COMP0026: Image Processing

Optical Flow

Many slides adapted from James Hays, Derek Hoeim, Lana Lazebnik, Silvio Saverse, who in turn adapted slides from Steve Seitz, Rick Szeliski, Martial Hebert, Mark Pollefeys, and others

UCL

Lectures will be Recorded

Recovering Motion

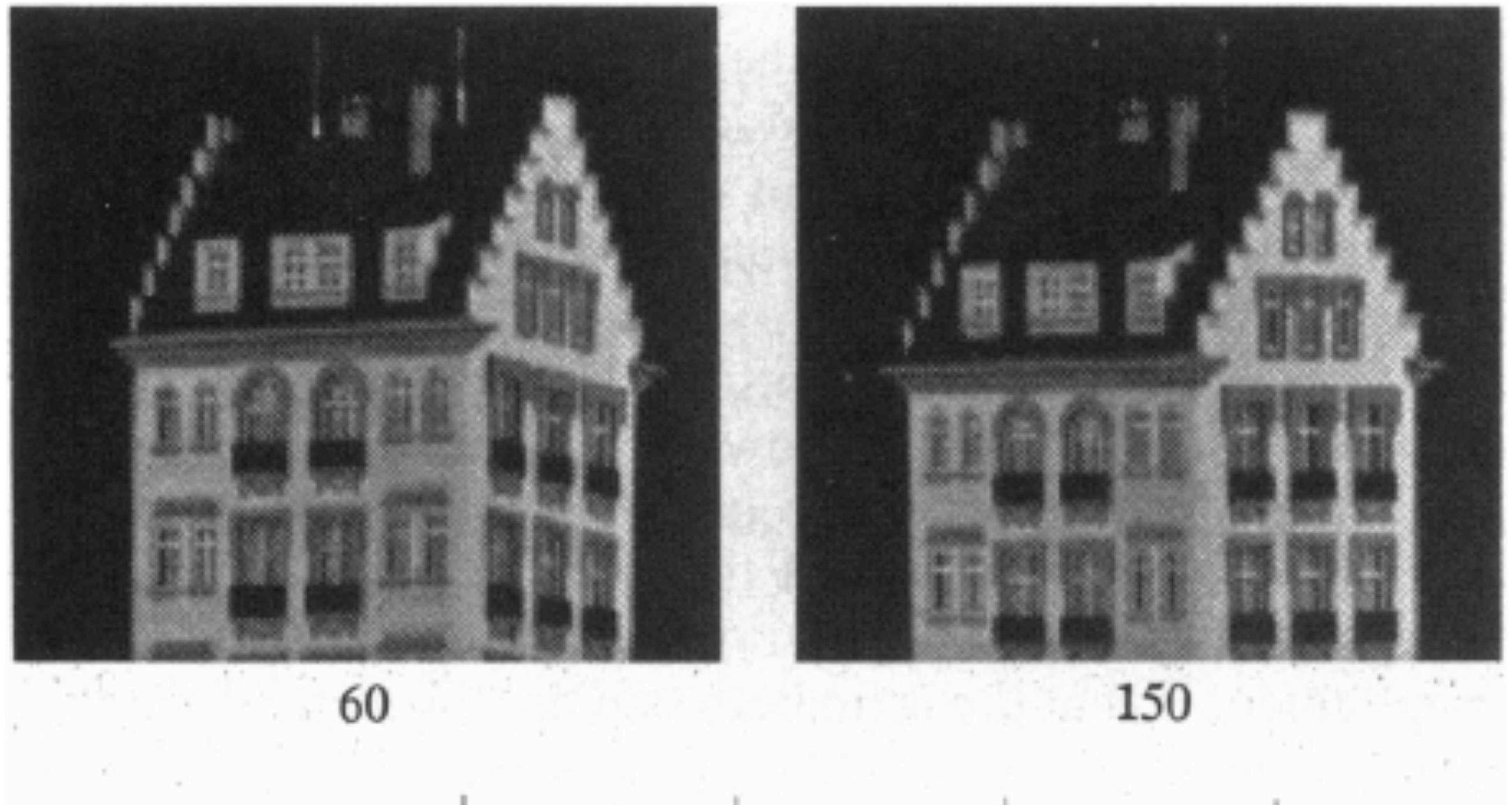
- **Feature-tracking**
  - Extract visual features (corners, textured areas) and "track" them over multiple frames
- **Optical flow**
  - Recover image motion at each pixel from spatio-temporal image brightness variations (optical flow)

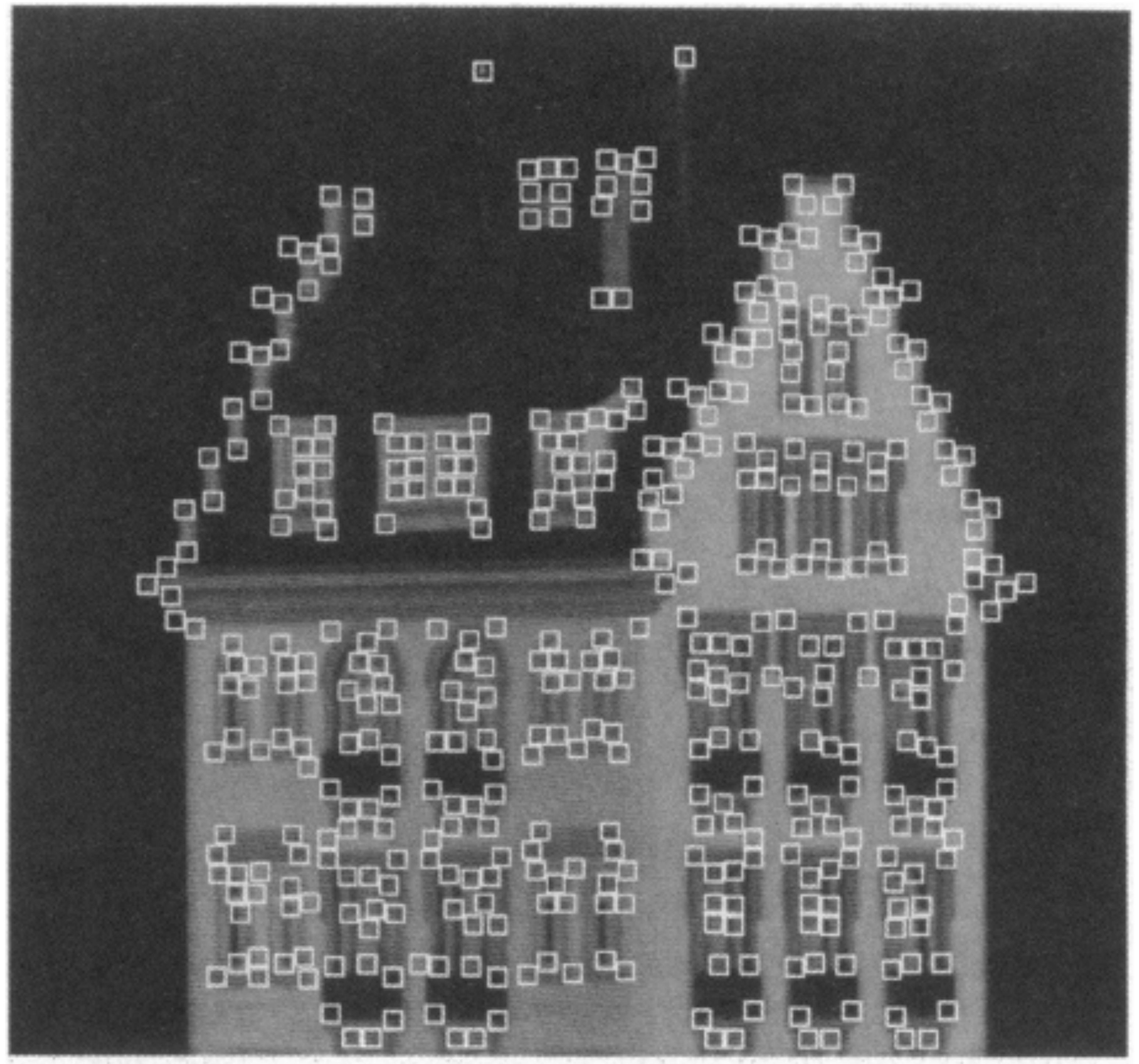
Two problems, one registration method

B. Lucas and T. Kanade. [An iterative image registration technique with an application to stereo vision.]() In *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 674–679, 1981.

Feature Tracking

- Many problems, such as structure from motion require
  matching points
- If *motion is small*, tracking is an easy way to get them





Challenges: Feature Tracking

Challenges: Feature Tracking

– Figure out which features can be tracked

Challenges: Feature Tracking

– Figure out which features can be tracked
– Efficiently track across frames

Challenges: Feature Tracking

- Figure out which features can be tracked
- Efficiently track across frames
- Some points may change appearance over time (e.g., due to rotation, moving into shadows, etc.)

Challenges: Feature Tracking

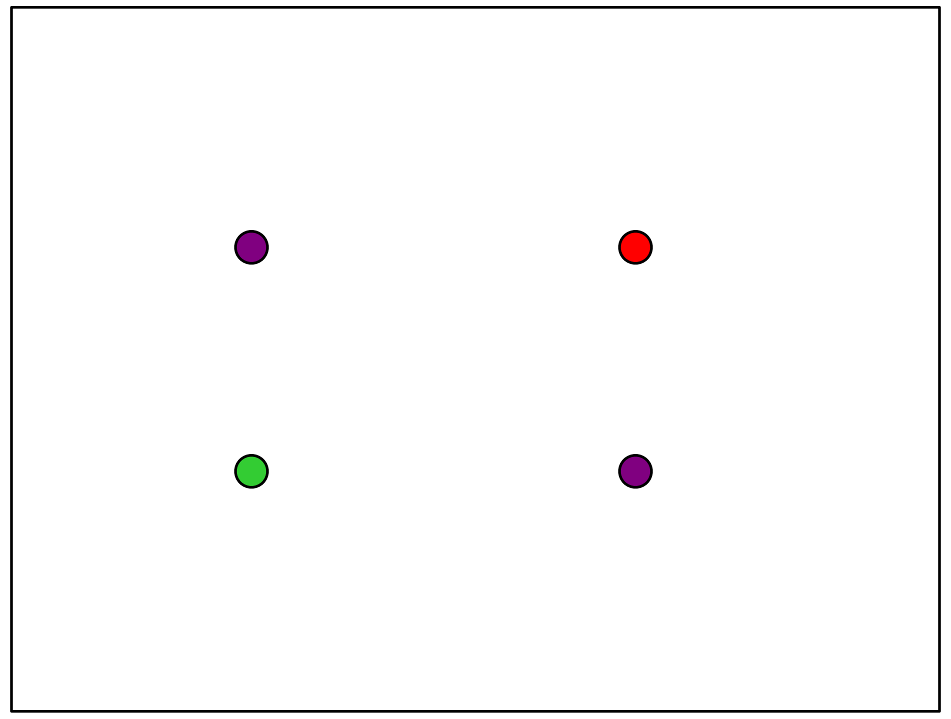
- Figure out which features can be tracked
- Efficiently track across frames
- Some points may change appearance over time
  (e.g., due to rotation, moving into shadows, etc.)
- Drift:
  small errors can accumulate as appearance model is updated

Challenges: Feature Tracking

- Figure out which features can be tracked
- Efficiently track across frames
- Some points may change appearance over time
  (e.g., due to rotation, moving into shadows, etc.)
- Drift:
  small errors can accumulate as appearance model is updated
- Points may appear or disappear:
  need to be able to add/delete tracked points

Feature tracking

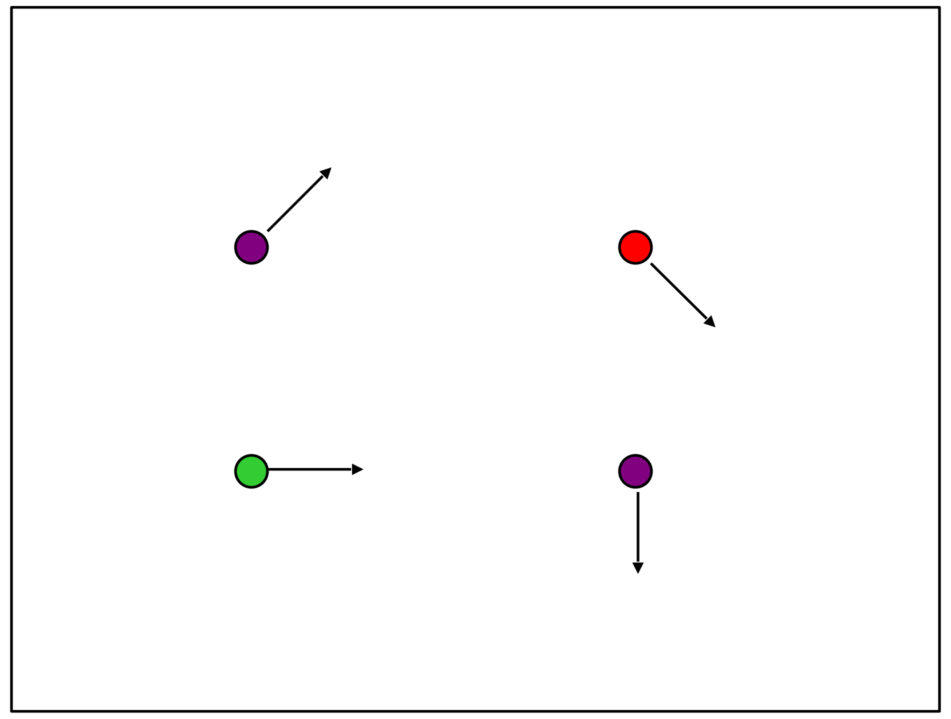
Given two subsequent frames, estimate the point translation



$I(x,y,t)$

Feature tracking

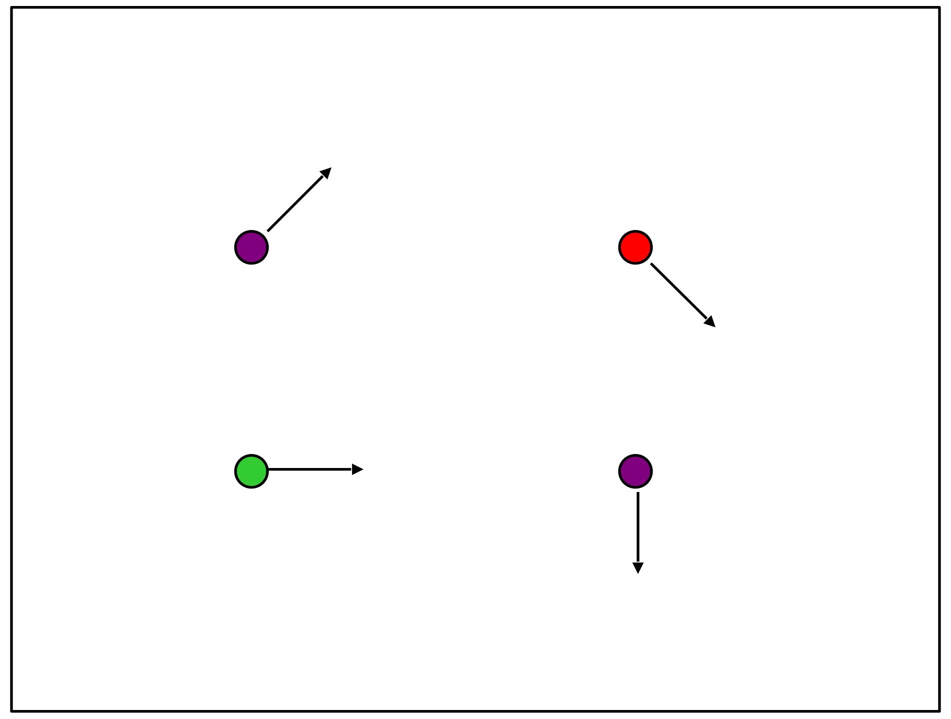
Given two subsequent frames, estimate the point translation



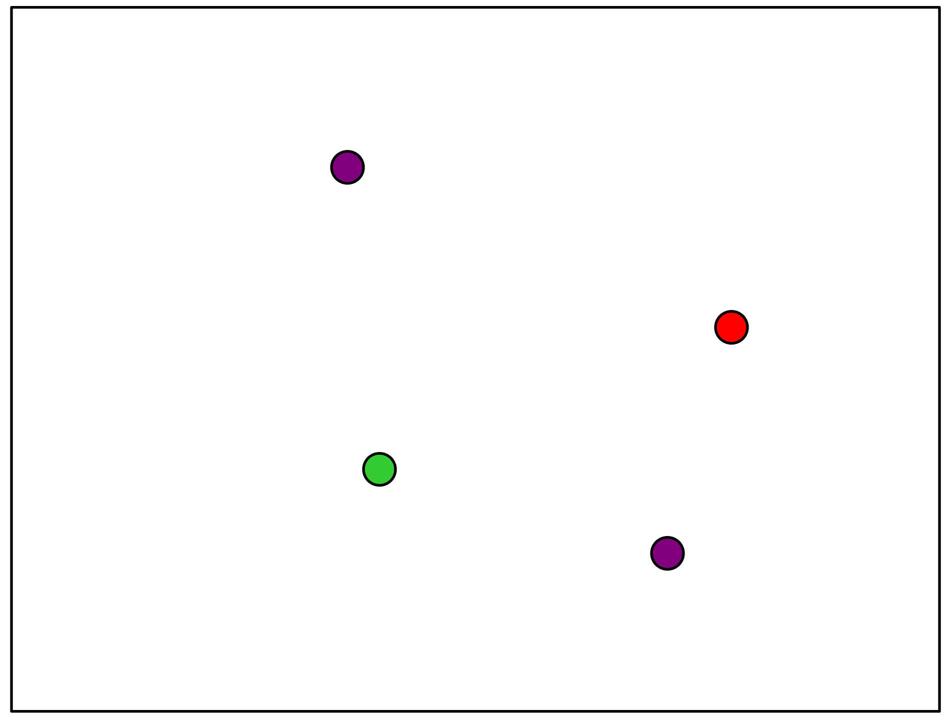
$I(x,y,t)$

Feature tracking

Given two subsequent frames, estimate the point translation



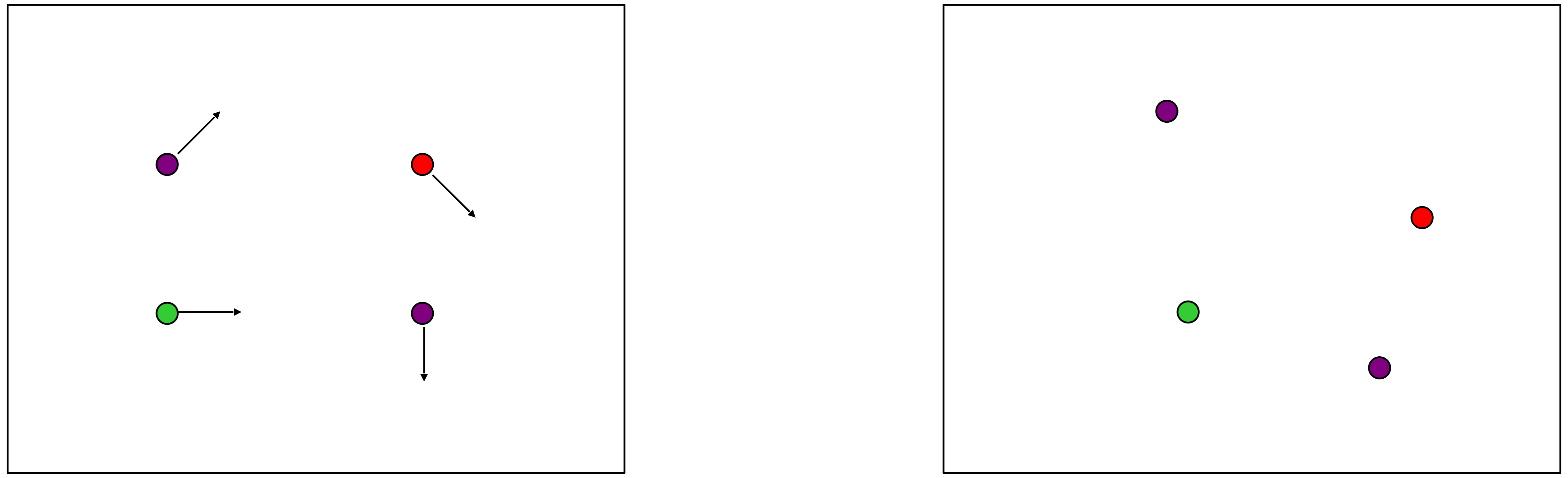
$I(x,y,t)$



$I(x,y,t+1)$

Feature tracking

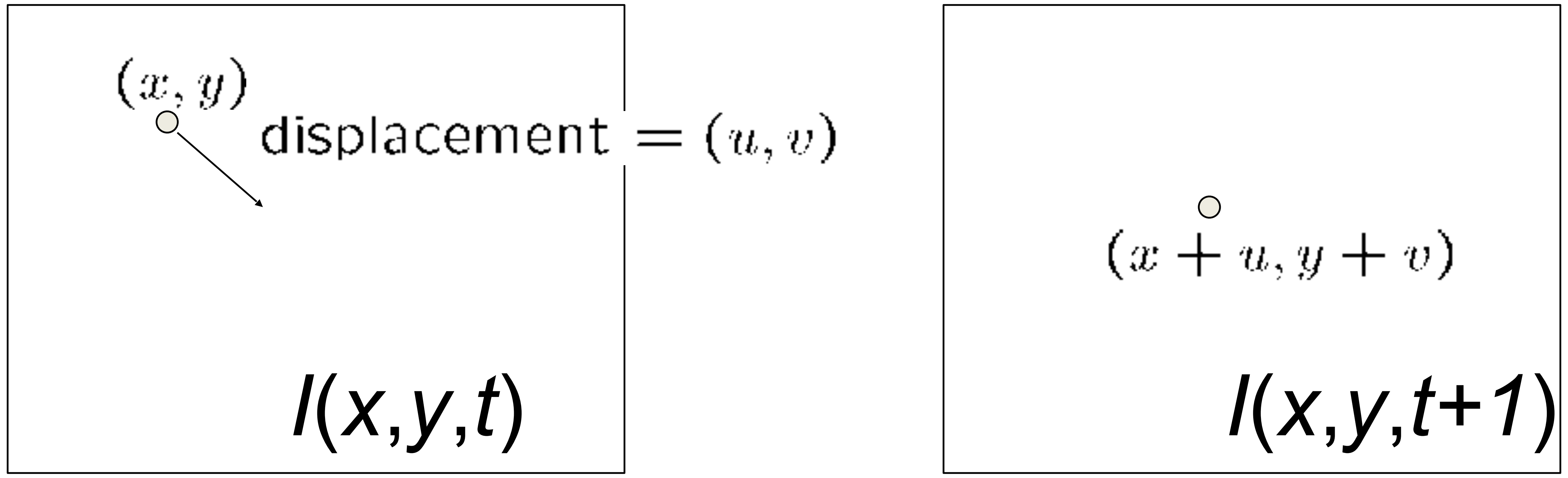
Given two subsequent frames, estimate the point translation



