Corner Detection & Optical Flow

CSC420
David Lindell
University of Toronto
cs.toronto.edu/~lindell/teaching/420
Slide credit: Babak Taati ←Ahmed Ashraf ←Sanja Fidler

TCIG
Toronto Computational Imaging Group

Logistics

- A2 due on Friday

- no lecture next Monday (reading week)

Overview

- Recap
- Image features
- Corner detection
- Optical flow

Recap

Review

- Images
  - what is an image?

Review

- Images
  - what is an image?
  - what do pixel values represent?

Review

- Filtering
  - what is correlation?

Review

- Filtering
  - what is correlation?
  - what is convolution?

Review

- Filtering
  - what is correlation?
  - what is convolution?
  - what is the convolution theorem?

Review

- Filtering
  - what is correlation?
  - what is convolution?
  - what is the convolution theorem?
  - what is the Nyquist theorem?

Review

- Filtering
  - what is correlation?
  - what is convolution?
  - what is the convolution theorem?
  - what is the Nyquist theorem?
  - how do we “smooth” an image?

Review

- Edges
  - how do we extract edges from an image?

Review

- Edges
  - how do we extract edges from an image?
  - advantages of using edges vs. a conventional image for computer vision?

Review

- Image resizing
  - what is an image pyramid?

Review

- Image resizing
  - what is an image pyramid?
  - what is aliasing?

Review

- Image resizing
  - what is an image pyramid?
  - what is aliasing?
  - how do we downsample an image?

Review

- Image resizing
  - what is an image pyramid?
  - what is aliasing?
  - how do we downsample an image?
  - how do we upsample an image?

Image Features:
Interest Point (Keypoint) Detection

Image Features

- What skyline is this?

Image Features

- What skyline is this?

Image Features

- What skyline is this?

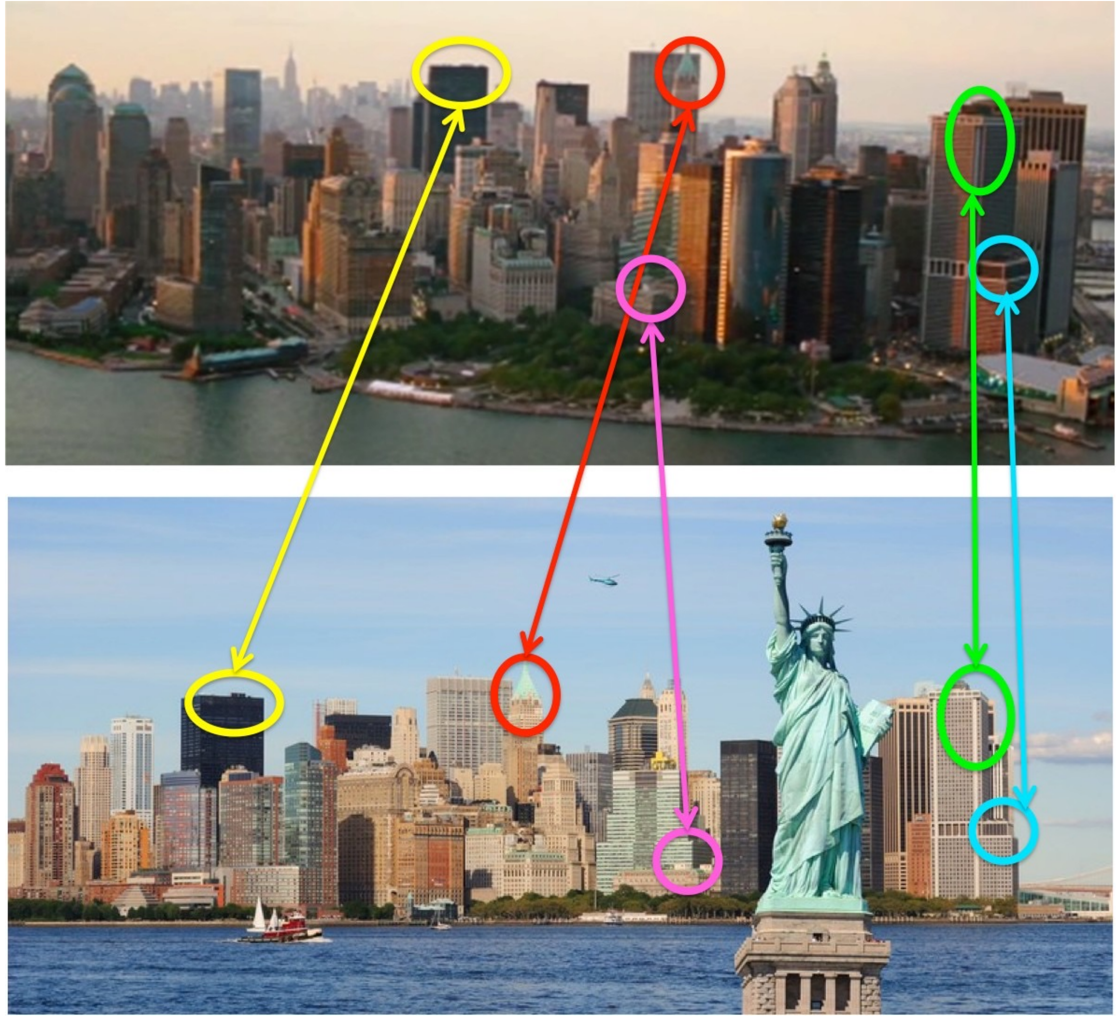
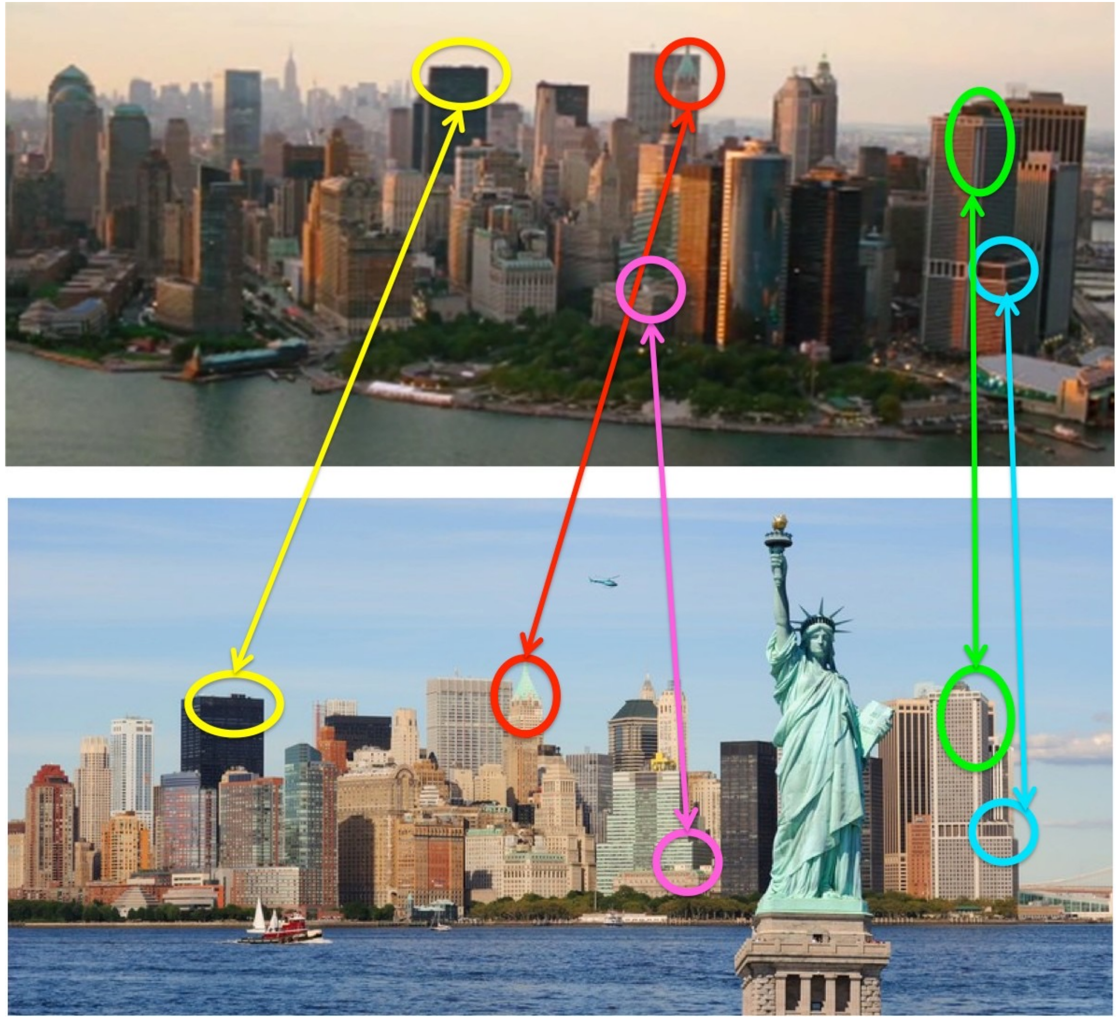


Image Features

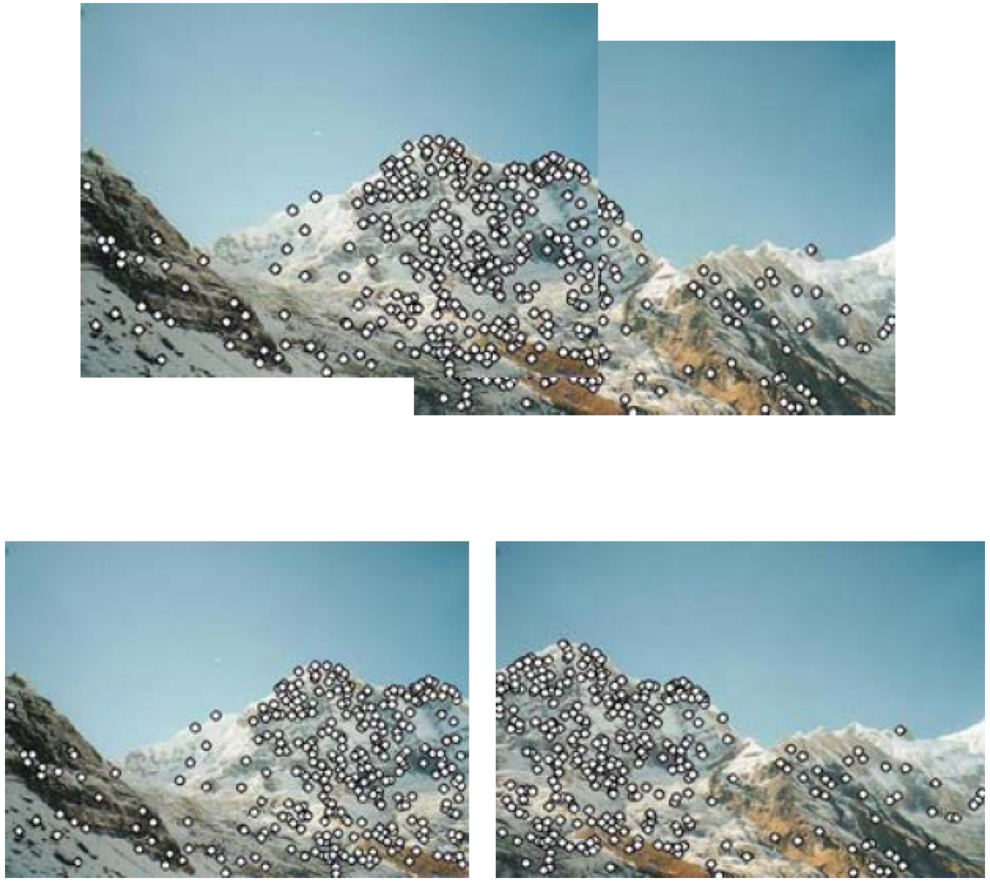
- What skyline is this?

We matched in:

- Distinctive locations: keypoints
- Distinctive features: descriptors



Application Example: Image Stitching



[Source: K. Grauman]

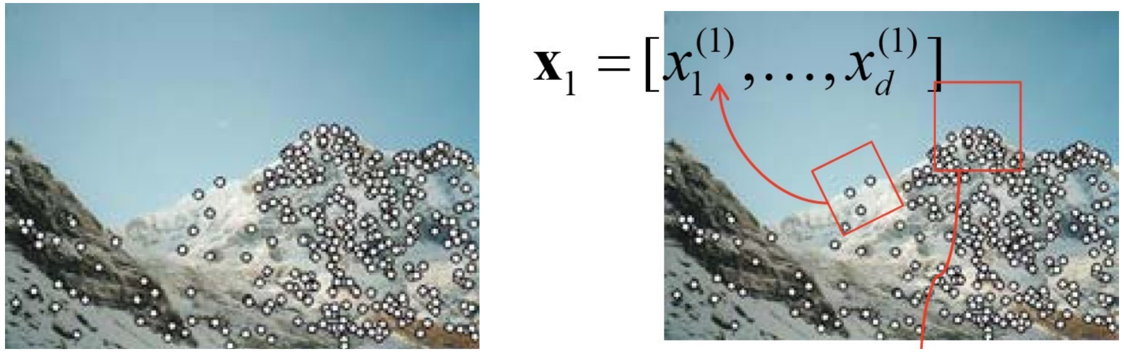
Application Example: Image Stitching

- Detection: Identify the interest points.

[Source: K. Grauman]

Application Example: Image Stitching

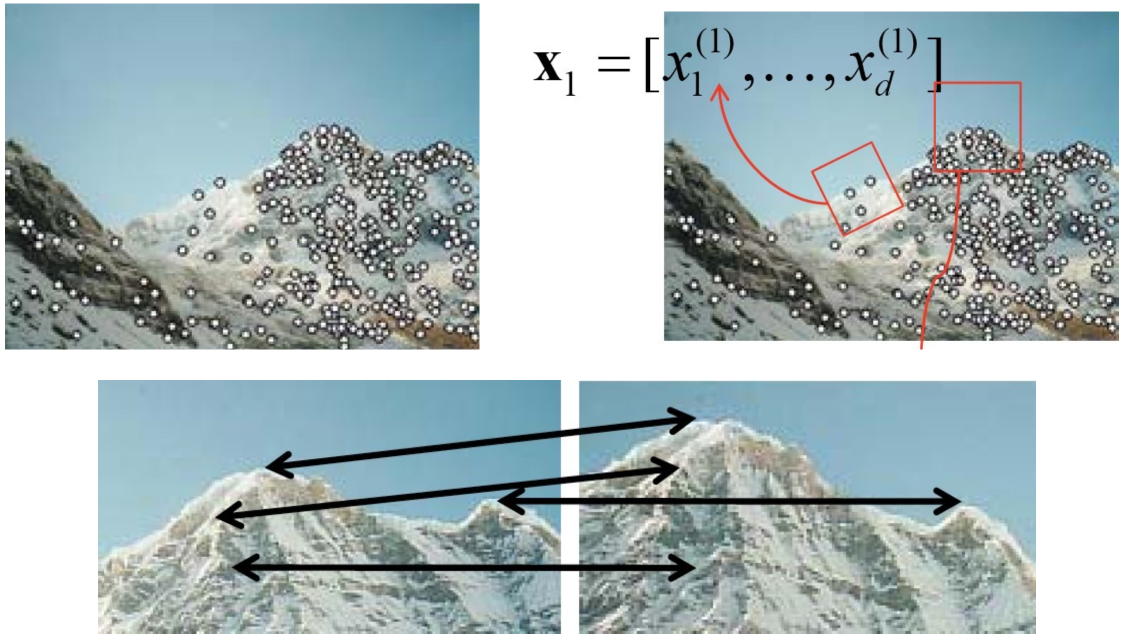
- Detection: Identify the interest points.
- Description: Extract feature vector descriptor around each interest point.



[Source: K. Grauman]

Application Example: Image Stitching

- Detection: Identify the interest points.
- Description: Extract feature vector descriptor around each interest point.
- Matching: Determine correspondence between descriptors in two views.



[Source: K. Grauman]

Goal: Repeatability of the Interest Point Operator

- Our goal is to detect (at least some of) the same points in both images
- We need to run the detection procedure independently per image

Figure: Too few keypoints → little chance to find the true matches

[Source: K. Grauman, slide credit: R. Urtasun]

Goal: Repeatability of the Interest Point Operator

- Our goal is to detect (at least some of) the same points in both images
- We need to run the detection procedure independently per image
- Is it better to detect more interest points or fewer interest points?

Figure: Too few keypoints → little chance to find the true matches

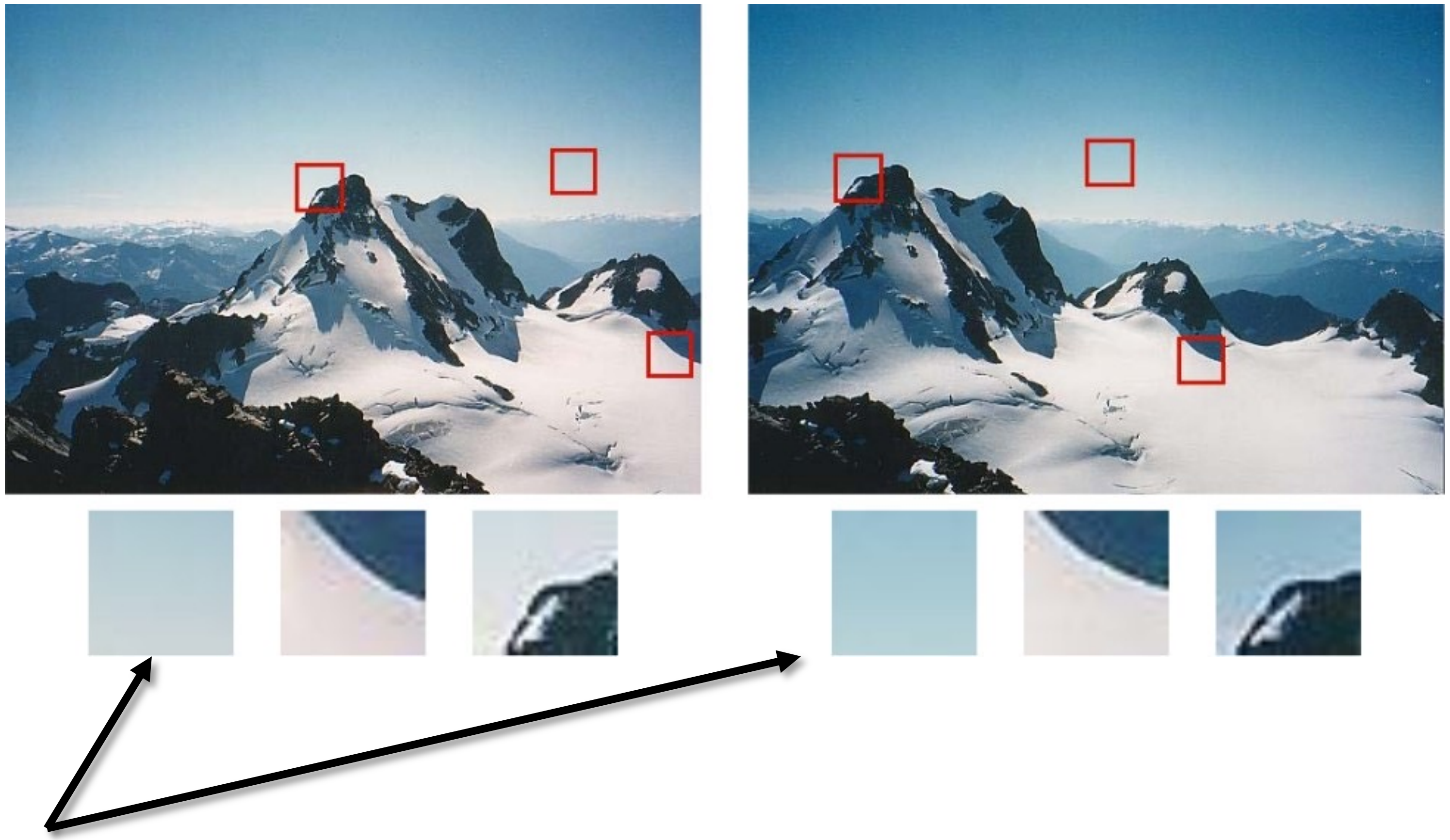
[Source: K. Grauman, slide credit: R. Urtasun]

What Points to Choose?

