\,[(E_x^{kl})^2 + (E_y^{kl})^2]} E_y^{kl}$$

Optical Flow (Lucas & Kanade)



- Object moves from $\boldsymbol{x} = (x, y)t$ to $\boldsymbol{x} + \boldsymbol{d}$.
- $\boldsymbol{d} = (u, v)\ t$

More Detail: Solving the aperture problem

- 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!

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

$$\underset{25\times 2}{\underbrace{\begin{bmatrix} I_x(\mathbf{p}_1) & I_y(\mathbf{p}_1) \\ I_x(\mathbf{p}_2) & I_y(\mathbf{p}_2) \\ \vdots & \vdots \\ I_x(\mathbf{p}_{25}) & I_y(\mathbf{p}_{25}) \end{bmatrix}}_{A}} \underset{2\times 1}{\underbrace{\begin{bmatrix} u \\ v \end{bmatrix}}_{d}} = - \underset{25\times 1}{\underbrace{\begin{bmatrix} I_t(\mathbf{p}_1) \\ I_t(\mathbf{p}_2) \\ \vdots \\ I_t(\mathbf{p}_{25}) \end{bmatrix}}_{b}}$$

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

Lukas-Kanade flow

- Prob: we have more equations than unknowns

$$\underset{25\text{x}2}{A} \quad \underset{25\text{x}1}{d} = \underset{25\text{x}1}{b} \longrightarrow \text{minimize } \|Ad - b\|^2$$

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

$$\underset{2\text{x}2}{(A^T A)} \; \underset{2\text{x}1}{d} = \underset{2\text{x}1}{A^T b}$$

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

- The summations are over all pixels in the K x K window
- This technique was first proposed by Lukas & Kanade (1981)
  - described in Trucco & Verri reading

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

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^TA** should be invertible
- **A^TA** should not be too small due to noise
  – eigenvalues λ_1 and λ_2 of **A^TA** should not be too small
- **A^TA** should be well-conditioned
  – λ_1 / λ_2 should not be too large (λ_1 = larger eigenvalue)

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

Iterative Refinement

- Iterative Lukas-Kanade Algorithm

1. Estimate velocity at each pixel by solving Lucas-Kanade equations
2. Warp I(t-1) towards I(t) using the estimated flow field

   - use image warping techniques

3. Repeat until convergence

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

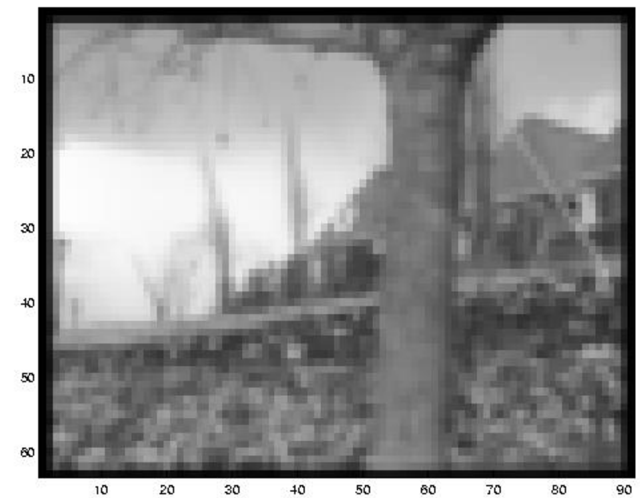
Computing Optical Flow

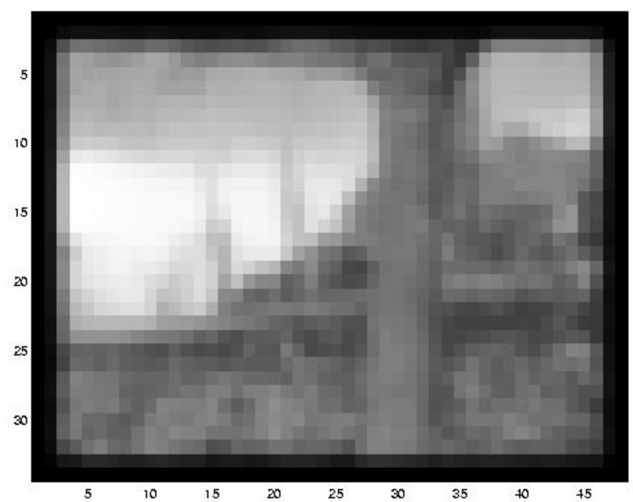
Additional constraints:

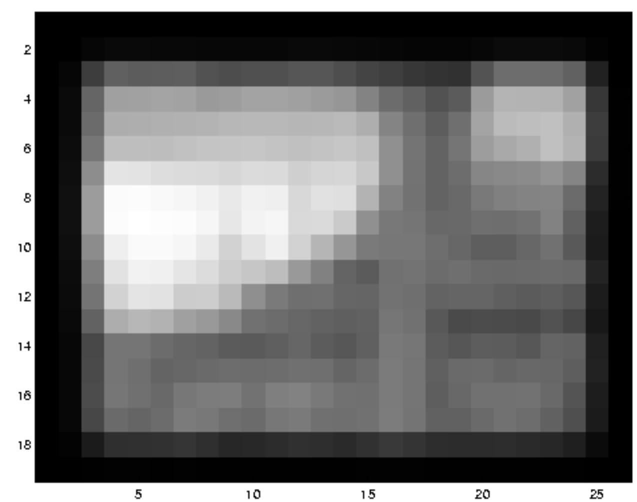
- **Increase aperture:** [e.g., Lucas & Kanade]

  Local singularities at degenerate image regions.

Reduce the resolution!

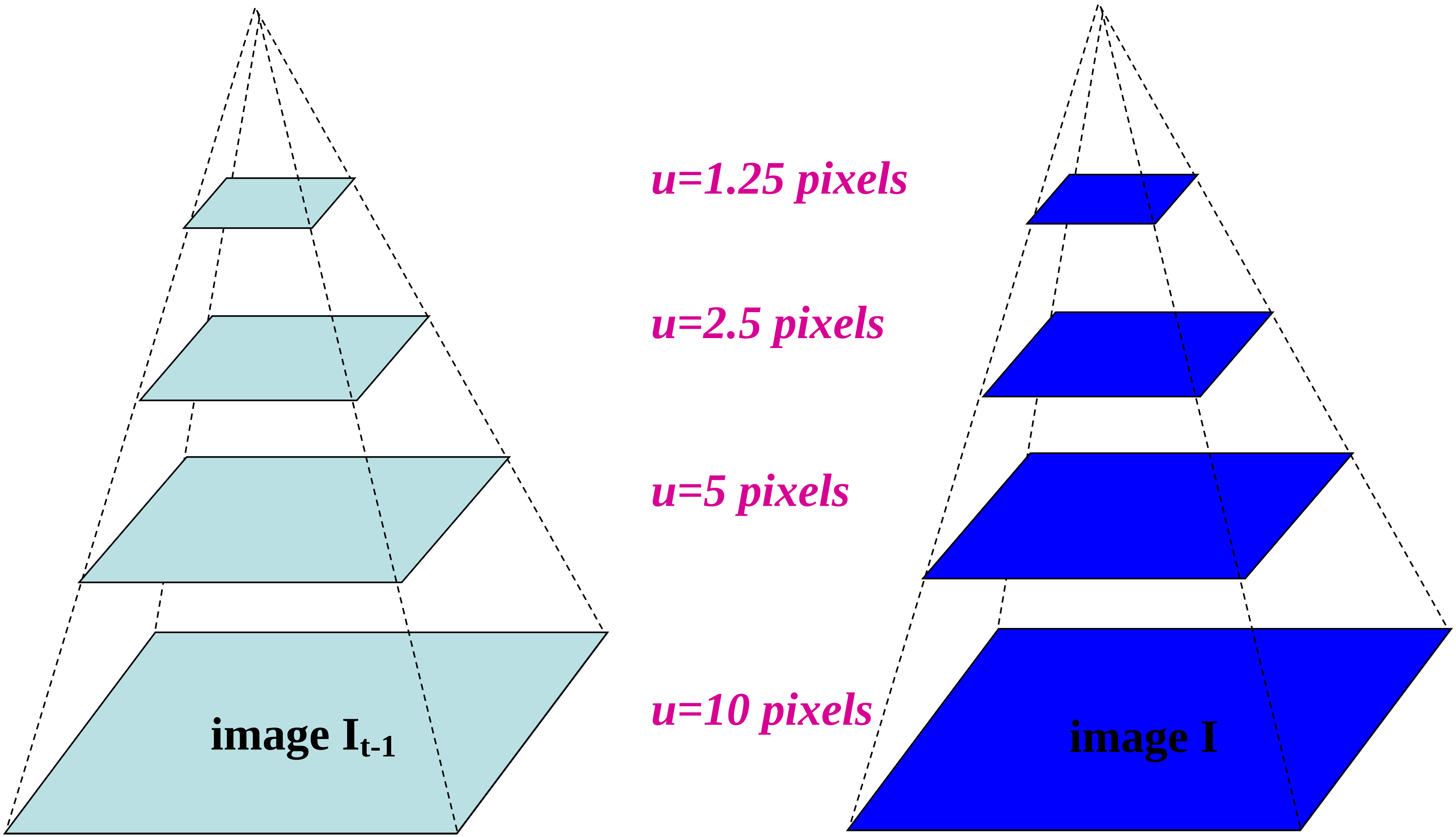






* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

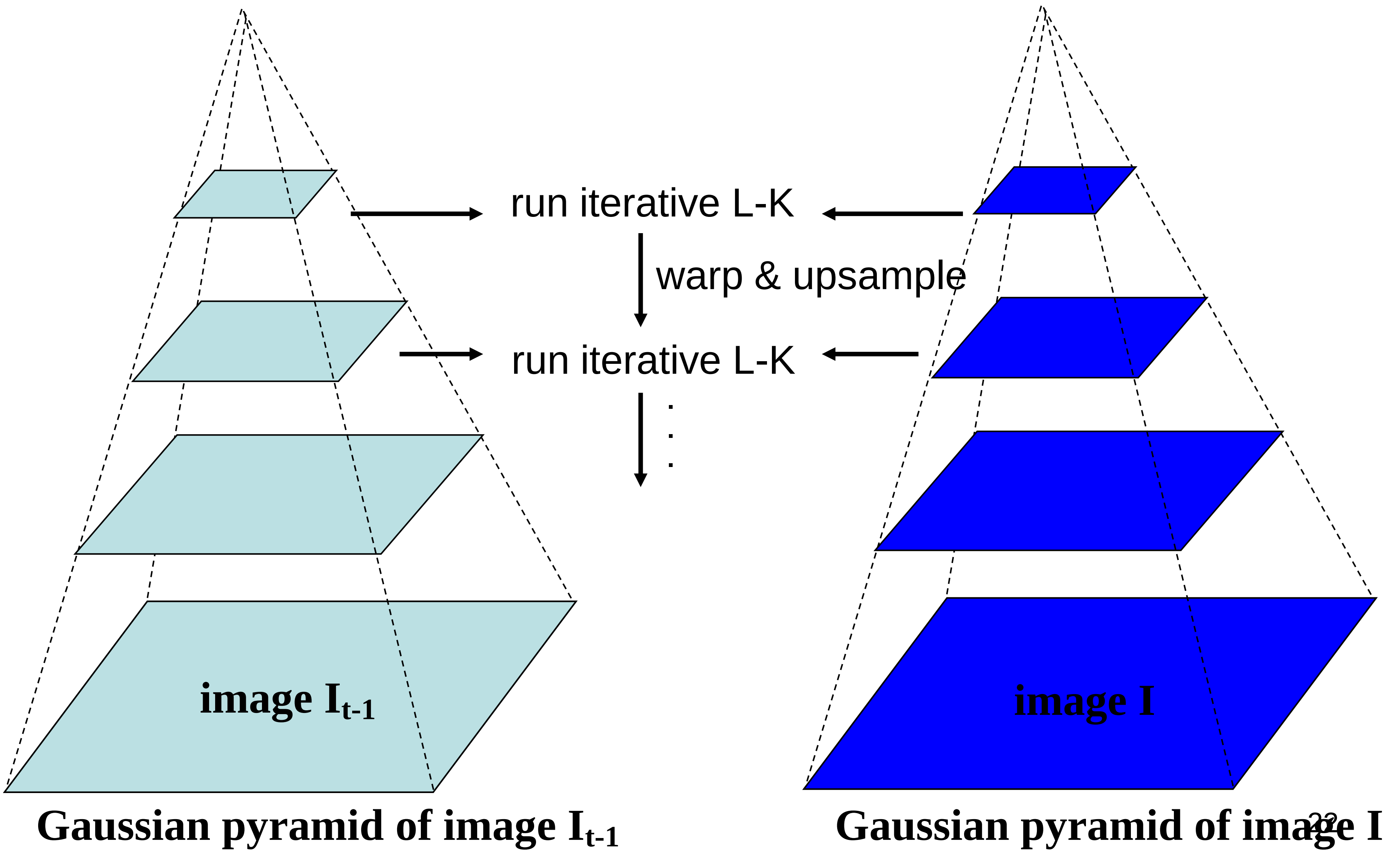
Coarse-to-fine optical flow estimation



Gaussian pyramid of image I_{t-1}

Gaussian pyramid of image I

Coarse-to-fine optical flow estimation



Multi-resolution Lucas Kanade Algorithm

- Compute 'simple' LK at highest level
- At level i
  - Take flow u_{i-1}, v_{i-1} from level i-1
  - bilinear interpolate it to create u_i^*, v_i^* matrices of twice resolution for level i
  - multiply u_i^*, v_i^* by 2
  - compute f_t from a block displaced by $u_i^*(x,y)$, $v_i^*(x,y)$
  - Apply LK to get $u_i'(x, y)$, $v_i'(x, y)$ (the correction in flow)
  - Add corrections u_i' v_i', *i.e.* $u_i = u_i^* + u_i'$, $v_i = v_i^* + v_i'$.