$I(x,y,t)$ $I(x,y,t+1)$

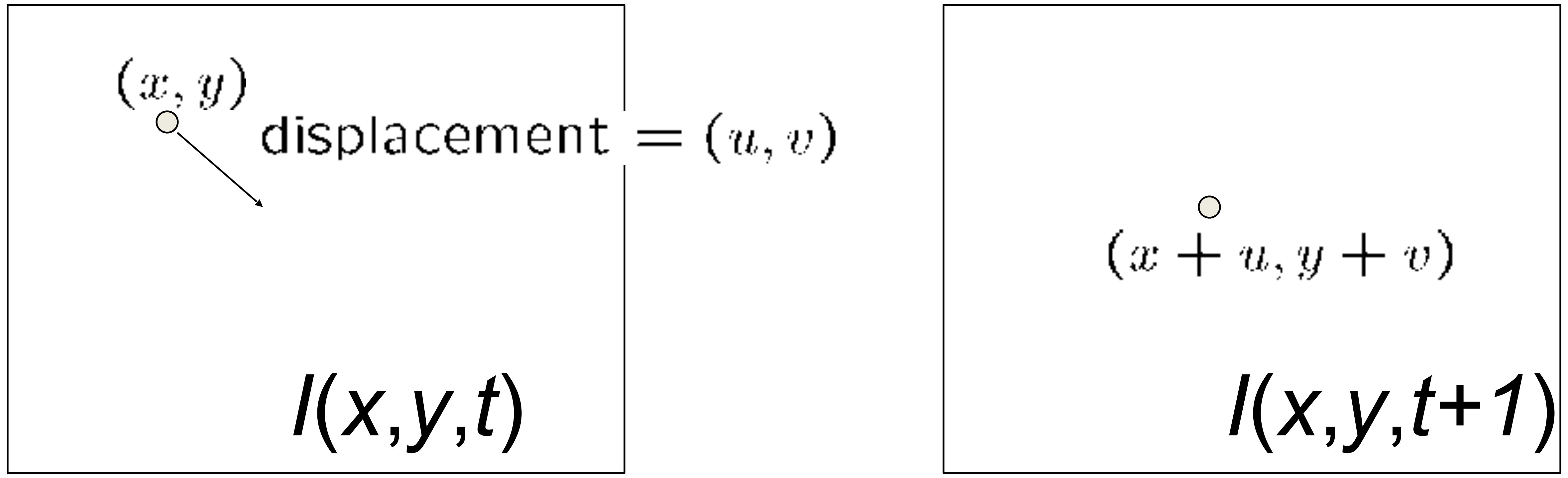
- Key assumptions of Lucas-Kanade Tracker
  - **Brightness constancy**: projection of the same point looks the same in every frame
  - **Small motion**: points do not move very far
  - **Spatial coherence**: points move like their neighbors

Brightness Constancy Equation



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

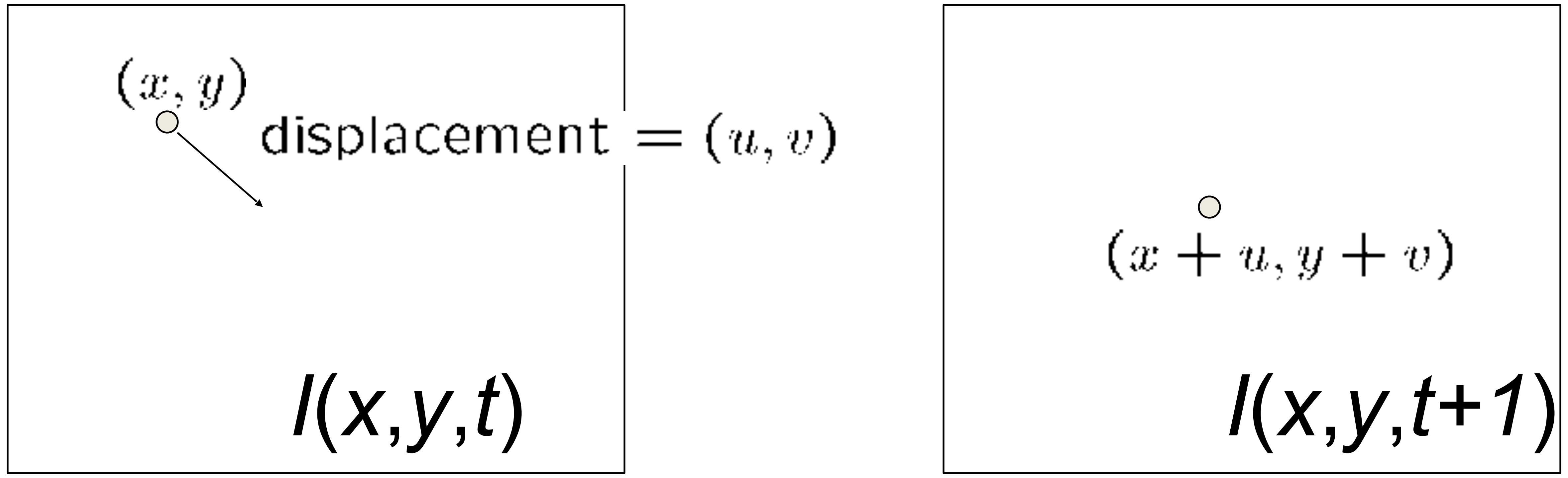
Brightness Constancy Equation



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

Take Taylor expansion of *I(x+u, y+v, t+1)* at *(x,y,t)* to linearize the right side:

Brightness Constancy Equation

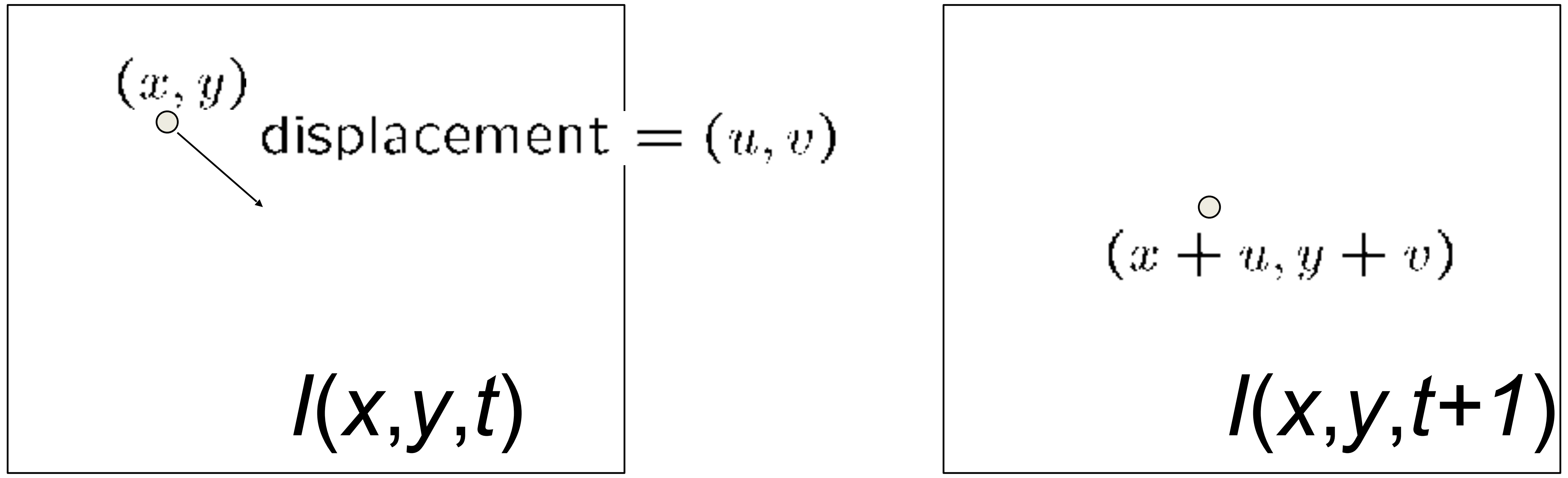


$$I(x, y, t) = I(x+u, y+v, t+1)$$

Take Taylor expansion of *I(x+u, y+v, t+1)* at *(x,y,t)* to linearize the right side:

$$I(x+u, y+v, t+1) \approx I(x, y, t) + I_x \cdot u + I_y \cdot v + I_t$$

Brightness Constancy Equation



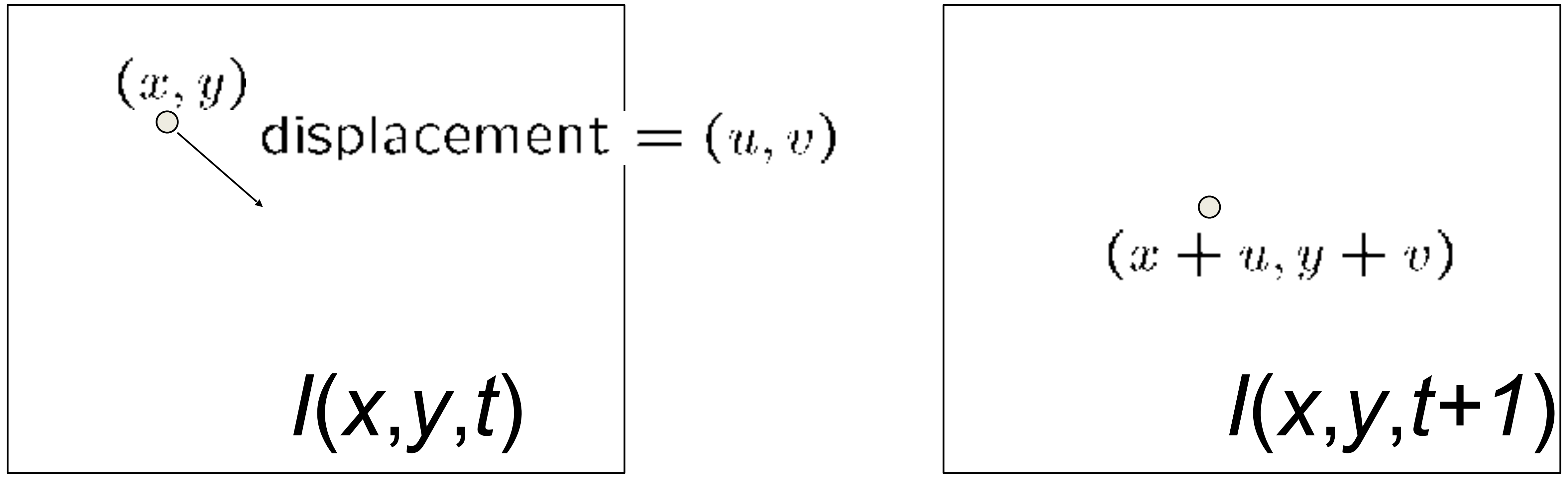
$$I(x,y,t) = I(x+u, y+v, t+1)$$

Take Taylor expansion of *I(x+u, y+v, t+1)* at *(x,y,t)* to linearize the right side:

Image derivative along x

$$I(x+u, y+v, t+1) \approx I(x,y,t) + \boxed{I_x} \cdot u + I_y \cdot v + I_t$$

Brightness Constancy Equation

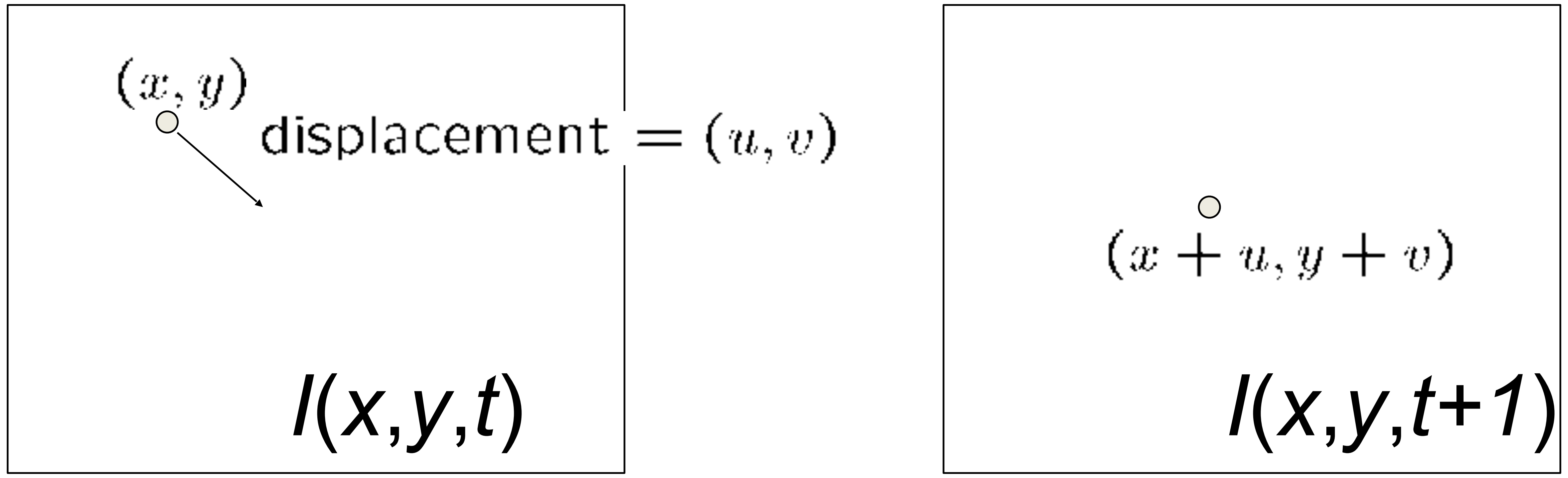


$$I(x, y, t) = I(x + u, y + v, t + 1)$$

Take Taylor expansion of *I(x+u, y+v, t+1)* at *(x,y,t)* to linearize the right side:

$$I(x + u, y + v, t + 1) \approx I(x, y, t) + \overset{\text{Image derivative along x}}{\boxed{I_x}} \cdot u + I_y \cdot v + \overset{\text{Difference over frames}}{\boxed{I_t}}$$

Brightness Constancy Equation



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

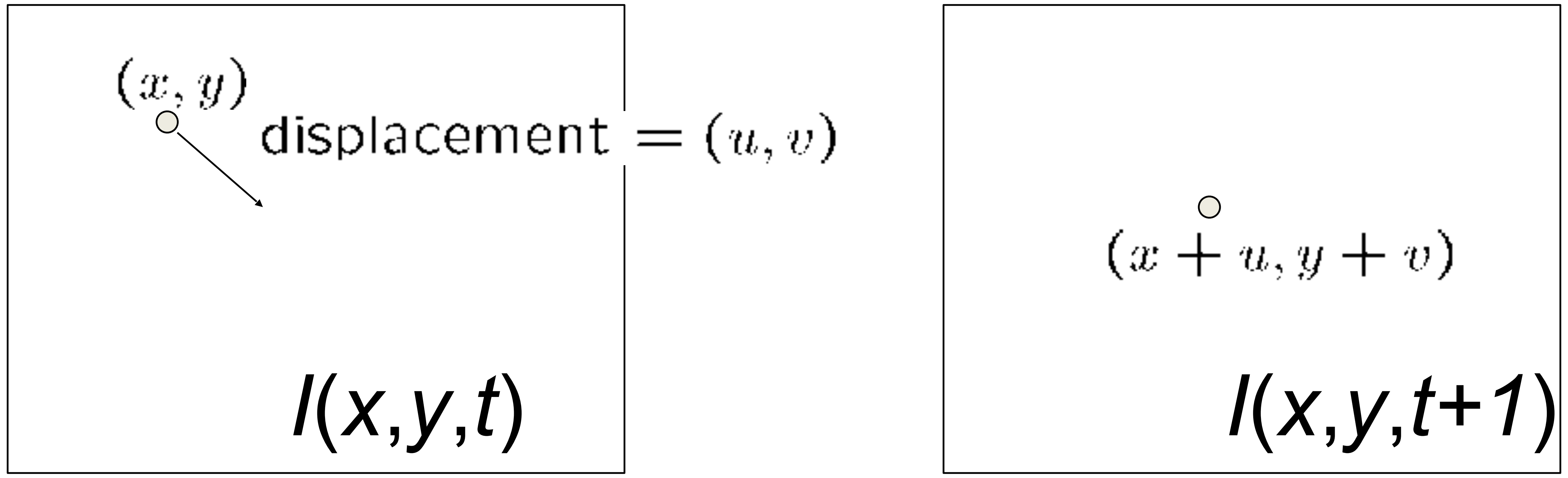
Take Taylor expansion of $I(x+u,\ y+v,\ t+1)$ at (x,y,t) to linearize the right side:

Image derivative along x   Difference over frames

$$I(x + u, y + v, t + 1) \approx I(x, y, t) + \boxed{I_x} \cdot u + I_y \cdot v + \boxed{I_t}$$

$$I(x + u, y + v, t + 1) - I(x, y, t) = +I_x \cdot u + I_y \cdot v + I_t$$

Brightness Constancy Equation



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

Take Taylor expansion of *I(x+u, y+v, t+1)* at *(x,y,t)* to linearize the right side:

Image derivative along x   Difference over frames

$$I(x+u, y+v, t+1) \approx I(x,y,t) + \boxed{I_x} \cdot u + I_y \cdot v + \boxed{I_t}$$

$$I(x+u, y+v, t+1) - I(x,y,t) = +I_x \cdot u + I_y \cdot v + I_t$$

Hence, $I_x \cdot u + I_y \cdot v + I_t \approx 0 \quad \rightarrow \nabla \mathrm{I} \cdot [\mathrm{u} \;\; \mathrm{v}]^{\mathrm{T}} + \mathrm{I_t} = 0$

What Does this Mean?

$$\nabla I \cdot \begin{bmatrix} u & v \end{bmatrix}^T + I_t = 0$$

- What do the static image gradients have to do with motion estimation?

What Does this Mean?

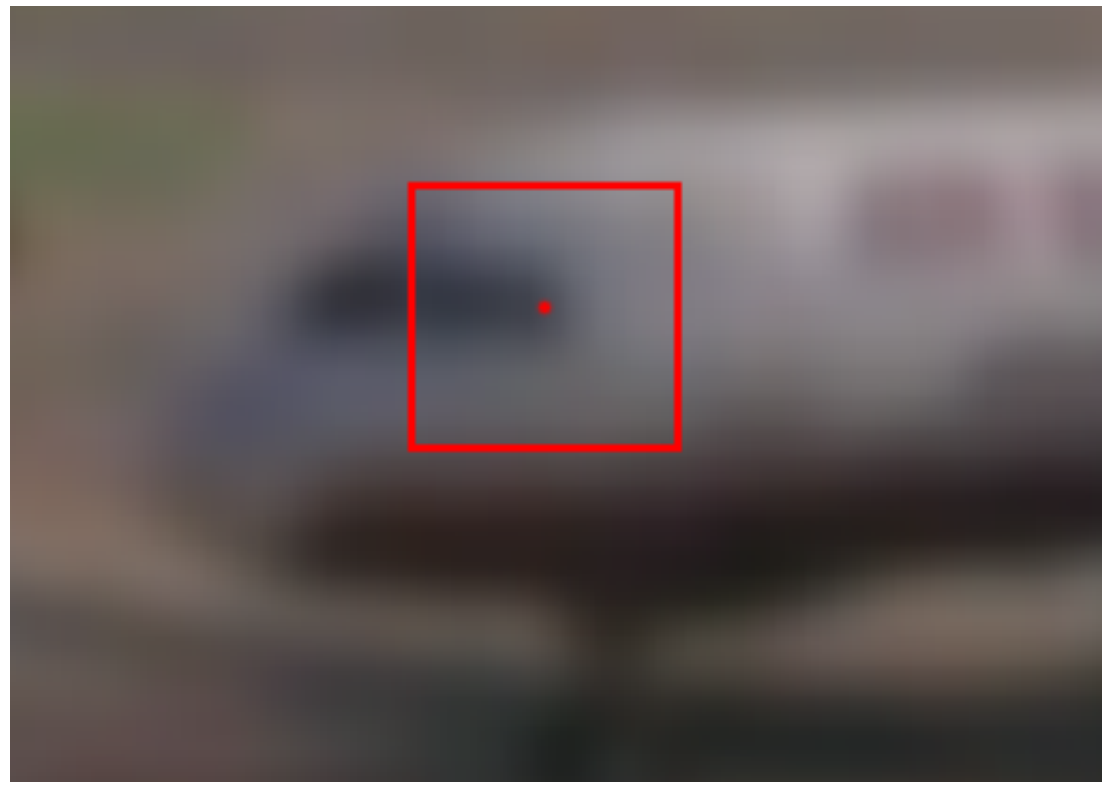
$$\nabla I \cdot \begin{bmatrix} u & v \end{bmatrix}^T + I_t = 0$$

- What do the static image gradients have to do with motion estimation?

What Does this Mean?

$$\nabla I \cdot \begin{bmatrix} u & v \end{bmatrix}^T + I_t = 0$$

- What do the static image gradients have to do with motion estimation?



Brightness Constancy Constraint

$$\nabla I \cdot \begin{bmatrix} u & v \end{bmatrix}^T + I_t = 0$$

Can we use this equation to recover image motion (u,v) at each pixel?

Brightness Constancy Constraint

How many equations and unknowns per pixel?

$$\nabla I \cdot \left[u \;\; v\right]^T + I_t = 0$$

Can we use this equation to recover image motion (u,v) at each pixel?

Brightness Constancy Constraint

How many equations and unknowns per pixel?

$$\nabla I \cdot \begin{bmatrix} u & v \end{bmatrix}^T + I_t = 0$$

Can we use this equation to recover image motion (u,v) at each pixel?

•One equation (this is a scalar equation!), two unknowns (u,v)

Brightness Constancy Constraint

How many equations and unknowns per pixel?

$$\nabla I \cdot \begin{bmatrix} u & v \end{bmatrix}^T + I_t = 0$$

Can we use this equation to recover image motion (u,v) at each pixel?

•One equation (this is a scalar equation!), two unknowns (u,v)

The component of the motion perpendicular to the gradient (i.e., parallel to the edge) cannot be measured

Brightness Constancy Constraint

How many equations and unknowns per pixel?

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

Can we use this equation to recover image motion (u,v) at each pixel?

- One equation (this is a scalar equation!), two unknowns (u,v)

The component of the motion perpendicular to the gradient (i.e., parallel to the edge) cannot be measured

If (u, v) satisfies the equation,
so does (u+u', v+v') if

$$\nabla I \cdot [u' \;\; v']^T = 0$$

Brightness Constancy Constraint

How many equations and unknowns per pixel?

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

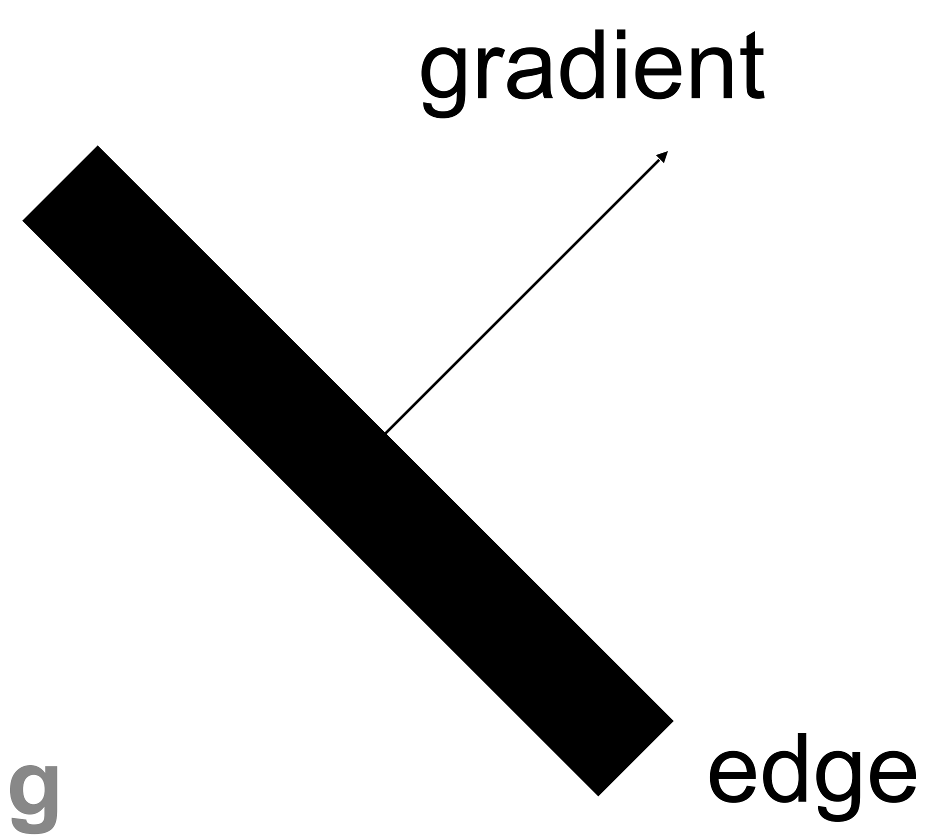
Can we use this equation to recover image motion (u,v) at each pixel?