[Source: K. Grauman]

What Points to Choose for matching?

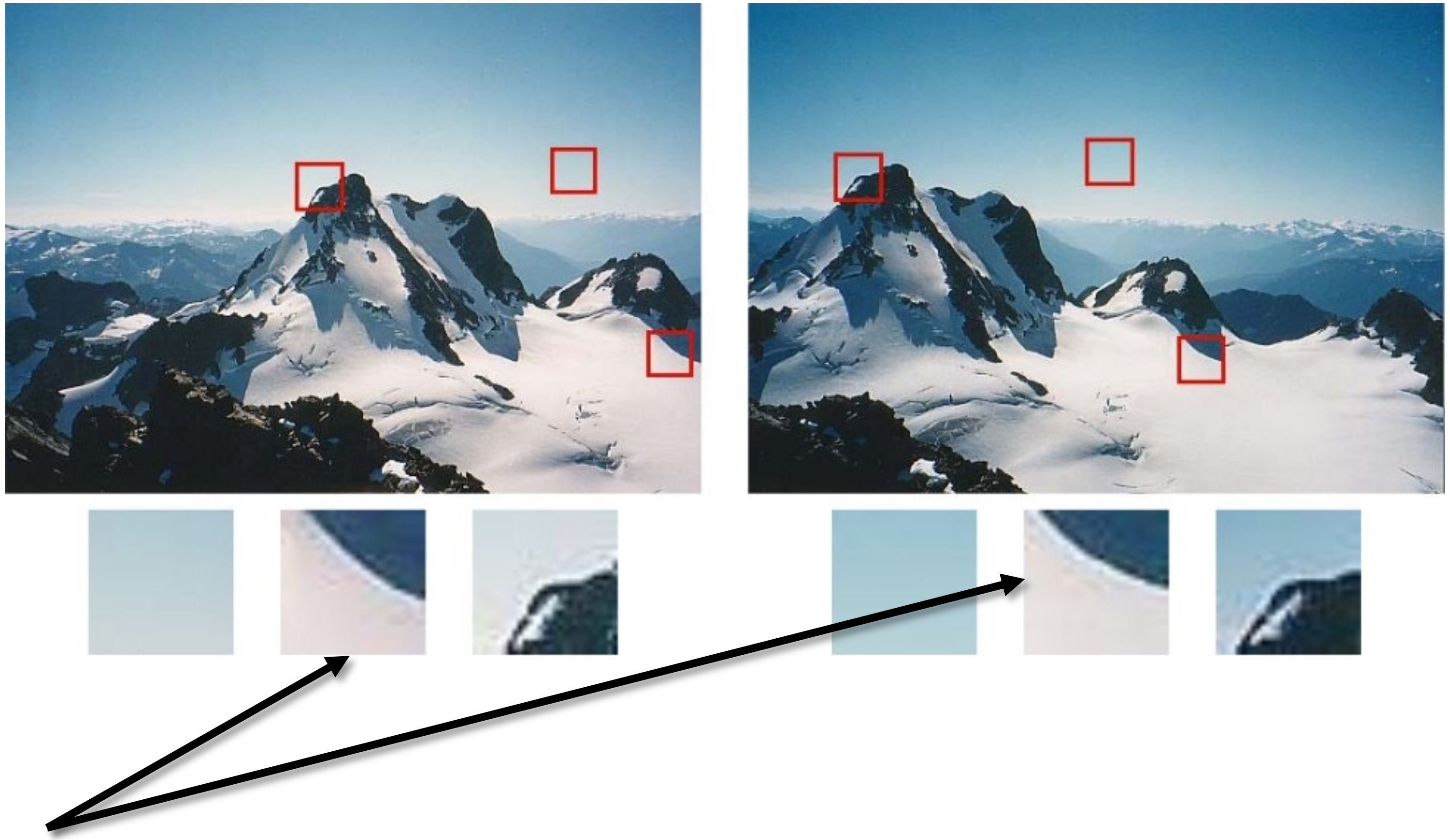
What Points to Choose for matching?



is this a good interest point?

[Adopted from: Szelski (Book)]

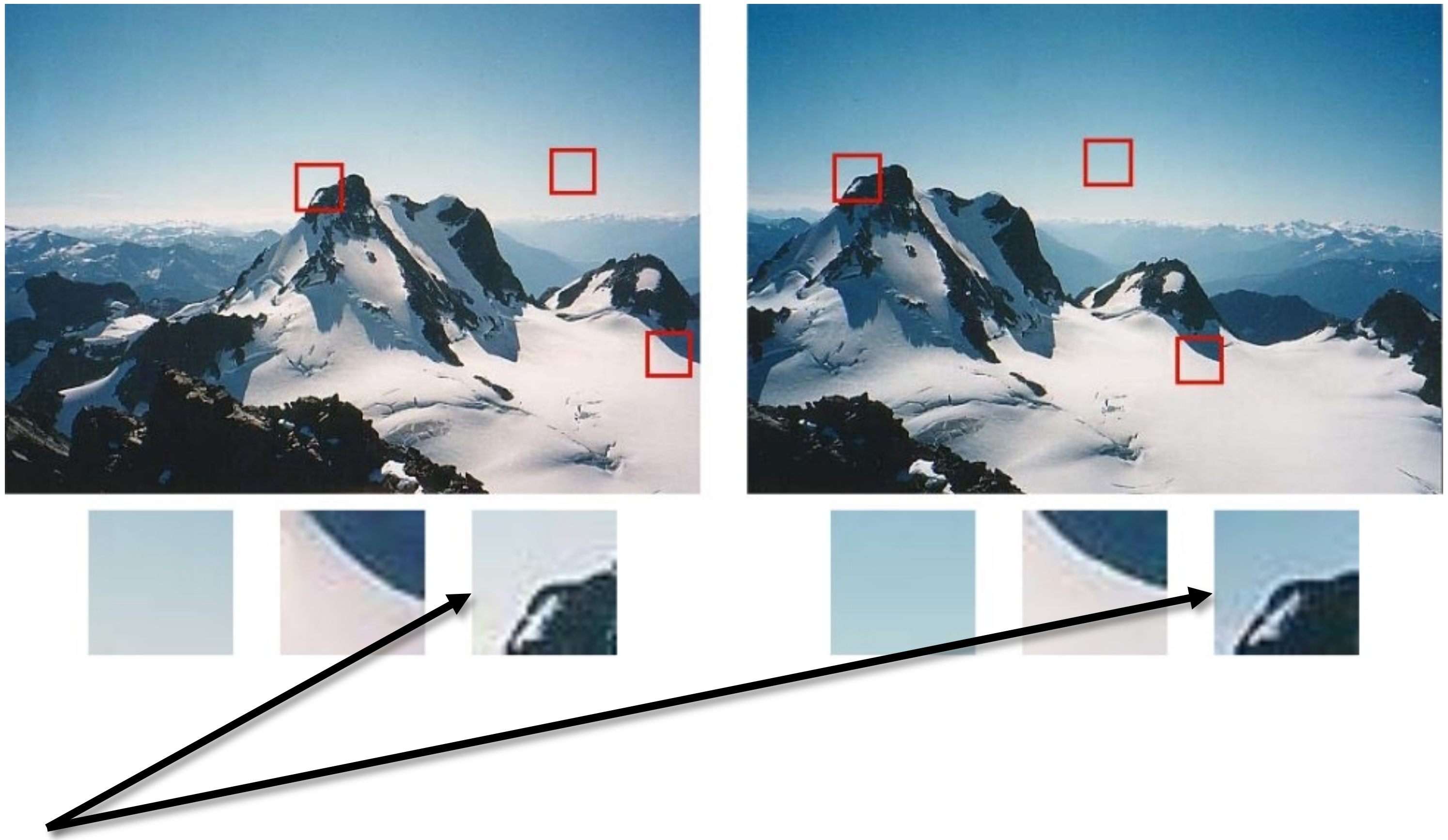
What Points to Choose for matching?



how about this one?

[Adopted from: Szelski (Book)]

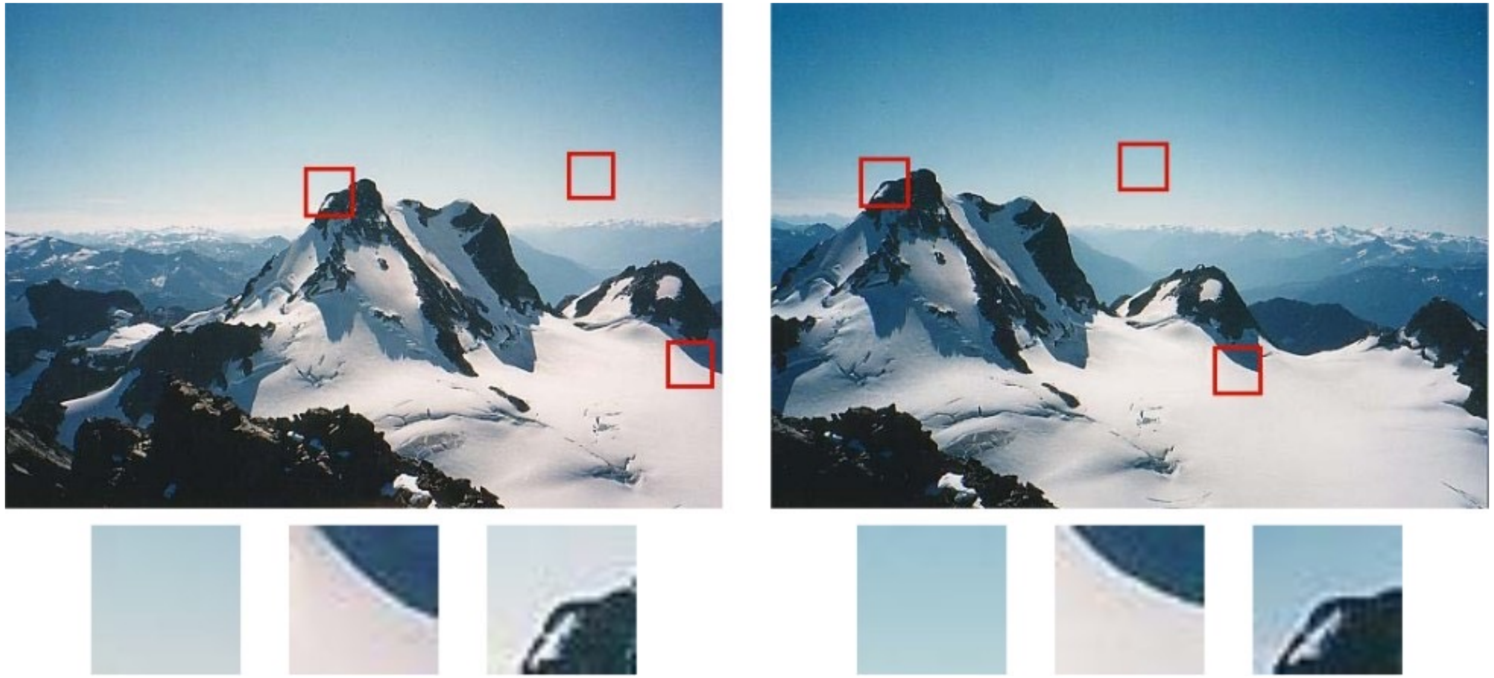
What Points to Choose for matching?



this one? which is best?

[Adopted from: Szelski (Book)]

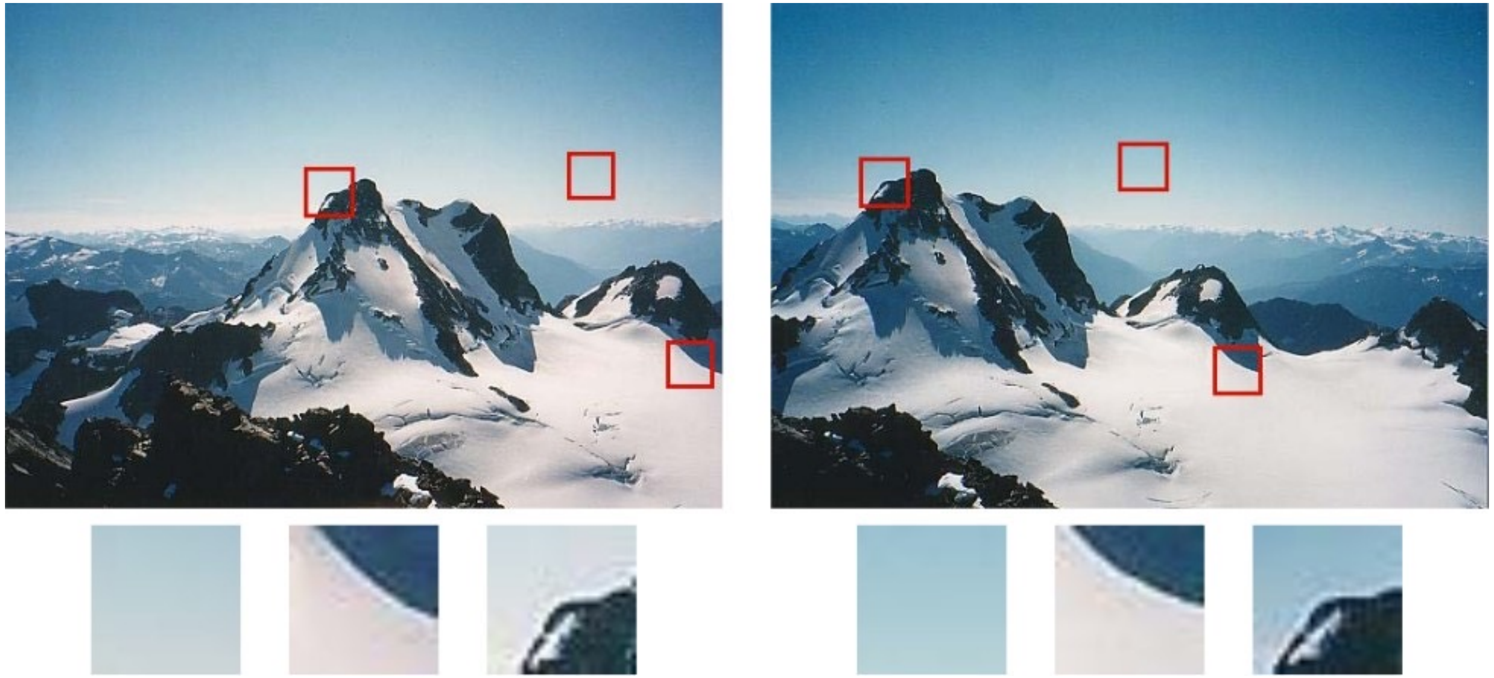
What Points to Choose for matching?



- textureless patches are nearly impossible to localize.

[Adopted from: Szelski (Book)]

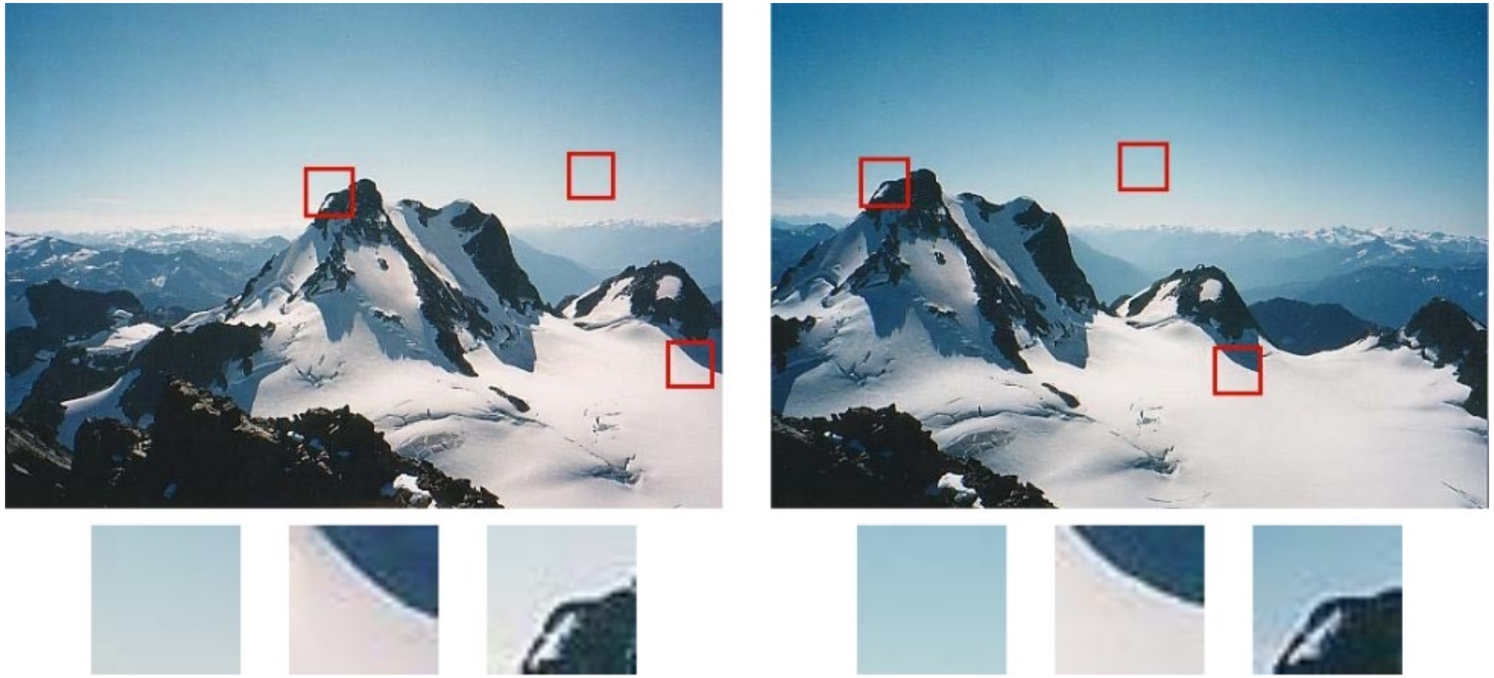
What Points to Choose for matching?



- textureless patches are nearly impossible to localize.
- large contrast changes (gradients) make it easier!

[Adopted from: Szelski (Book)]

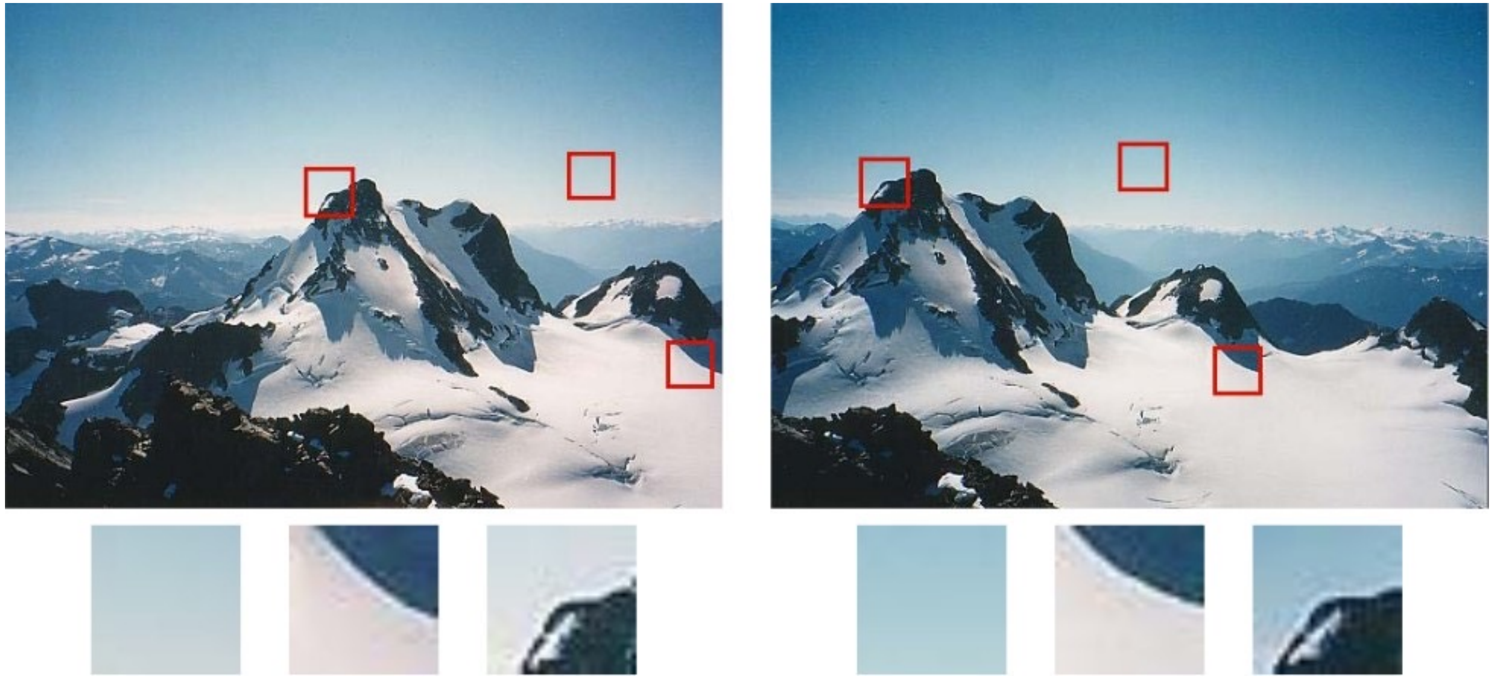
What Points to Choose for matching?



