Computer Vision

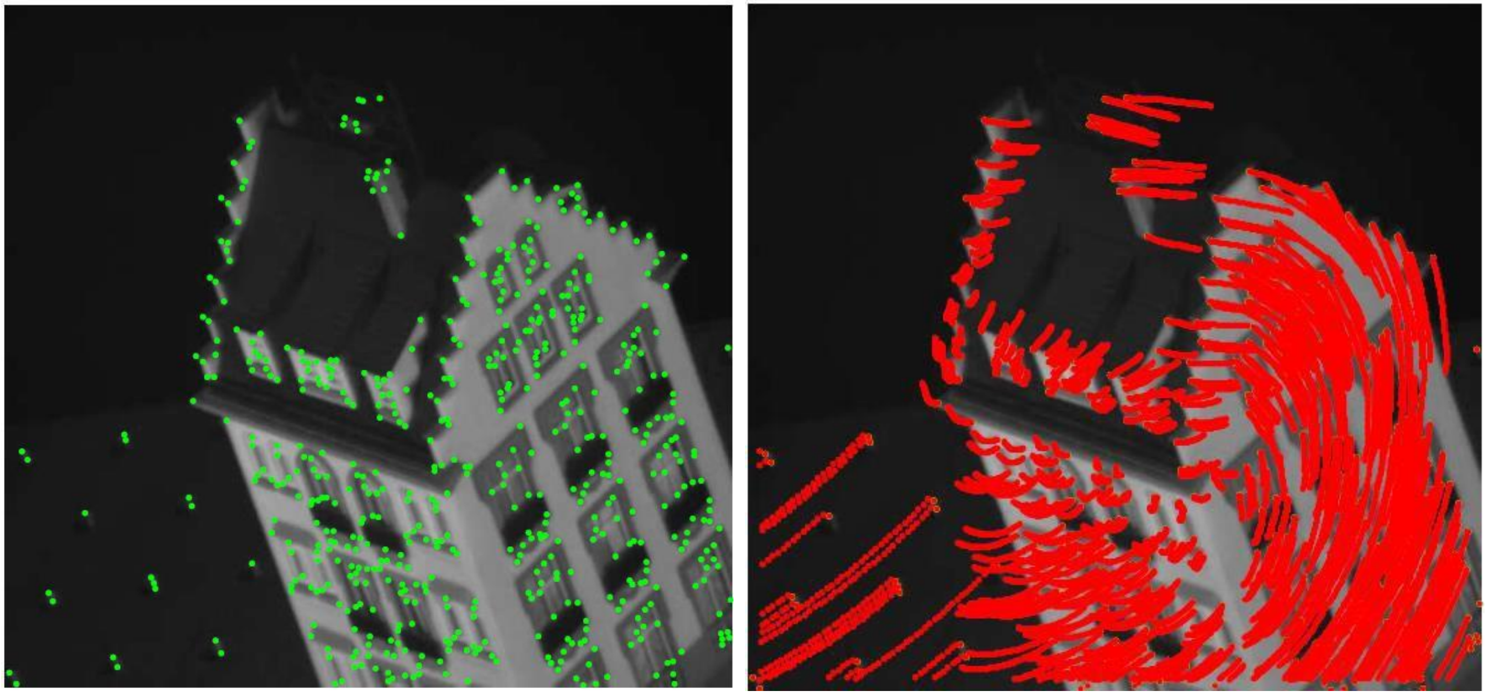
Feature Tracking and Optical Flow

Dr. Mrinmoy Ghorai

Indian Institute of Information Technology

Sri City, Chittoor

Feature Tracking and Optical Flow



This class: 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

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*, 1981.

Feature tracking - Challenges

- Figure out which features can be tracked

Feature tracking - Challenges

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

Feature tracking - Challenges

- 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.)

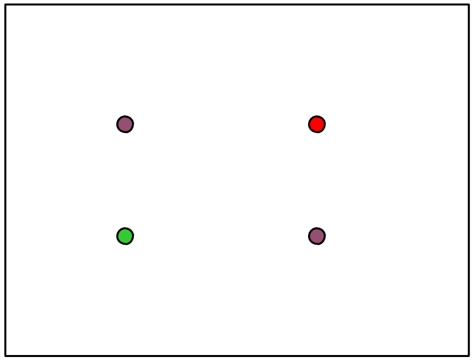
Feature tracking - Challenges

- 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.)
- Points may appear or disappear: need to be able to add/delete tracked points

Feature tracking - Challenges

- 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.)
- Points may appear or disappear: need to be able to add/delete tracked points
- Drift: small errors can accumulate as appearance model is updated

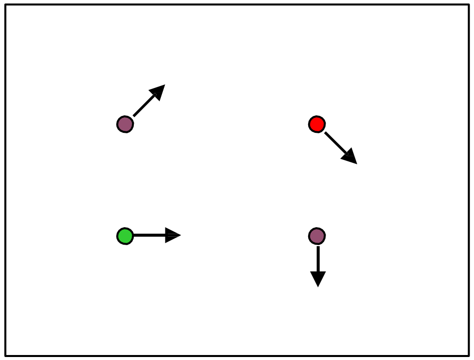
Feature tracking: Lucas-Kanade



$I(x,y,t)$

- Given two subsequent frames, estimate the point translation

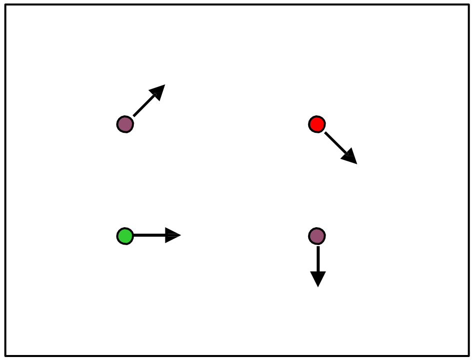
Feature tracking: Lucas-Kanade



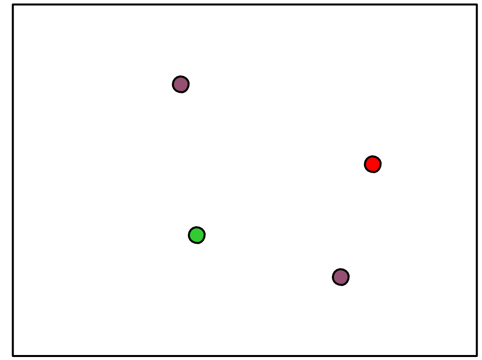
$I(x,y,t)$

- Given two subsequent frames, estimate the point translation

Feature tracking: Lucas-Kanade



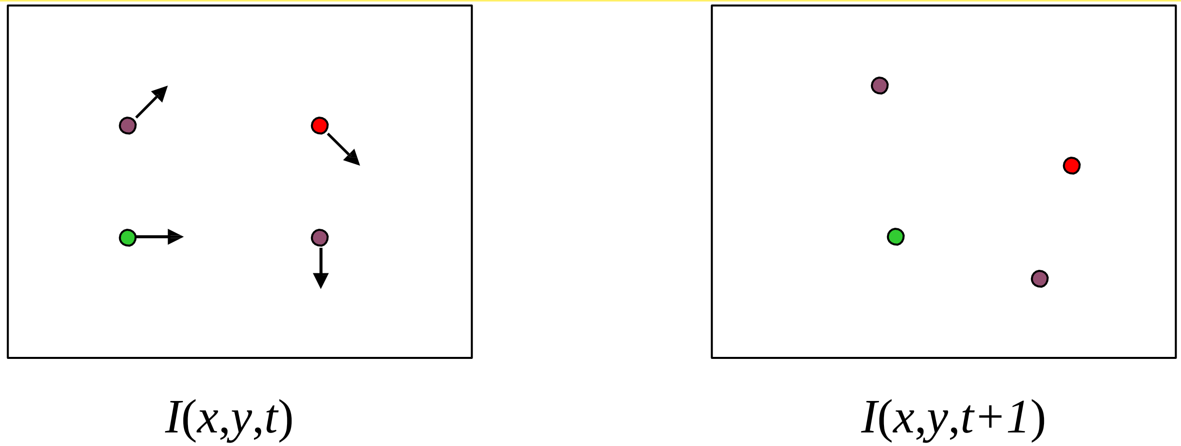
$I(x,y,t)$



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

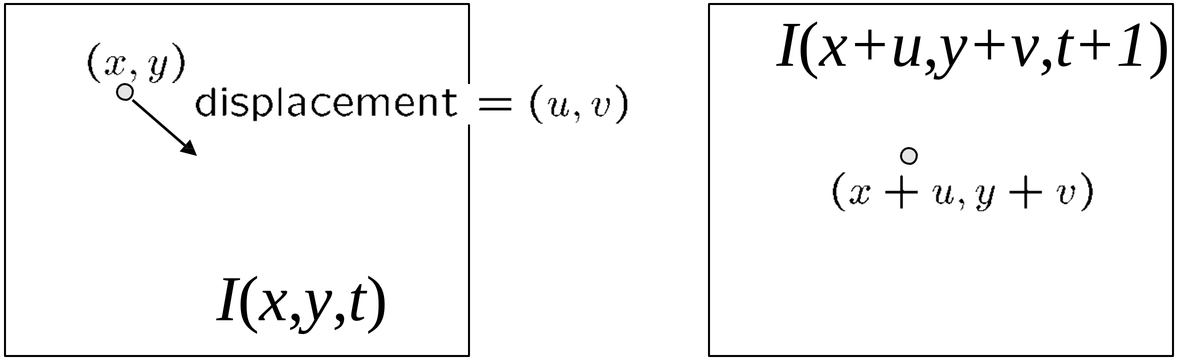
- Given two subsequent frames, estimate the point translation

Feature tracking: Lucas-Kanade



- Given two subsequent frames, estimate the point translation
- 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

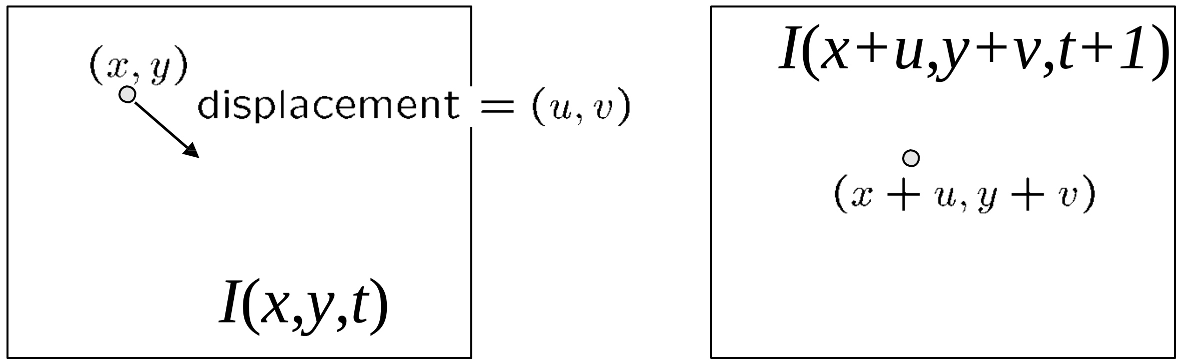
The brightness constancy constraint



- Brightness Constancy Equation:

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

The brightness constancy constraint



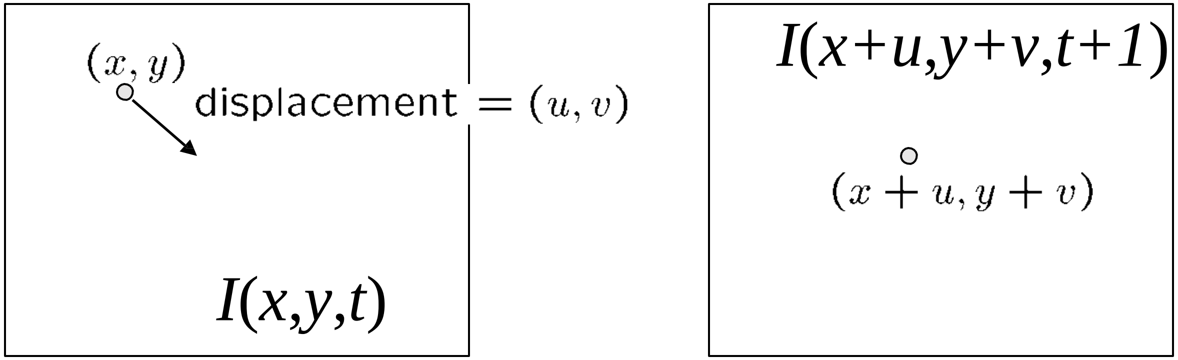
- Brightness Constancy Equation:

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

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

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

The brightness constancy constraint



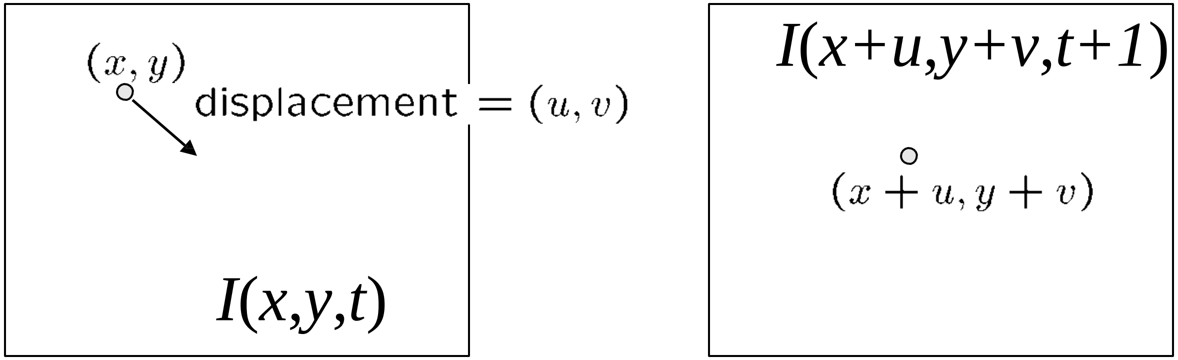
- Brightness Constancy Equation:

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

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

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

The brightness constancy constraint



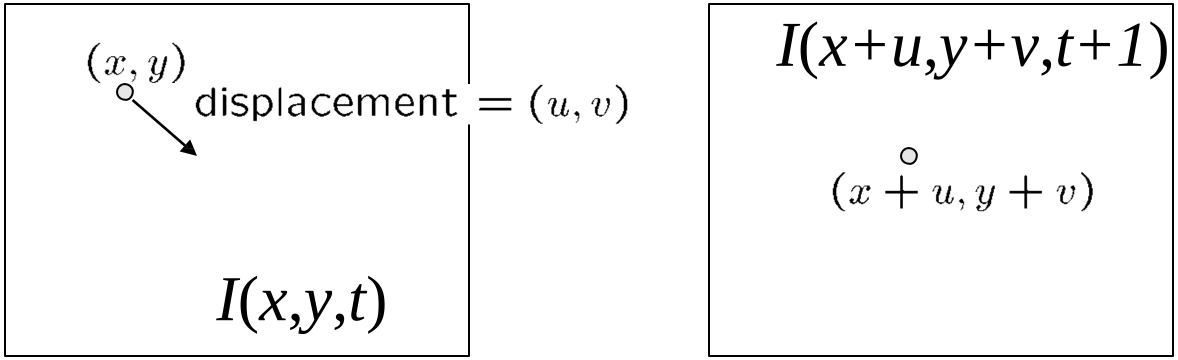
- Brightness Constancy Equation:

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

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

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

The brightness constancy constraint



- Brightness Constancy Equation:

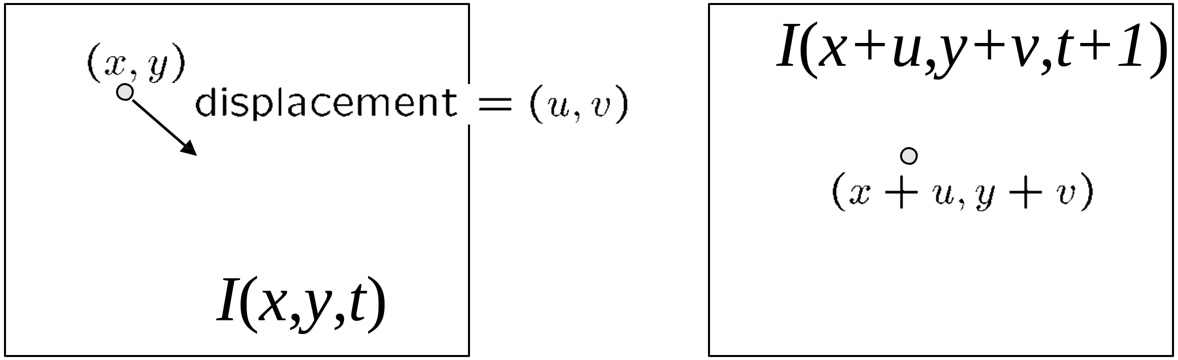
$$I(x, y, t) = I(x+u, y+v, t+1)$$

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

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

derivative along t i.e., difference over frames

The brightness constancy constraint



- 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:

derivative along x

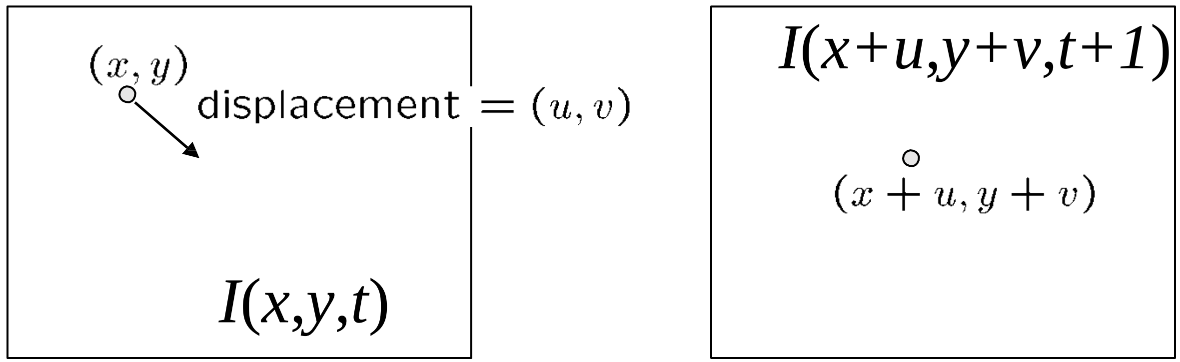
derivative along y

derivative along t i.e., difference over frames

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

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

The brightness constancy constraint



- 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:

derivative along x

derivative along y

derivative along t i.e., difference over frames

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

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

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

The brightness constancy constraint

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

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

The brightness constancy constraint

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

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

- How many equations and unknowns per pixel?

The brightness constancy constraint

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

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

- How many equations and unknowns per pixel?
    - One equation (this is a scalar equation!), two unknowns (u,v)

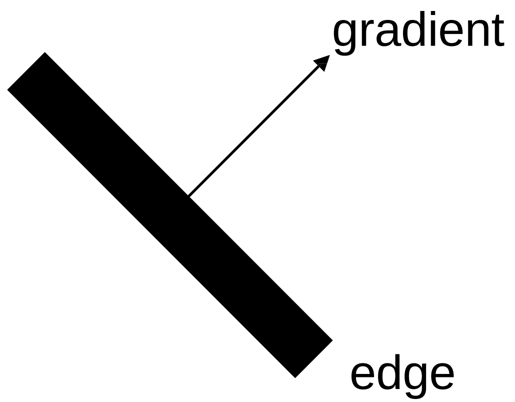
The brightness constancy constraint

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

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

- How many equations and unknowns per 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



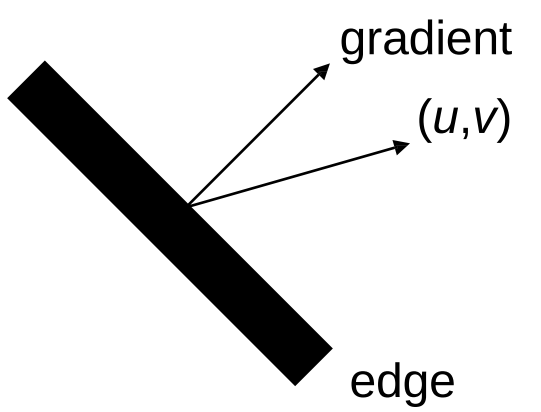
The brightness constancy constraint

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

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

- How many equations and unknowns per 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



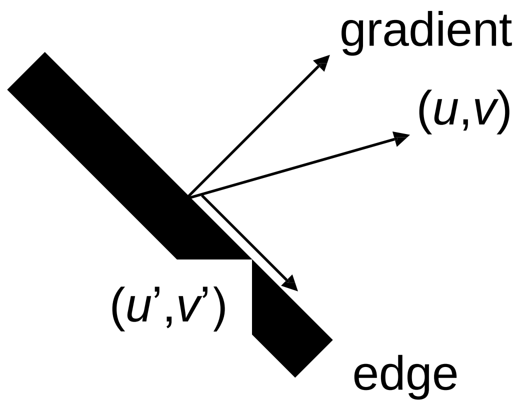
The brightness constancy constraint

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

$$\nabla \mathrm{I} \cdot [\mathrm{u} \;\; \mathrm{v}]^{\mathrm{T}} + \mathrm{I}_{\mathrm{t}} = 0$$

- How many equations and unknowns per 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



The brightness constancy constraint

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

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

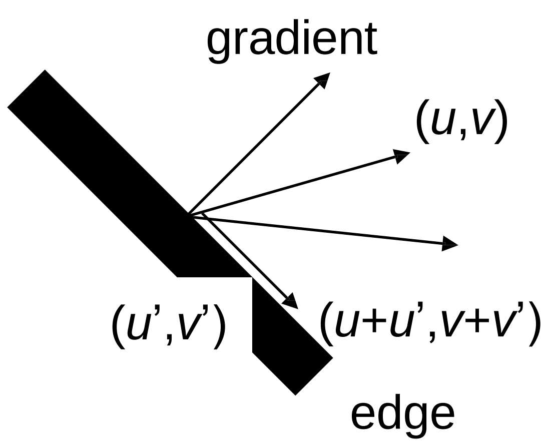
- How many equations and unknowns per pixel?