•One equation (this is a scalar equation!), two unknowns (u,v)

The component of the motion perpendicular to the gradient (i.e., parallel to the edge) cannot be measured

If (*u*, *v*) satisfies the equation, so does (*u*+*u’*, *v*+*v’*) if

$$\nabla I \cdot [u' \;\; v']^T = 0$$



Brightness Constancy Constraint

How many equations and unknowns per pixel?

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

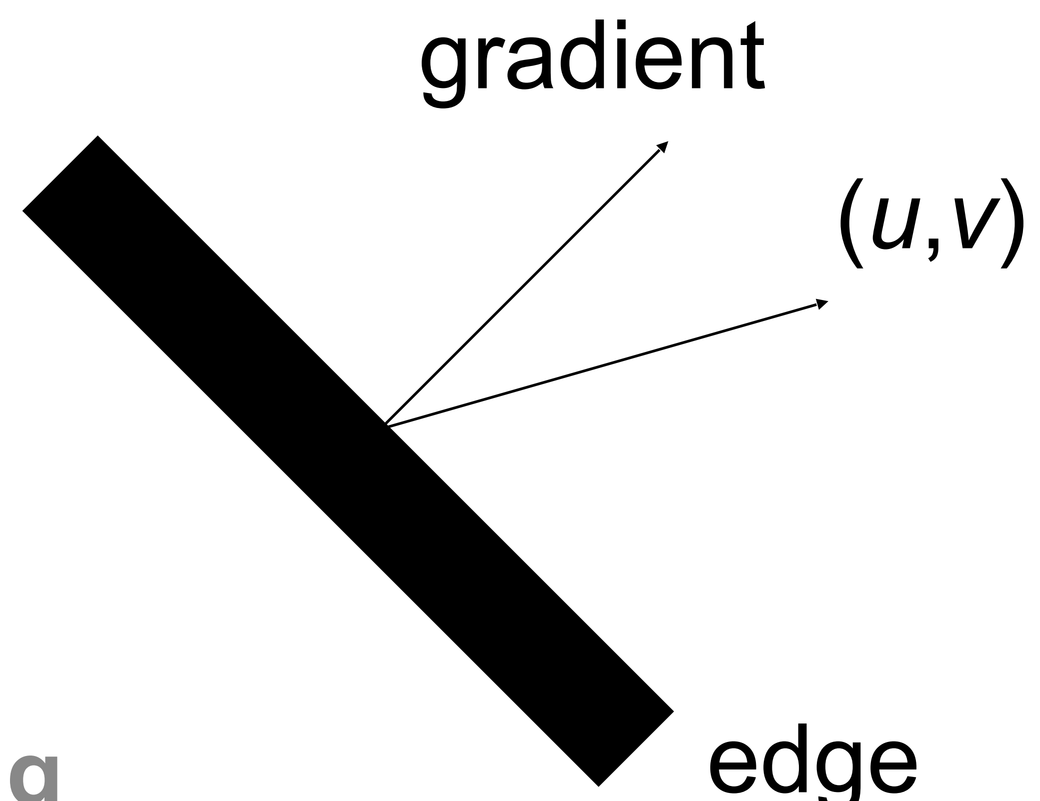
Can we use this equation to recover image motion (u,v) at each pixel?

•One equation (this is a scalar equation!), two unknowns (u,v)

The component of the motion perpendicular to the gradient (i.e., parallel to the edge) cannot be measured

If (u, v) satisfies the equation, so does ($u+u'$, $v+v'$) if

$$\nabla I \cdot [u' \;\; v']^T = 0$$



Brightness Constancy Constraint

How many equations and unknowns per pixel?

$$\nabla I \cdot \begin{bmatrix} u & v \end{bmatrix}^T + I_t = 0$$

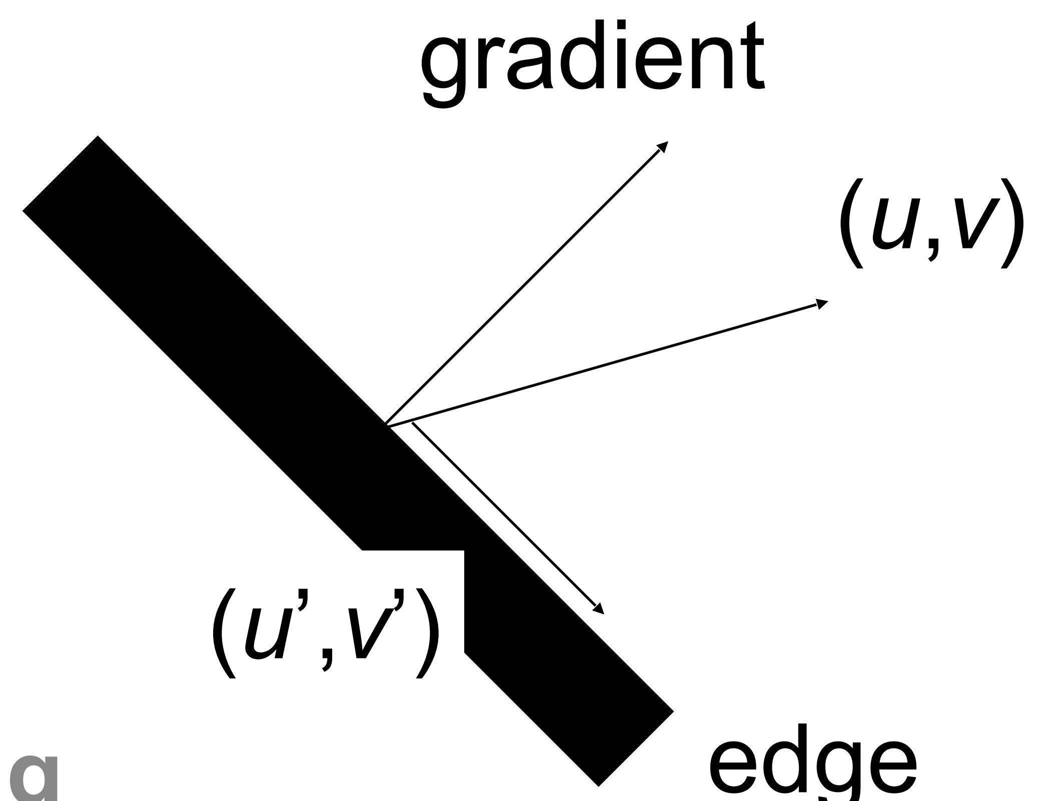
Can we use this equation to recover image motion (u,v) at each pixel?

•One equation (this is a scalar equation!), two unknowns (u,v)

The component of the motion perpendicular to the gradient (i.e., parallel to the edge) cannot be measured

If (u, v) satisfies the equation, so does ($u+u'$, $v+v'$) if

$$\nabla I \cdot \begin{bmatrix} u' & v' \end{bmatrix}^T = 0$$



Brightness Constancy Constraint

How many equations and unknowns per pixel?

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

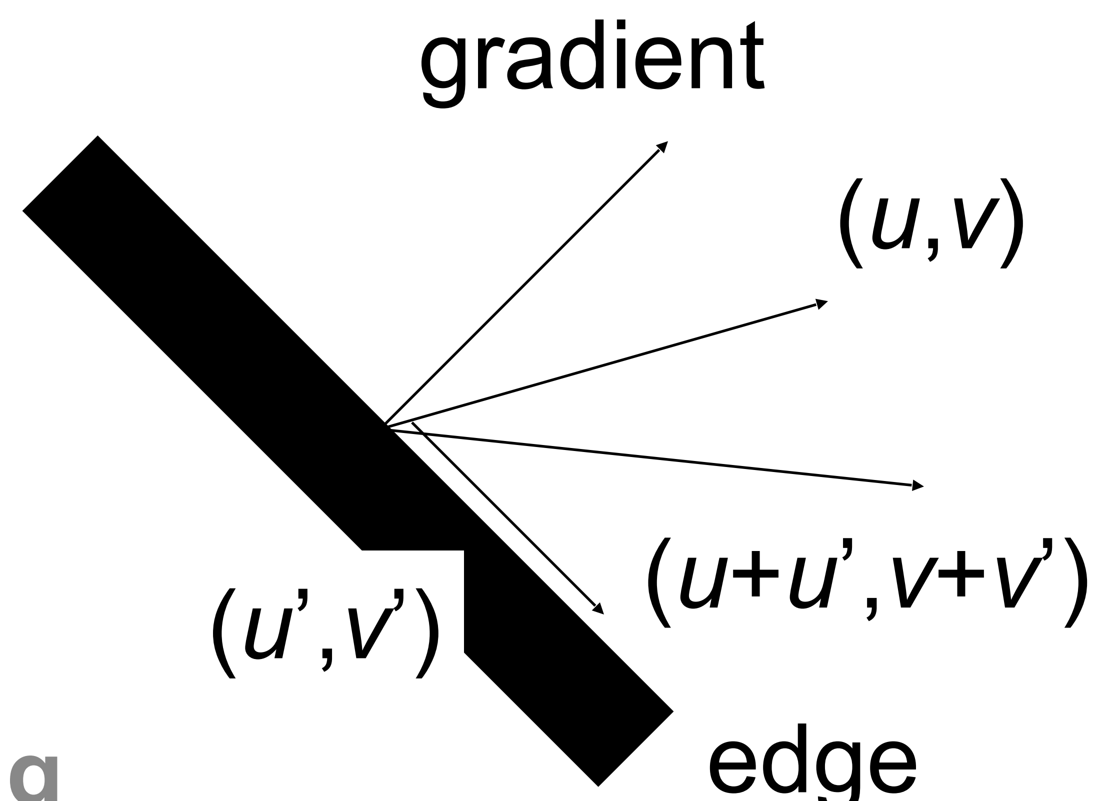
Can we use this equation to recover image motion (u,v) at each pixel?

•One equation (this is a scalar equation!), two unknowns (u,v)

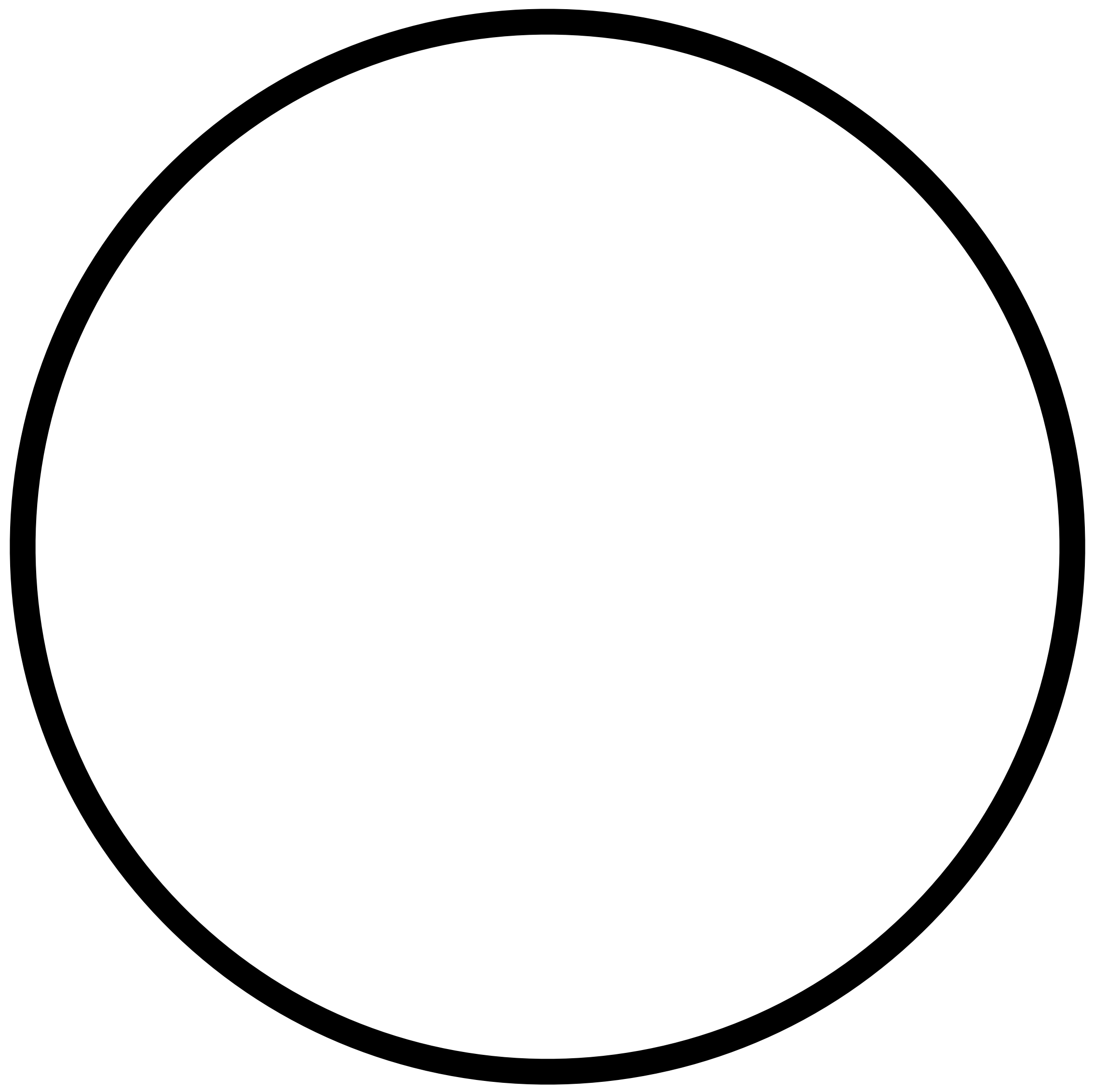
The component of the motion perpendicular to the gradient (i.e., parallel to the edge) cannot be measured