- textureless patches are nearly impossible to localize.
- large contrast changes (gradients) make it easier!
    - can we localize with a single horizontal/vertical/diagonal edge?

[Adopted from: Szelski (Book)]

What Points to Choose for matching?

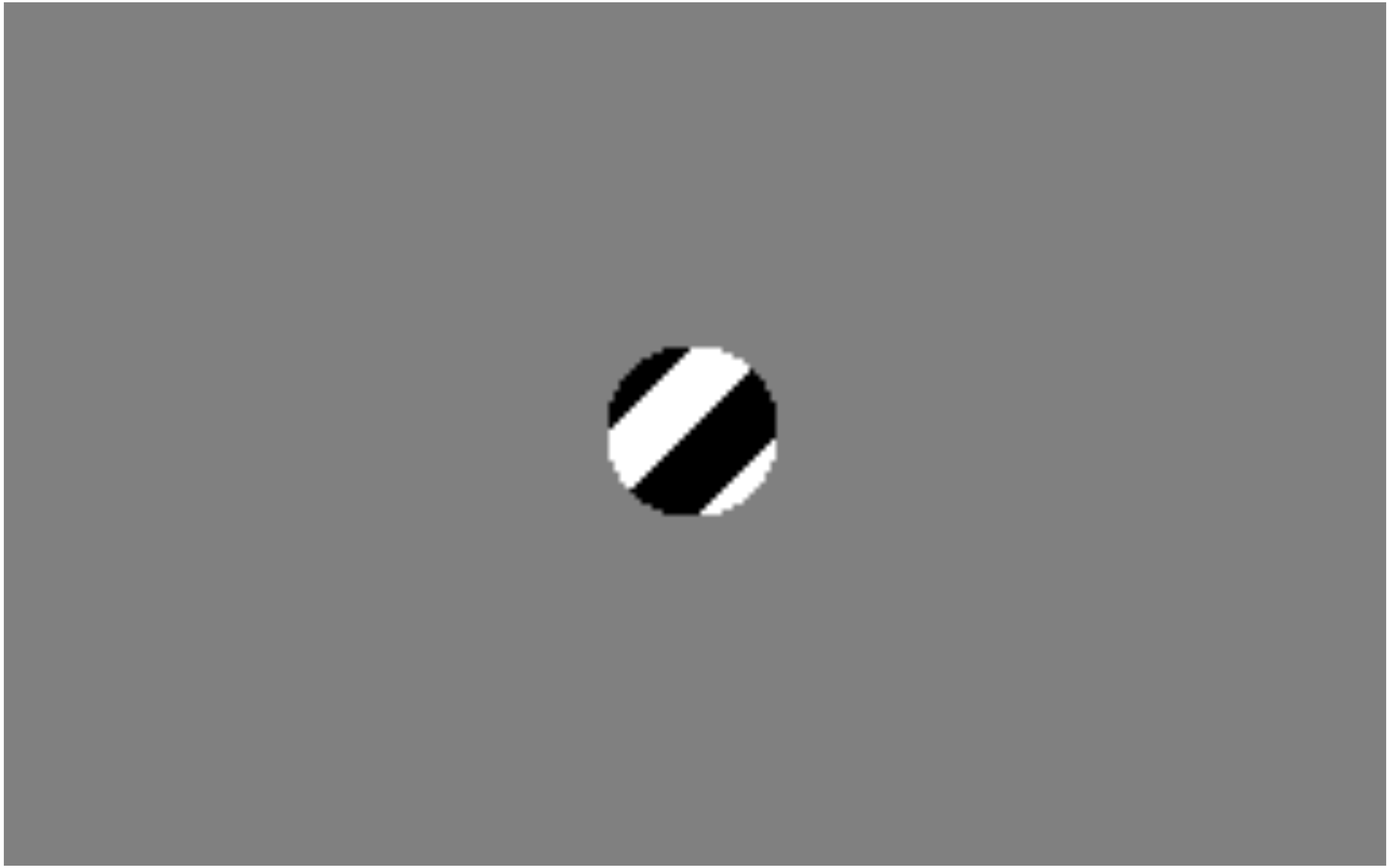


- textureless patches are nearly impossible to localize.
- large contrast changes (gradients) make it easier!
  - can we localize with a single horizontal/vertical/diagonal edge?
  - no—gradients with at least two orientations are easiest (corners)

[Adopted from: Szelski (Book)]

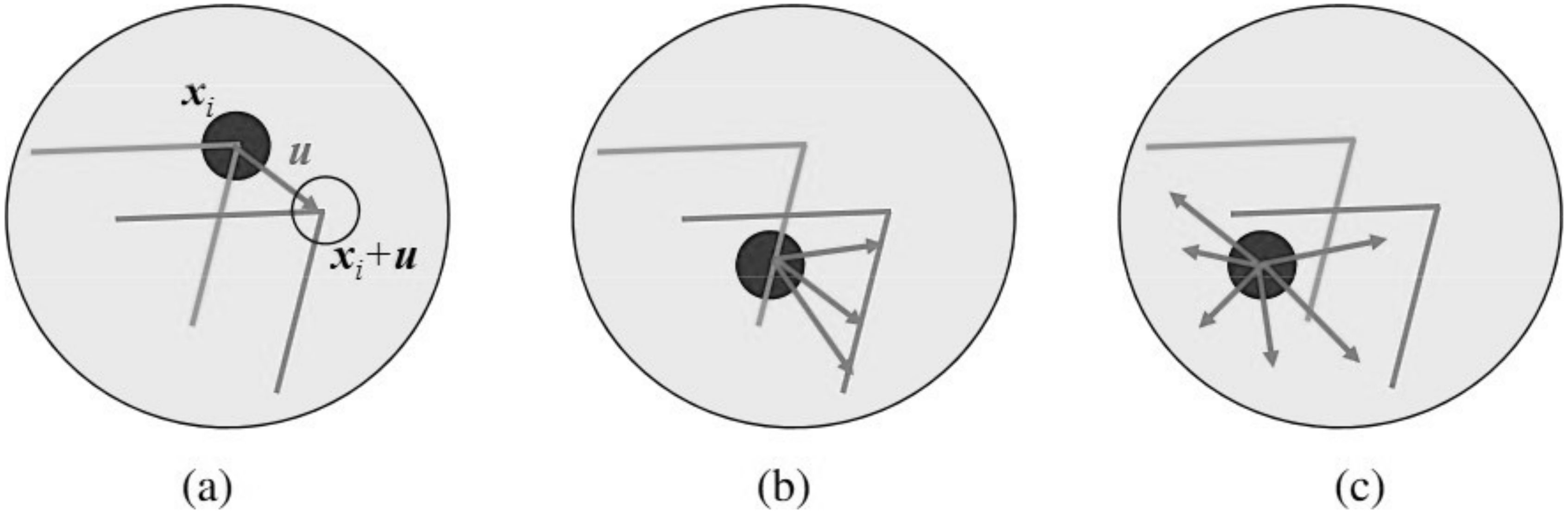
Aperture Problem

Aperture Problem



Aperture Problem

Aperture Problem



- "Corner-like" patch can be reliably matched
- A straight line patch can have multiple matches (Aperture Problem)
- Zero texture, useless, can have infinite matches

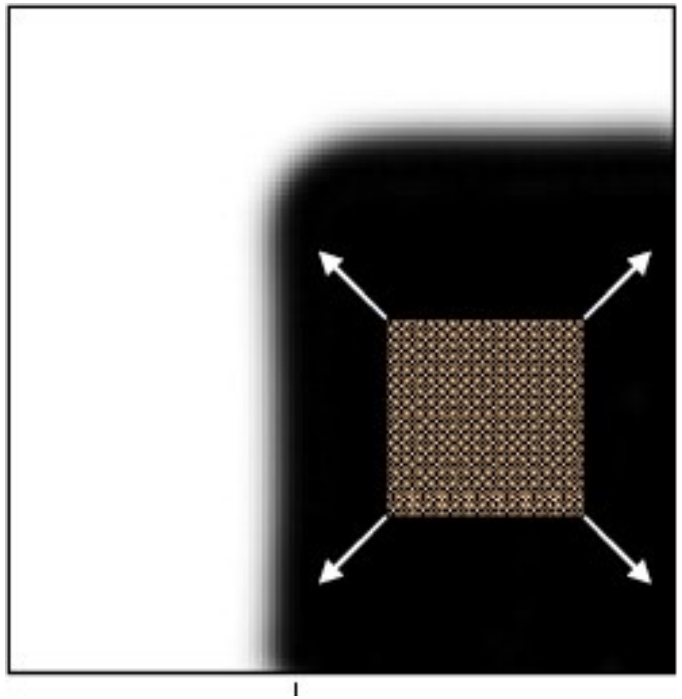
[Source: K. Grauman]

Corner Detection

Interest Points: Corners

- How can we find corners in an image?

Interest Points: Corners

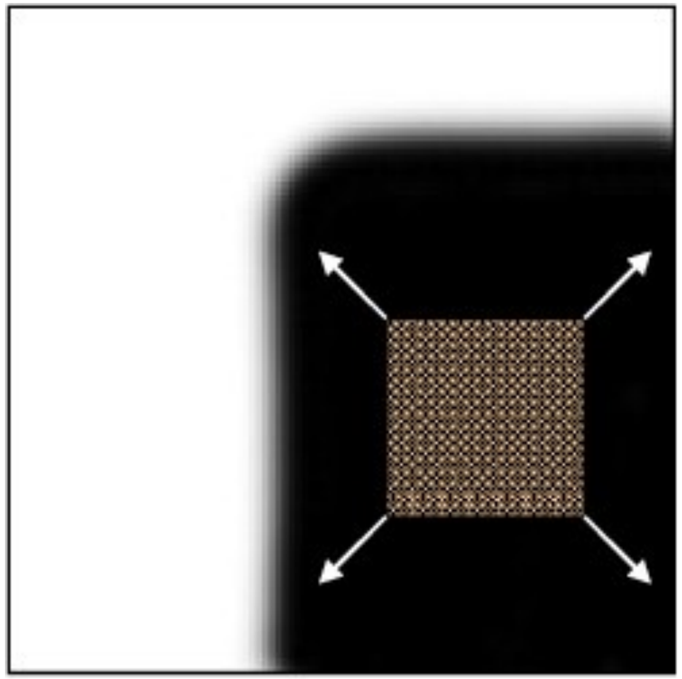


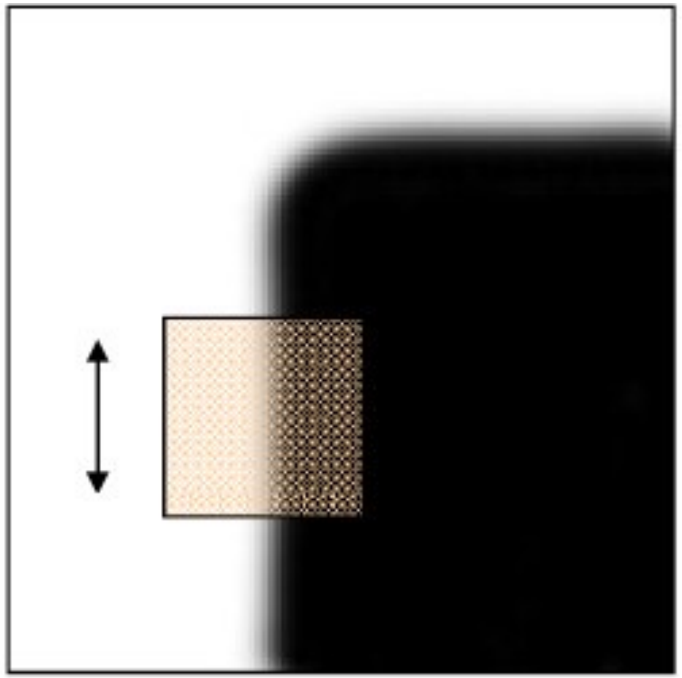
What if we use a small window?

What happens to the intensity variation within the window if we change it's location?

[Source: Alyosha Efros, Darya Frolova, Denis Simakov]

Interest Points: Corners



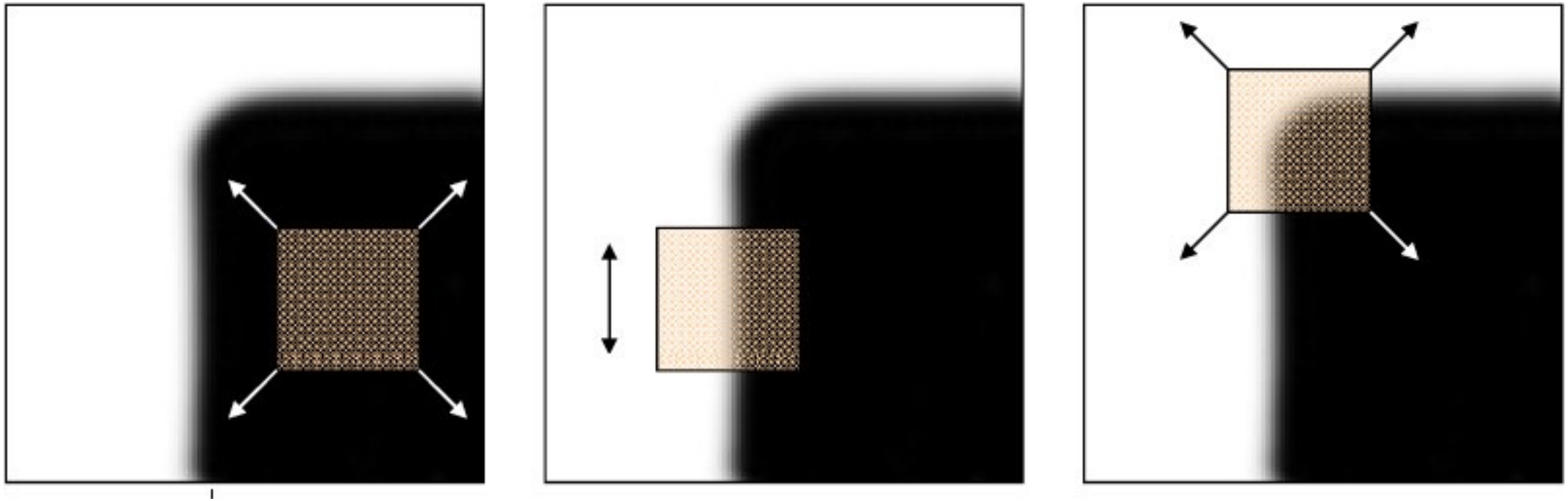


What if we use a small window?

What happens to the intensity variation within the window if we change it's location?

[Source: Alyosha Efros, Darya Frolova, Denis Simakov]

Interest Points: Corners



What if we use a small window?

What happens to the intensity variation within the window if we change it's location?

[Source: Alyosha Efros, Darya Frolova, Denis Simakov]

Interest Points: Corners

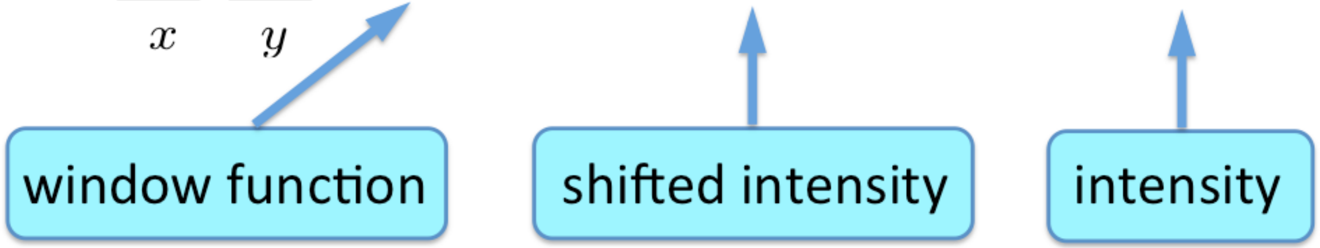
- Compare two image patches using (weighted) summed square difference
- Measures change in appearance of window w(x, y) for the shift

[Source: J. Hays]

Interest Points: Corners

- Compare two image patches using (weighted) summed square difference
- Measures change in appearance of window w(x, y) for the shift

$$E_{\mathrm{WSSD}}(u, v) = \sum_x \sum_y w(x, y)[I(x + u, y + v) - I(x, y)]^2$$



[Source: J. Hays]

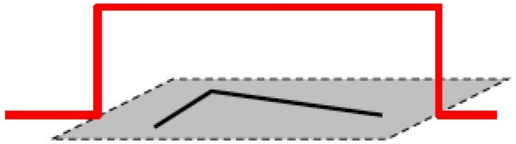
Interest Points: Corners

- Compare two image patches using (weighted) summed square difference
- Measures change in appearance of window w(x, y) for the shift

$$E_{\mathrm{WSSD}}(u,v) = \sum_x \sum_y w(x,y)\left[I(x+u,y+v) - I(x,y)\right]^2$$

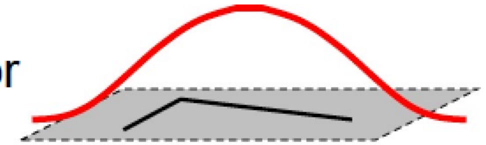
window function — shifted intensity — intensity

Window function $w(x,y)$ =



1 in window, 0 outside

or



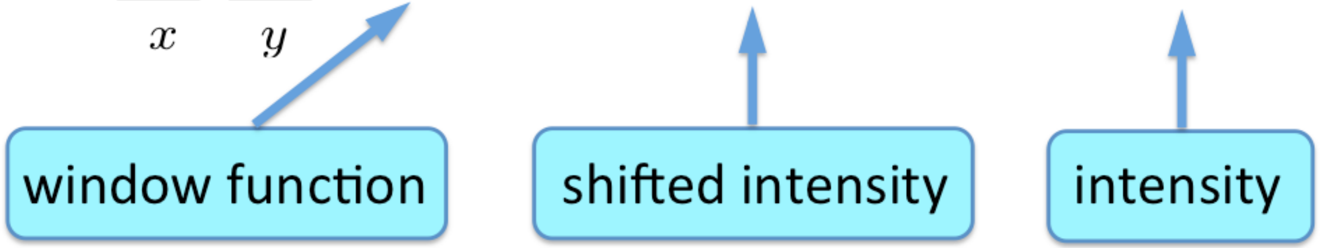
Gaussian

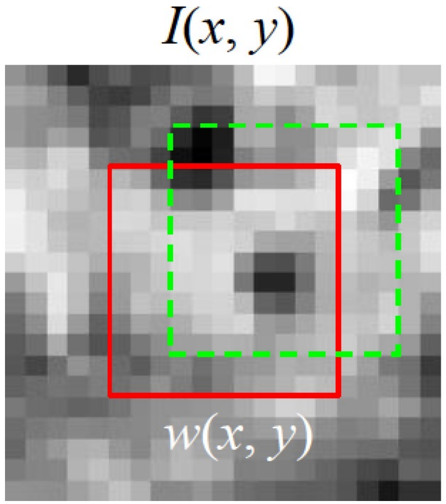
[Source: J. Hays]