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

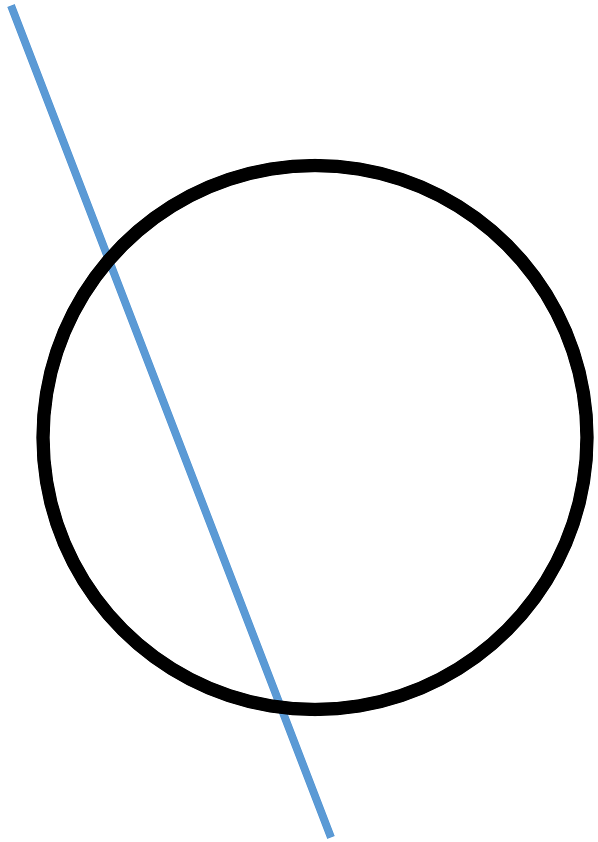
The component of the motion perpendicular to the gradient (i.e., parallel to the edge) cannot be measured