If (u, v) satisfies the equation, so does ($u+u'$, $v+v'$) if

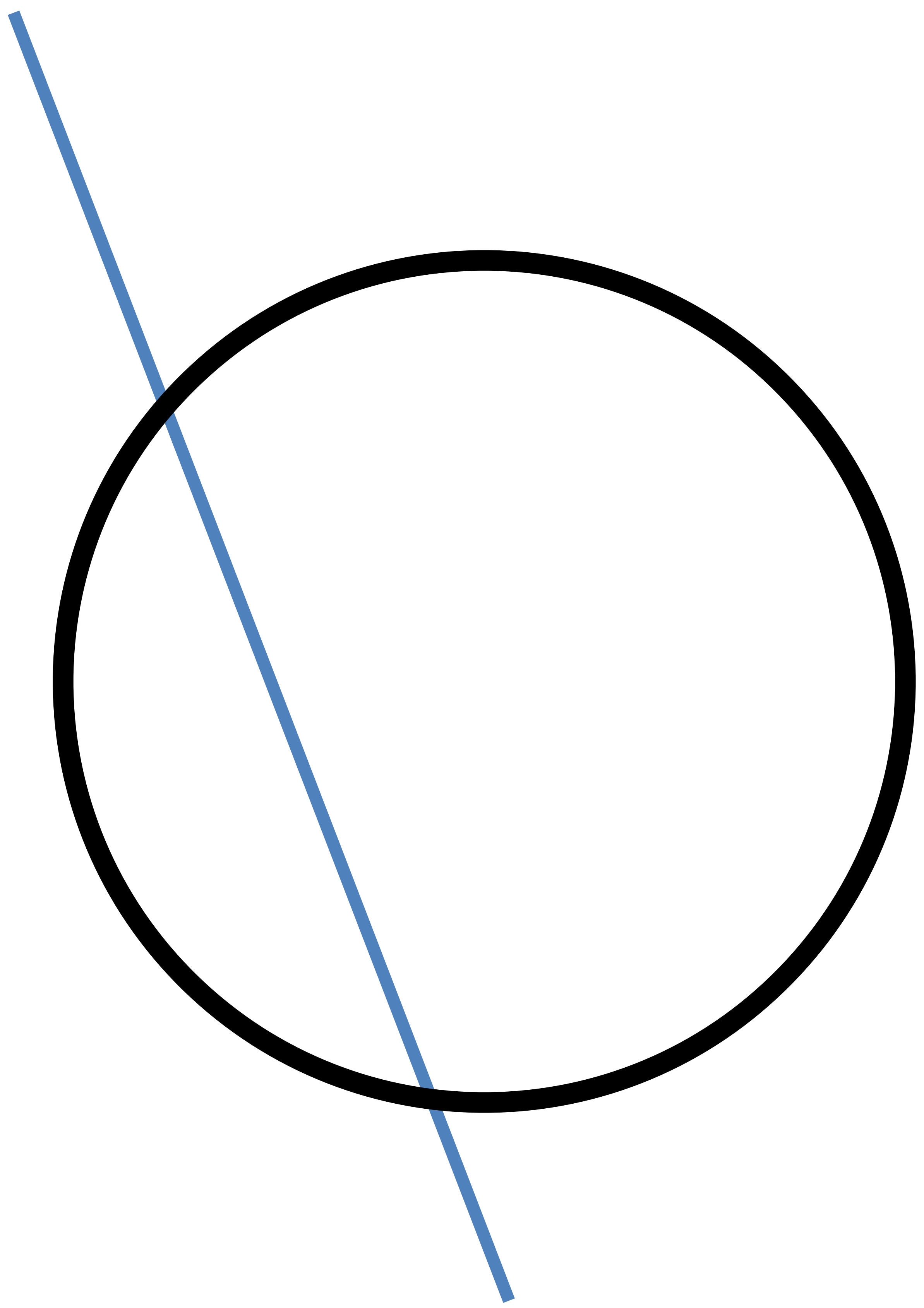
$$\nabla I \cdot [u' \;\; v']^T = 0$$



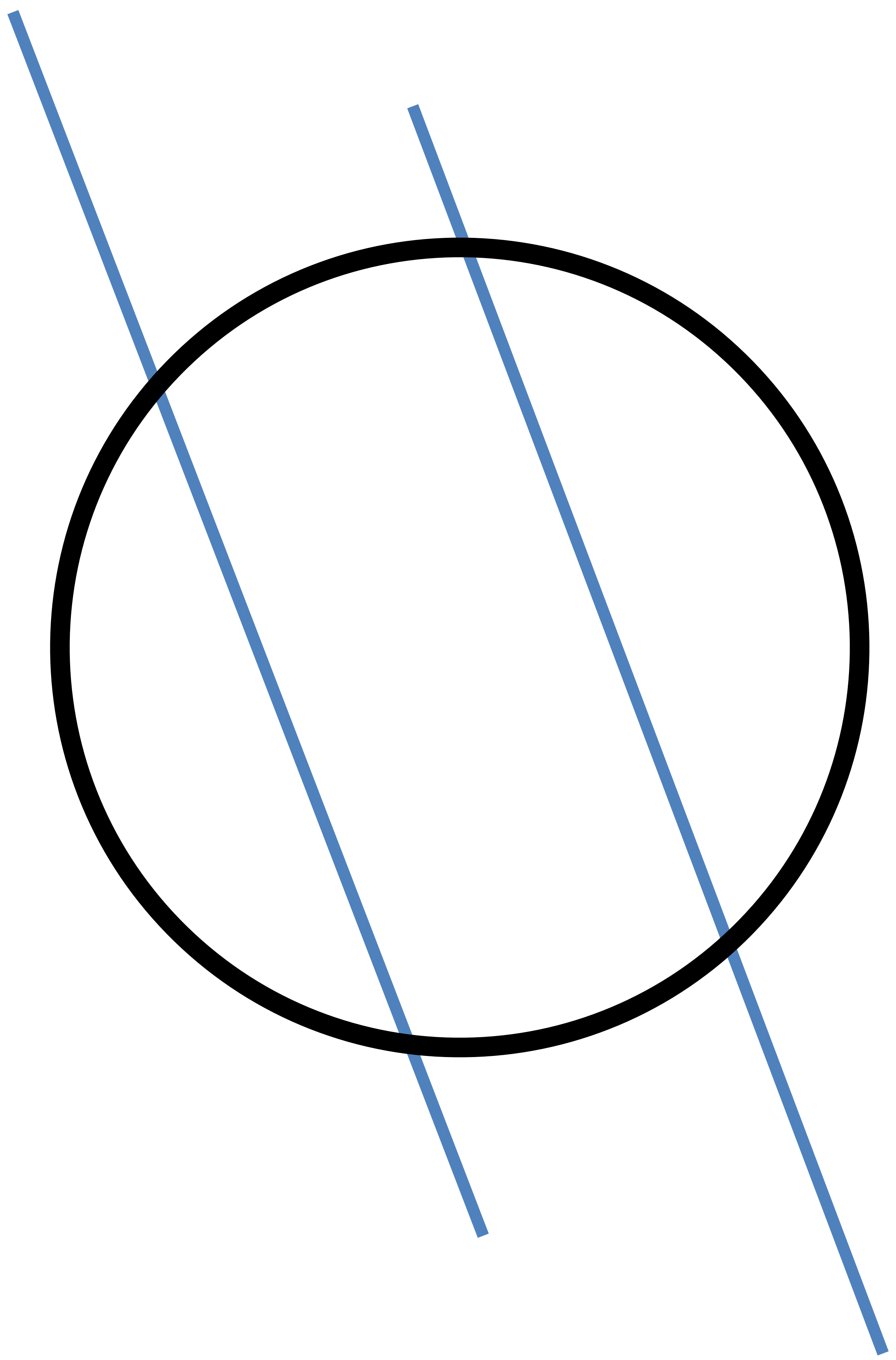
The Aperture Problem



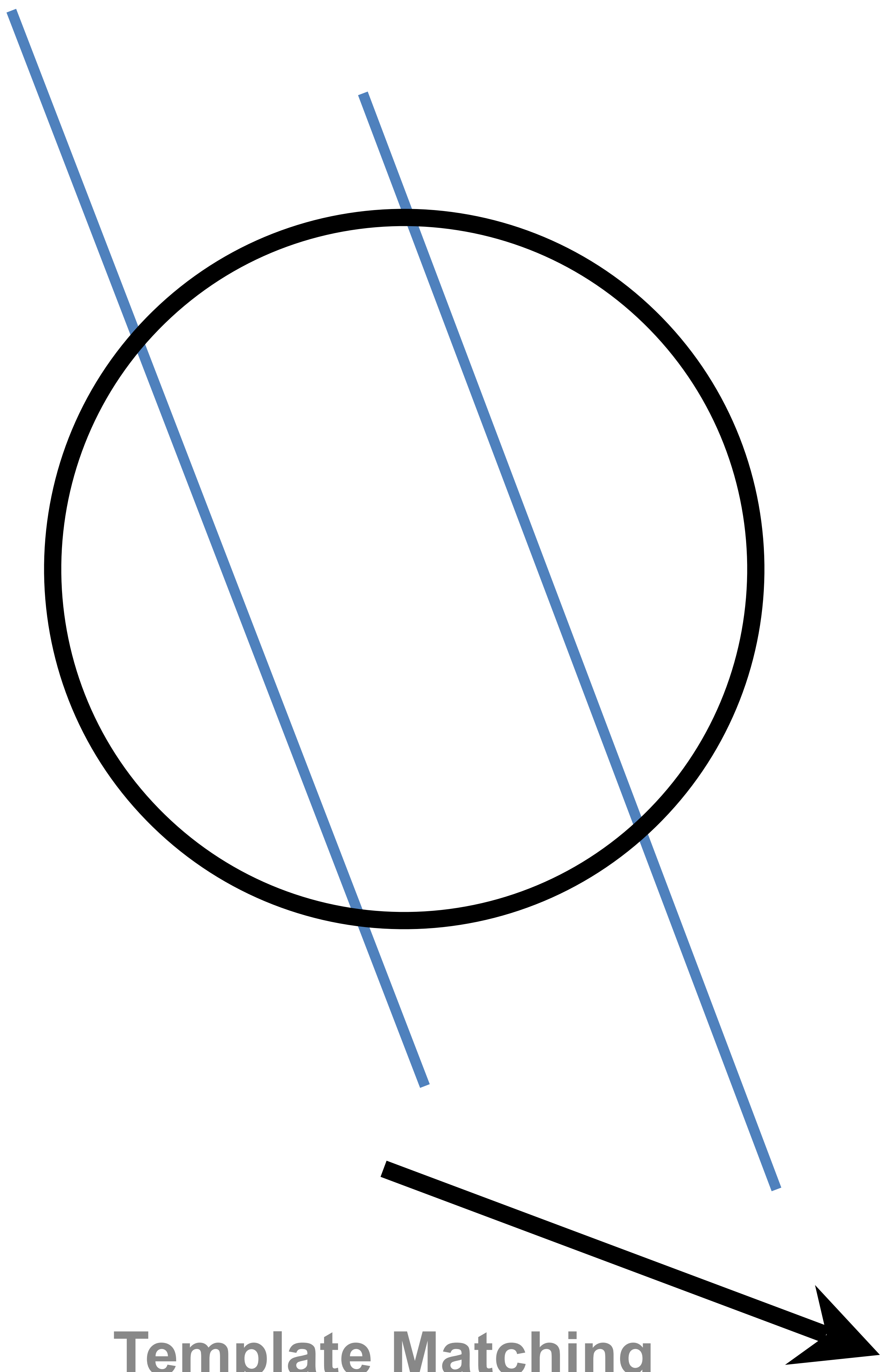
The Aperture Problem



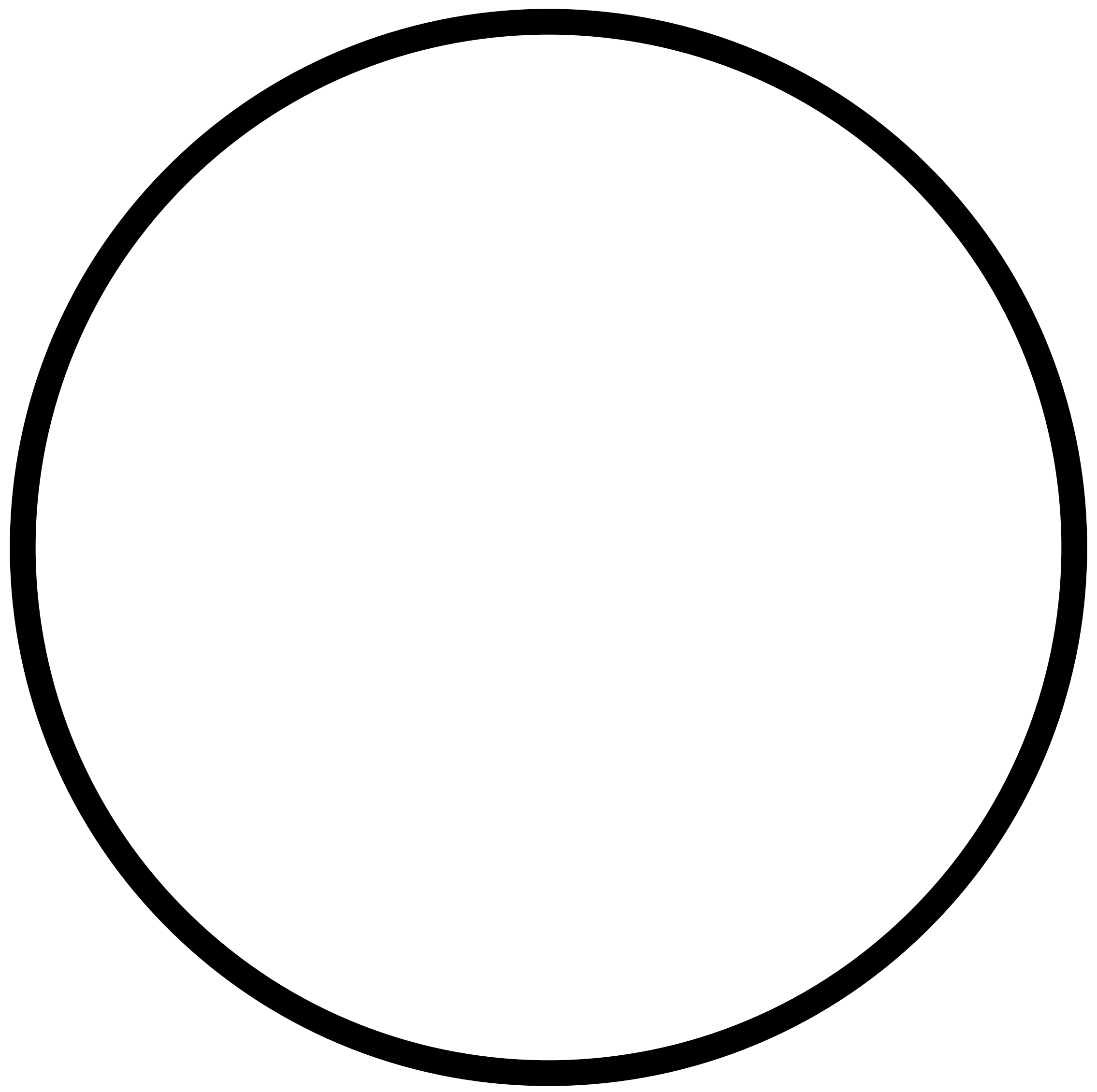
The Aperture Problem



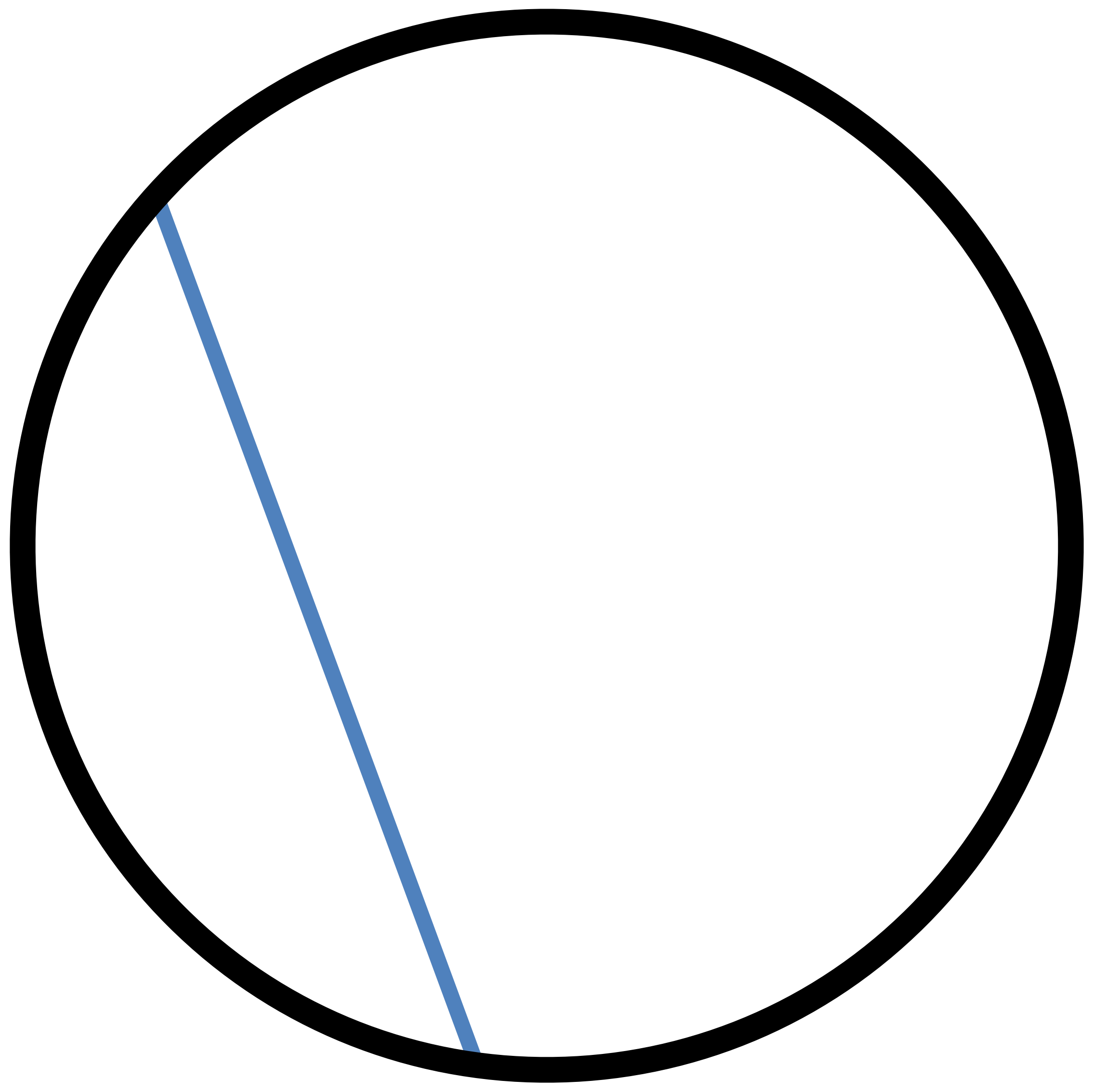
The Aperture Problem



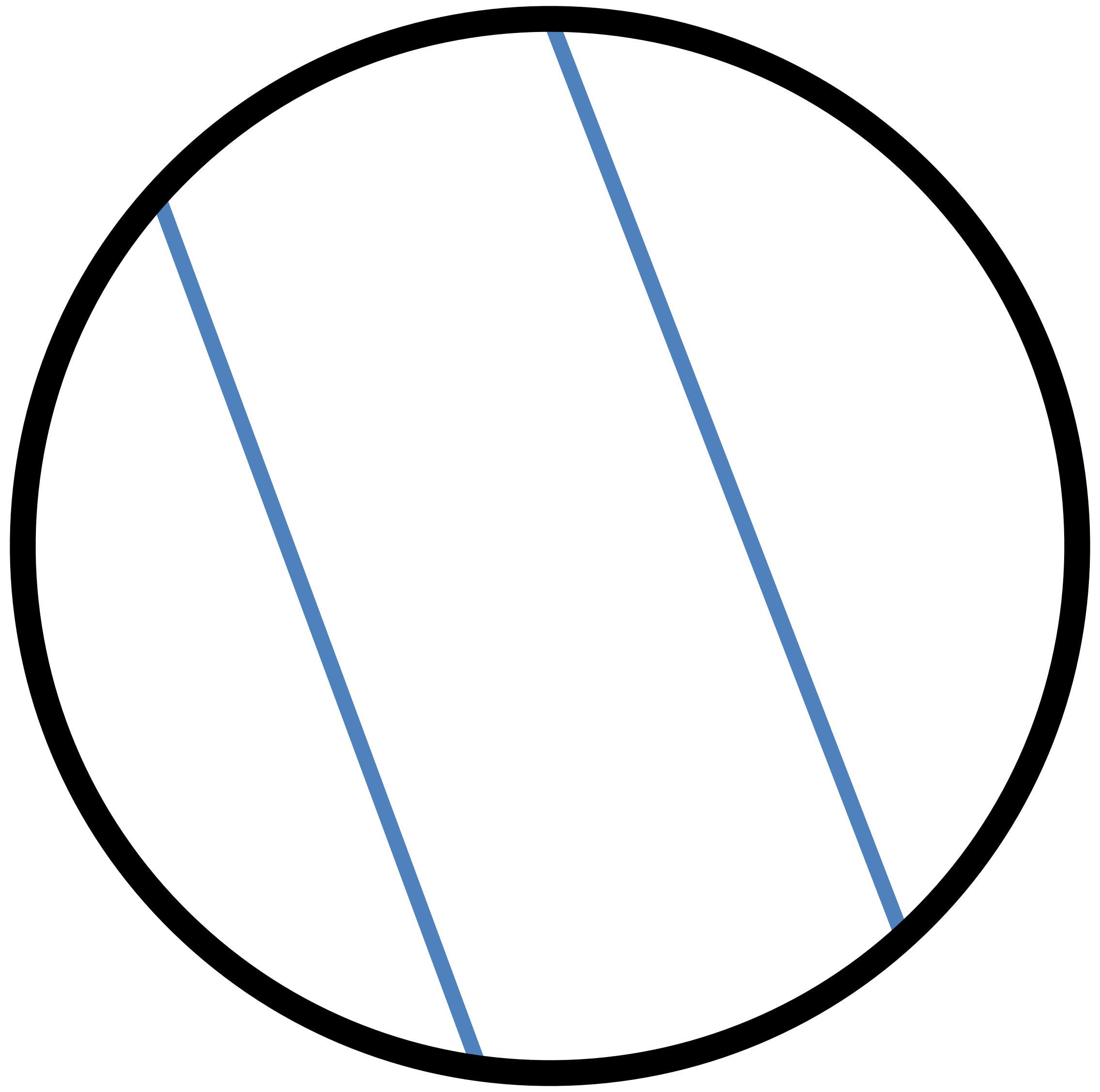
The Aperture Problem



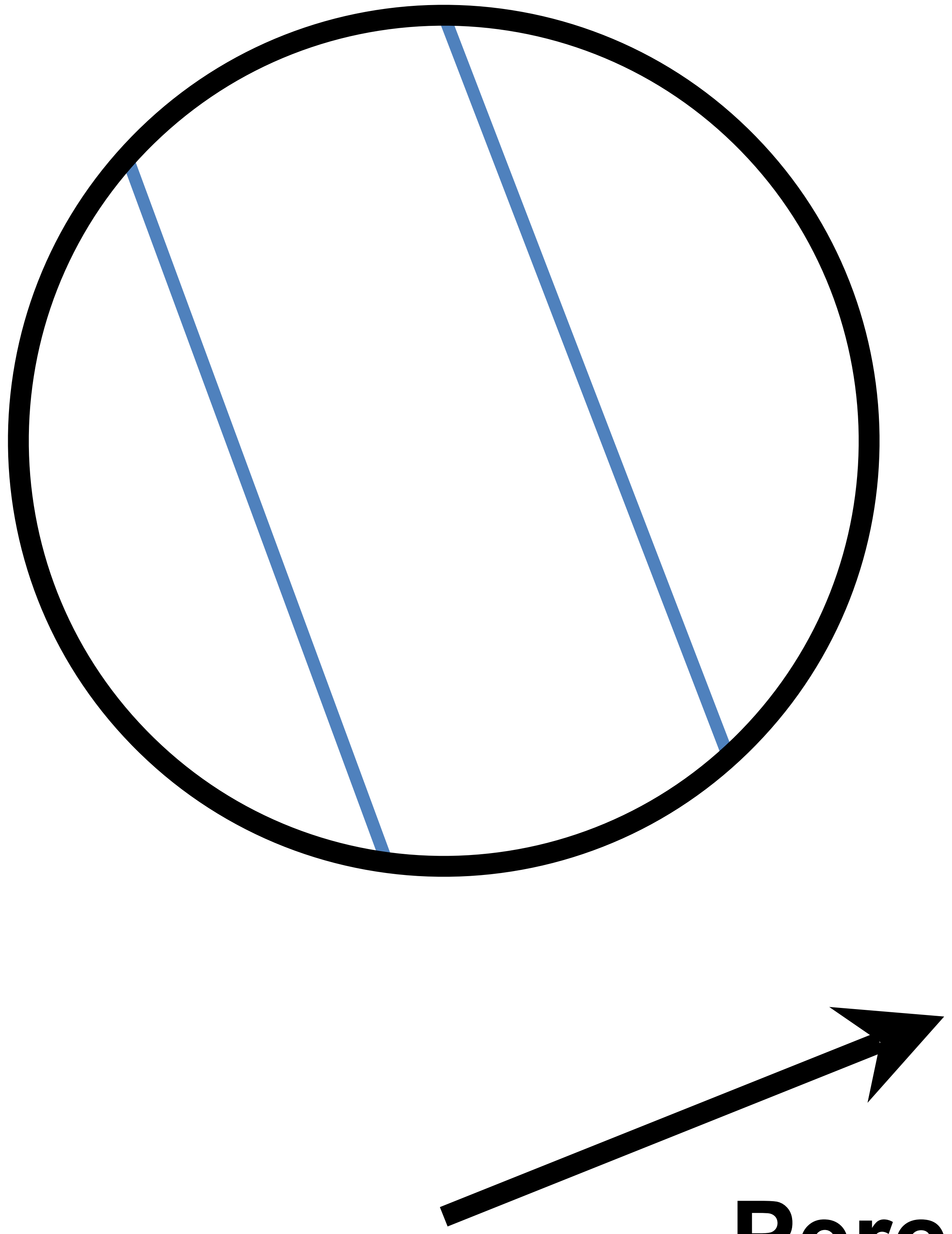
The Aperture Problem



The Aperture Problem

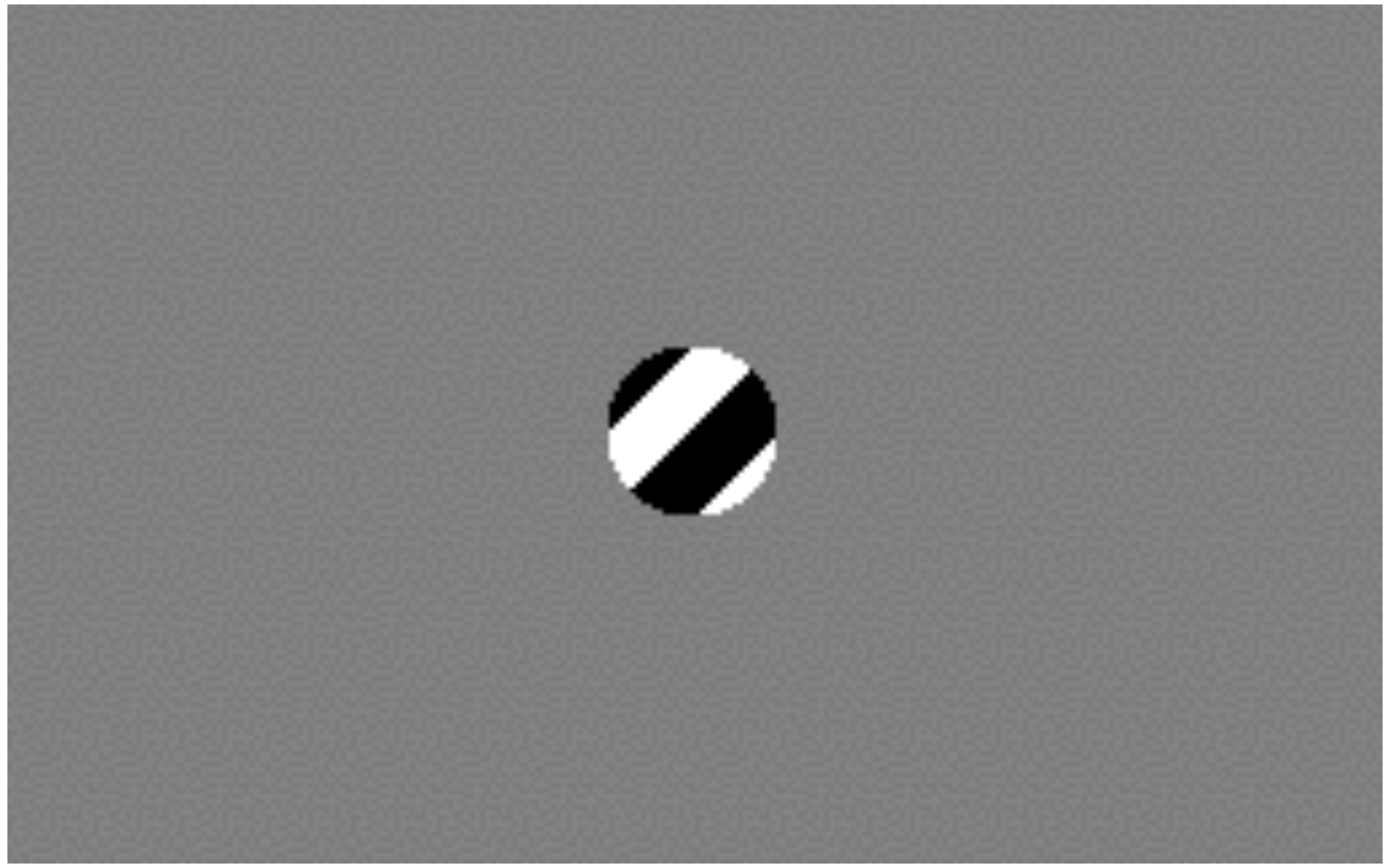


The Aperture Problem



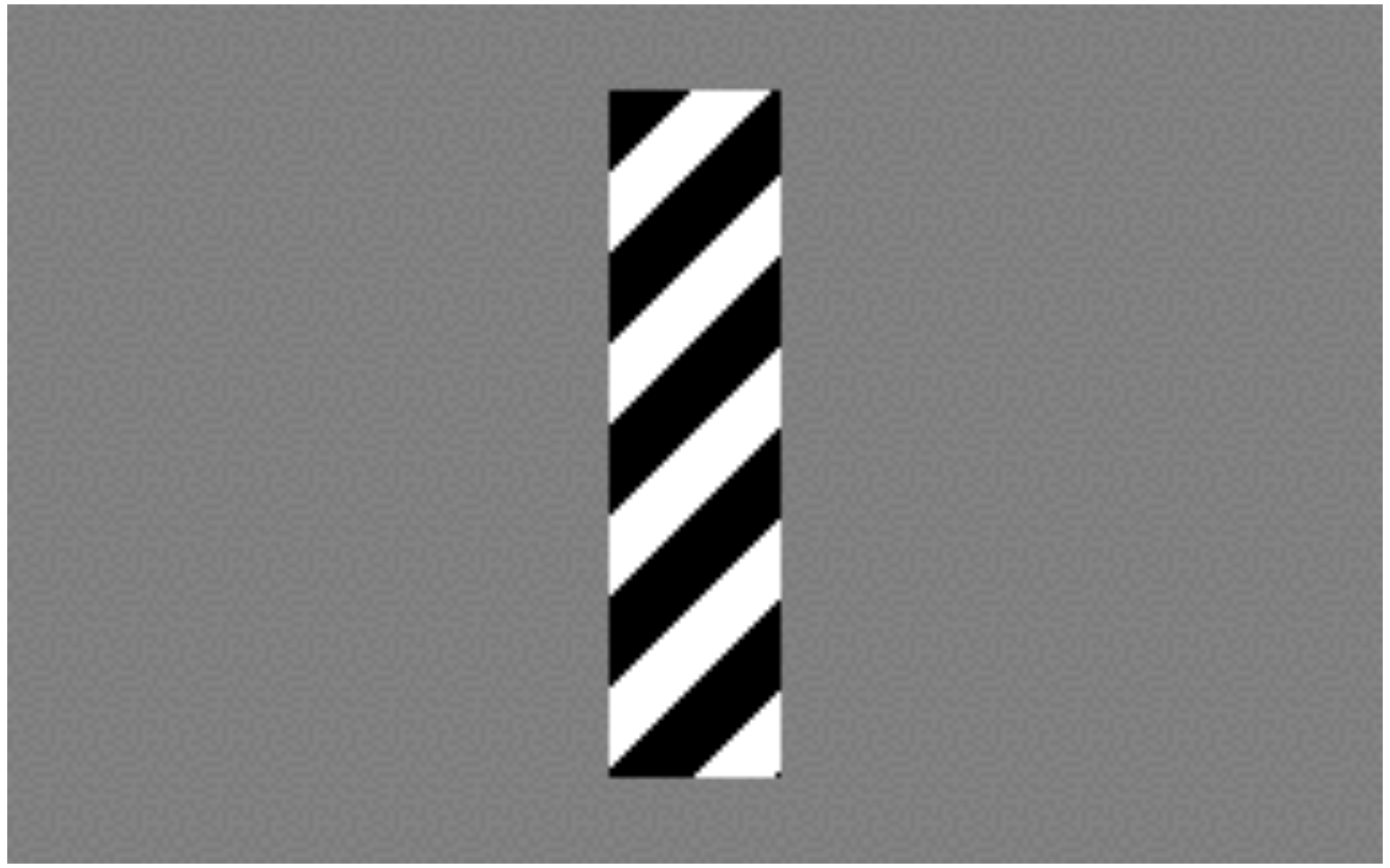
UCL

Barber Pole Illusion



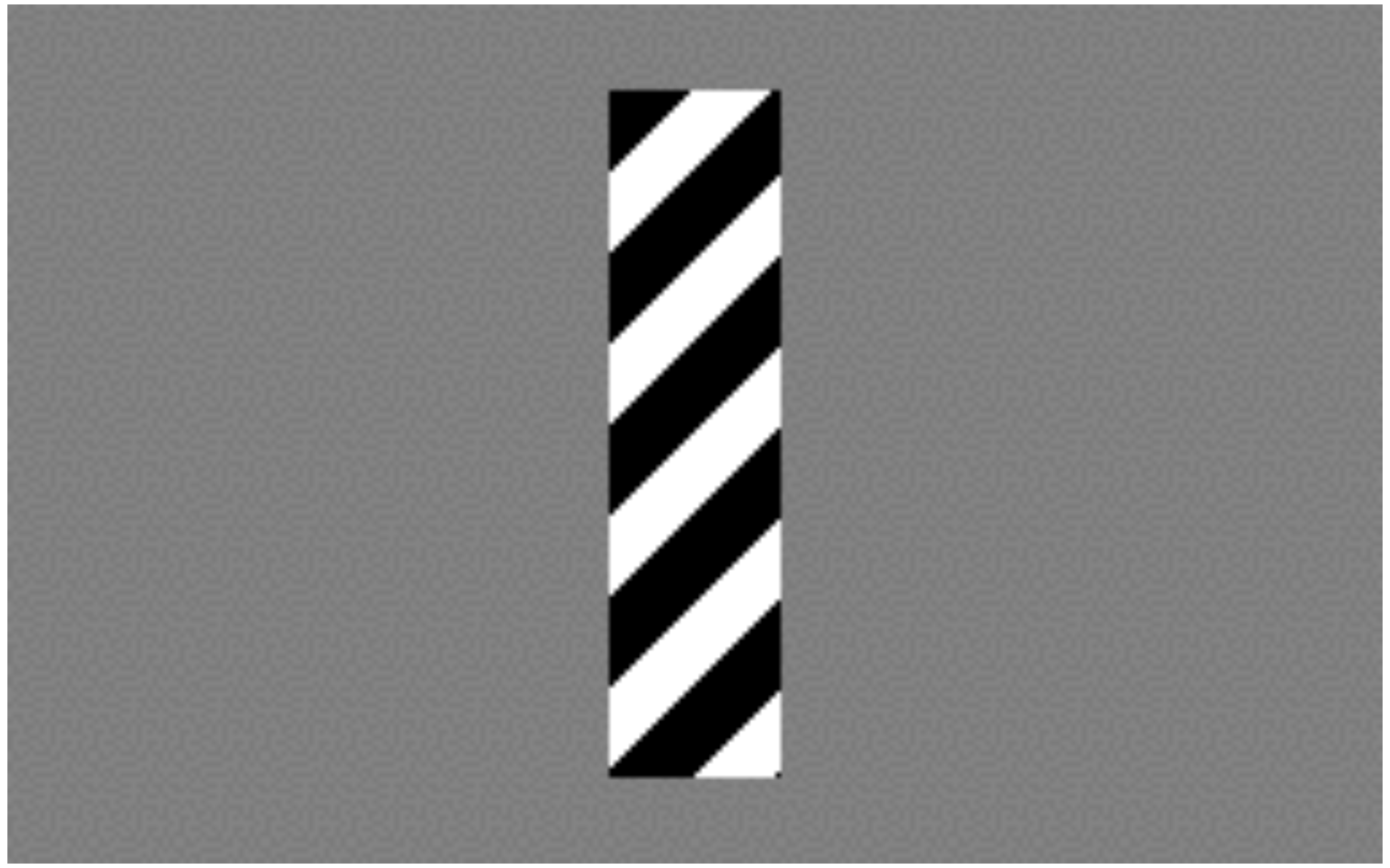
http://en.wikipedia.org/wiki/Barberpole_illusion

Barber Pole Illusion



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

Barber Pole Illusion



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

Addressing the Ambiguity...

B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 674–679, 1981.

Addressing the Ambiguity...

B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 674–679, 1981.

- **How to get more equations for a pixel?**

Addressing the Ambiguity...

B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 674–679, 1981.

- **How to get more equations for a pixel?**
- **Spatial coherence constraint**

Addressing the Ambiguity...

B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 674–679, 1981.

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

Addressing the Ambiguity…

B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 674–679, 1981.

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

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

Addressing the Ambiguity…

B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 674–679, 1981.

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

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

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

Addressing the Ambiguity…

- **Least squares problem:**

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

Matching Patches Across Images

$$\begin{bmatrix} I_x(\mathrm{p}_1) & I_y(\mathrm{p}_1) \\ I_x(\mathrm{p}_2) & I_y(\mathrm{p}_2) \\ \vdots & \vdots \\ I_x(\mathrm{p}_{25}) & I_y(\mathrm{p}_{25}) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(\mathrm{p}_1) \\ I_t(\mathrm{p}_2) \\ \vdots \\ I_t(\mathrm{p}_{25}) \end{bmatrix} \qquad \underset{25\text{x}2}{A} \; \underset{2\text{x}1}{d} = \underset{25\text{x}1}{b}$$

Least squares solution for *d* given by

Matching Patches Across Images

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

Least squares solution for *d* given by $(A^T A)\ d = A^T b$

Matching Patches Across Images

$$\begin{bmatrix} I_x(\mathrm{p}_1) & I_y(\mathrm{p}_1) \\ I_x(\mathrm{p}_2) & I_y(\mathrm{p}_2) \\ \vdots & \vdots \\ I_x(\mathrm{p}_{25}) & I_y(\mathrm{p}_{25}) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(\mathrm{p}_1) \\ I_t(\mathrm{p}_2) \\ \vdots \\ I_t(\mathrm{p}_{25}) \end{bmatrix} \qquad \underset{25\times2}{A}\ \underset{2\times1}{d} = \underset{25\times1}{b}$$

Least squares solution for *d* given by $(A^T A)\ d = A^T b$

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

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

Conditions for Solvability

Optimal (u, v) satisfies Lucas-Kanade equation

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