Interest Points: Corners

- Compare two image patches using (weighted) summed square difference
- Measures change in appearance of window w(x, y) for the shift

$$E_{\mathrm{WSSD}}(u, v) = \sum_x \sum_y w(x, y)[I(x + u, y + v) - I(x, y)]^2$$





[Source: J. Hays]

Interest Points: Corners

- Compare two image patches using (weighted) summed square difference
- Measures change in appearance of window w(x, y) for the shift

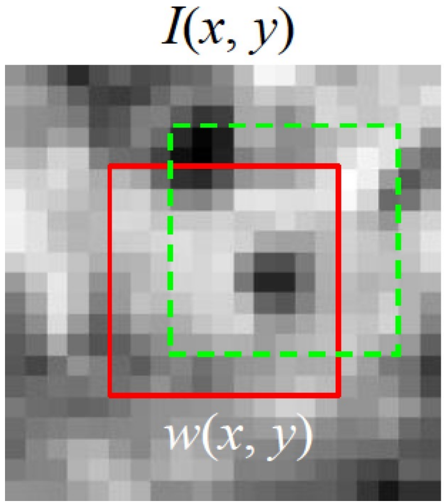
$$E_{\mathrm{WSSD}}(u, v) = \sum_x \sum_y w(x, y)[I(x + u, y + v) - I(x, y)]^2$$

what does E_{WSSD} look like?

window function

shifted intensity

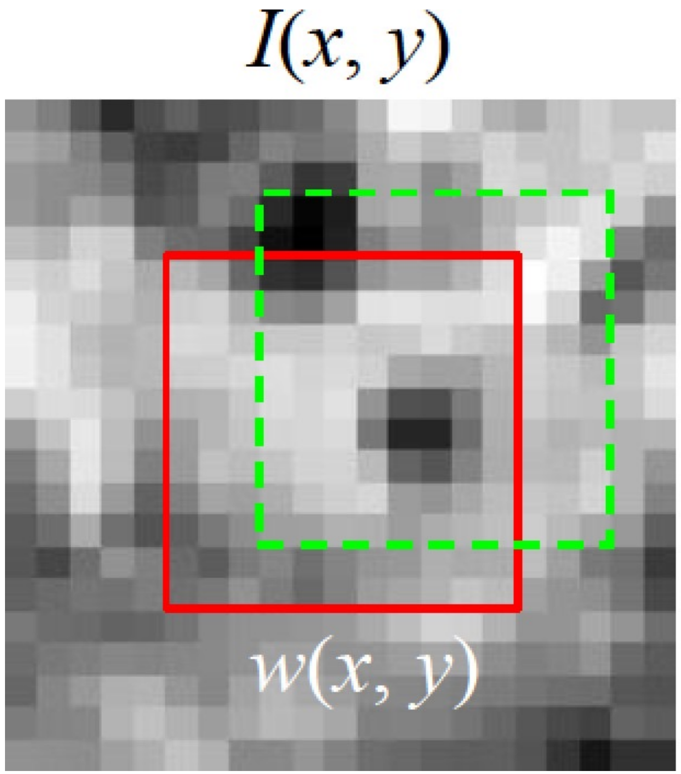
intensity

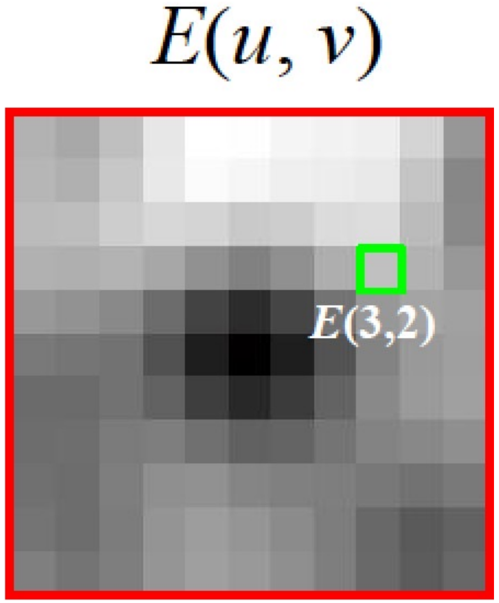


[Source: J. Hays]

Interest Points: Corners

- Compare two image patches using (weighted) summed square difference
- Measures change in appearance of window w(x, y) for the shift

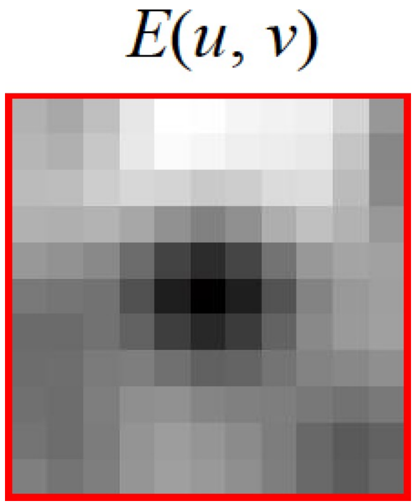




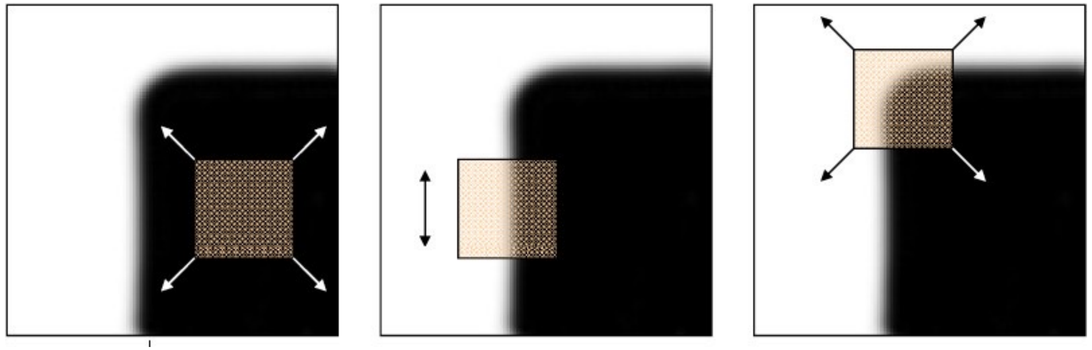
[Source: J. Hays]

Interest Points: Corners

- Let's look at E_{WSSD}
- We want to find out how this function behaves for small shifts



- Remember our goal to detect corners:



Interest Points: Corners

- Using a simple first order Taylor series expansion about *x, y* :

$$I(x+u, y+v) \approx I(x,y) + u \cdot \frac{\partial I}{\partial x}(x,y) + v \cdot \frac{\partial I}{\partial y}(x,y)$$

Interest Points: Corners

- Using a simple first order Taylor series expansion about *x, y* :

$$I(x+u, y+v) \approx I(x,y) + u \cdot \frac{\partial I}{\partial x}(x,y) + v \cdot \frac{\partial I}{\partial y}(x,y)$$

- Using a series of polynomials to approximate I, more info on Taylor Series [here]

Interest Points: Corners

- Using a simple first order Taylor series expansion about *x, y* :

$$I(x+u, y+v) \approx I(x,y) + u \cdot \frac{\partial I}{\partial x}(x,y) + v \cdot \frac{\partial I}{\partial y}(x,y)$$

- Using a series of polynomials to approximate I, more info on Taylor Series here
- And plugging it in our expression for E_{WSSD}:

$$E_{\text{WSSD}}(u,v) \quad = \quad \sum_x \sum_y w(x,y)\Big(I(x+u, y+v) - I(x,y)\Big)^2$$

Interest Points: Corners

- Using a simple first order Taylor series expansion about *x, y* :

$$I(x+u, y+v) \approx I(x,y) + u \cdot \frac{\partial I}{\partial x}(x,y) + v \cdot \frac{\partial I}{\partial y}(x,y)$$

- Using a series of polynomials to approximate I, more info on Taylor Series here
- And plugging it in our expression for E_{WSSD}:

$$\begin{aligned} E_{\text{WSSD}}(u,v) &= \sum_x \sum_y w(x,y)\Big(I(x+u,y+v) - I(x,y)\Big)^2 \\ &\approx \sum_x \sum_y w(x,y)\Big(I(x,y) + u \cdot I_x + v \cdot I_y - I(x,y)\Big)^2 \end{aligned}$$

Interest Points: Corners

- Using a simple first order Taylor series expansion about *x, y*:

$$I(x+u, y+v) \approx I(x,y) + u \cdot \frac{\partial I}{\partial x}(x,y) + v \cdot \frac{\partial I}{\partial y}(x,y)$$

- Using a series of polynomials to approximate I, more info on Taylor Series here
- And plugging it in our expression for E_{WSSD}:

$$\begin{aligned} E_{\text{WSSD}}(u,v) &= \sum_x \sum_y w(x,y)\Big(I(x+u,y+v) - I(x,y)\Big)^2 \\ &\approx \sum_x \sum_y w(x,y)\Big(I(x,y) + u \cdot I_x + v \cdot I_y - I(x,y)\Big)^2 \\ &= \sum_x \sum_y w(x,y)\Big(u^2 I_x^2 + 2u \cdot v \cdot I_x \cdot I_y + v^2\ I_y^2\Big) \end{aligned}$$

Interest Points: Corners

- Using a simple first order Taylor series expansion about *x, y* :

$$I(x+u, y+v) \approx I(x,y) + u \cdot \frac{\partial I}{\partial x}(x,y) + v \cdot \frac{\partial I}{\partial y}(x,y)$$

- Using a series of polynomials to approximate I, more info on Taylor Series here
- And plugging it in our expression for E_{WSSD}:

$$\begin{aligned} E_{WSSD}(u,v) &= \sum_x \sum_y w(x,y)\Big(I(x+u,y+v) - I(x,y)\Big)^2 \\ &\approx \sum_x \sum_y w(x,y)\Big(I(x,y) + u \cdot I_x + v \cdot I_y - I(x,y)\Big)^2 \\ &= \sum_x \sum_y w(x,y)\Big(u^2 I_x^2 + 2u \cdot v \cdot I_x \cdot I_y + v^2\, I_y^2\Big) \\ &= \sum_x \sum_y w(x,y) \cdot \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} I_x^2 & I_x \cdot I_y \\ I_x \cdot I_y & I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} \end{aligned}$$

Interest Points: Corners

$$E_{\text{WSSD}}(u,v) = \sum_x \sum_y w(x,y) \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} I_x^2 & I_x \cdot I_y \\ I_x \cdot I_y & I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

Interest Points: Corners

$$E_{\text{WSSD}}(u,v) \quad = \quad \sum_x \sum_y w(x,y) \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} I_x^2 & I_x \cdot I_y \\ I_x \cdot I_y & I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

- Since (*u, v*) doesn't depend on (*x, y*) we can rewrite it slightly:

Interest Points: Corners

$$E_{\text{WSSD}}(u,v) = \sum_x \sum_y w(x,y) \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} I_x^2 & I_x \cdot I_y \\ I_x \cdot I_y & I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

- Since (*u, v*) doesn't depend on (*x, y*) we can rewrite it slightly:

$$= \begin{bmatrix} u & v \end{bmatrix} \underbrace{\left(\sum_x \sum_y w(x,y) \begin{bmatrix} I_x^2 & I_x \cdot I_y \\ I_x \cdot I_y & I_y^2 \end{bmatrix} \right)}_{\text{Let's denotes this with } M} \begin{bmatrix} u \\ v \end{bmatrix}$$

Interest Points: Corners

$$E_{\text{WSSD}}(u,v) = \sum_x \sum_y w(x,y) \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} I_x^2 & I_x \cdot I_y \\ I_x \cdot I_y & I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

- Since (*u, v*) doesn't depend on (*x, y*) we can rewrite it slightly:

$$= \begin{bmatrix} u & v \end{bmatrix} \underbrace{\left(\sum_x \sum_y w(x,y) \begin{bmatrix} I_x^2 & I_x \cdot I_y \\ I_x \cdot I_y & I_y^2 \end{bmatrix} \right)}_{\text{Let's denotes this with } M} \begin{bmatrix} u \\ v \end{bmatrix}$$

$$= \begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix}$$

Interest Points: Corners

$$E_{\text{WSSD}}(u,v) = \sum_x \sum_y w(x,y) \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} I_x^2 & I_x \cdot I_y \\ I_x \cdot I_y & I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

- Since (u, v) doesn't depend on (x, y) we can rewrite it slightly:

$$= \begin{bmatrix} u & v \end{bmatrix} \underbrace{\left(\sum_x \sum_y w(x,y) \begin{bmatrix} I_x^2 & I_x \cdot I_y \\ I_x \cdot I_y & I_y^2 \end{bmatrix} \right)}_{\text{Let's denotes this with } M} \begin{bmatrix} u \\ v \end{bmatrix}$$

$$= \begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix}$$

what is M?

Interest Points: Corners

$$E_{\text{WSSD}}(u,v) = \sum_x \sum_y w(x,y) \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} I_x^2 & I_x \cdot I_y \\ I_x \cdot I_y & I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

- Since (*u, v*) doesn't depend on (*x, y*) we can rewrite it slightly:

$$= \begin{bmatrix} u & v \end{bmatrix} \underbrace{\left(\sum_x \sum_y w(x,y) \begin{bmatrix} I_x^2 & I_x \cdot I_y \\ I_x \cdot I_y & I_y^2 \end{bmatrix} \right)}_{\text{Let's denotes this with } M} \begin{bmatrix} u \\ v \end{bmatrix}$$

$$= \begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix}$$

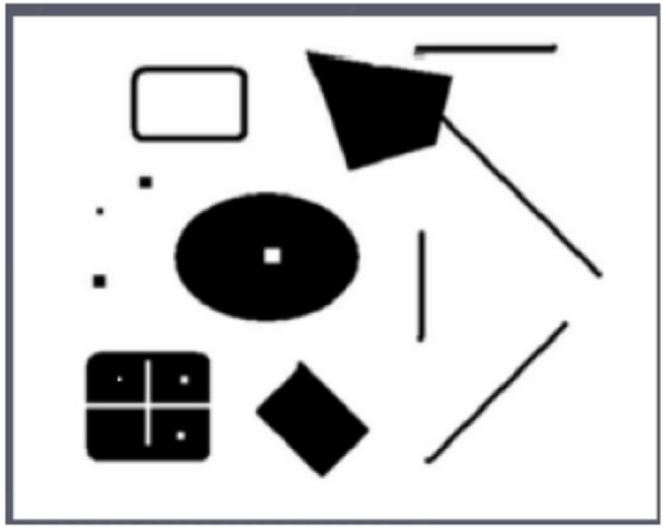
- M is a 2x2 second moment matrix computed from image gradients

$$M = \sum_x \sum_y w(x,y) \begin{bmatrix} I_x^2 & I_x \cdot I_y \\ I_x \cdot I_y & I_y^2 \end{bmatrix}$$

How Do I Compute M ?

How Do I Compute M ?

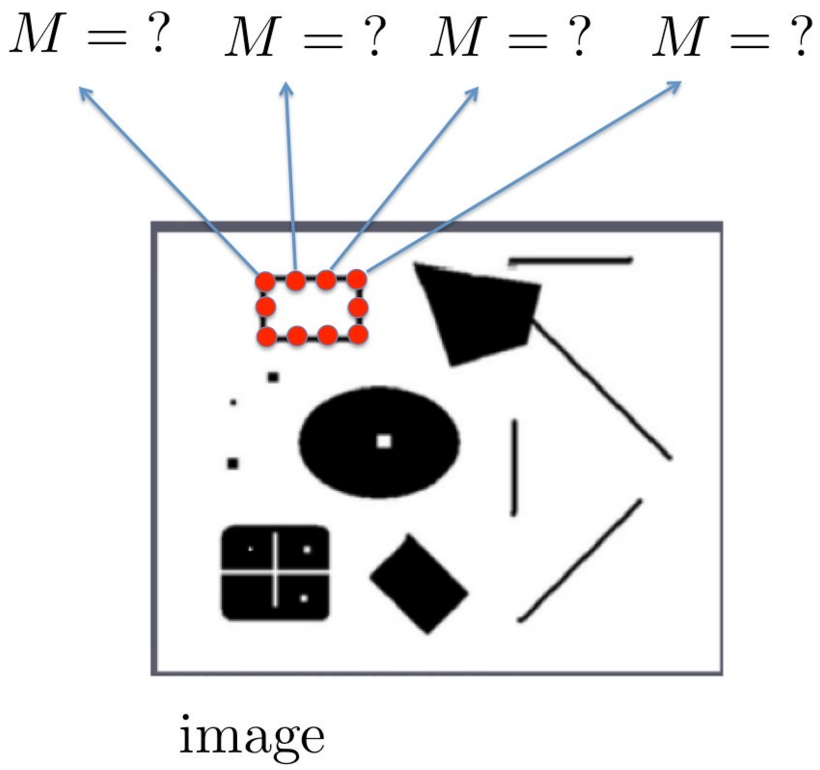
- Let’s say I have this image



image

How Do I Compute M ?

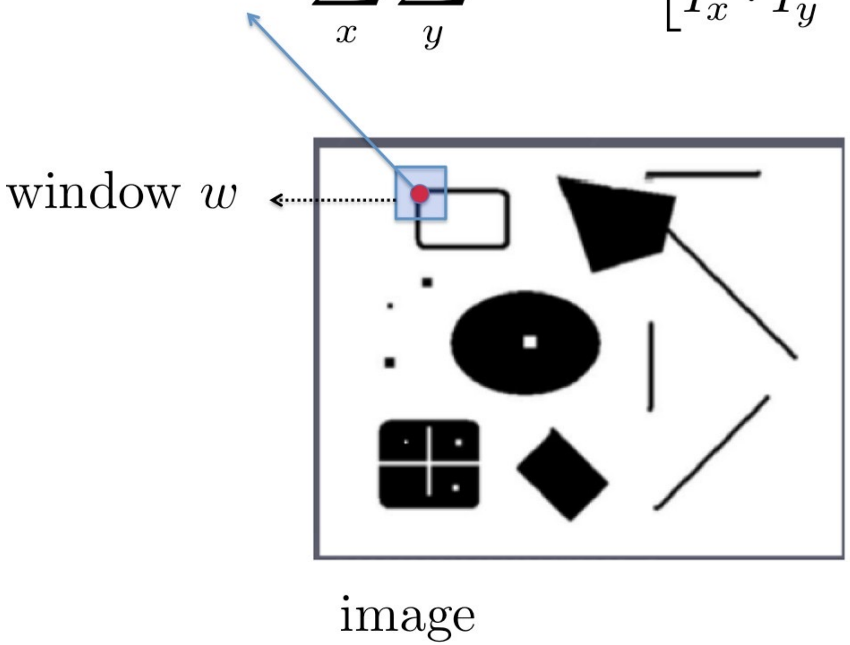
- Let's say I have this image
- I need to compute a 2 × 2 second moment matrix in each image location



How Do I Compute M ?

- Let's say I have this image
- I need to compute a 2 × 2 second moment matrix in each image location
- In a particular location I need to compute M as a weighted average of gradients in a window

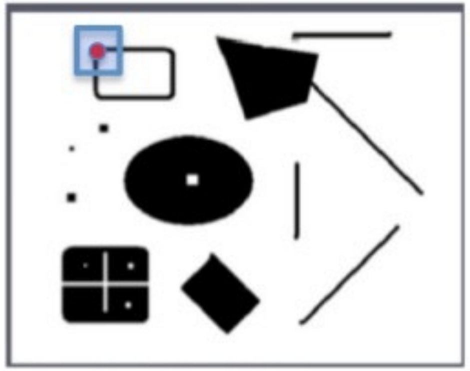
$$M = \sum_x \sum_y w(x, y) \begin{bmatrix} I_x^2 & I_x \cdot I_y \\ I_x \cdot I_y & I_y^2 \end{bmatrix}$$



How Do I Compute M ?