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

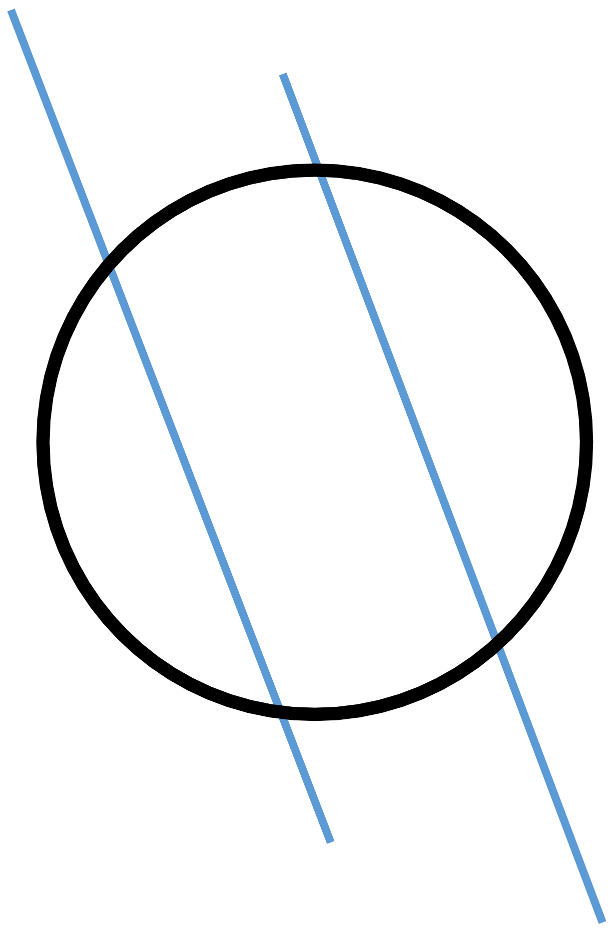
$$\nabla I \cdot [u' \;\; v']^T = 0$$



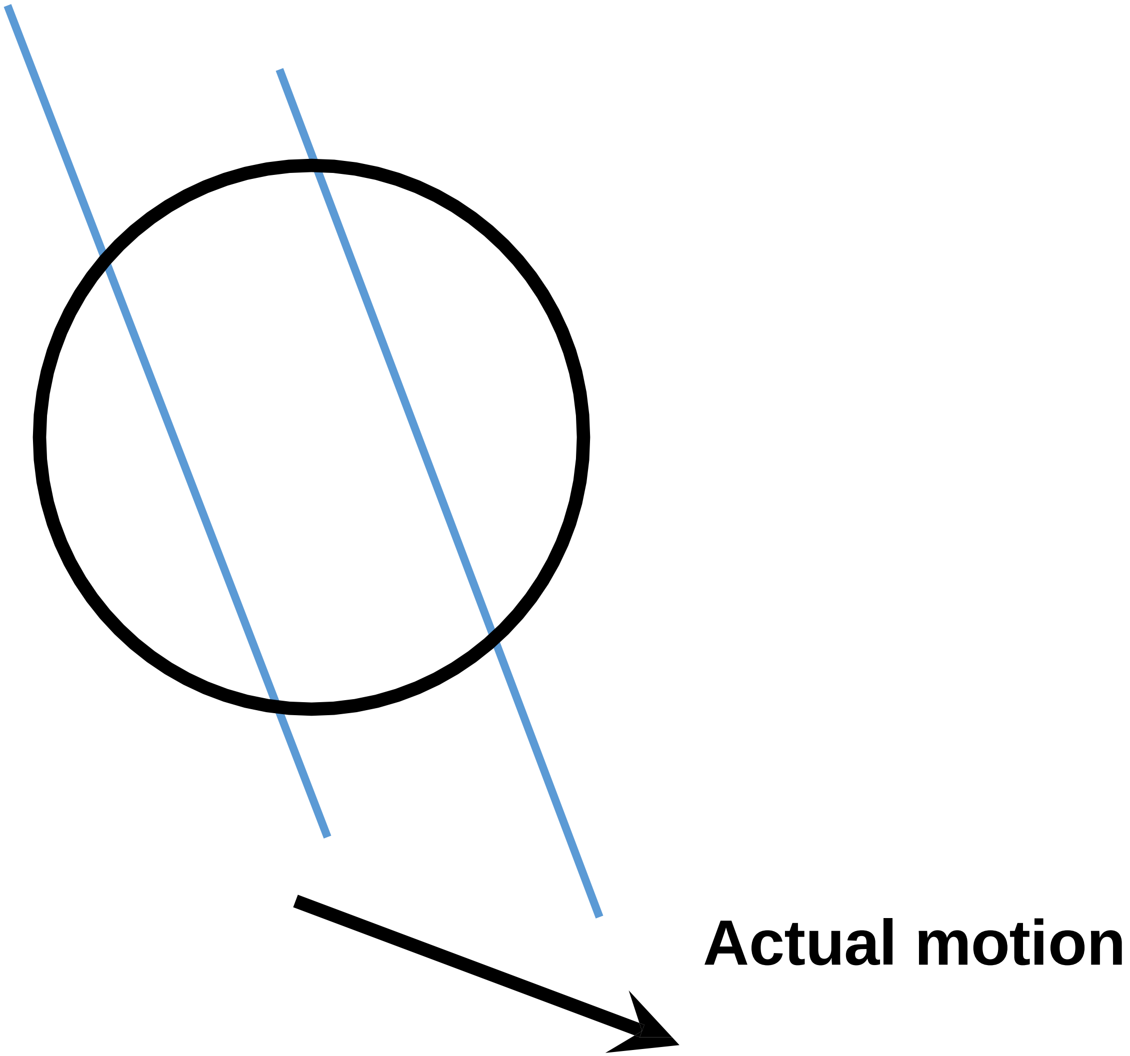
The aperture problem



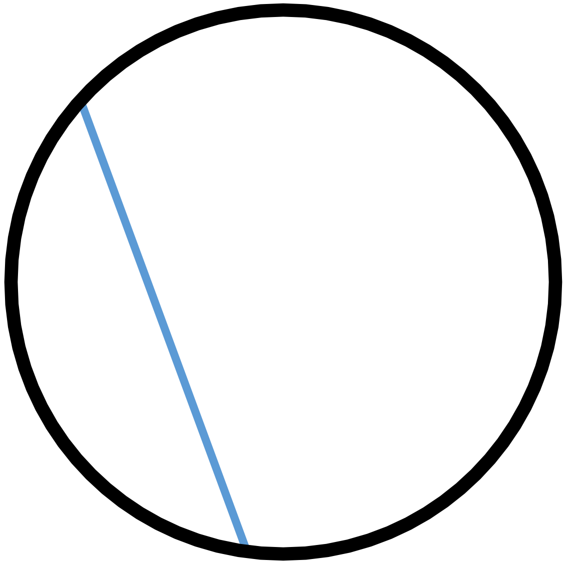
The aperture problem



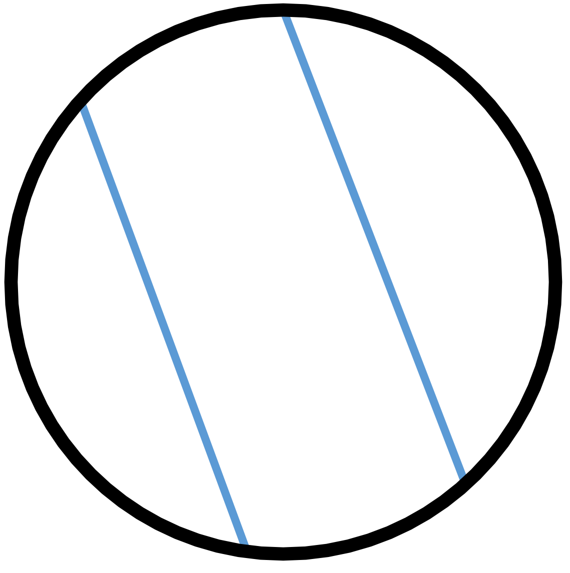
The aperture problem



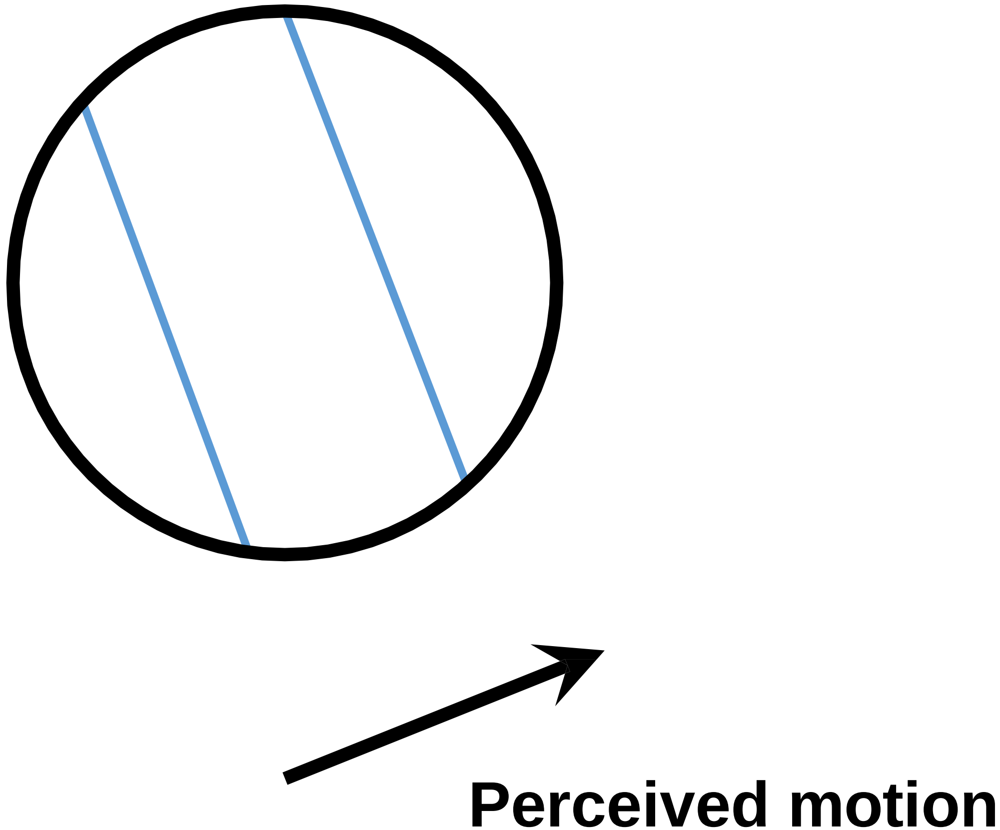
The aperture problem



The aperture problem



The aperture problem



Solving 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.

Solving 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?

Solving 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

Solving 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

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

Solving 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

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

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

Matching patches across images

- Overconstrained linear system

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

$$\underset{25\text{x}2}{A} \; \underset{2\text{x}1}{d} = \underset{25\text{x}1}{b}$$

Matching patches across images

- Overconstrained linear system

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

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

Matching patches across images

- Overconstrained linear system

$$\begin{bmatrix} I_x(\mathbf{p}_1) & I_y(\mathbf{p}_1) \\ I_x(\mathbf{p}_2) & I_y(\mathbf{p}_2) \\ \vdots & \vdots \\ I_x(\mathbf{p}_{25}) & I_y(\mathbf{p}_{25}) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(\mathbf{p}_1) \\ I_t(\mathbf{p}_2) \\ \vdots \\ I_t(\mathbf{p}_{25}) \end{bmatrix}$$

$$\underset{25\text{x}2}{A} \quad \underset{2\text{x}1}{d} = \underset{25\text{x}1}{b}$$

Least squares solution for *d* is 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

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

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

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?

- $\mathbf{A^T A}$ 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?

- $\mathbf{A^T A}$ 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

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)

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?

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

Low-texture region

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

– gradients have small magnitude

– small λ_1, small λ_2

Edge

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

– gradients very large or very small

– large λ_1, small λ_2

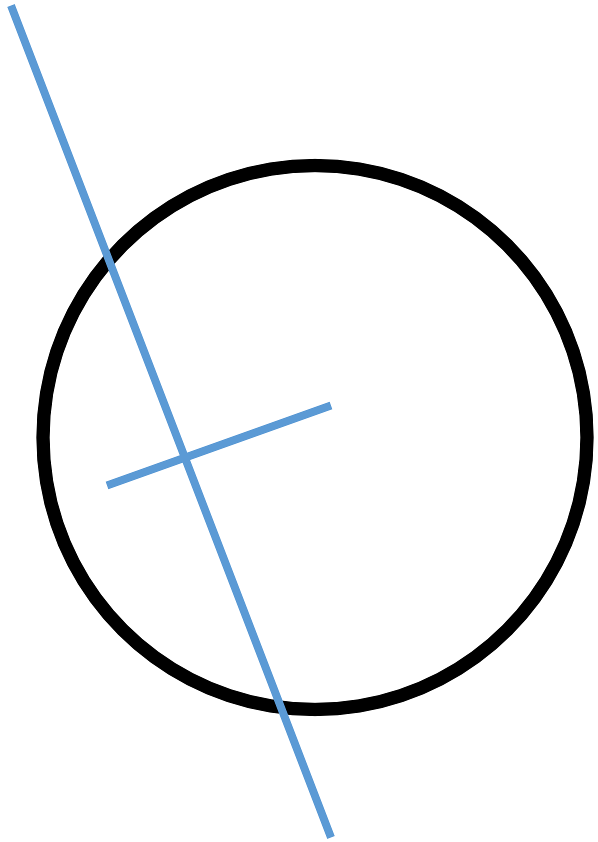
High-texture region

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

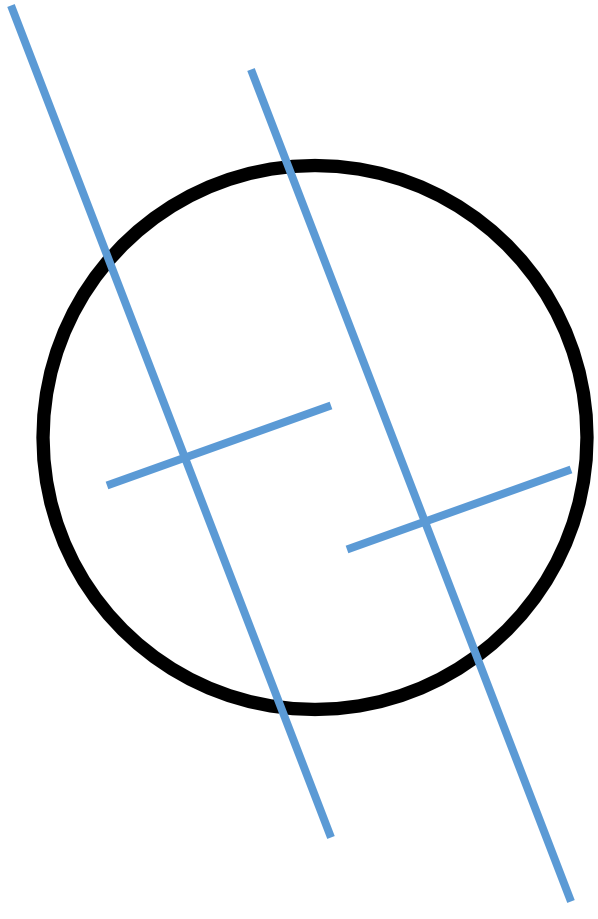
– gradients are different, large magnitudes