Conditions for Solvability

Optimal (u, v) satisfies Lucas-Kanade equation

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

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

When is this solvable? I.e., what are good points to track?

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? I.e., what are good points to track?

- **$A^T A$** should be invertible

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? I.e., what are good points to track?

- **A^TA** should be invertible
- **A^TA** should not be too small due to noise
  - eigenvalues λ_1 and λ_2 of $\mathbf{A^T A}$ should not be too small

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? I.e., what are good points to track?

- **$A^T A$** should be invertible
- **$A^T A$** should not be too small due to noise
  - 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)

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? I.e., what are good points to track?

- **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)

Does this remind you of anything?

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? I.e., what are good points to track?

- **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)

Does this remind you of anything?

Criteria for Harris corner detector

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? I.e., what are good points to track?

- $\mathbf{A^T A}$ should be invertible
- $\mathbf{A^T A}$ should not be too small due to noise
  - eigenvalues λ_1 and λ_2 of $\mathbf{A^T A}$ should not be too small
- $\mathbf{A^T A}$ should be well-conditioned
  - λ_1 / λ_2 should not be too large (λ_1 = larger eigenvalue)

Does this remind you of anything?

Criteria for Harris corner detector

Harris Corner Detector (revisited)

$$A^T A = \begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} = \sum \begin{bmatrix} I_x \\ I_y \end{bmatrix} [I_x \ I_y] = \sum \nabla I (\nabla I)^T$$

M = $A^T A$ is the *second moment matrix* !

(Harris corner detector…)

Harris Corner Detector (revisited)

$$A^T A = \left[\begin{array}{cc} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{array} \right] = \sum \left[\begin{array}{c} I_x \\ I_y \end{array} \right] [I_x \; I_y] = \sum \nabla I (\nabla I)^T$$

- Eigenvectors and eigenvalues of $A^T A$ relate to edge direction and magnitude

M = $A^T A$ is the *second moment matrix* !

(Harris corner detector…)

Harris Corner Detector (revisited)

$$A^T A = \left[\begin{array}{cc} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{array} \right] = \sum \left[\begin{array}{c} I_x \\ I_y \end{array} \right] [I_x \ I_y] = \sum \nabla I (\nabla I)^T$$

- Eigenvectors and eigenvalues of $A^T A$ relate to edge direction and magnitude
  - The eigenvector associated with the larger eigenvalue points in the direction of fastest intensity change

M = $A^T A$ is the *second moment matrix* !

(Harris corner detector…)

Harris Corner Detector (revisited)

$$A^T A = \left[\begin{array}{cc} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{array} \right] = \sum \left[\begin{array}{c} I_x \\ I_y \end{array} \right] [I_x \ I_y] = \sum \nabla I (\nabla I)^T$$

- Eigenvectors and eigenvalues of $A^T A$ relate to edge direction and magnitude
  - The eigenvector associated with the larger eigenvalue points in the direction of fastest intensity change
  - The other eigenvector is orthogonal to it

M = $A^T A$ is the *second moment matrix* !

(Harris corner detector…)