- Let’s say I have this image
- I need to compute a 2 × 2 second moment matrix in each image location
- In a particular location I need to compute M as a weighted average of gradients in a window

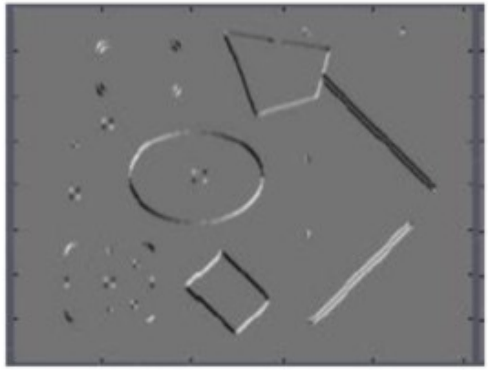
$$M = \sum_x \sum_y w(x,y) \begin{bmatrix} I_x^2 & I_x \cdot I_y \\ I_x \cdot I_y & I_y^2 \end{bmatrix}$$



image

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

$$I_y = \frac{\partial I}{\partial y}$$

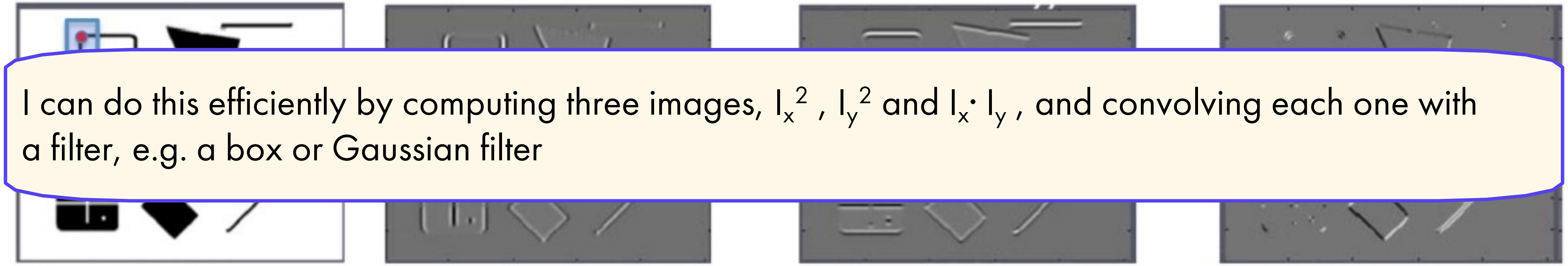


$$I_x \cdot I_y$$

How Do I Compute M ?

- Let's say I have this image
- I need to compute a 2 × 2 second moment matrix in each image location
- In a particular location I need to compute M as a weighted average of gradients in a window

$$M = \sum_x \sum_y w(x,y) \begin{bmatrix} I_x^2 & I_x \cdot I_y \\ I_x \cdot I_y & I_y^2 \end{bmatrix}$$



I can do this efficiently by computing three images, I_x^2 , I_y^2 and $I_x \cdot I_y$, and convolving each one with a filter, e.g. a box or Gaussian filter

image

$$I_x = \frac{\partial I}{\partial x} \qquad I_y = \frac{\partial I}{\partial y} \qquad I_x \cdot I_y$$

How Do I Compute M ?

- Let's take a "slice" of $E_{WSSD}(u, v)$:

$$\begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix} = \text{const}$$

what is this the equation for?

How Do I Compute M ?

- Let's take a "slice" of $E_{WSSD}(u, v)$:

$$\begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix} = \text{const}$$

- This is the equation of an ellipse

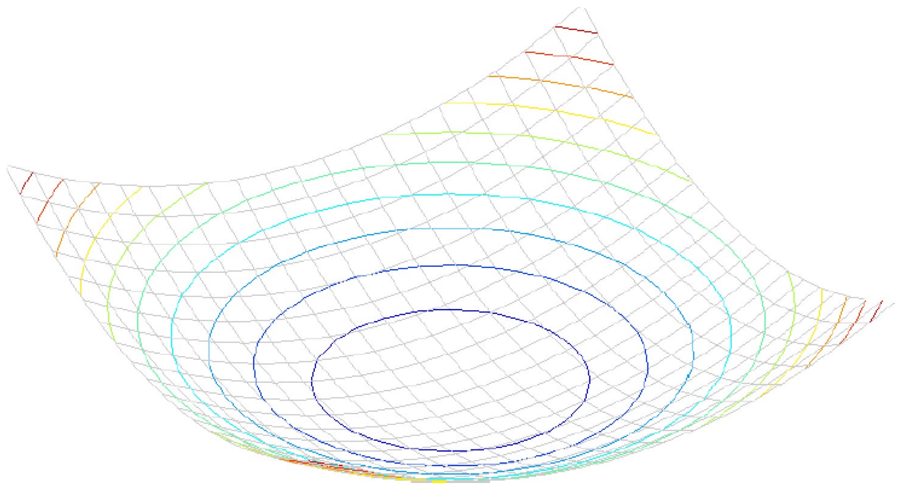


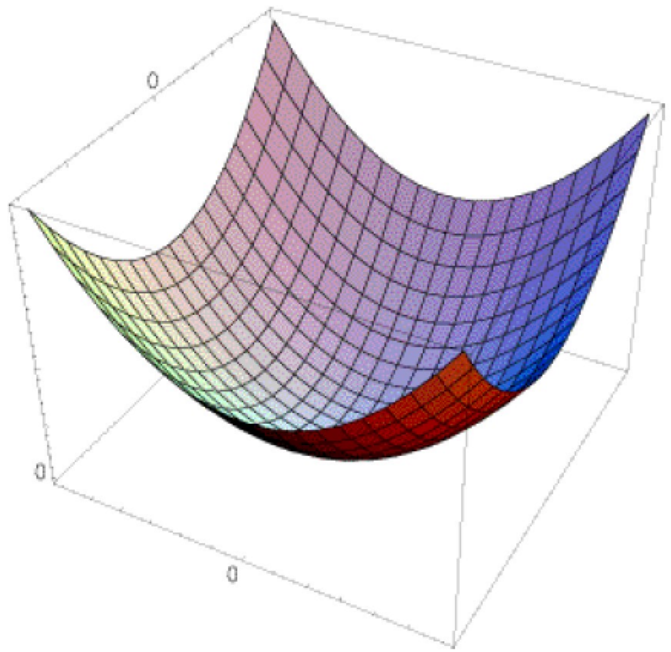
Figure: Different ellipses obtain by different horizontal "slices"

How Do I Compute M ?

- We now have *M* computed in each image location
- Our E_{WSSD} is a quadratic function where *M* implies its shape

$$E_{\text{WSSD}}(u, v) \;=\; \begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix}$$

$$M = \sum_x \sum_y w(x, y) \begin{bmatrix} I_x^2 & I_x \cdot I_y \\ I_x \cdot I_y & I_y^2 \end{bmatrix}$$



[Source: J. Hays]

How Do I Compute M ?

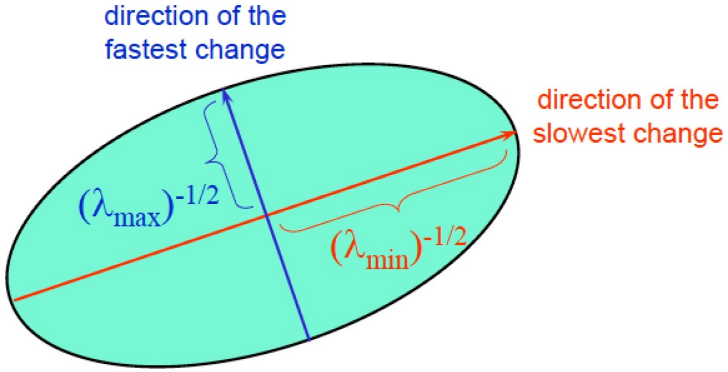
- Our matrix M is symmetric:

$$M = \sum_x \sum_y w(x,y) \begin{bmatrix} I_x^2 & I_x \cdot I_y \\ I_x \cdot I_y & I_y^2 \end{bmatrix}$$

- And thus we can diagonalize it (in Matlab: [V,D] = eig(M)):

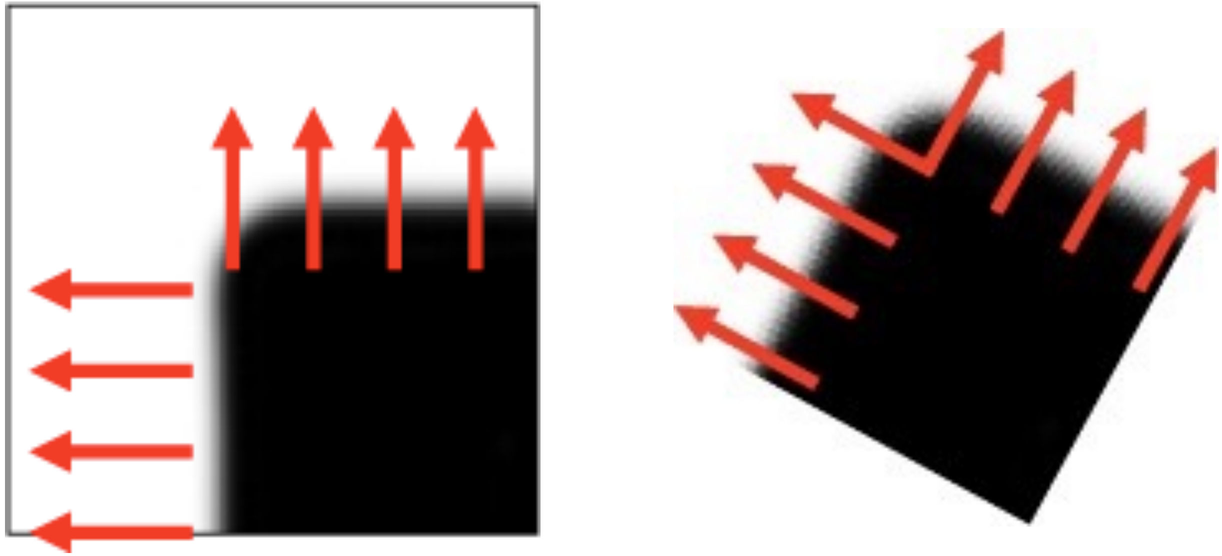
$$M = V \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} V^{-1}$$

- Columns of V are major and minor axes of ellipse, the lengths of the radii proportional to $\lambda^{-1/2}$



How Do I Compute M ?

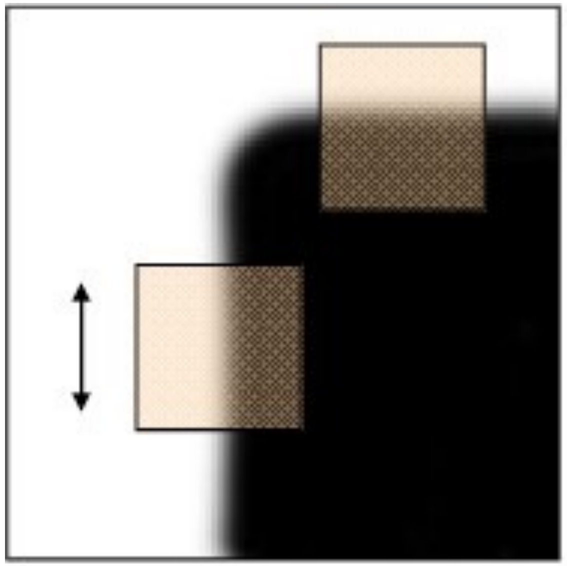
- for these images, what will the eigenvalues and eigenvectors look like?



[Source: R. Szeliski, slide credit: R. Urtasun]

How Do I Compute M ?

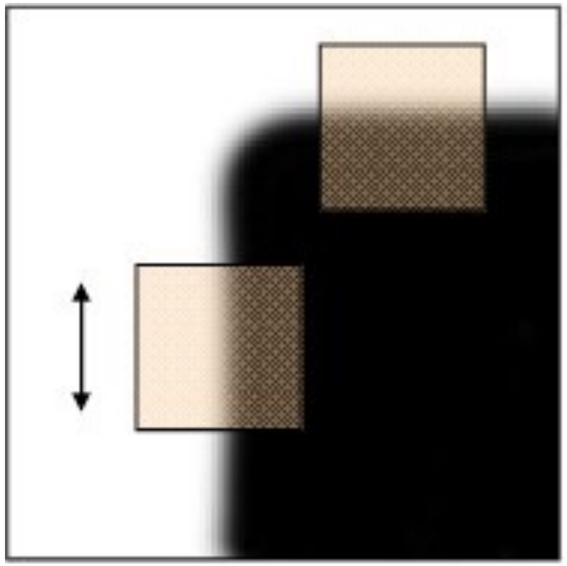
- how about for these windows?



[Source: K. Grauman, slide credit: R. Urtasun]

How Do I Compute M ?

- how about for these windows?



"edge":

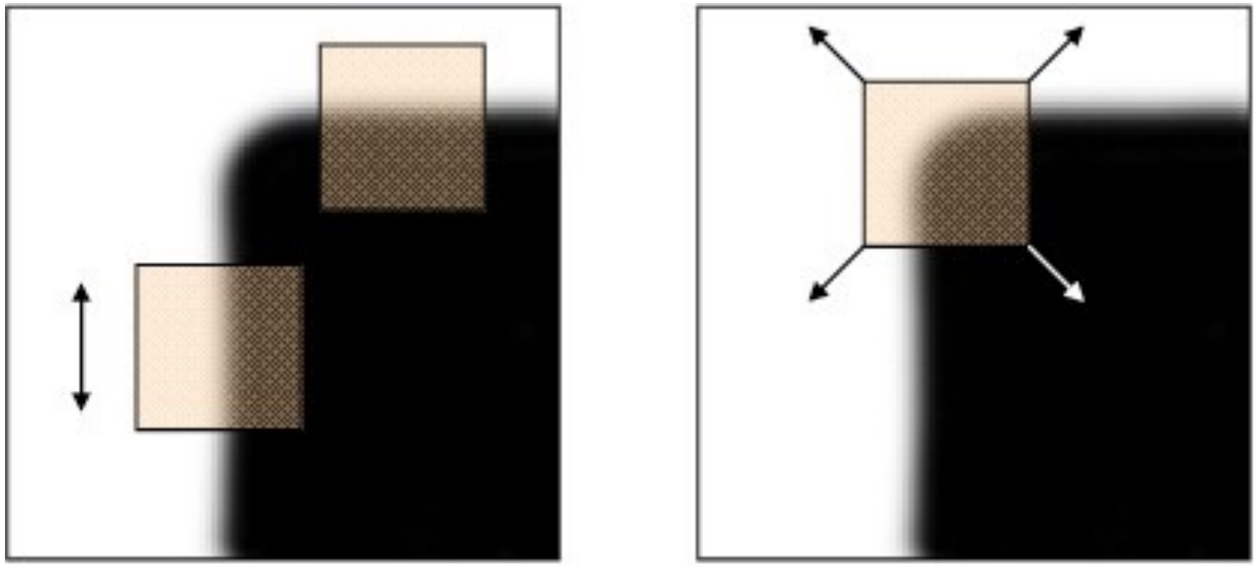
$\lambda_1 >> \lambda_2$

$\lambda_2 >> \lambda_1$

[Source: K. Grauman, slide credit: R. Urtasun]

How Do I Compute M ?

- how about for these windows?



“edge”:

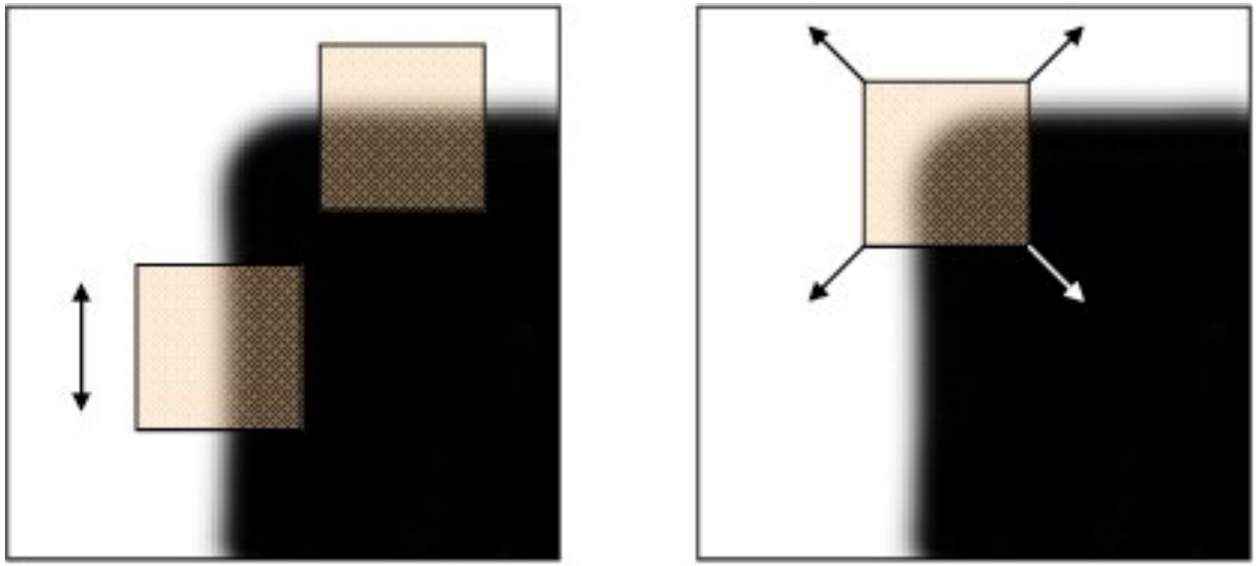
$\lambda_1 >> \lambda_2$

$\lambda_2 >> \lambda_1$

[Source: K. Grauman, slide credit: R. Urtasun]

How Do I Compute M ?

- how about for these windows?