– large λ_1, large λ_2

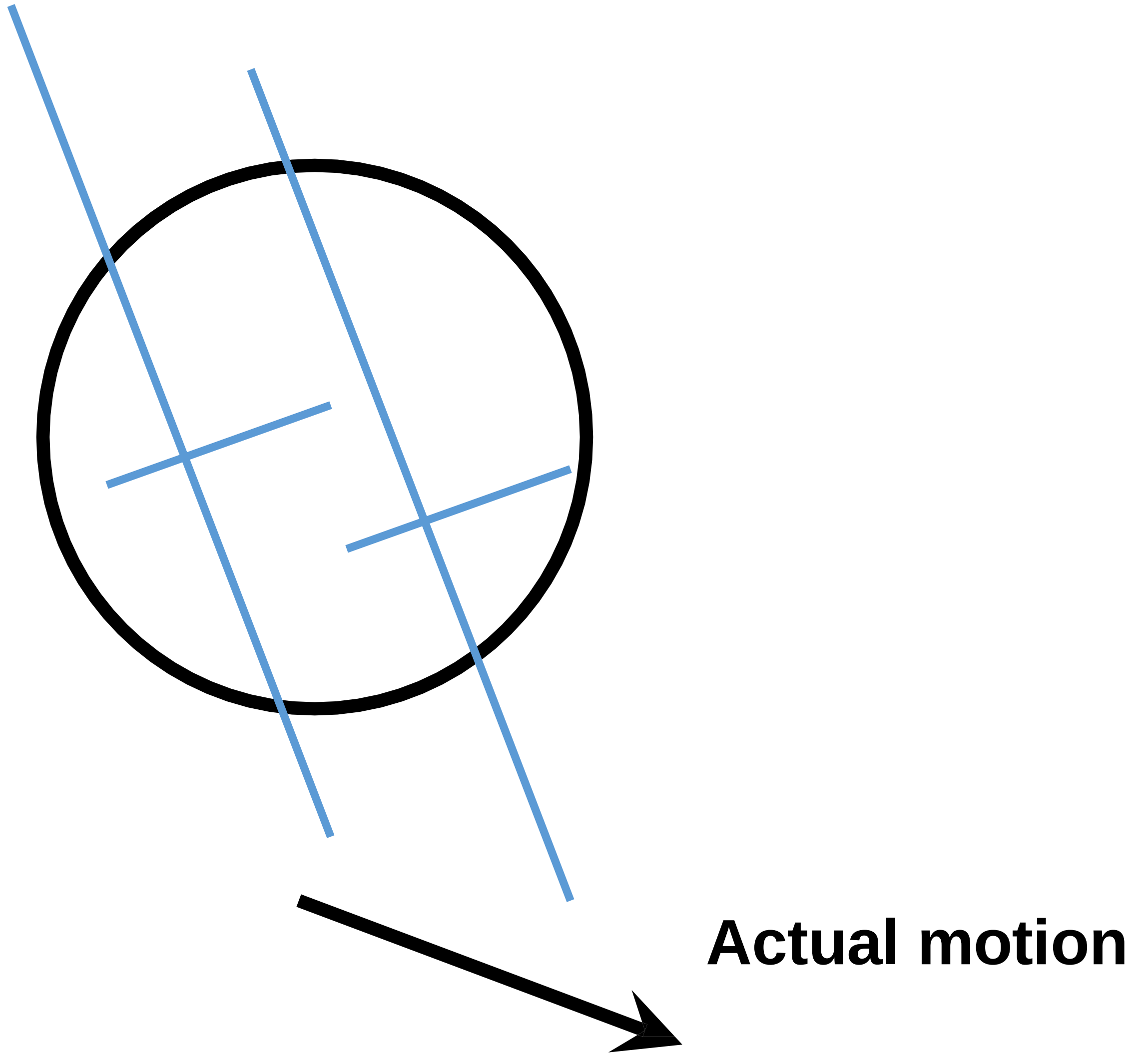
The aperture problem resolved



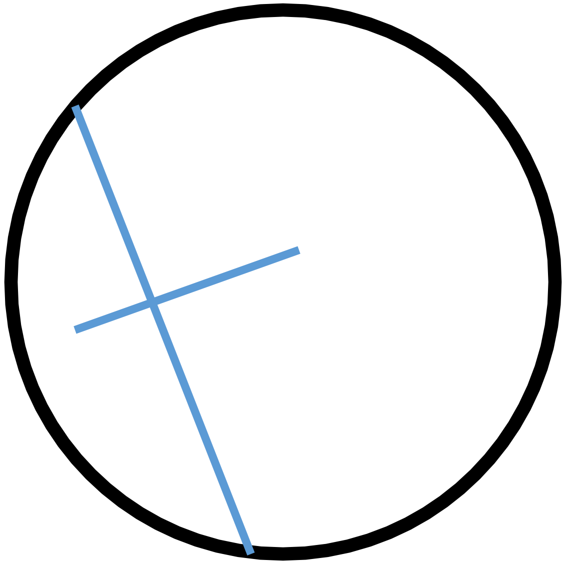
The aperture problem resolved



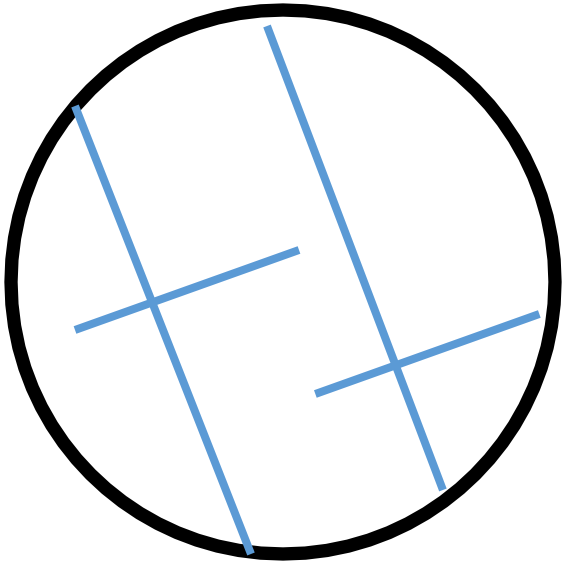
The aperture problem resolved



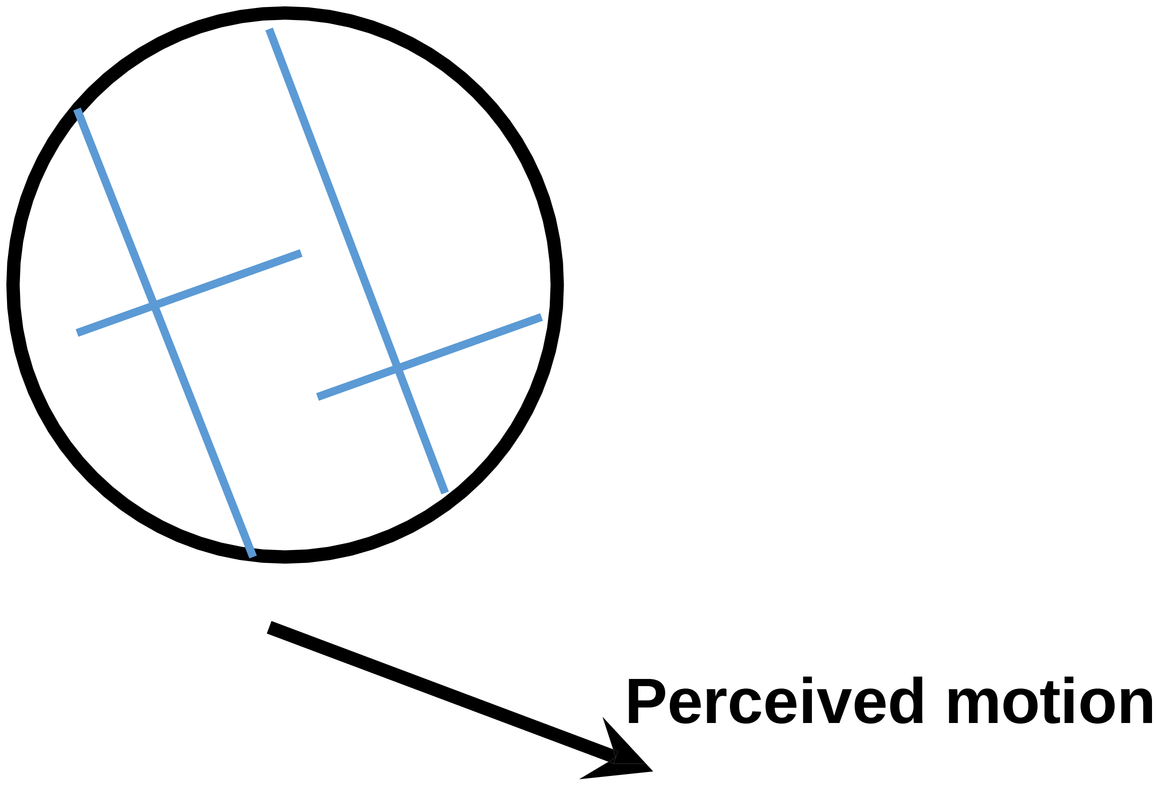
The aperture problem resolved



The aperture problem resolved



The aperture problem resolved



Dealing with larger movements: Iterative refinement

Original (x,y) position

1. Initialize (x',y') = (x,y)

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

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

$$\underbrace{\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix}}_{\text{2nd moment matrix for feature patch in first image}} \underbrace{\begin{bmatrix} u \\ v \end{bmatrix}}_{\text{displacement}} = - \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$$

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

$$\left[\begin{array}{cc} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{array}\right] \left[\begin{array}{c} u \\ v \end{array}\right] = - \left[\begin{array}{c} \sum I_x I_t \\ \sum I_y I_t \end{array}\right]$$

2nd moment matrix for feature patch in first image

displacement

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;`

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

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

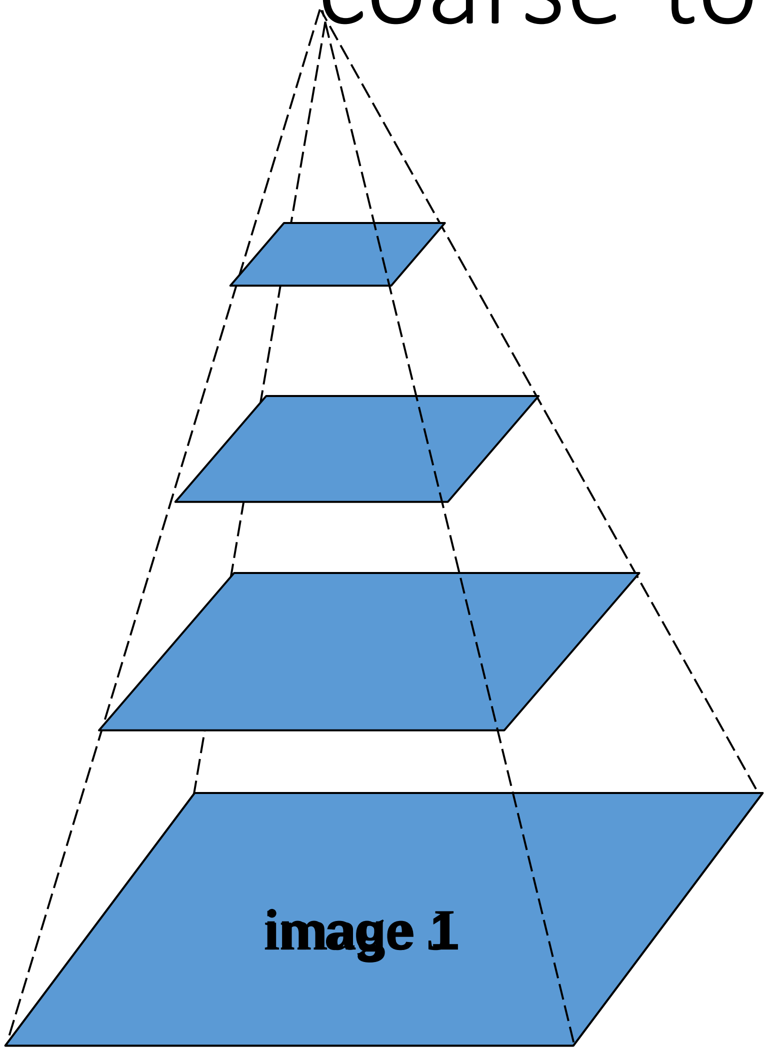
$$\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$$