Low-texture Region

$$\sum \nabla I (\nabla I)^T$$

– gradients have small magnitude
– small λ_1, small λ_2

Edge Feature

$$\sum \nabla I (\nabla I)^T$$

– gradients very large or very small

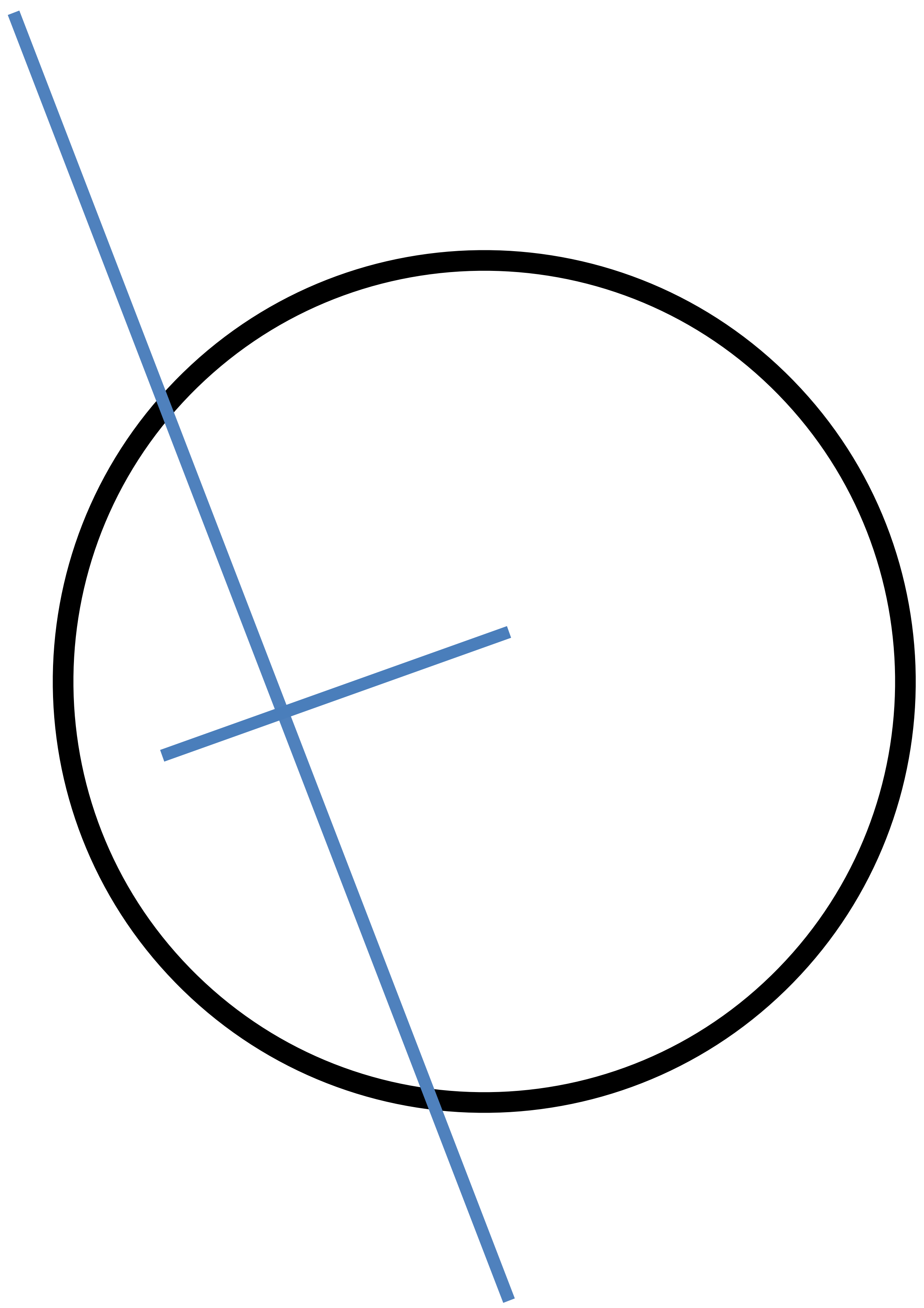
– large λ_1, small λ_2

High-texture Region

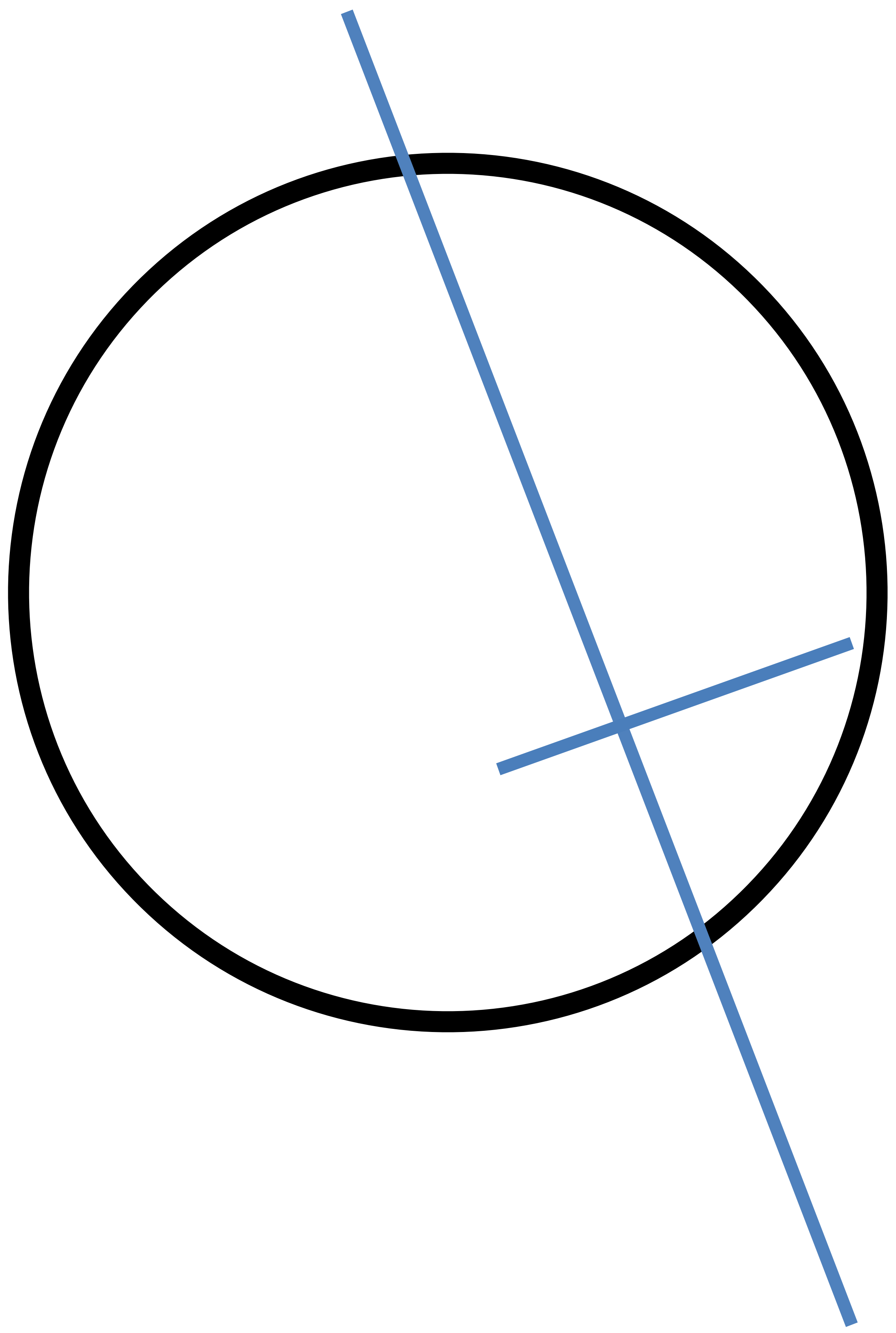
$$\sum \nabla I (\nabla I)^T$$

– gradients are different, large magnitudes
– large λ_1, large λ_2

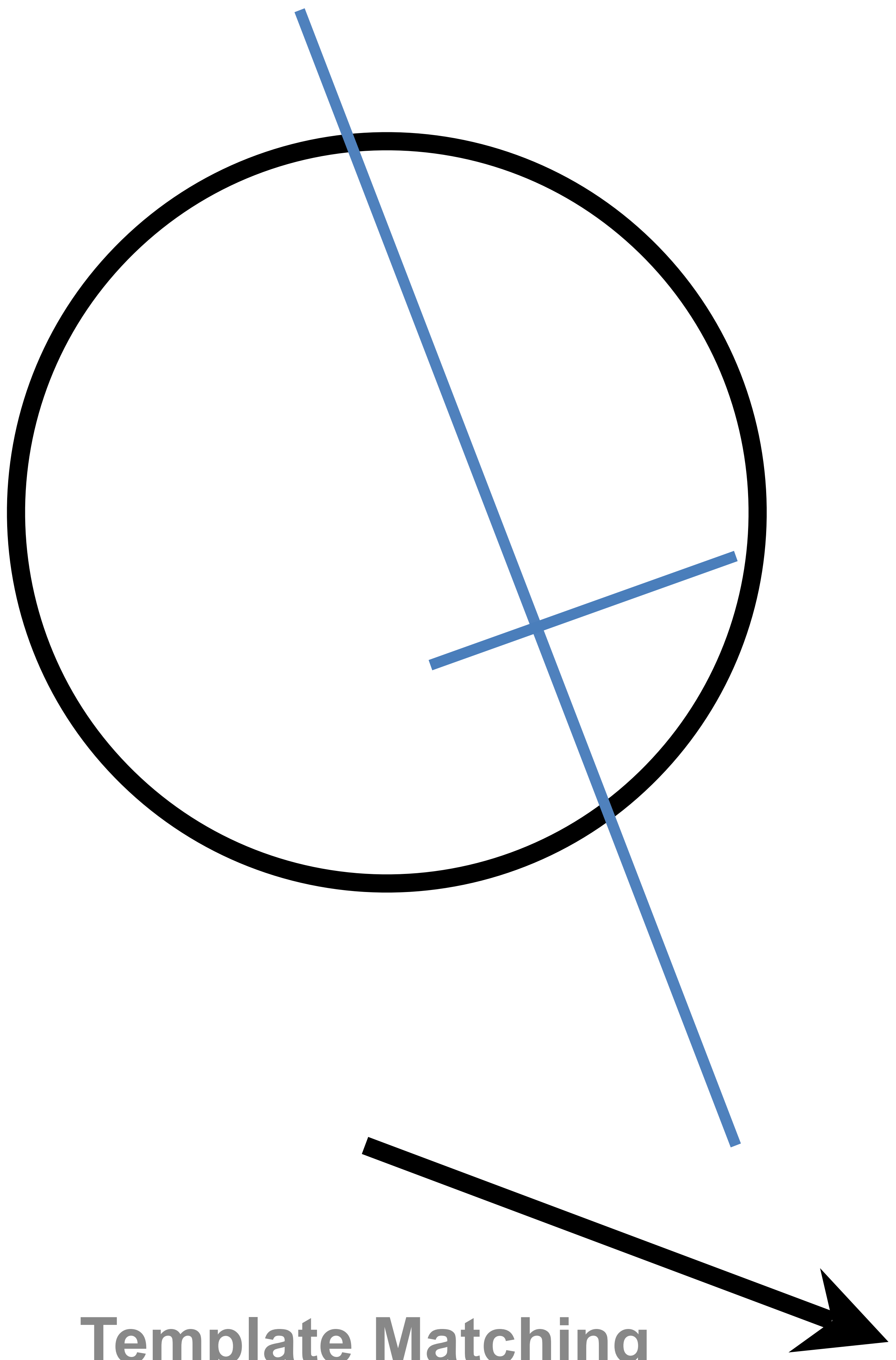
The Aperture Problem Resolved



The Aperture Problem Resolved

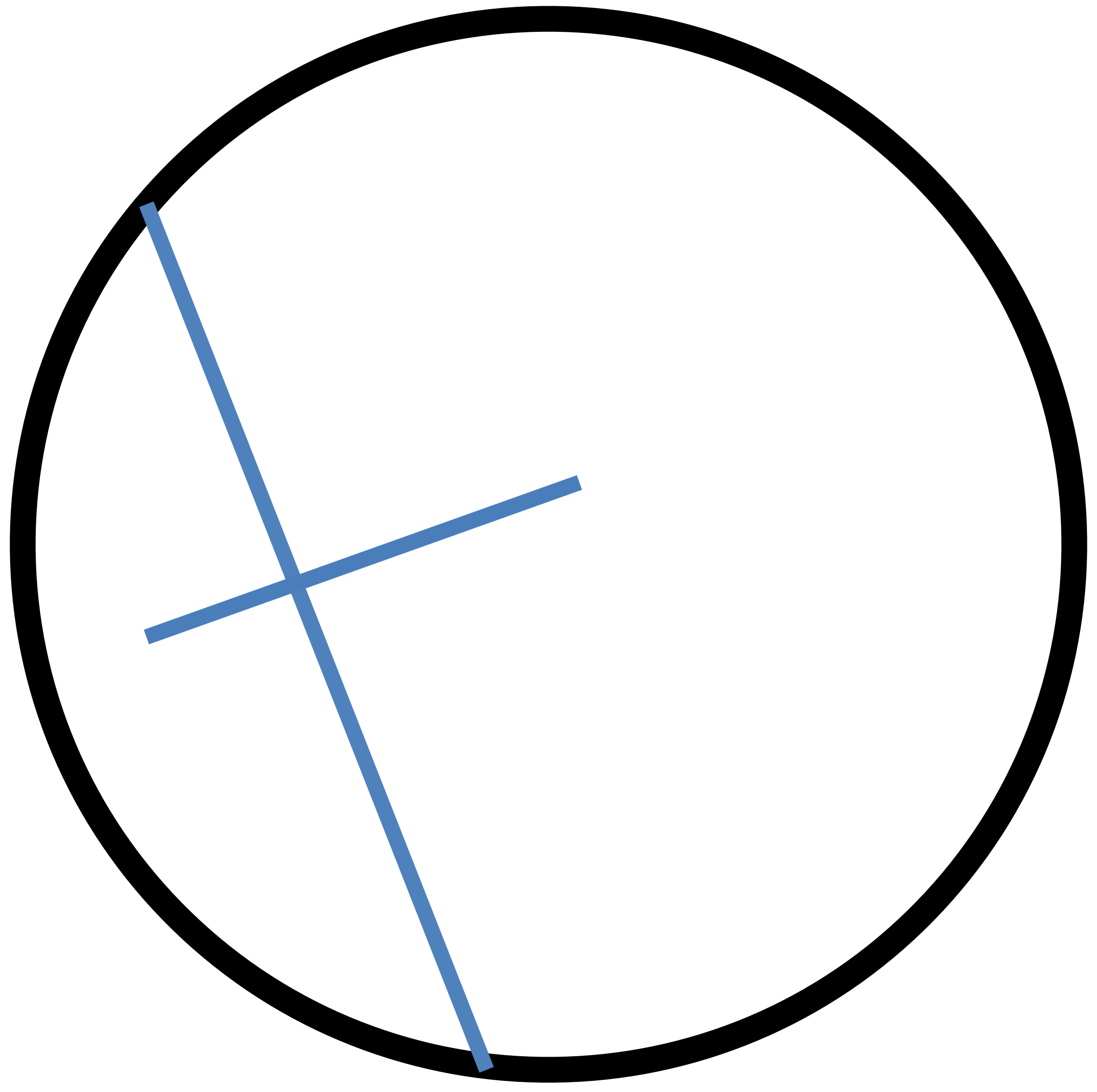


The Aperture Problem Resolved

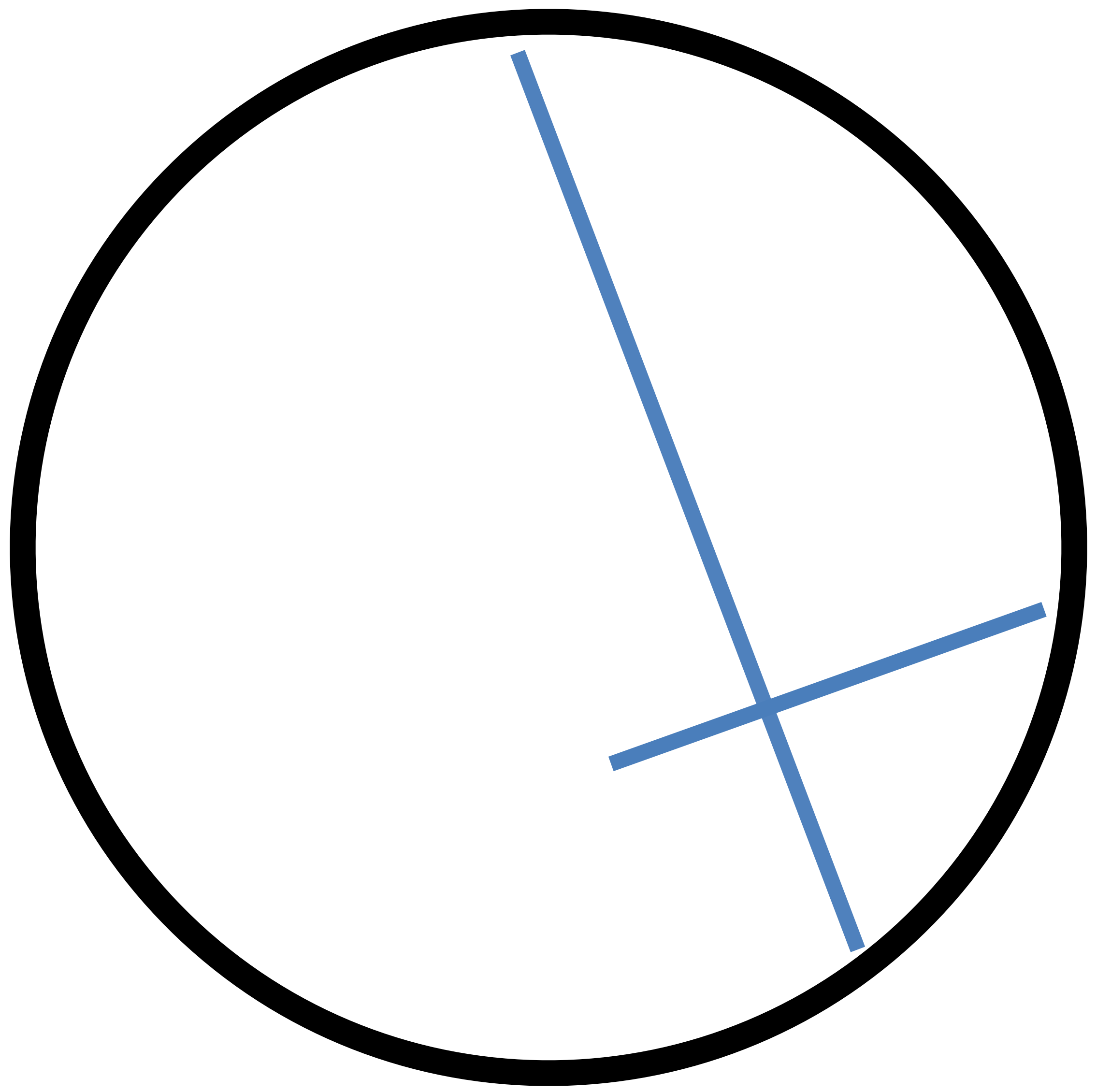


Actual motion

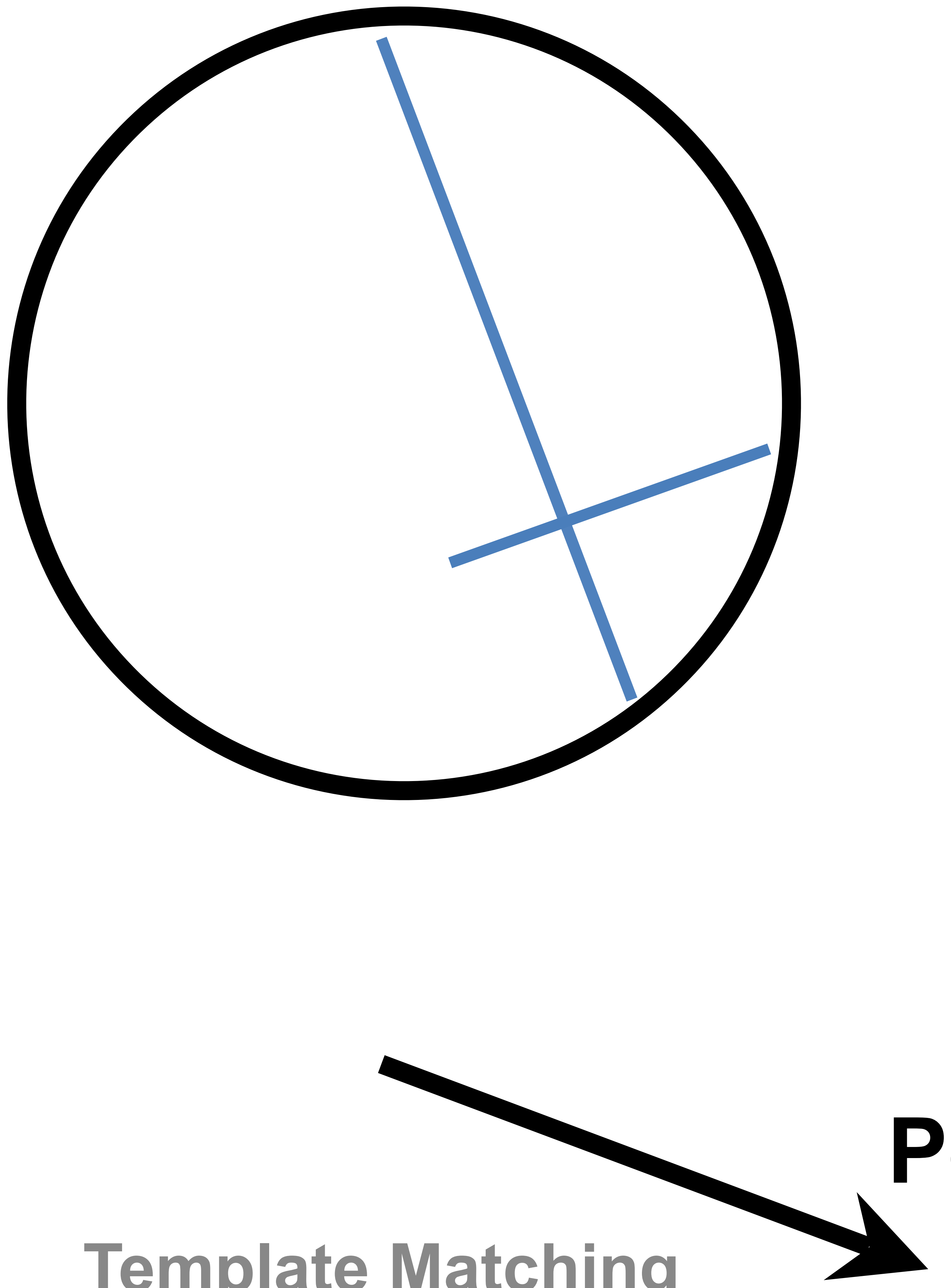
The Aperture Problem Resolved



The Aperture Problem Resolved



The Aperture Problem Resolved



Perceived motion

Dealing with Larger Movements: Iterative Refinement

Original (x,y) position

$I_t = I(x', y', t+1) - I(x, y, t)$

1. **Initialize (x',y') = (x,y)**
2. **Compute (u,v) by**

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

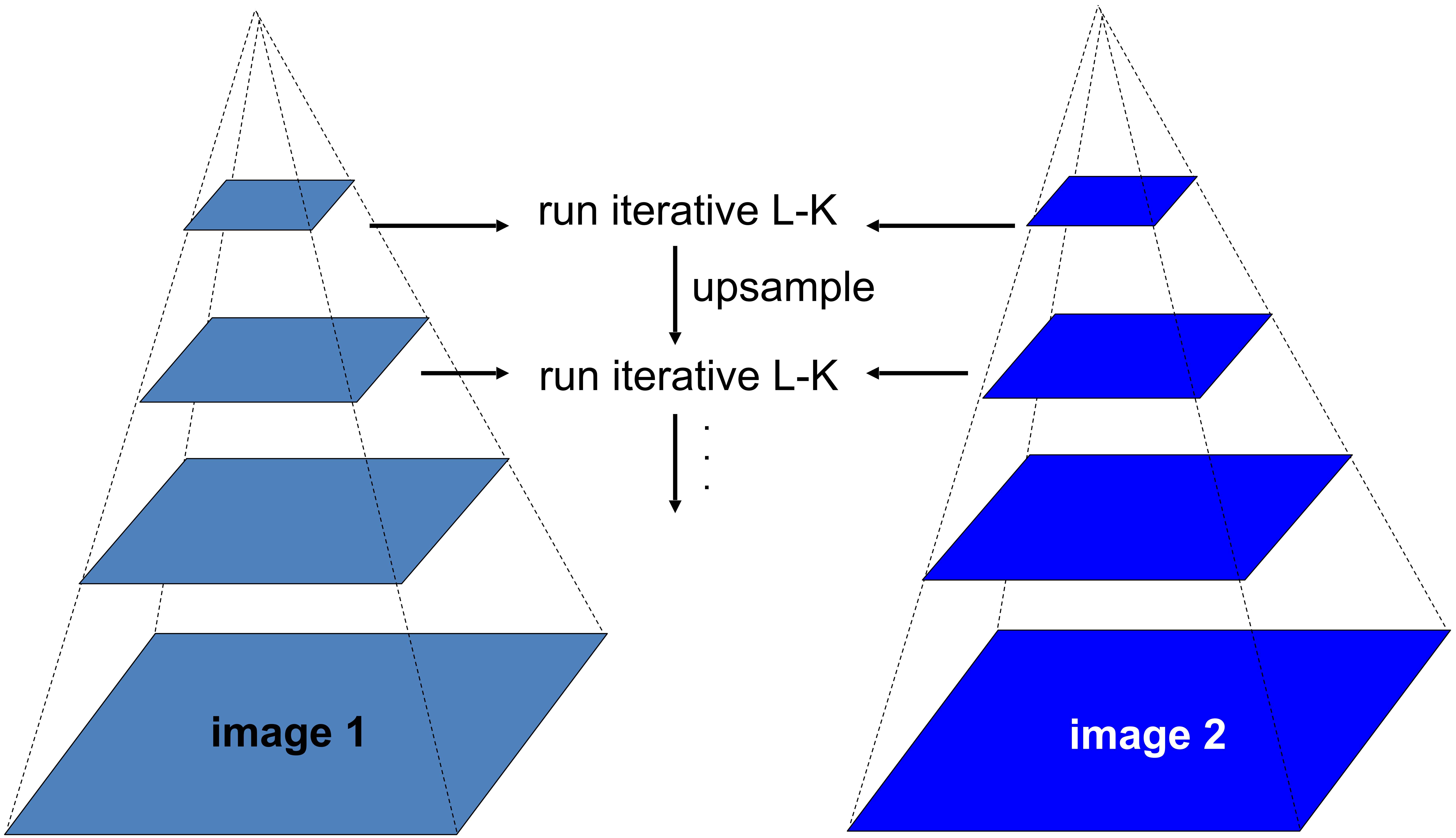
2nd moment matrix for feature patch in first image

displacement

3. **Shift window by (u, v): x'=x'+u; y'=y'+v;**
4. **Recalculate I_t**
5. **Repeat steps 2-4 until small change**

- **Use interpolation for subpixel values**

Dealing with Larger Movements: Coarse-to-fine Registration



Gaussian pyramid of image 1 (t) **Gaussian pyramid of image 2 (t+1)**

Shi-Tomasi Feature Tracker

J. Shi and C. Tomasi. [Good Features to Track](). CVPR 1994.

Shi-Tomasi Feature Tracker

- Find good features using eigenvalues of second-moment matrix (e.g., Harris detector or threshold on the smallest eigenvalue)
  - Key idea: “good” features to track are the ones whose motion can be estimated reliably

J. Shi and C. Tomasi. Good Features to Track. CVPR 1994.

Shi-Tomasi Feature Tracker

- Find good features using eigenvalues of second-moment matrix (e.g., Harris detector or threshold on the smallest eigenvalue)
  - Key idea: “good” features to track are the ones whose motion can be estimated reliably

- Track from frame to frame with Lucas-Kanade
  - This amounts to assuming a translation model for frame-to-frame feature movement

J. Shi and C. Tomasi. Good Features to Track. CVPR 1994.

Shi-Tomasi Feature Tracker

- Find good features using eigenvalues of second-moment matrix (e.g., Harris detector or threshold on the smallest eigenvalue)
  - Key idea: “good” features to track are the ones whose motion can be estimated reliably

- Track from frame to frame with Lucas-Kanade
  - This amounts to assuming a translation model for frame-to-frame feature movement

- Check consistency of tracks by *affine* registration to the first observed instance of the feature
  - Affine model is more accurate for larger displacements
  - Comparing to the first frame helps to minimize drift

J. Shi and C. Tomasi. [Good Features to Track](). CVPR 1994.

Tracking example

Figure 1: Three frame details from Woody Allen's *Manhattan*. The details are from the 1st, 11th, and 21st frames of a subsequence from the movie.

Figure 2: The traffic sign windows from frames 1,6,11,16,21 as tracked (top), and warped by the computed deformation matrices (bottom).

J. Shi and C. Tomasi. [Good Features to Track](). CVPR 1994.

Summary of KLT tracking

- Find a good point to track (Harris corner)

Summary of KLT tracking

- Find a good point to track (Harris corner)

- Use intensity second moment matrix and difference across frames to find displacement

Summary of KLT tracking

- Find a good point to track (Harris corner)
- Use intensity second moment matrix and difference across frames to find displacement
- Iterate and use coarse-to-fine search to deal with larger movements

Summary of KLT tracking

- Find a good point to track (Harris corner)
- Use intensity second moment matrix and difference across frames to find displacement
- Iterate and use coarse-to-fine search to deal with larger movements
- When creating long tracks, check appearance of registered patch against appearance of initial patch to find points that have drifted

Implementation Details

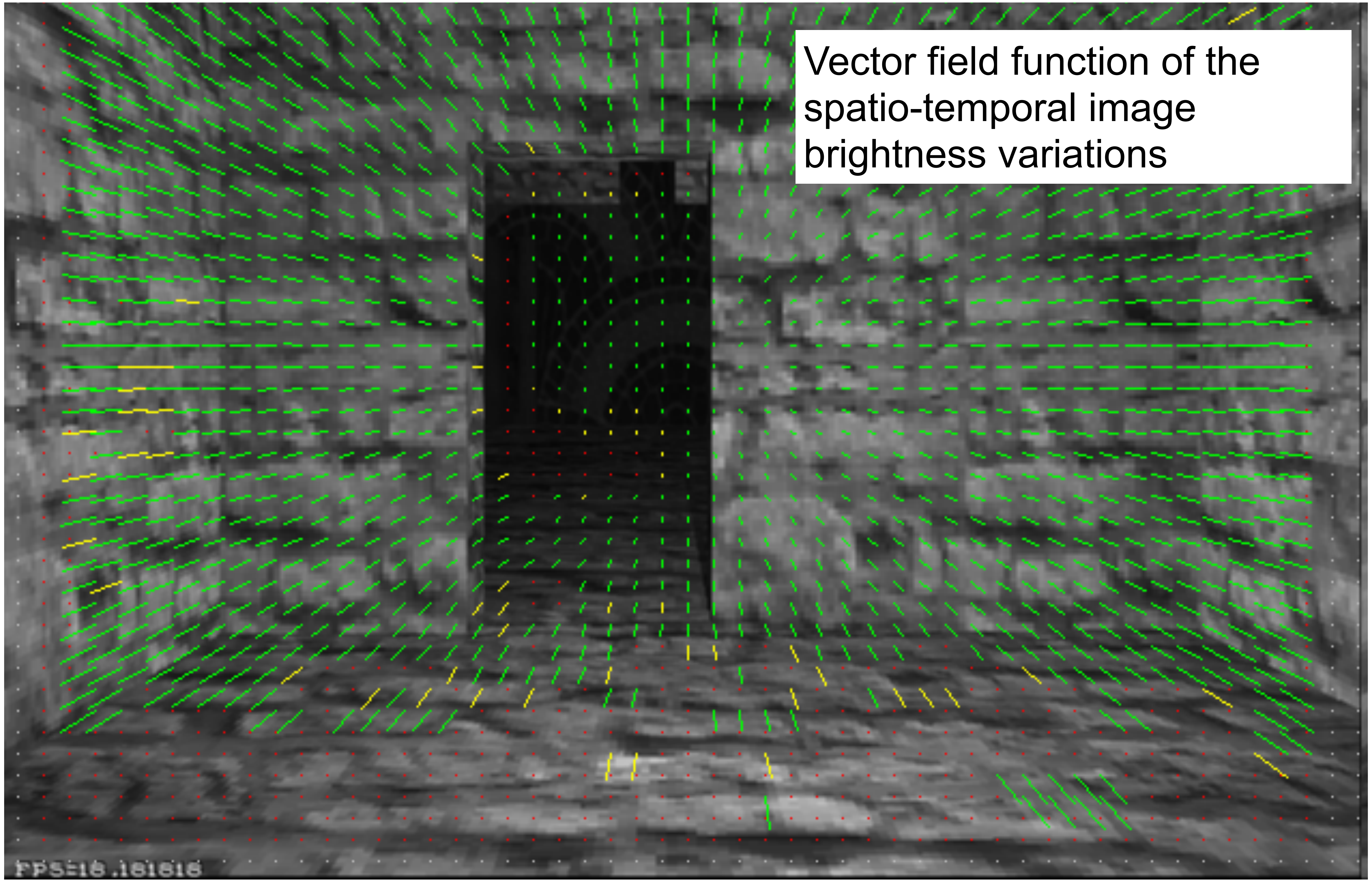
Implementation Details

- **Window size**
  - Small window more sensitive to noise and may miss larger motions (without pyramid)
  - Large window more likely to cross an occlusion boundary (and it's slower)
  - 15x15 to 31x31 seems typical

Implementation Details

- **Window size**
  - Small window more sensitive to noise and may miss larger motions (without pyramid)
  - Large window more likely to cross an occlusion boundary (and it’s slower)
  - 15x15 to 31x31 seems typical

- **Weighting the window**
  - Common to apply weights so that center matters more (e.g., with Gaussian)



Picture courtesy of Selim Temizer - Learning and Intelligent Systems (LIS) Group, MIT

Applications

Applications

- Estimating 3D structure

Applications

- Estimating 3D structure
- Segmenting objects based on motion cues

Applications

- Estimating 3D structure
- Segmenting objects based on motion cues
- Learning and tracking dynamical models

Applications

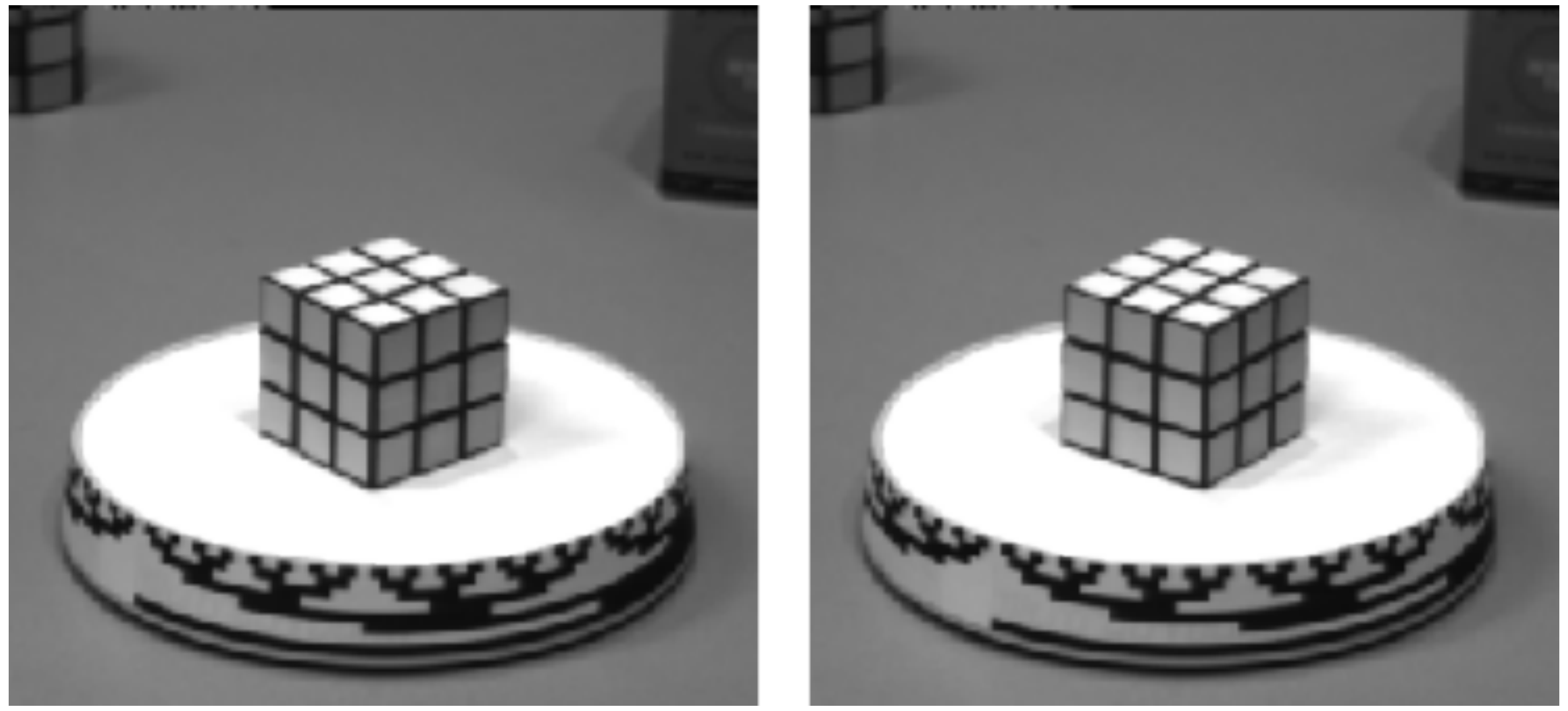
- Estimating 3D structure
- Segmenting objects based on motion cues
- Learning and tracking dynamical models
- Recognizing events and activities