"edge":

$\lambda_1 >> \lambda_2$

$\lambda_2 >> \lambda_1$

"corner":

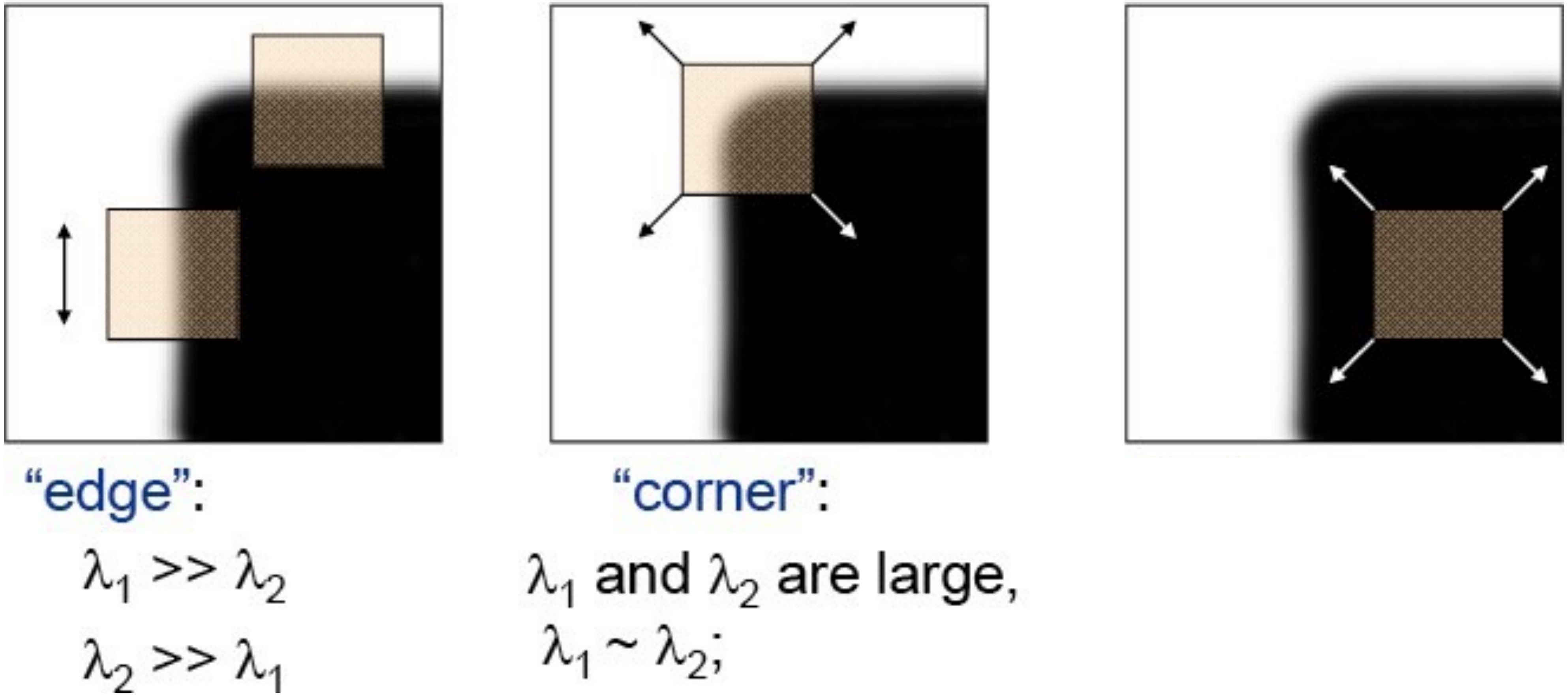
λ_1 and λ_2 are large,

$\lambda_1 \sim \lambda_2$;

[Source: K. Grauman, slide credit: R. Urtasun]

How Do I Compute M ?

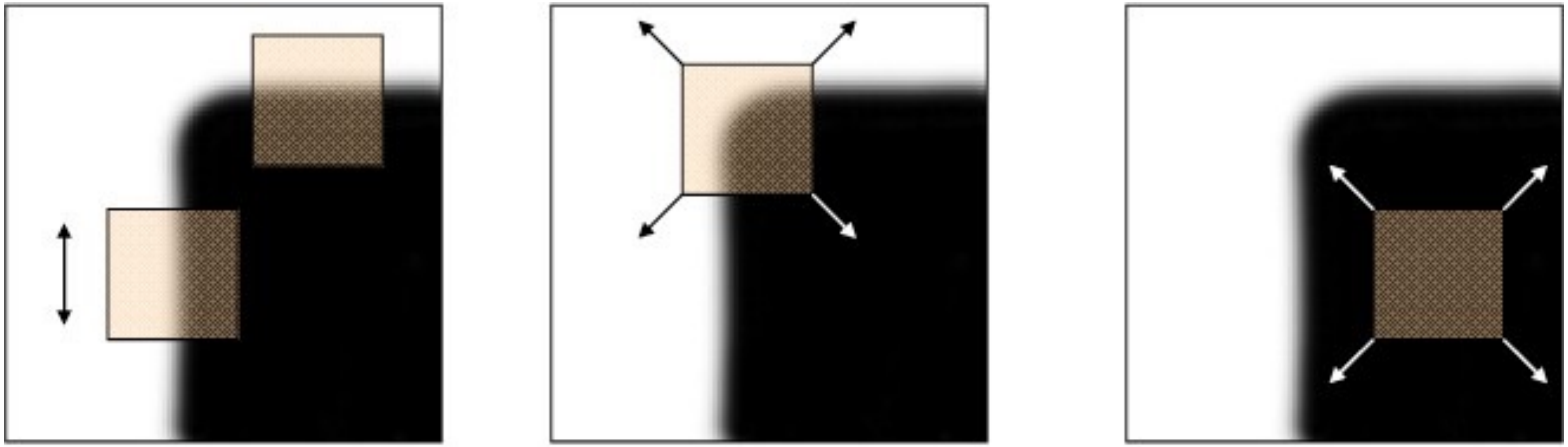
- how about for these windows?



[Source: K. Grauman, slide credit: R. Urtasun]

How Do I Compute M ?

- how about for these windows?



"edge":

$\lambda_1 >> \lambda_2$

$\lambda_2 >> \lambda_1$

"corner":

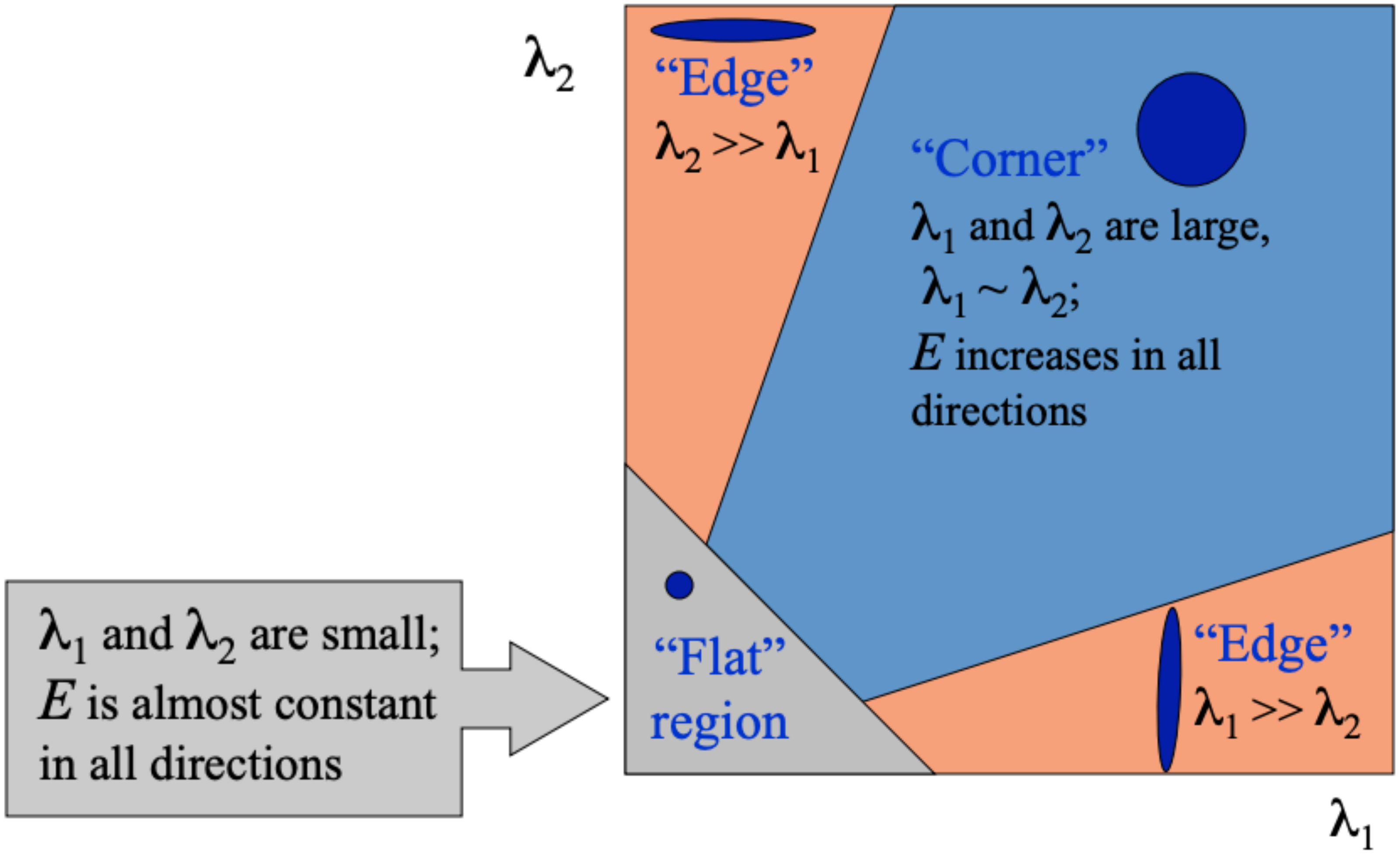
λ_1 and λ_2 are large, $\lambda_1 \sim \lambda_2$;

"flat" region

λ_1 and λ_2 are small;

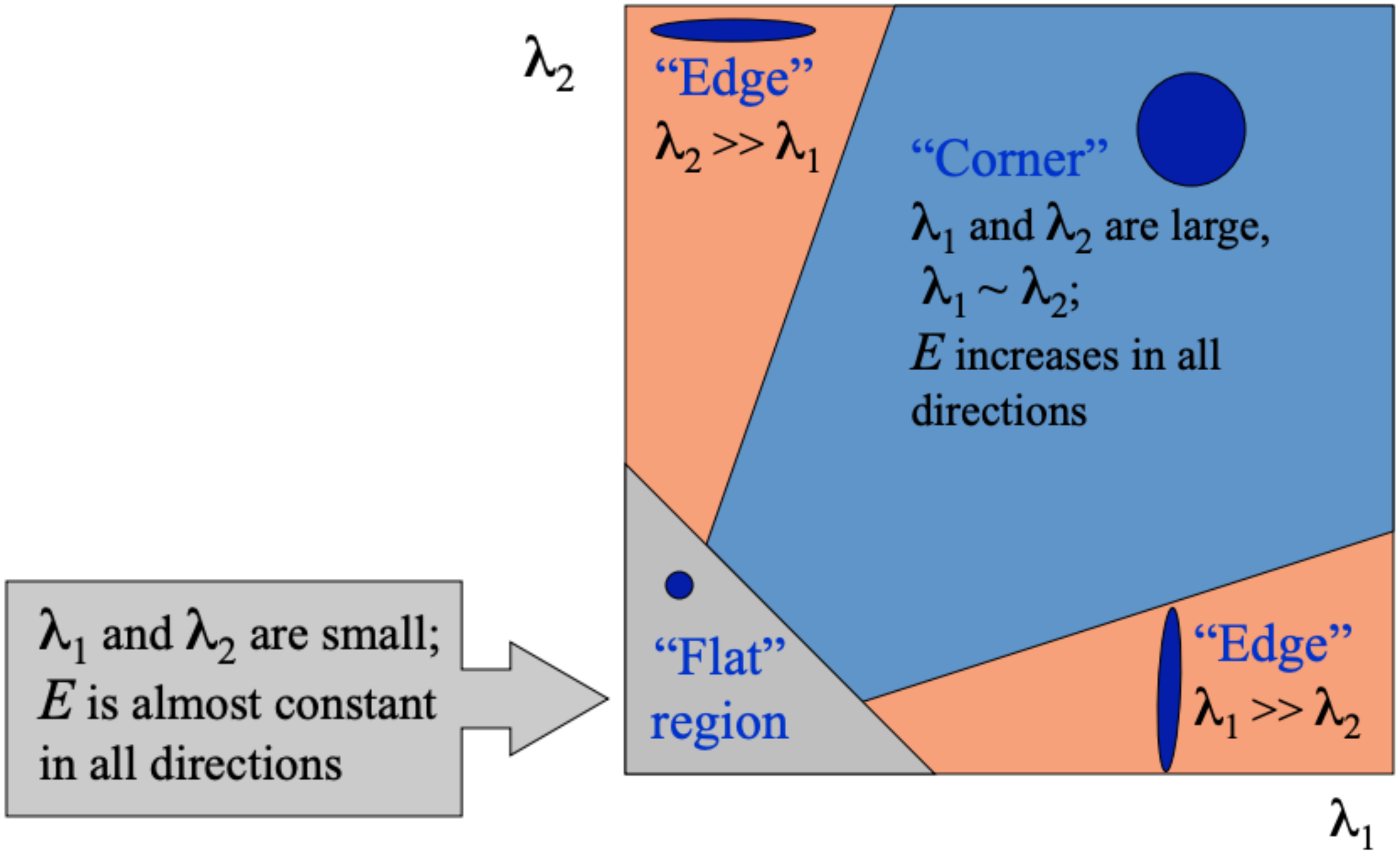
[Source: K. Grauman, slide credit: R. Urtasun]

Interest Points: Criteria to Find Corners



[Source: K. Bala]

Interest Points: Criteria to Find Corners



- can you write an equation that uses the eigenvalues to detect a corner?

[Source: K. Bala]

Interest Points: Criteria to Find Corners

- Harris and Stephens, '88, is rotationally invariant and downweighs edge-like features where $\lambda_1 \gg \lambda_0$

$$R = \lambda_0 \lambda_1 - \alpha(\lambda_0 + \lambda_1)^2 = \text{det}(M) - \alpha \cdot \text{trace}(M)^2$$

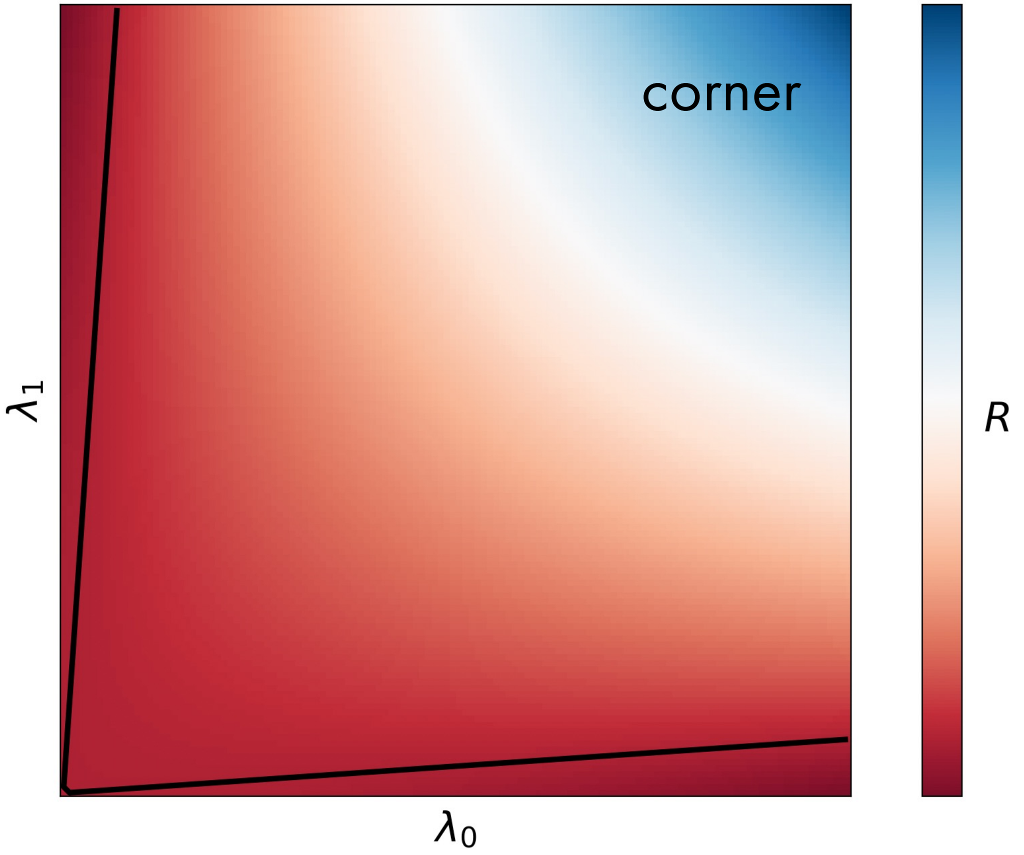
- α a constant (0.04 to 0.06)

Interest Points: Criteria to Find Corners

- Harris and Stephens, ’88, is rotationally invariant and downweighs edge-like features where $\lambda_1 \gg \lambda_0$

$$R = \lambda_0\lambda_1 - \alpha(\lambda_0 + \lambda_1)^2 = \text{det}(M) - \alpha \cdot \text{trace}(M)^2$$

- α a constant (0.04 to 0.06)

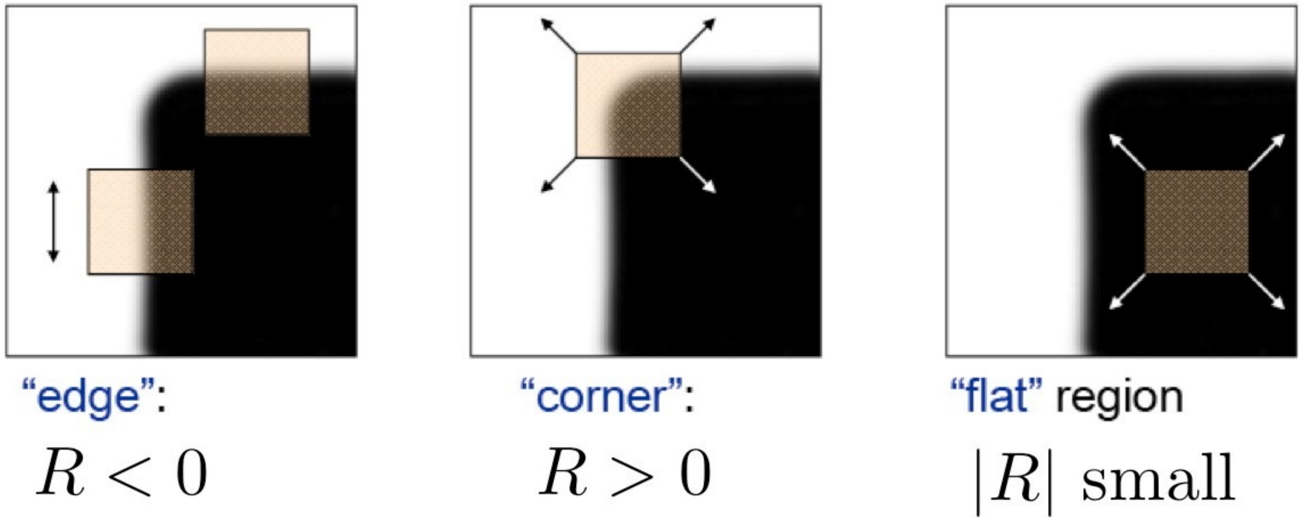


Interest Points: Criteria to Find Corners

- Harris and Stephens, '88, is rotationally invariant and downweighs edge-like features where $\lambda_1 \gg \lambda_0$

$$R = \lambda_0\lambda_1 - \alpha(\lambda_0 + \lambda_1)^2 = \mathrm{det}(M) - \alpha \cdot \mathrm{trace}(M)^2$$

- α a constant (0.04 to 0.06)



- The corresponding detector is called Harris corner detector

Interest Points: Criteria to Find Corners

- Harris & Stephens (1998)

$$R = \lambda_0 \lambda_1 - \alpha(\lambda_0 + \lambda_1)^2 = \mathrm{det}(M) - \alpha \cdot \mathrm{trace}(M)^2$$

- Kande & Tomasi (1994)

$$R = \min(\lambda_1, \lambda_2)$$

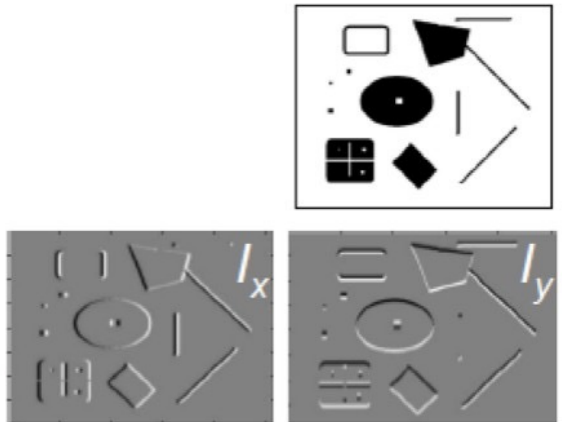
- Nobel (1998)

$$R = \frac{\det(M)}{\mathrm{trace}(M) + \epsilon}$$

[Source Mubarak Shah, Szelski]