2nd moment matrix for feature patch in first image

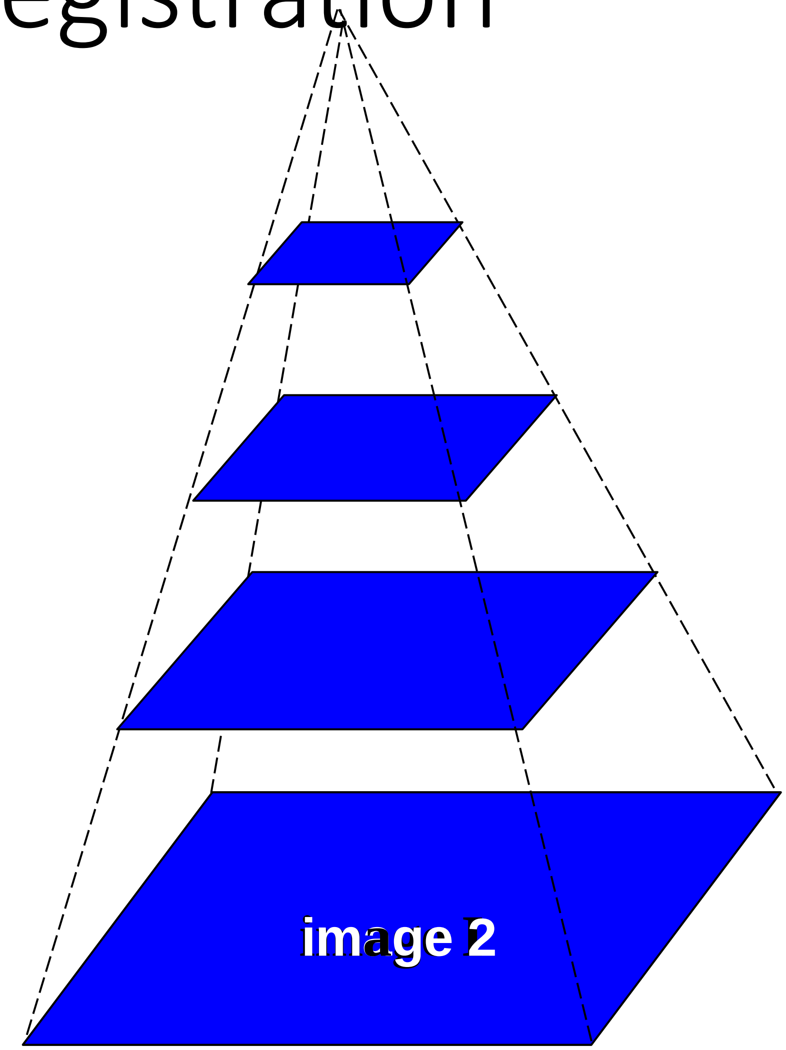
displacement

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

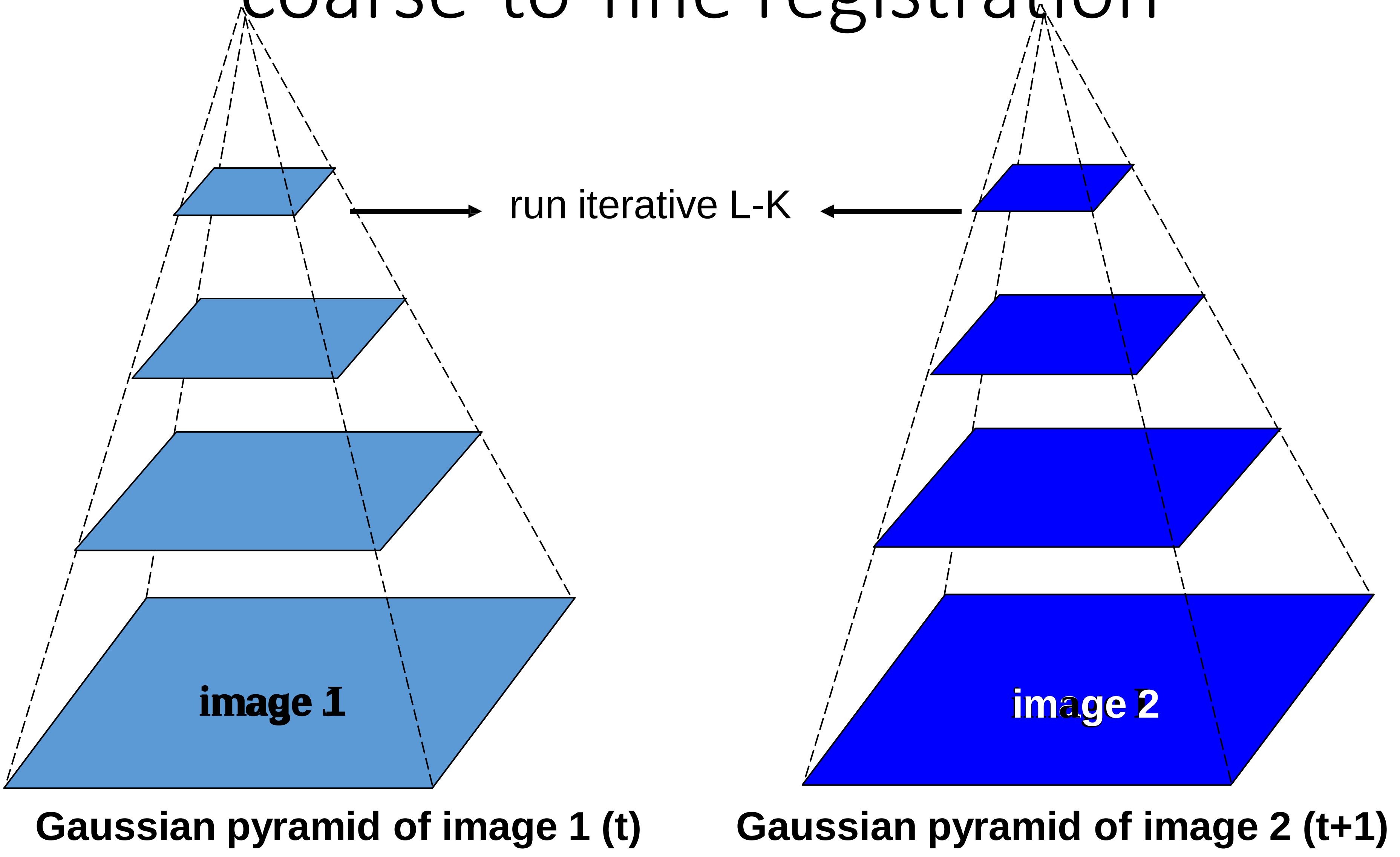


Gaussian pyramid of image 1 (t)

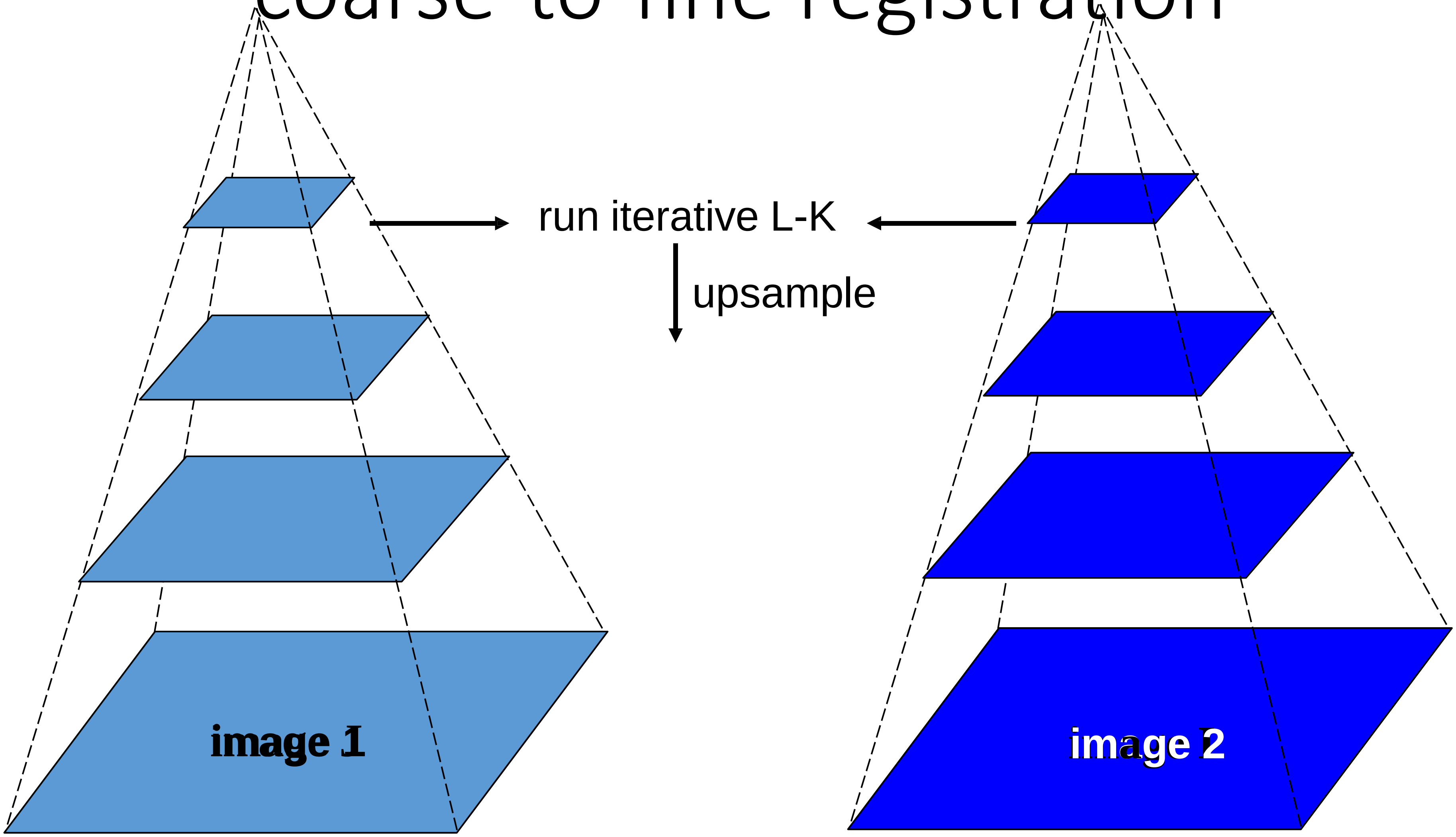


Gaussian pyramid of image 2 (t+1)