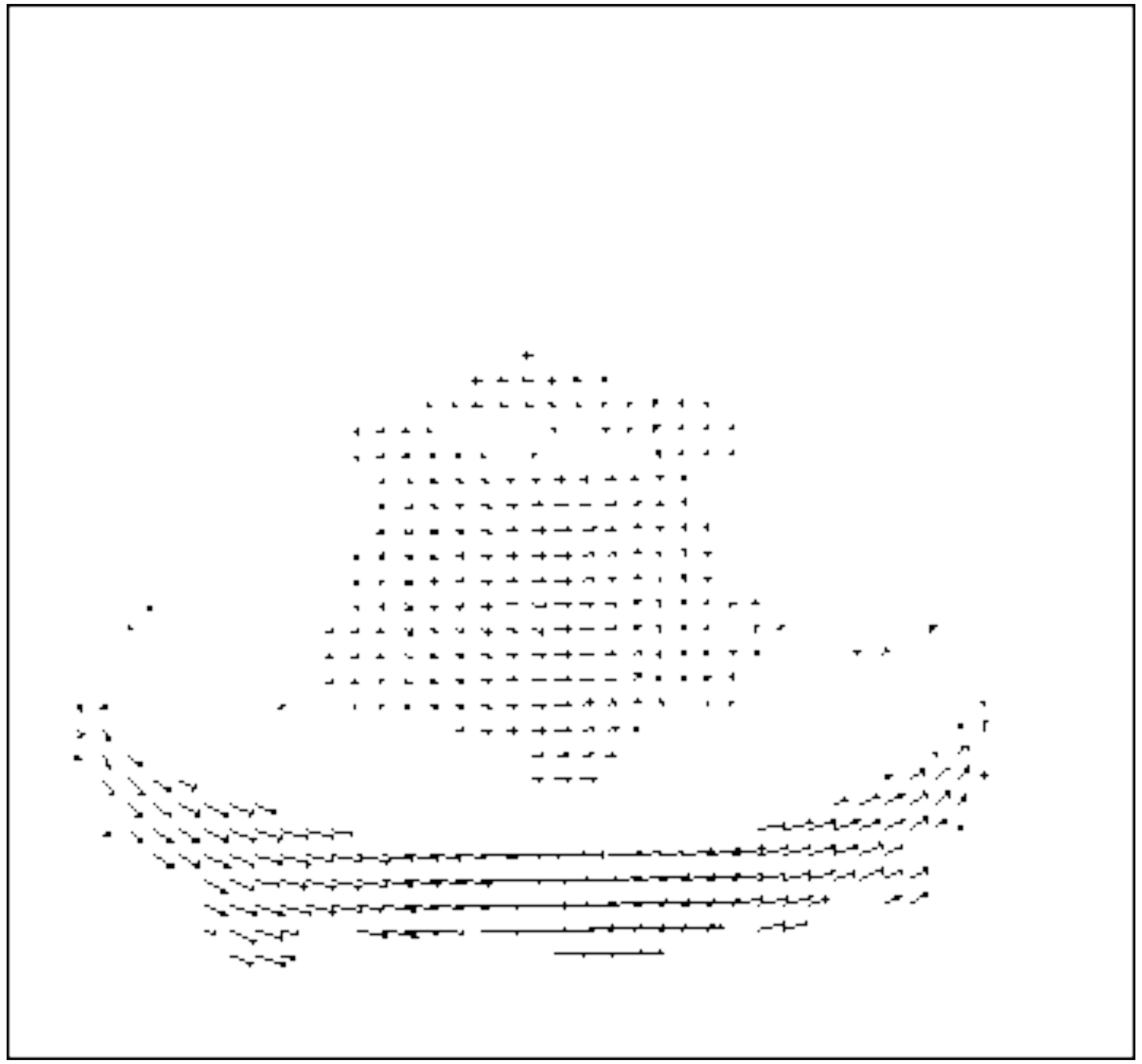
Applications

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

Motion Field

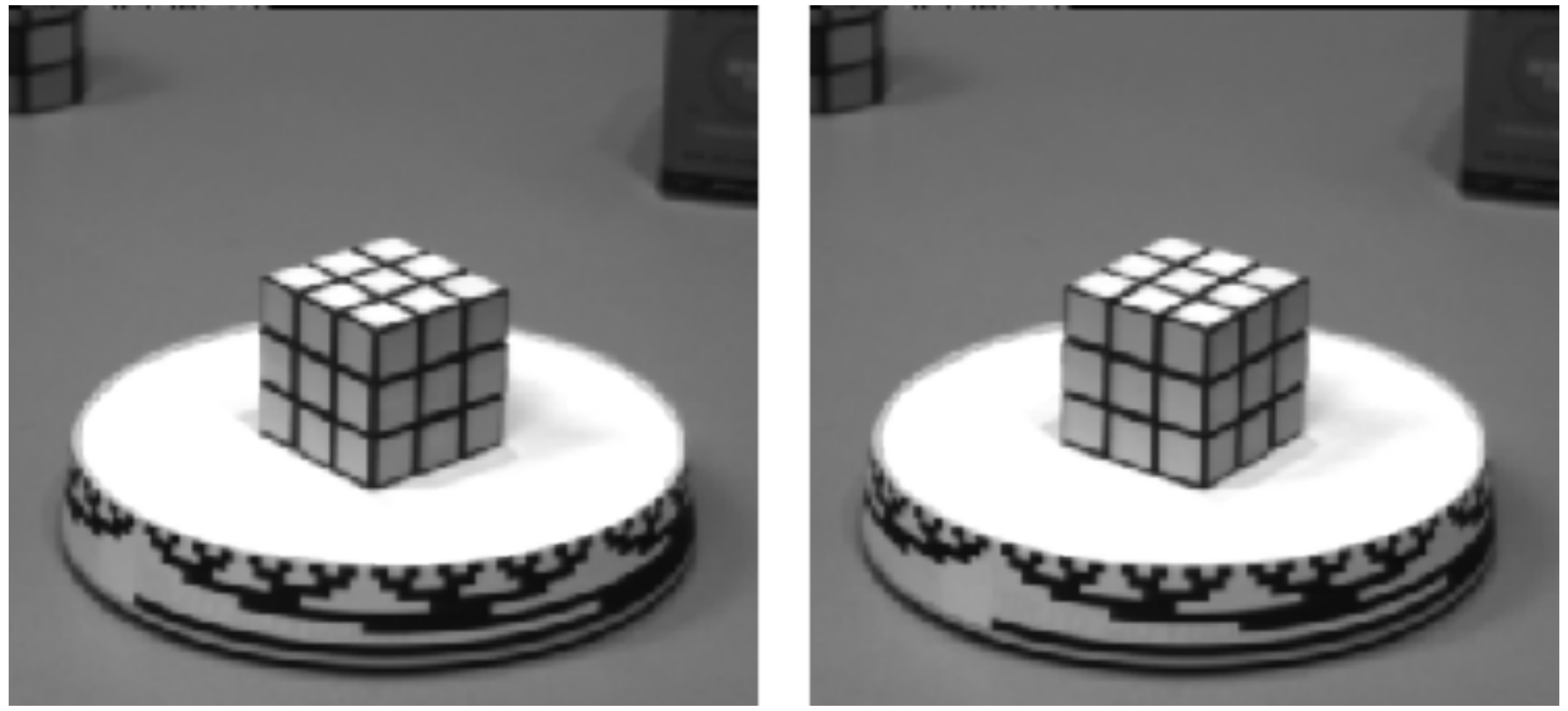
- The motion field is the projection of the 3D scene motion into the image

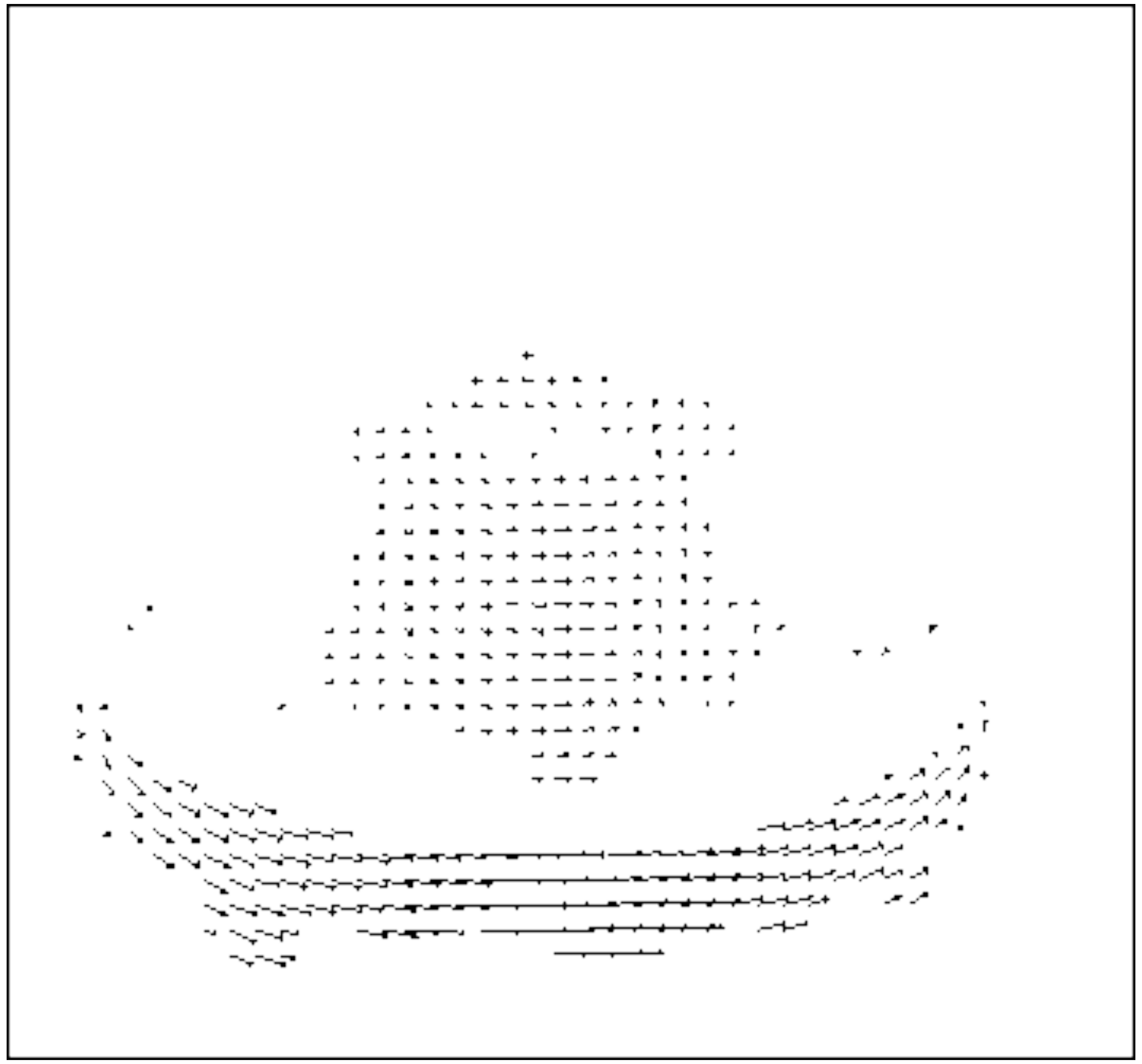




Motion Field

- **The motion field is the projection of the 3D scene motion into the image**





What would the motion field of a non-rotating ball moving towards the camera look like?

Apparent Motion

- Definition: optical flow is the *apparent* motion of brightness patterns in the image

Apparent Motion

- Definition: optical flow is the *apparent* motion of brightness patterns in the image
- Ideally, optical flow would be the same as the motion field

Apparent Motion

- Definition: optical flow is the *apparent* motion of brightness patterns in the image
- Ideally, optical flow would be the same as the motion field
- Have to be careful: apparent motion can be caused by lighting changes without any actual motion

Apparent Motion

- Definition: optical flow is the *apparent* motion of brightness patterns in the image
- Ideally, optical flow would be the same as the motion field
- Have to be careful: apparent motion can be caused by lighting changes without any actual motion
  - Think of a uniform rotating sphere under fixed lighting vs. a stationary sphere under moving illumination

Lucas-Kanade Optical Flow

- Same as Lucas-Kanade feature tracking, but for each pixel
  - As we saw, works better for textured pixels
- Operations can be done one frame at a time, rather than pixel by pixel
  - Efficient

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.* $u_i = u_i^* + u_i'$,

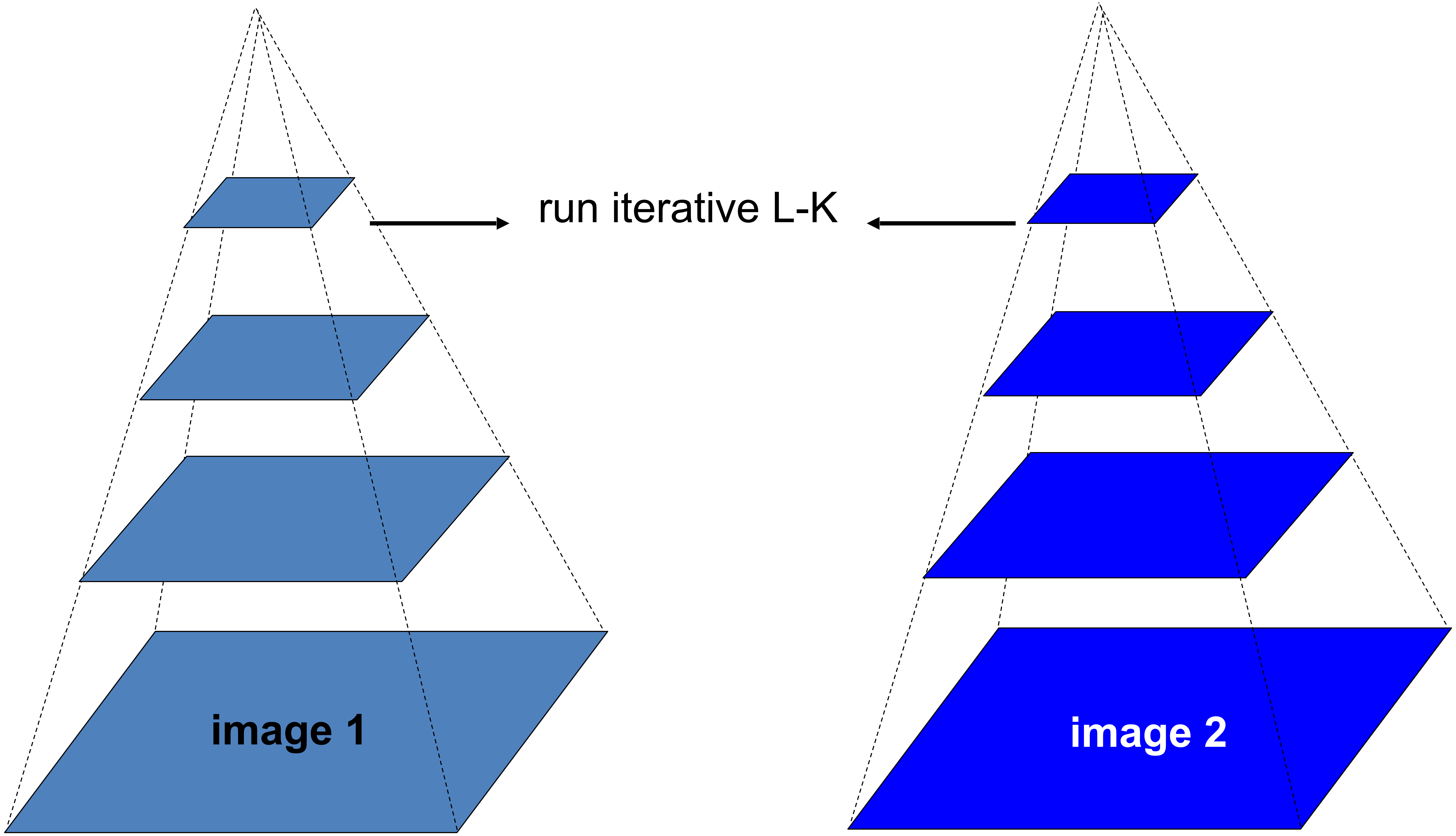
$v_i = v_i^* + v_i'$.

Iterative Refinement

- Iterative Lukas-Kanade Algorithm
  1. Estimate displacement at each pixel by solving Lucas-Kanade equations
  2. Warp I(t) towards I(t+1) using the estimated flow field
     - Basically, just interpolation
  3. Repeat until convergence

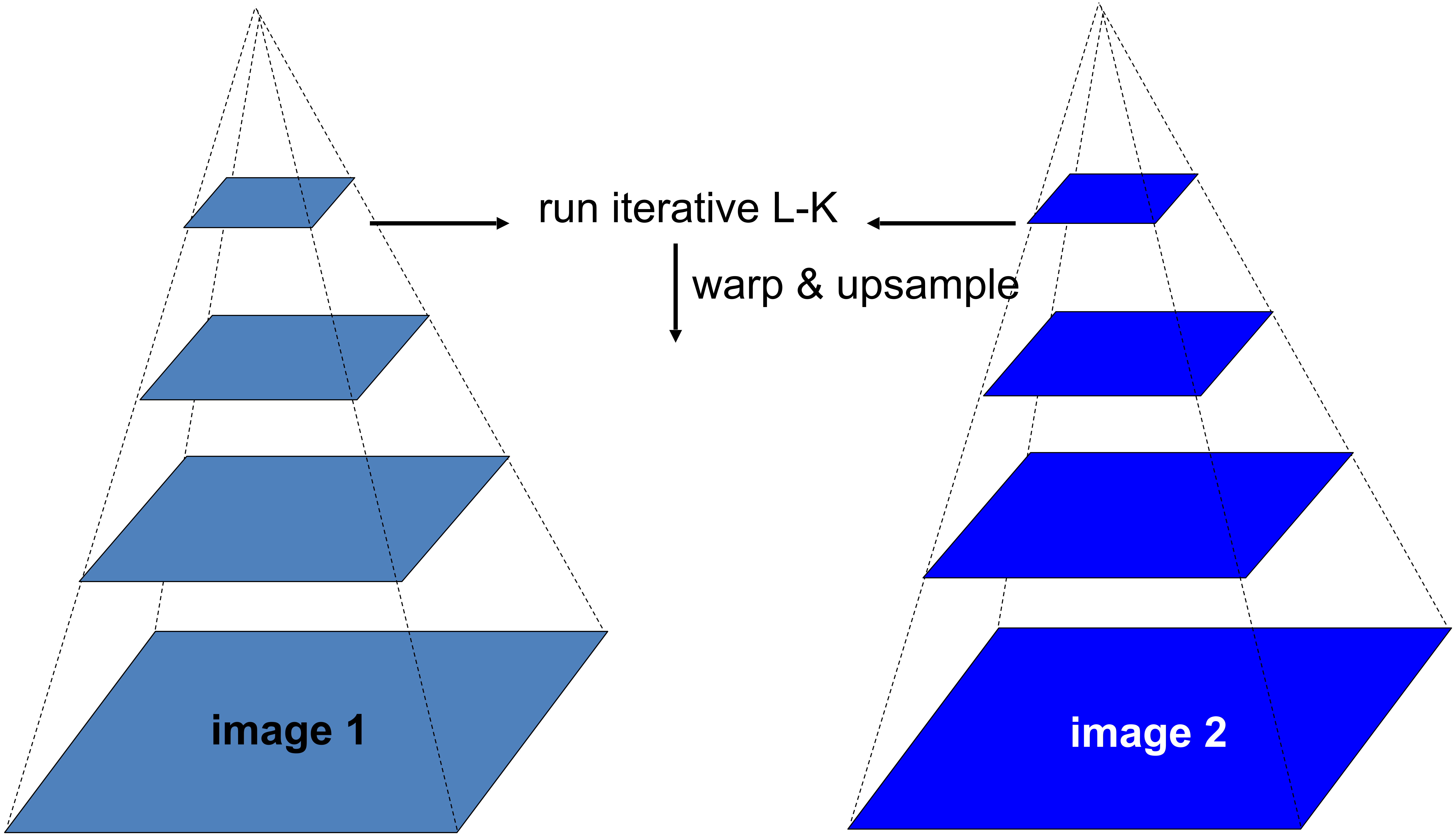
* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

Coarse-to-fine Optical Flow



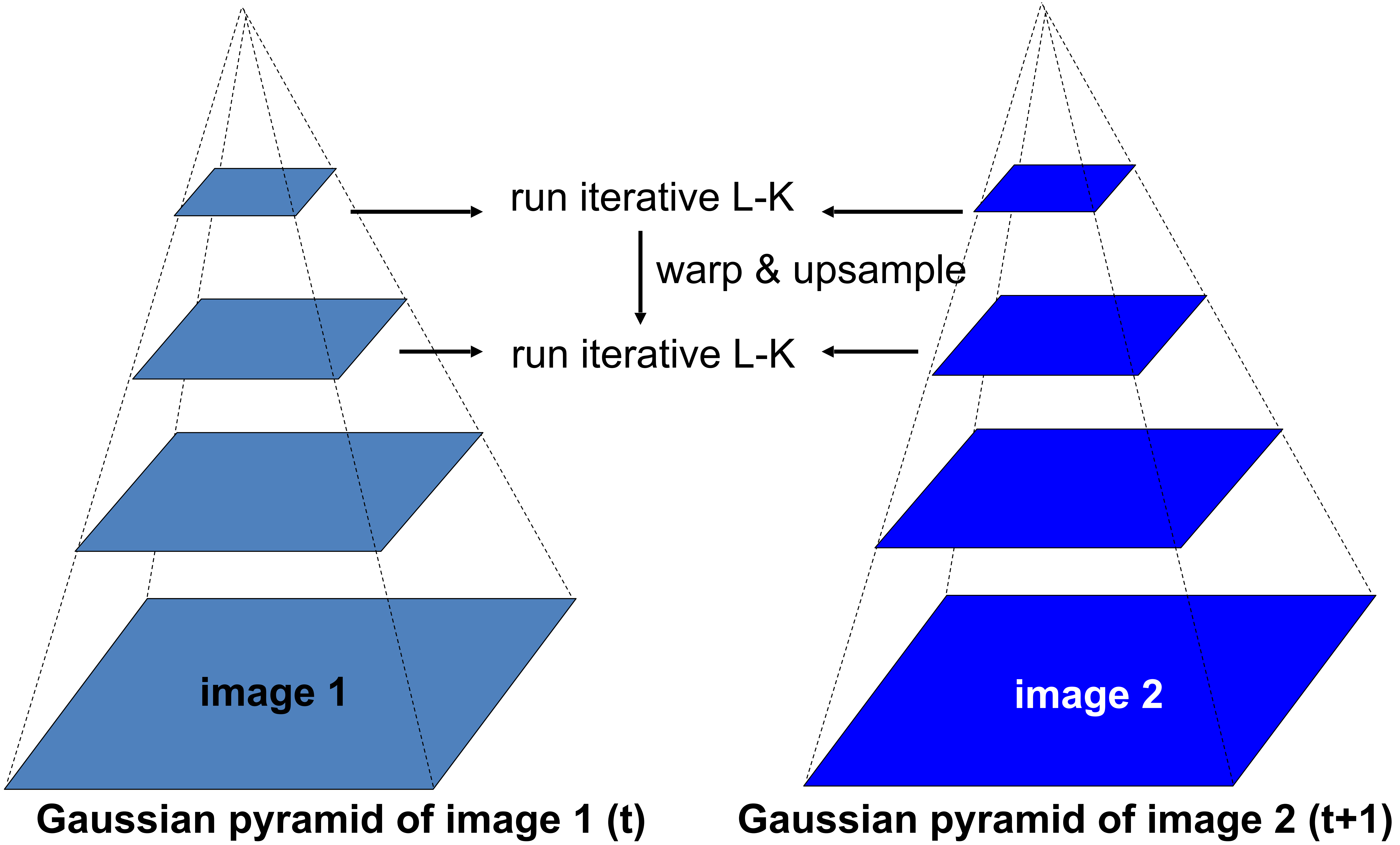
Gaussian pyramid of image 1 (t) **Gaussian pyramid of image 2 (t+1)**