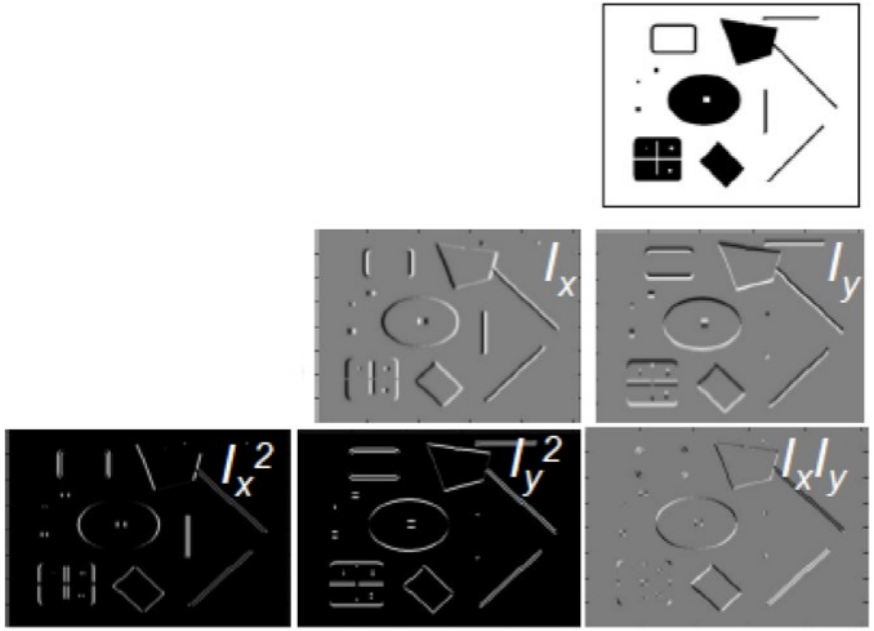
Harris Corner detector

1. Compute gradients I_x and I_y



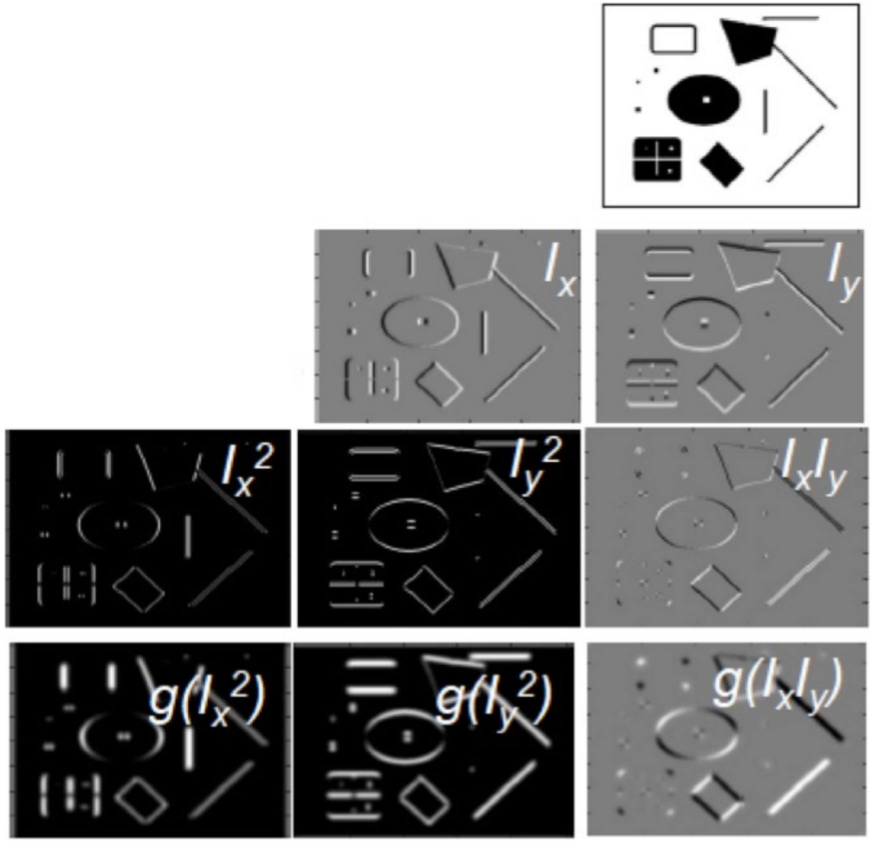
Harris Corner detector

1. Compute gradients I_x and I_y
2. Compute I_x^2, I_y^2 , $I_x \cdot I_y$



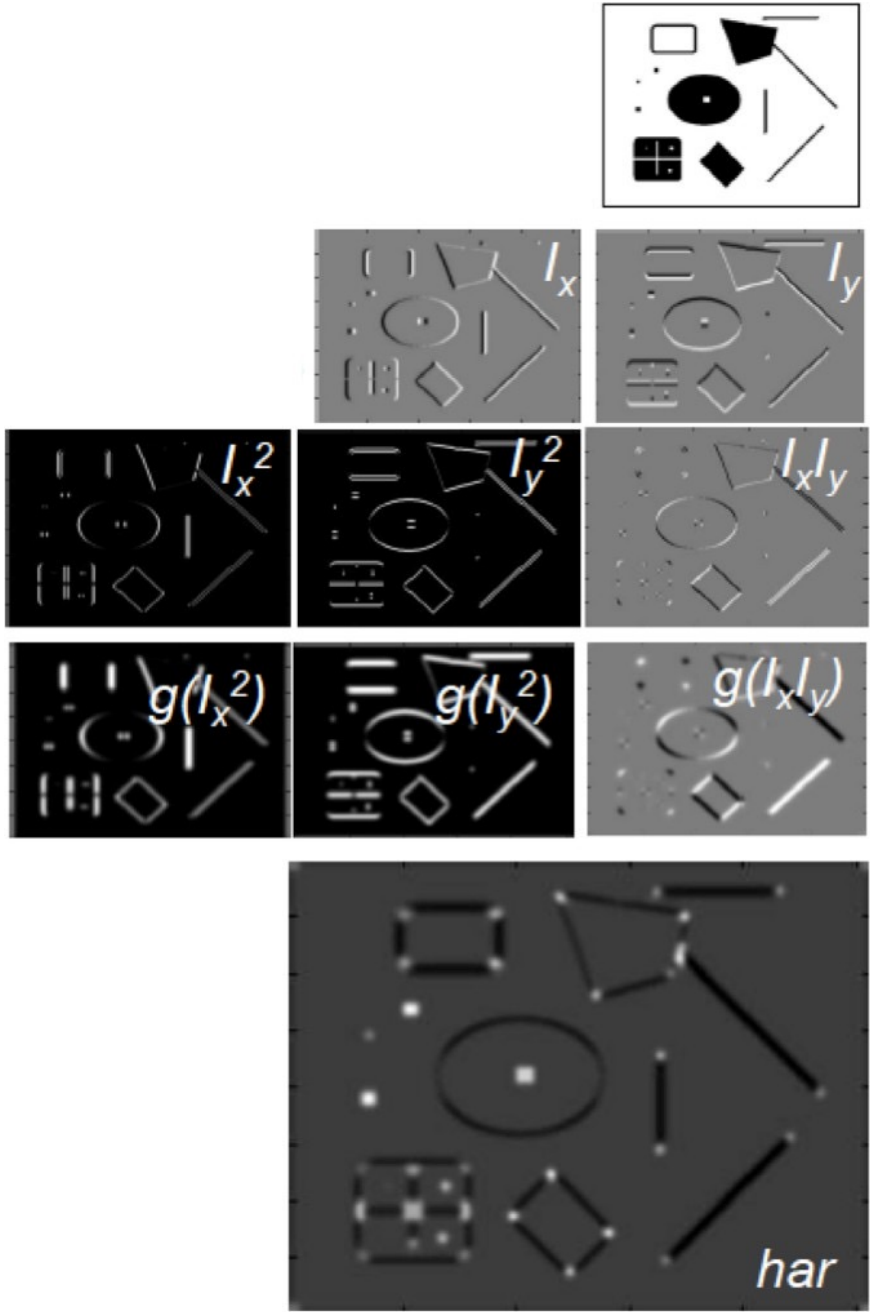
Harris Corner detector

1. Compute gradients I_x and I_y
2. Compute I_x^2, I_y^2 , $I_x \cdot I_y$
3. Average (Gaussian) → gives M per voxel



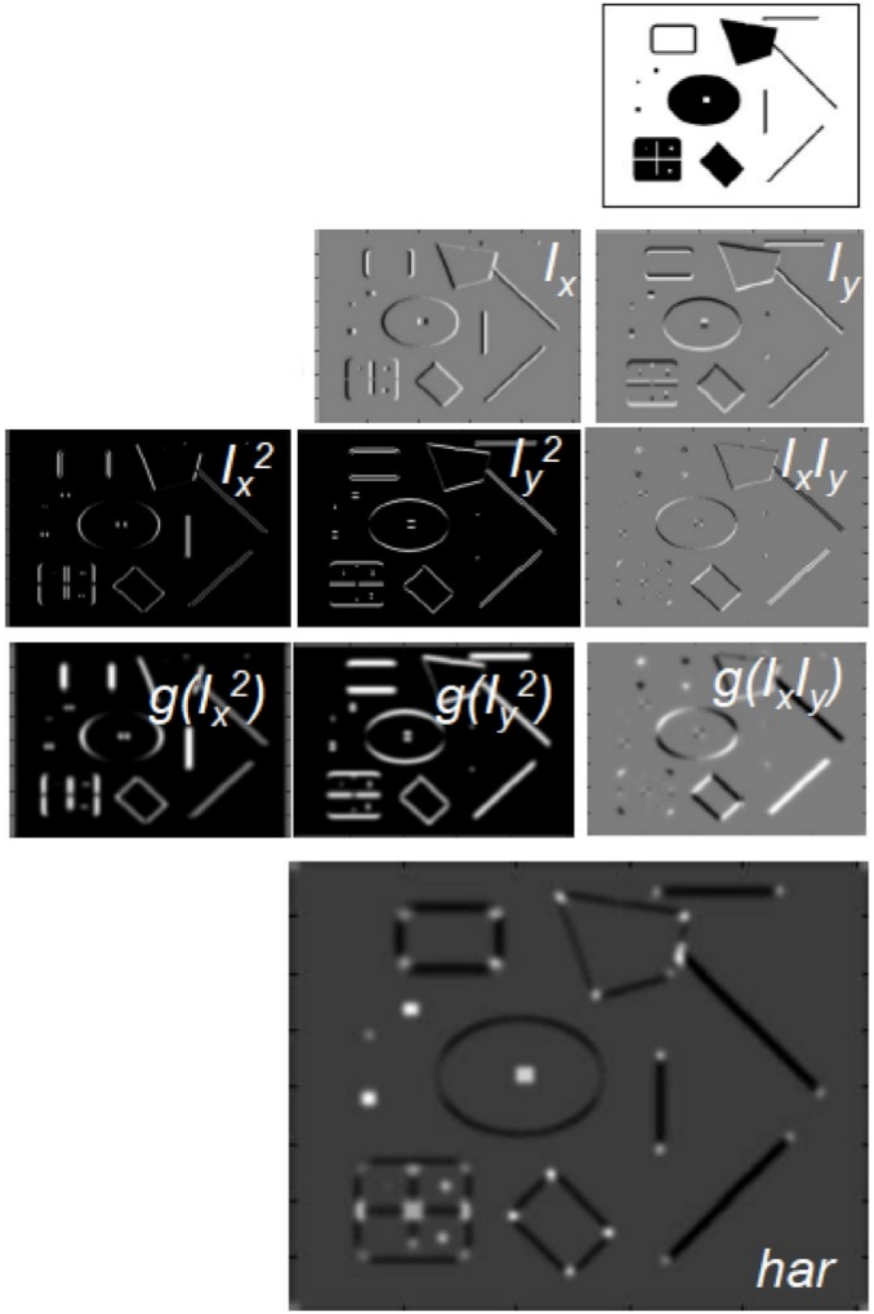
Harris Corner detector

1. Compute gradients I_x and I_y
2. Compute I_x^2, I_y^2 , $I_x \cdot I_y$
3. Average (Gaussian) → gives M per voxel
4. Compute $R = \det(M) - \alpha\, \mathrm{trace}(M)^2$ for each image window (cornerness score)



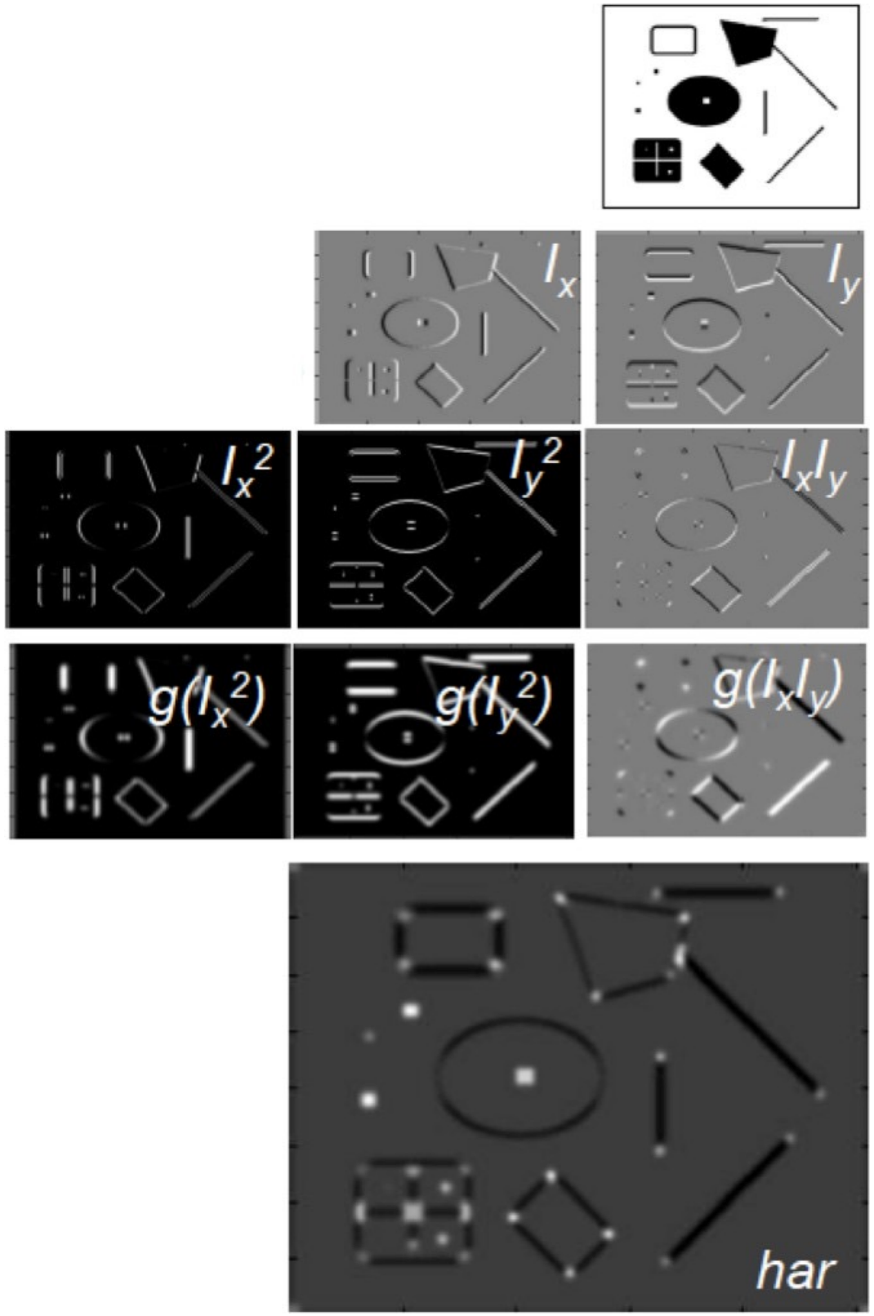
Harris Corner detector

1. Compute gradients I_x and I_y
2. Compute I_x^2, I_y^2 , $I_x \cdot I_y$
3. Average (Gaussian) $\rightarrow$ gives M per voxel
4. Compute $R = \det(M) - \alpha\,\mathrm{trace}(M)^2$ for each image window (cornerness score)
5. Find points with large R ($R >$ threshold).



Harris Corner detector

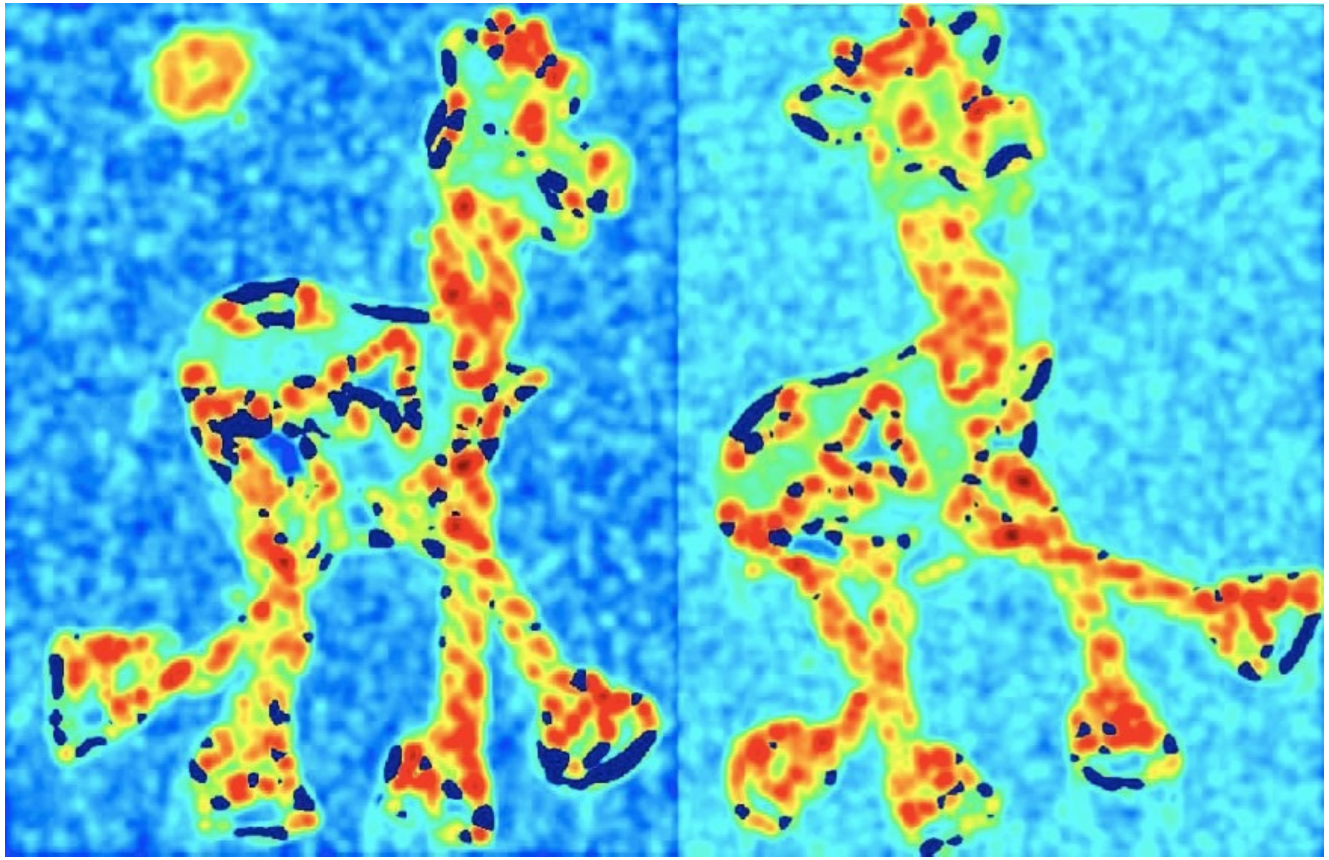
1. Compute gradients I_x and I_y
2. Compute I_x^2, I_y^2 , $I_x \cdot I_y$
3. Average (Gaussian) → gives M per voxel
4. Compute $R = \det(M) - \alpha \, \mathrm{trace}(M)^2$ for each image window (cornerness score)
5. Find points with large R ($R >$ threshold).
6. Take only points of local maxima, i.e., perform non-maximum suppression