Dealing with larger movements: coarse-to-fine registration



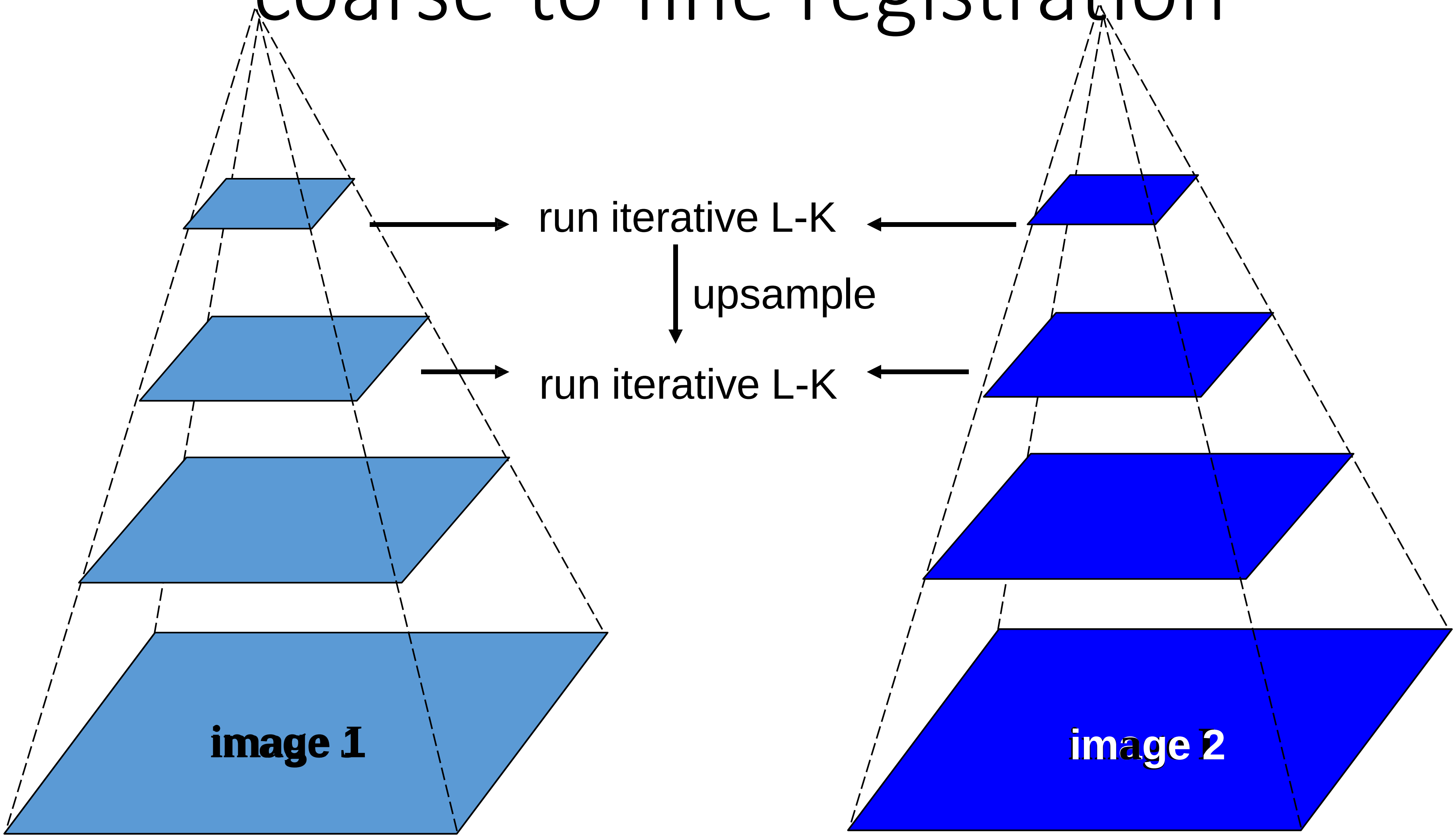
Dealing with larger movements: coarse-to-fine registration



Gaussian pyramid of image 1 (t)

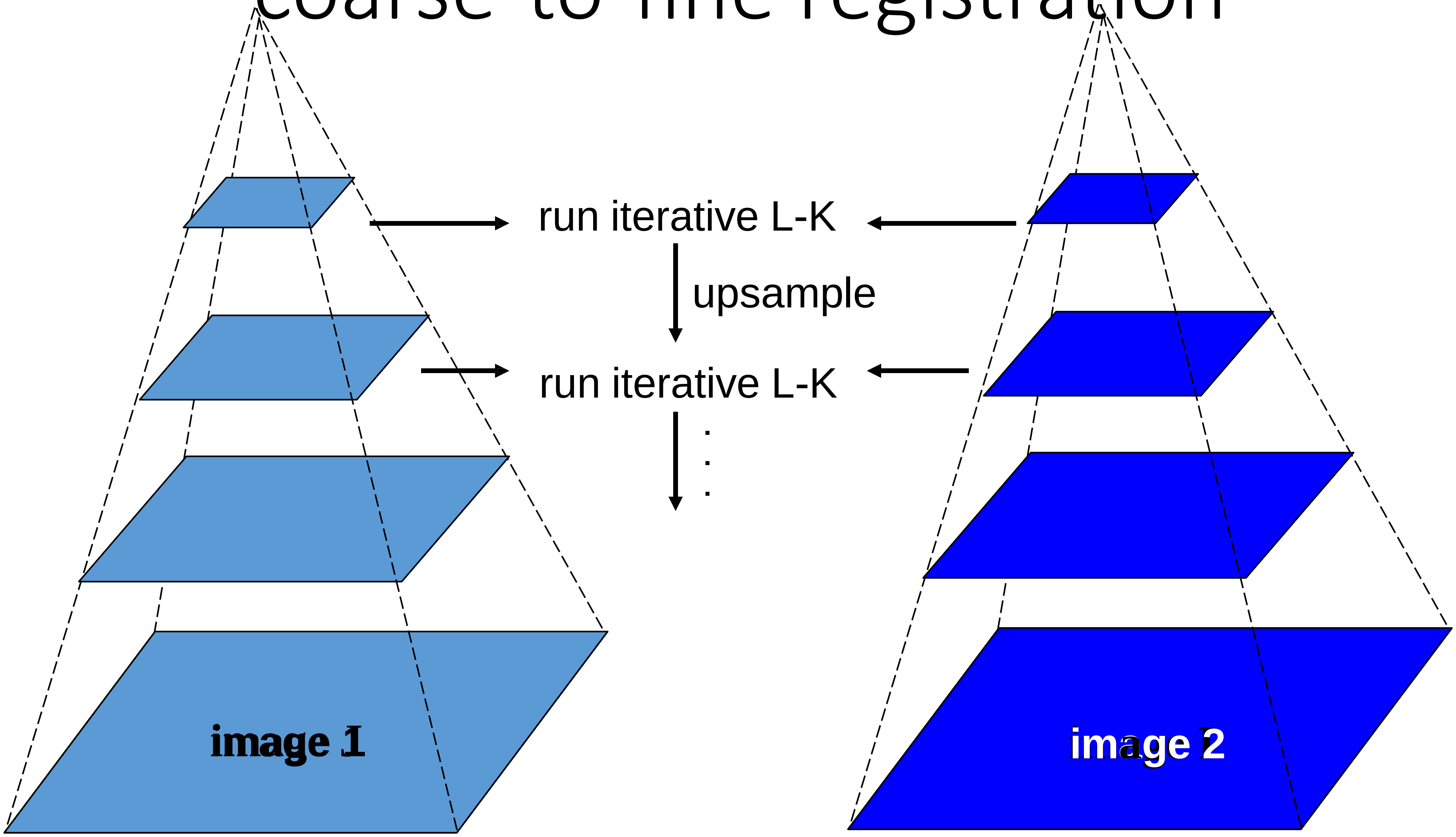
Gaussian pyramid of image 2 (t+1)

Dealing with larger movements: coarse-to-fine registration



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

Dealing with larger movements: coarse-to-fine registration



Gaussian pyramid of image 1 (t)

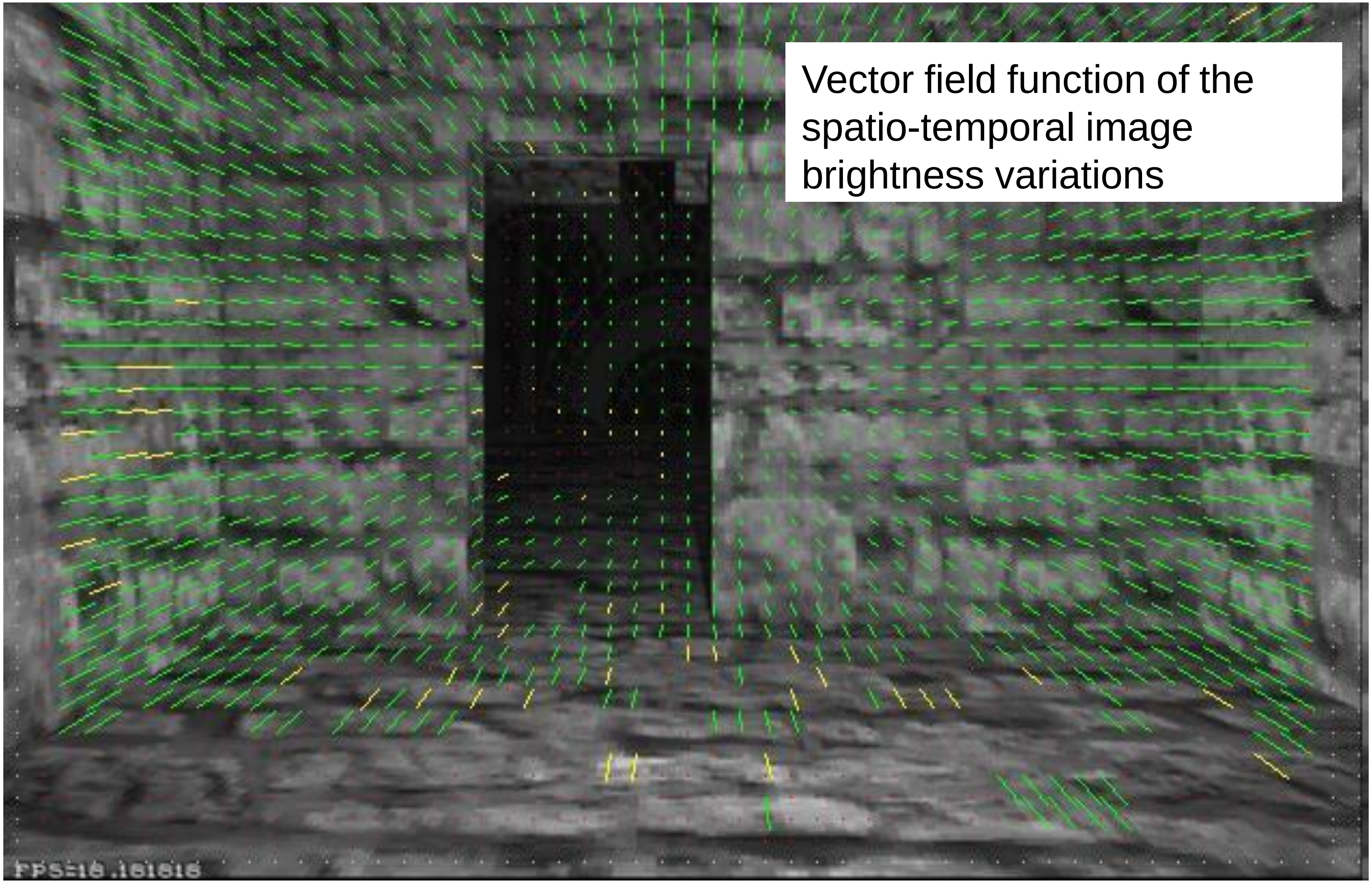
Gaussian pyramid of image 2 (t+1)

Summary of Lucas & Kanade tracking

- Find a good point to track
- 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

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*, 1981.