Coarse-to-fine Optical Flow

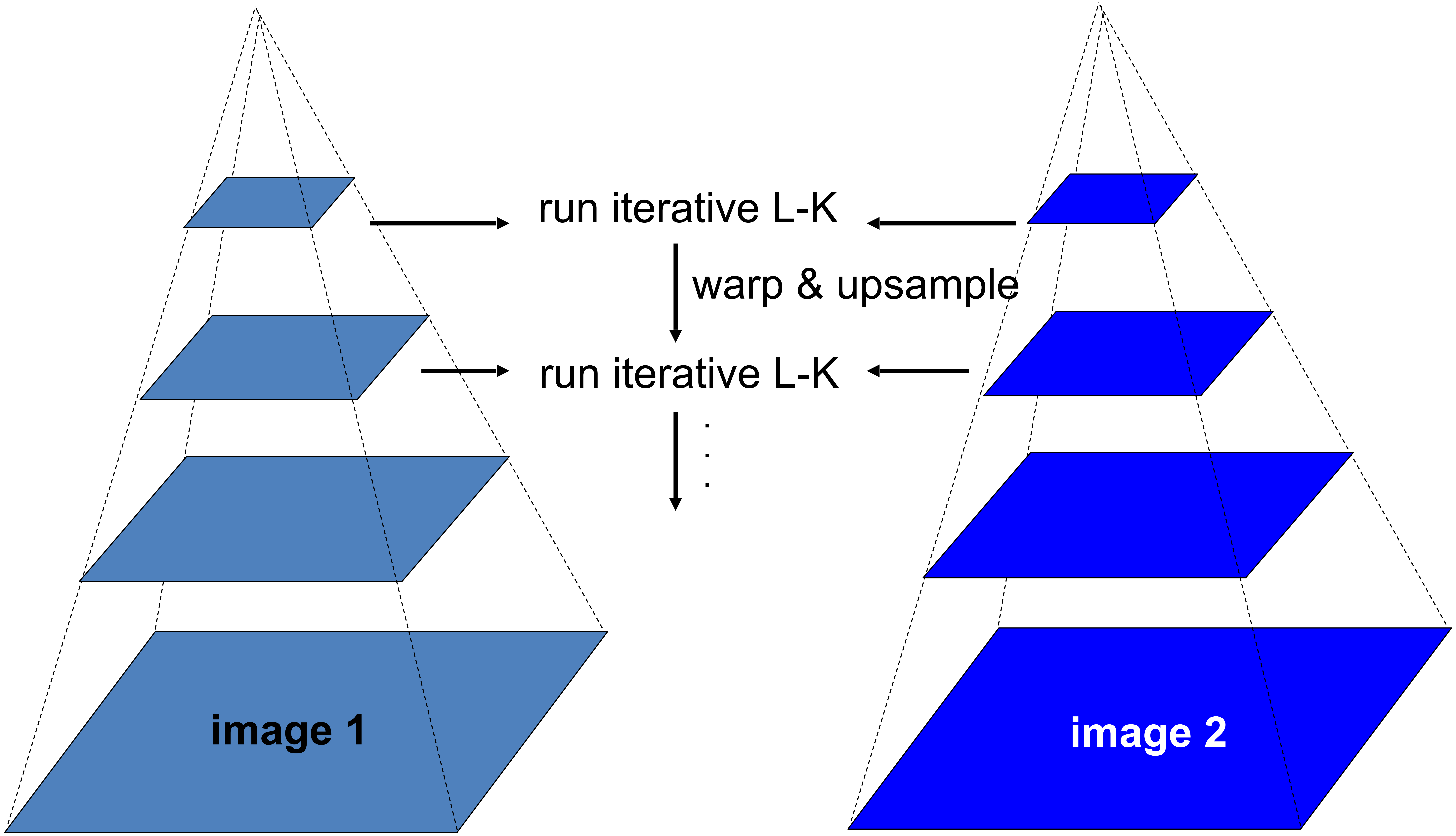


Gaussian pyramid of image 1 (t) **Gaussian pyramid of image 2 (t+1)**

Coarse-to-fine Optical Flow



Coarse-to-fine Optical Flow

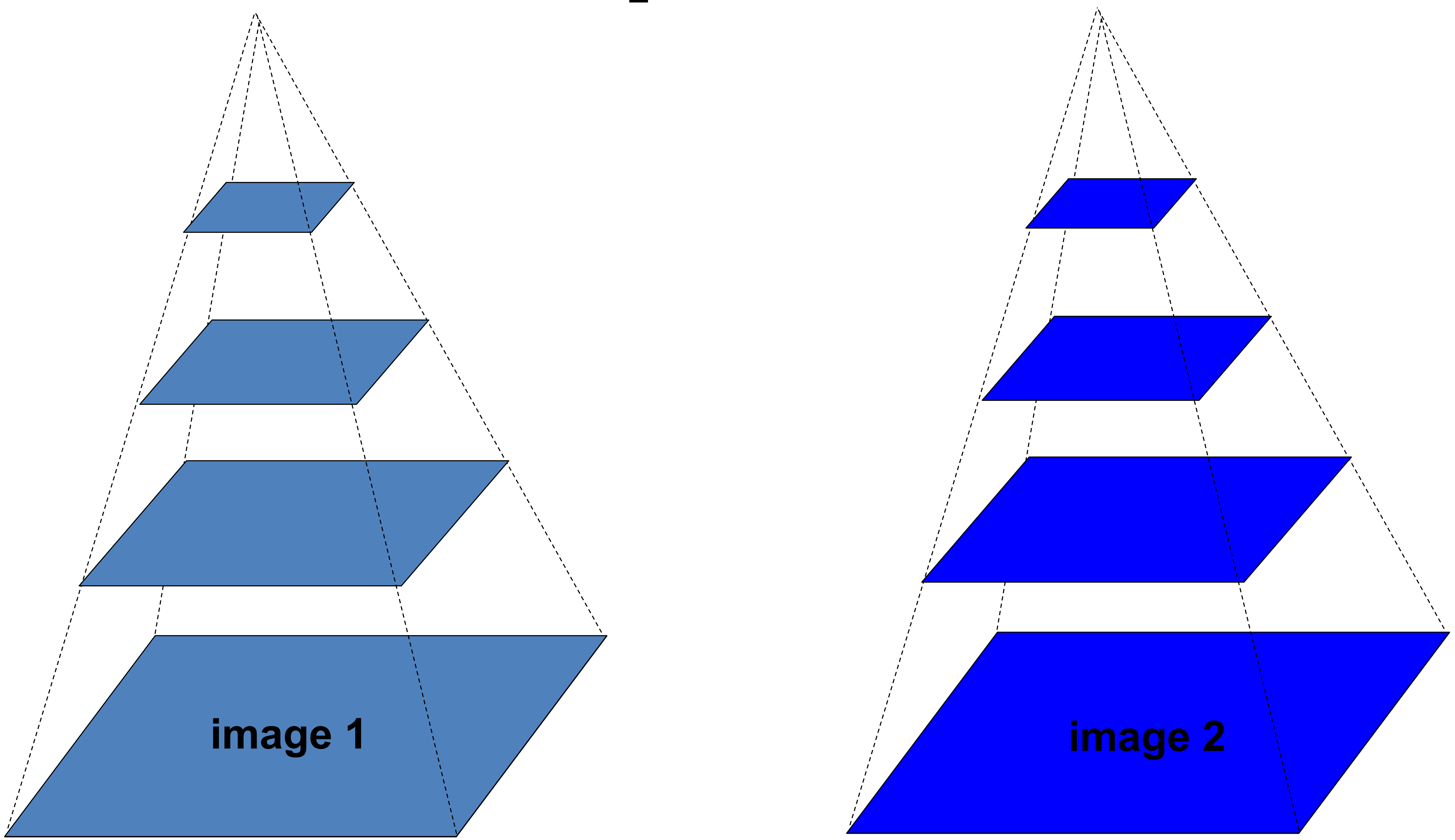


Gaussian pyramid of image 1 (t)

Gaussian pyramid of image 2 (t+1)

Coarse-to-fine Optical Flow

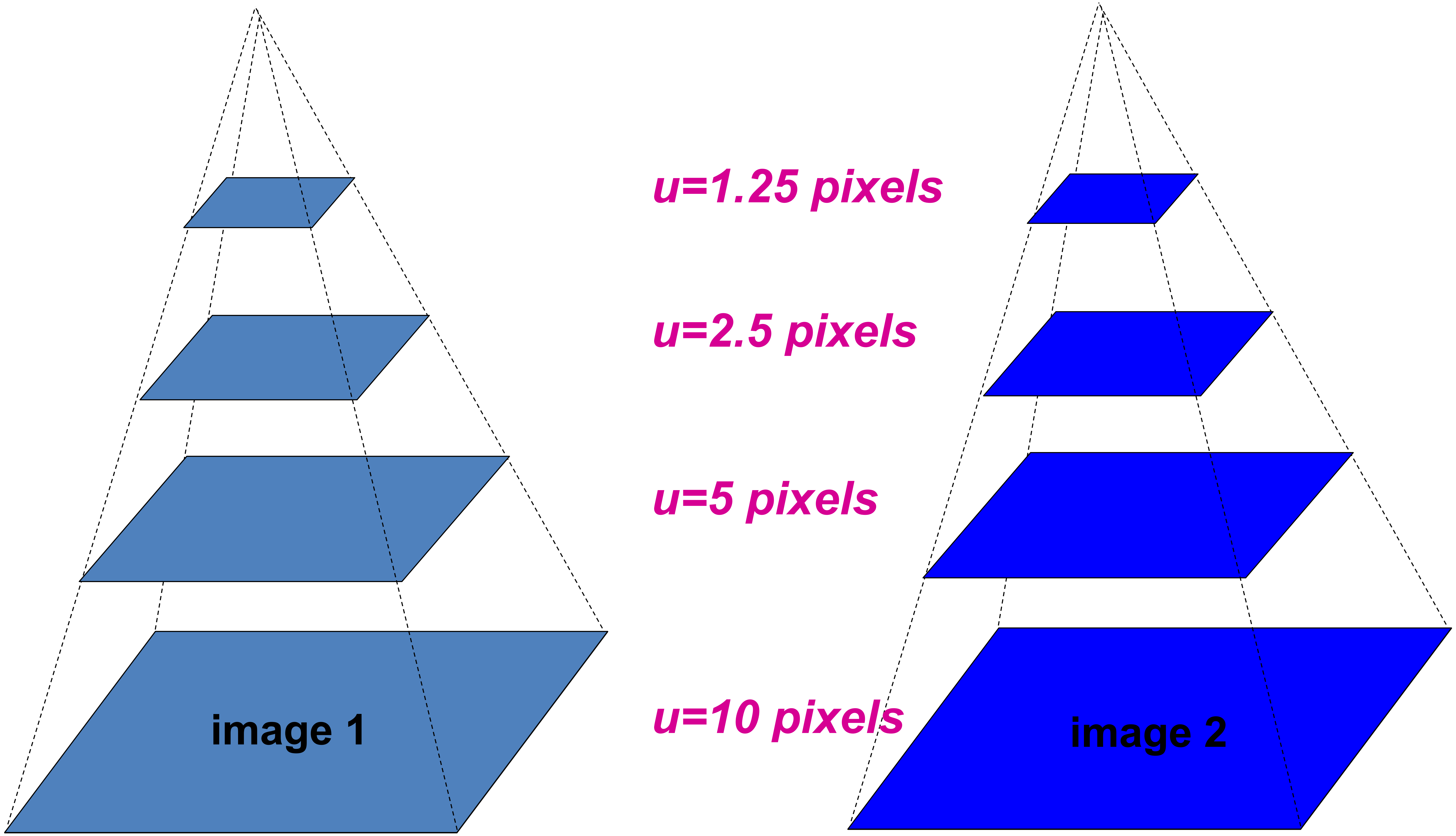
Coarse-to-fine Optical Flow



Gaussian pyramid of image 1

Gaussian pyramid of image 2

Coarse-to-fine Optical Flow



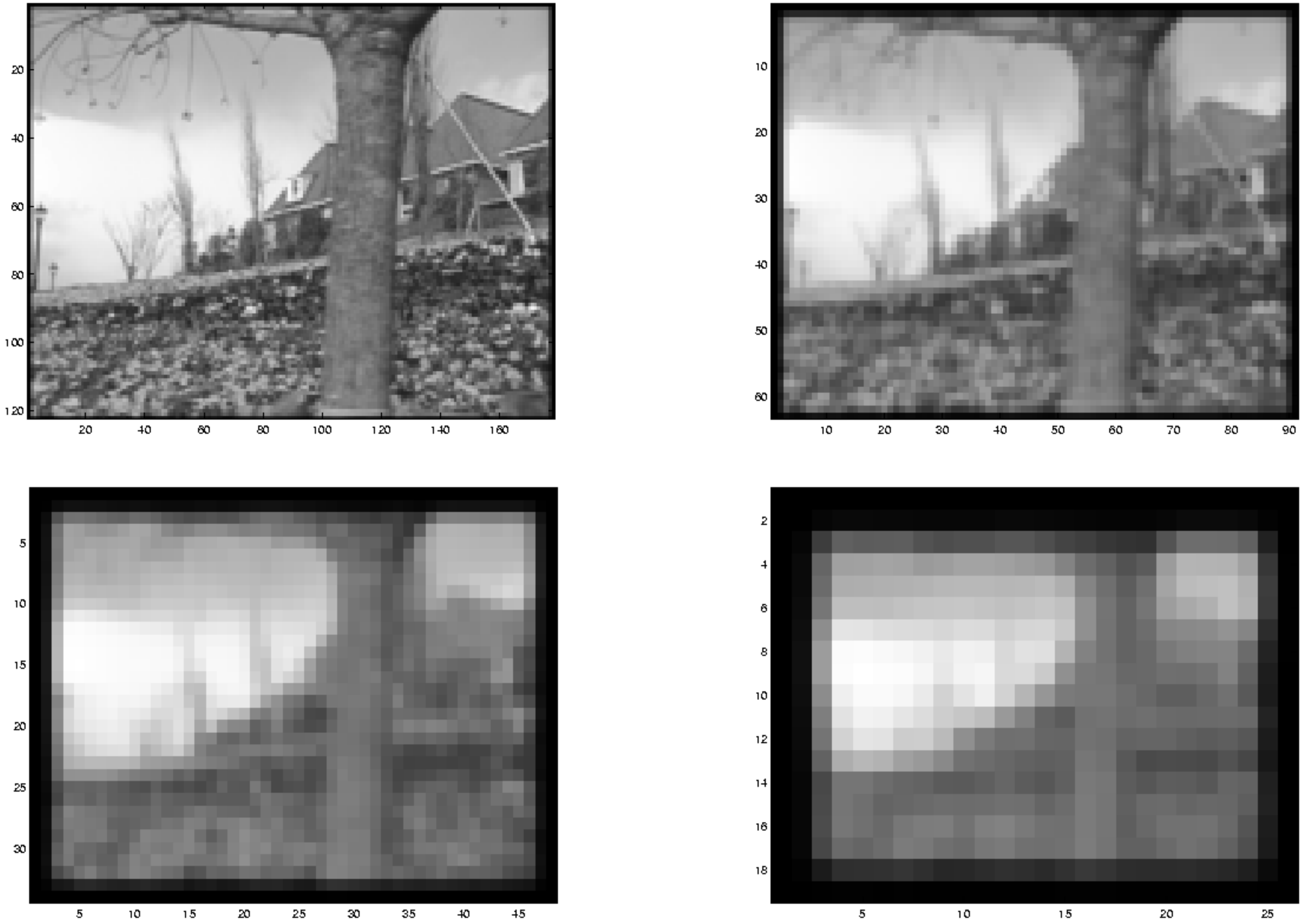
Gaussian pyramid of image 1

Gaussian pyramid of image 2

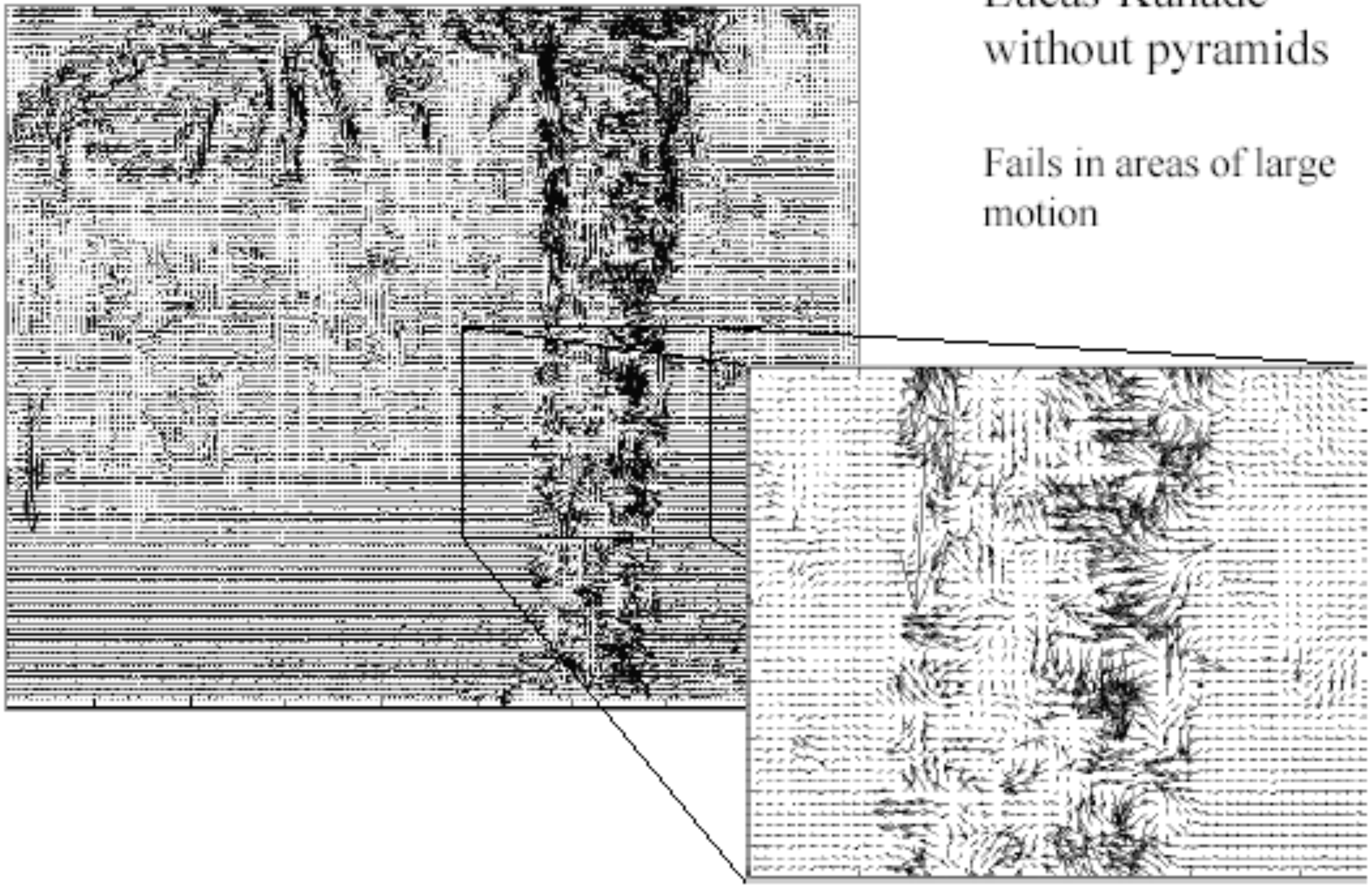
Example

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

Multi-resolution registration



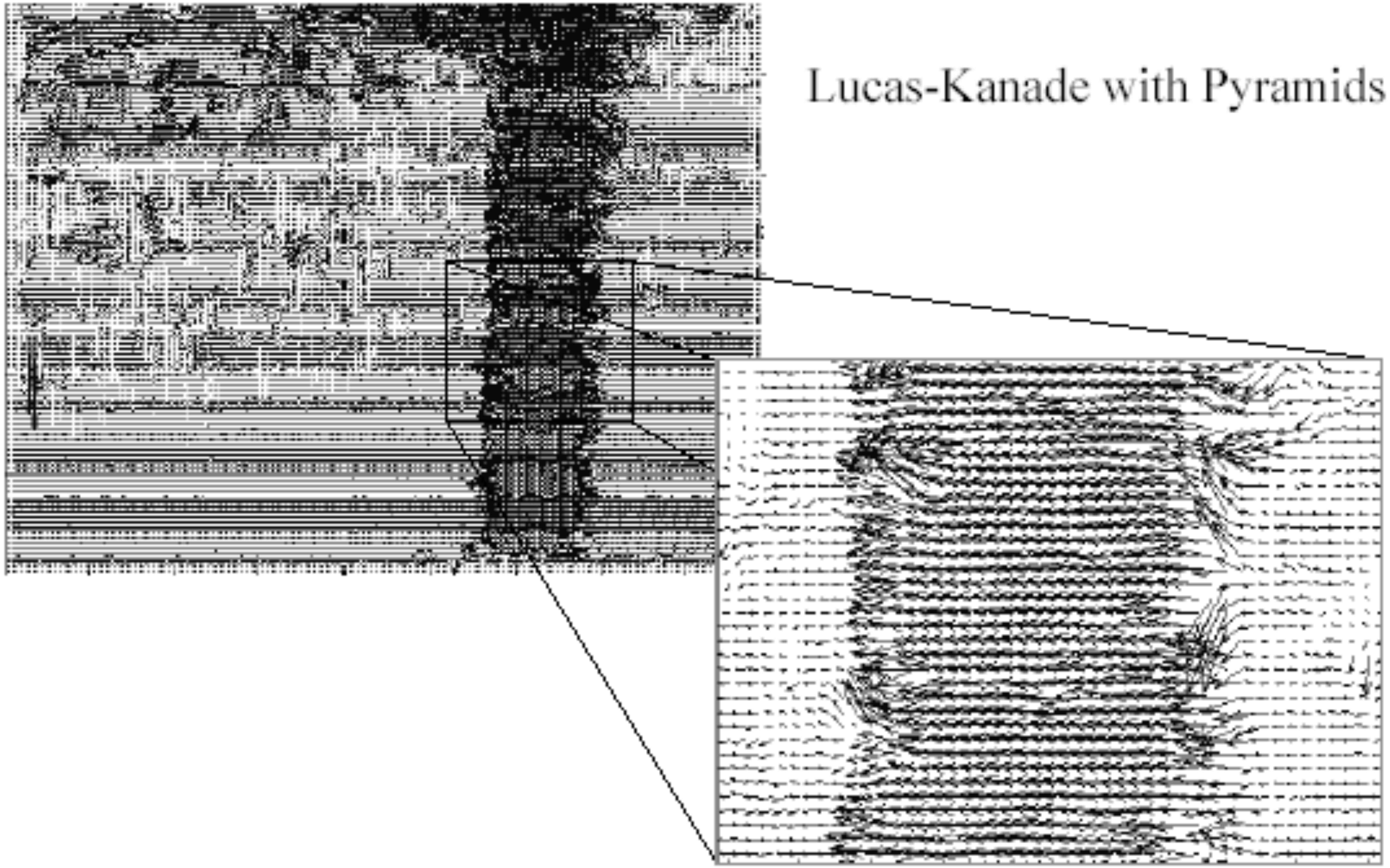
Optical Flow Results



Lucas-Kanade without pyramids

Fails in areas of large motion

Optical Flow Results



Errors in Lucas-Kanade

Errors in Lucas-Kanade

- The motion is large

Errors in Lucas-Kanade

- The motion is large
  - Possible Fix: Keypoint matching

Errors in Lucas-Kanade

- The motion is large
  - Possible Fix: Keypoint matching
- A point does not move like its neighbors

Errors in Lucas-Kanade

- **The motion is large**
  - **Possible Fix: Keypoint matching**
- **A point does not move like its neighbors**
  - **Possible Fix: Region-based matching**

Errors in Lucas-Kanade

- The motion is large
  - Possible Fix: Keypoint matching
- A point does not move like its neighbors
  - Possible Fix: Region-based matching
- Brightness constancy does not hold

Errors in Lucas-Kanade

- The motion is large
  - Possible Fix: Keypoint matching
- A point does not move like its neighbors
  - Possible Fix: Region-based matching
- Brightness constancy does not hold
  - Possible Fix: Gradient constancy

Summary

- Major contributions from Lucas, Tomasi, Kanade
  - Tracking feature points
  - Optical flow
- Key ideas
  - By assuming brightness constancy, truncated Taylor expansion leads to simple and fast patch matching across frames
  - Coarse-to-fine registration