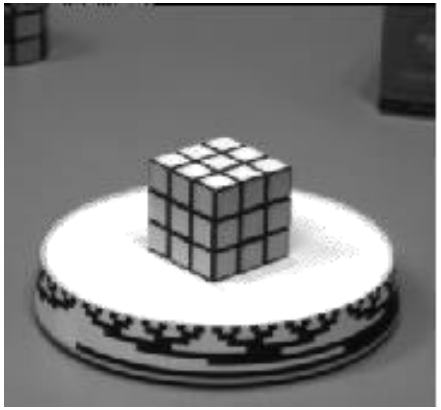
Optical flow

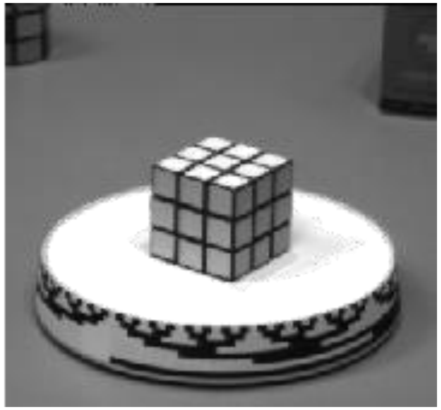


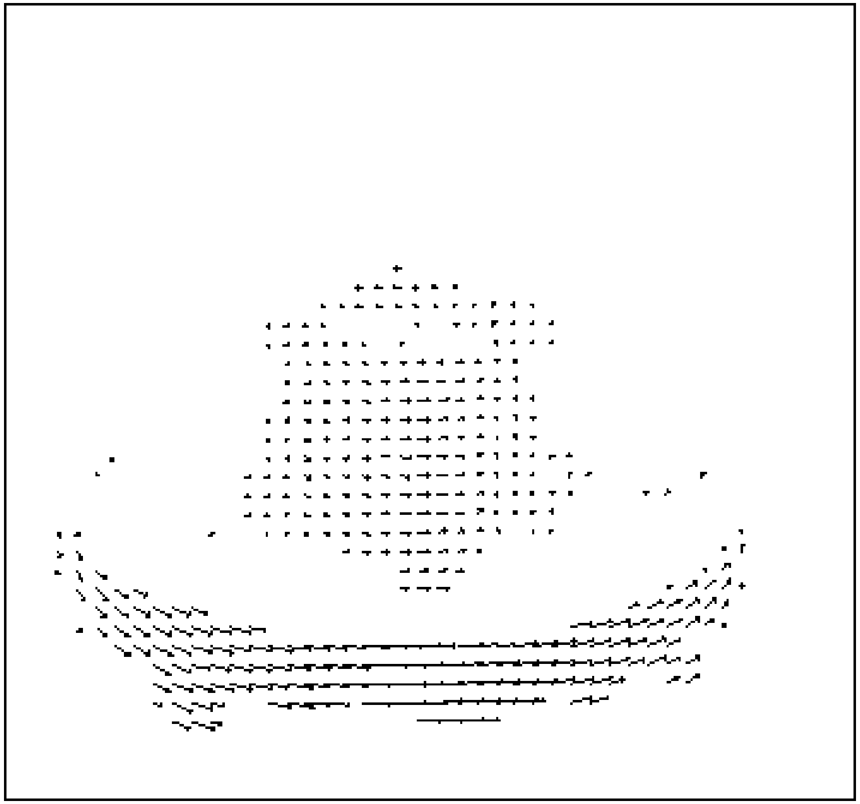
Picture courtesy of Selim Temizer - Learning and Intelligent Systems (LIS) Group, MIT

Motion field

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







Optical flow

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

Optical flow

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

Optical flow

- 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

Optical flow

- 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

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

State-of-the-art optical flow

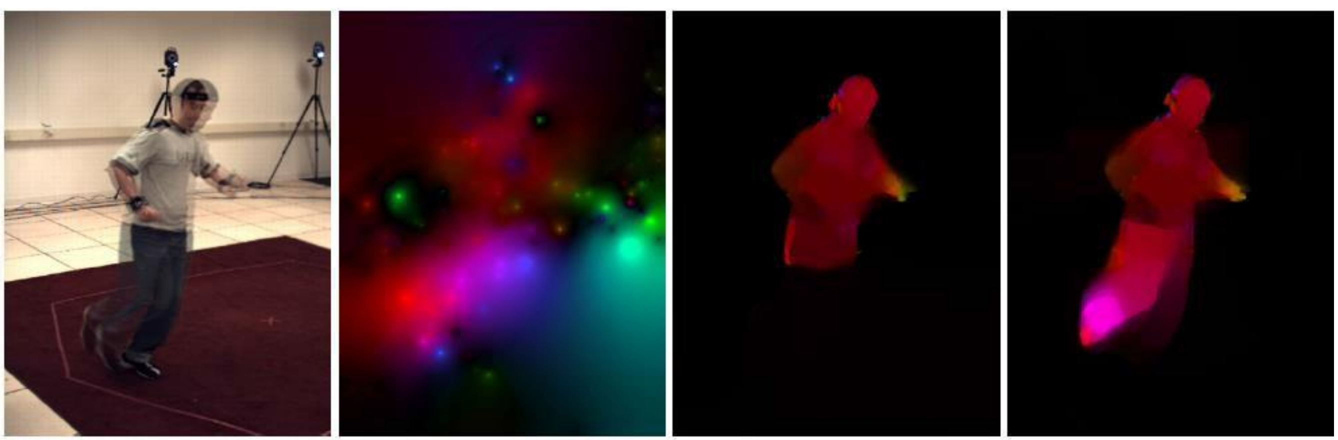
Start with something similar to Lucas-Kanade

+ gradient constancy

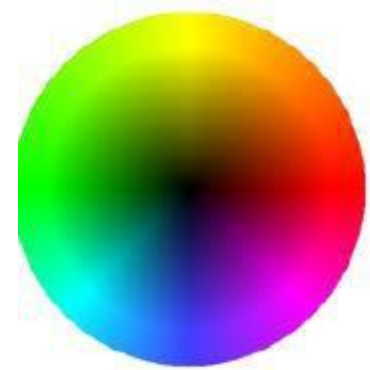
+ energy minimization with smoothing term

+ region matching

+ keypoint matching (long-range)

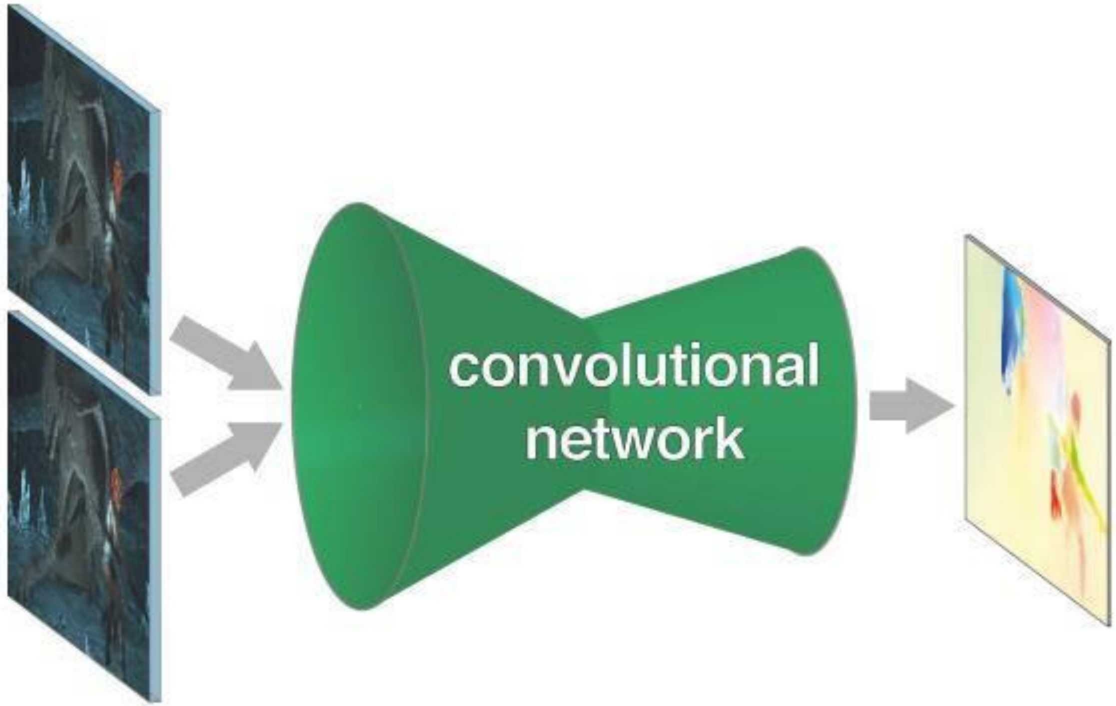


Region-based +Pixel-based +Keypoint-based



Large displacement optical flow, Brox et al., CVPR 2009

Recent Trends



DeepFlow: Large displacement optical flow with deep matching. ICCV 2013
FlowNet: Learning Optical Flow with Convolutional Networks. ICCV 2015
Flow fields: Dense correspondence fields for highly accurate large displacement optical flow estimation. ICCV 2015
A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. CVPR 2016
FlowNet 2.0: Evolution of Optical Flow Estimation with Deep Networks. CVPR 2017
Optical flow estimation using a spatial pyramid network. CVPR 2017
Unsupervised Deep Learning for Optical Flow Estimation. AAAI 2017
Semi-supervised learning for optical flow with generative adversarial networks. NIPS 2017

Acknowledgements

- Thanks to the following researchers for making their teaching/research material online
  - Forsyth
  - Steve Seitz
  - Noah Snavely
  - J.B. Huang
  - Derek Hoiem
  - D. Lowe
  - A. Bobick
  - S. Lazebnik
  - K. Grauman
  - R. Zaleski
  - Leibe
  - And many more ………..