Example

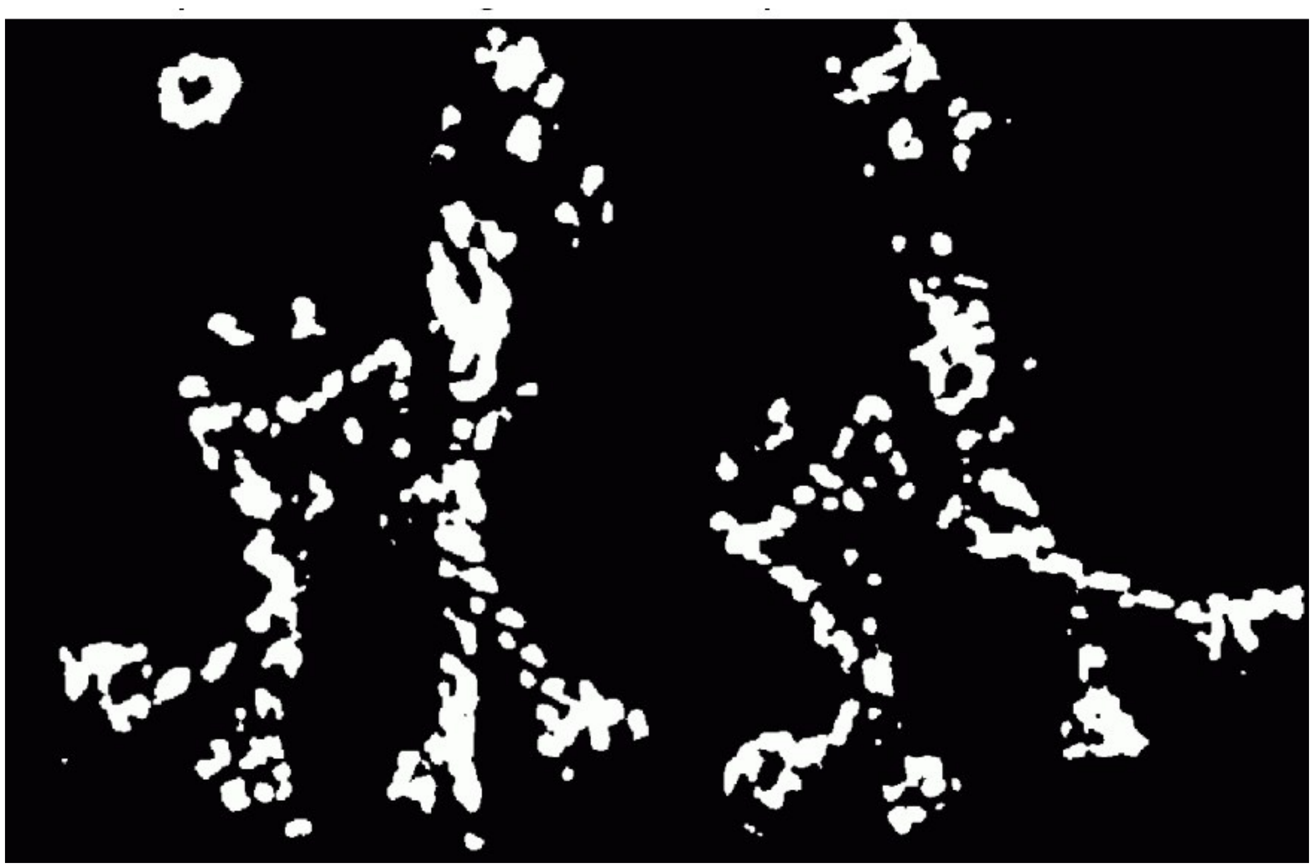
[Source: K. Grauman]

1) Compute Cornerness



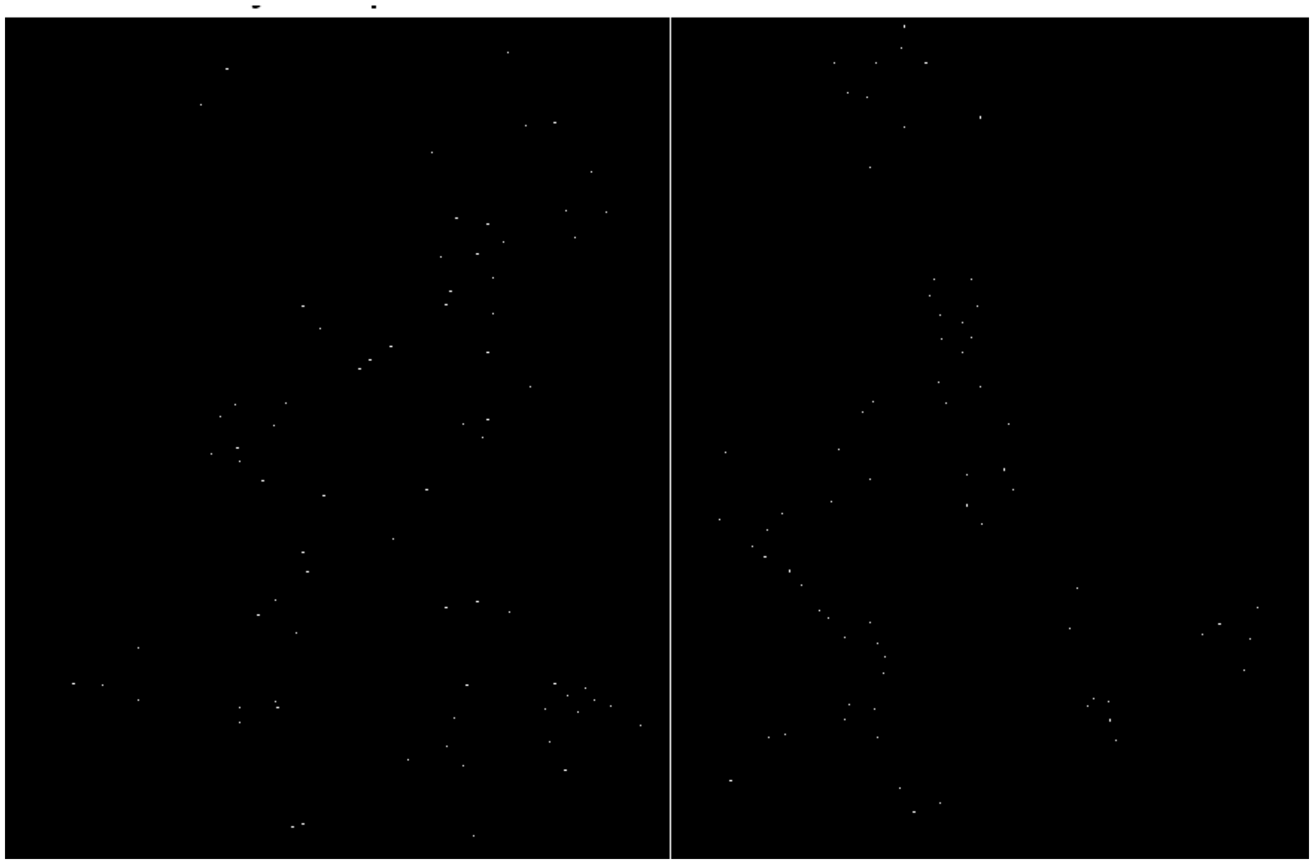
[Source: K. Grauman]

2) Find High Response



[Source: K. Grauman]

3) Non-maxima Suppresion



[Source: K. Grauman]

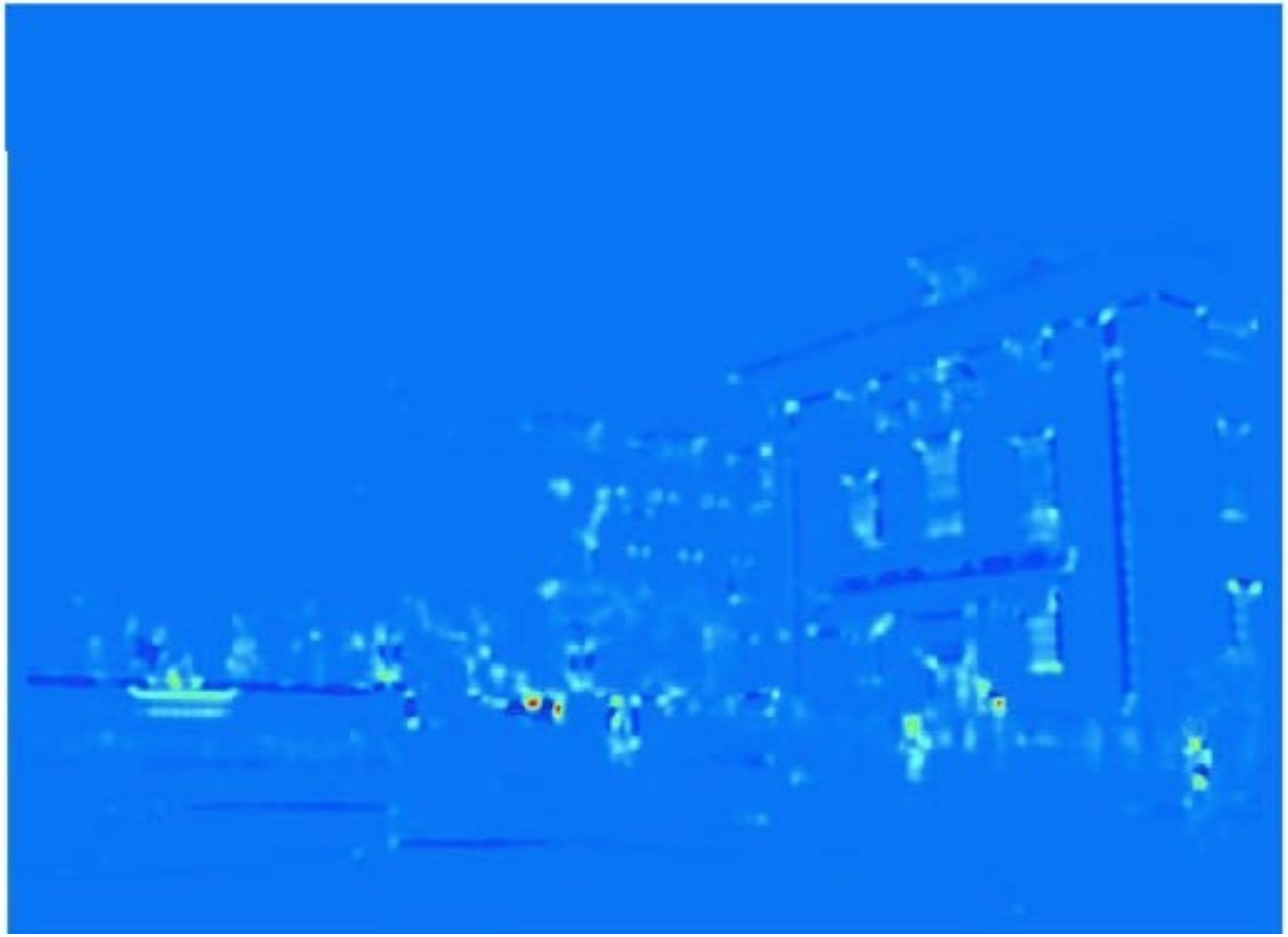
Results

[Source: K. Grauman]

Another Example

[Source: K. Grauman]

Cornerness



[Source: K. Grauman]

Interest Points

[Source: K. Grauman]

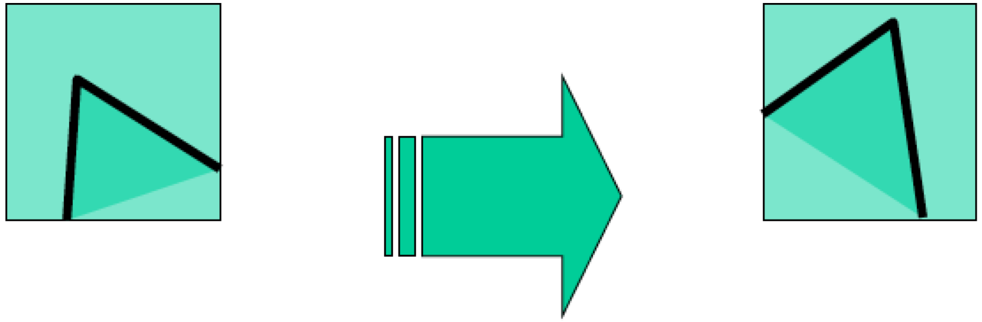
Properties of Harris Corner Detector

- Is the Harris corner detector rotation invariant? Shift invariant?

[Source: J. Hays]

Properties of Harris Corner Detector

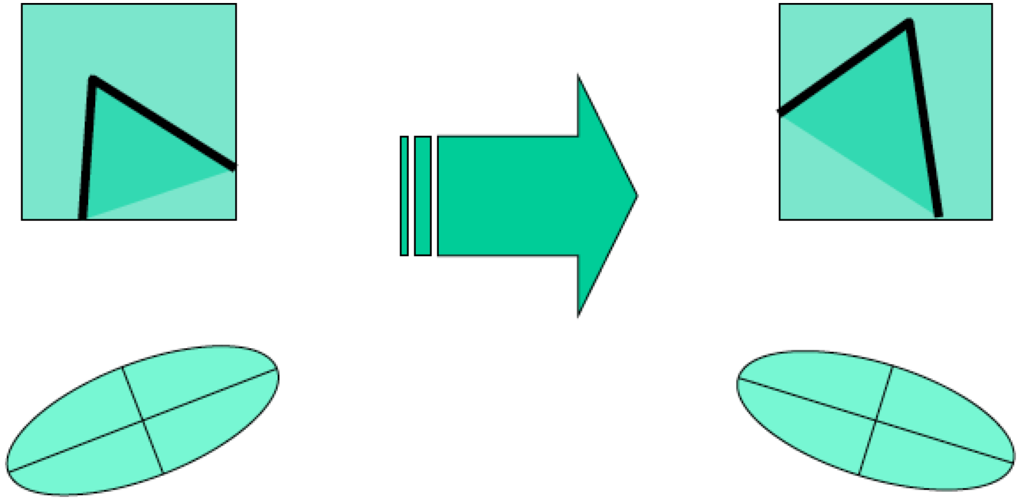
- Is the Harris corner detector rotation invariant? Shift invariant?



[Source: J. Hays]

Properties of Harris Corner Detector

- Is the Harris corner detector rotation invariant? Shift invariant?

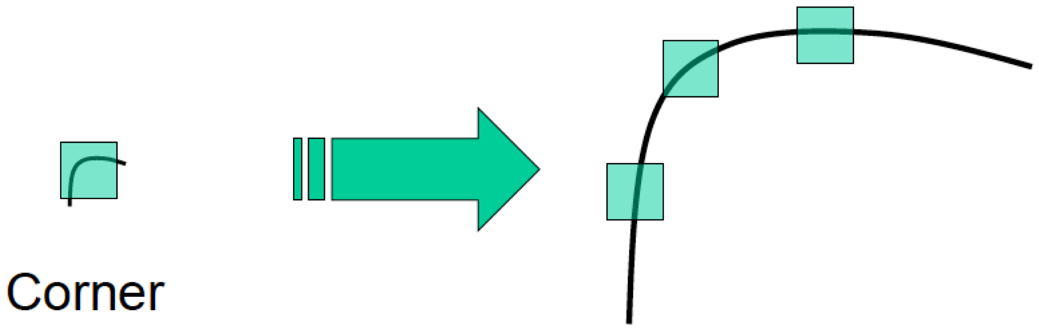


- Second moment ellipse rotates but its shape (i.e. eigenvalues) remains the same
- Harris corner detector is rotation-covariant
- what about scale?

[Source: J. Hays]

Properties of Harris Corner Detector

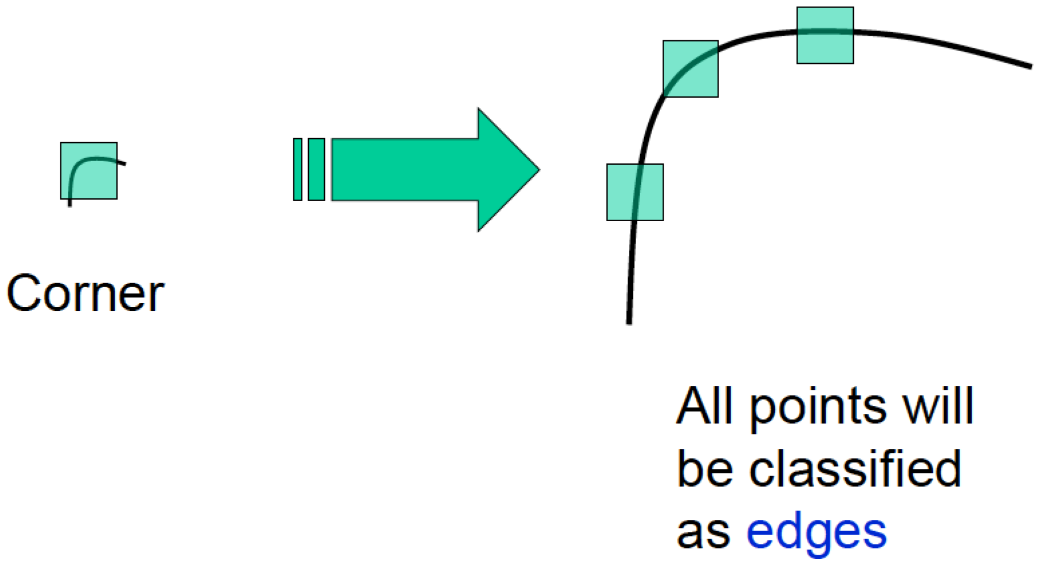
- Scale?



[Source: J. Hays]

Properties of Harris Corner Detector

- Scale?



- Corner location is not scale invariant/covariant!

[Source: J. Hays]

Optical Flow

Slide Credit: Ali Farhadi

We live in a moving world

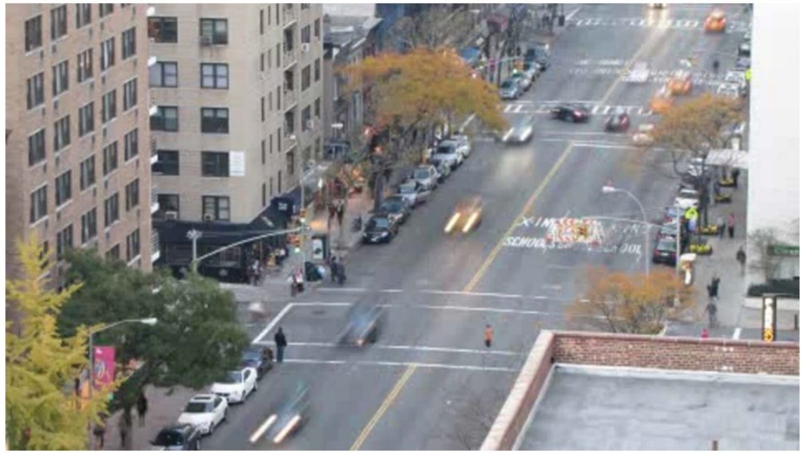
- Perceiving, understanding and predicting motion is an important part of our daily lives

Motion scenarios (priors)

Static camera, moving scene

Moving camera, static scene

Moving camera, moving scene



Static camera, moving scene, moving light

How can we recover motion?

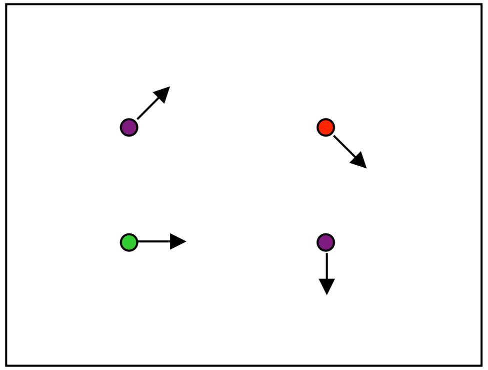
- Extract visual features (corners, textured areas) and “track” them over multiple frames.

- Recover image motion at each pixel from spatio-temporal image brightness variations (optical flow).

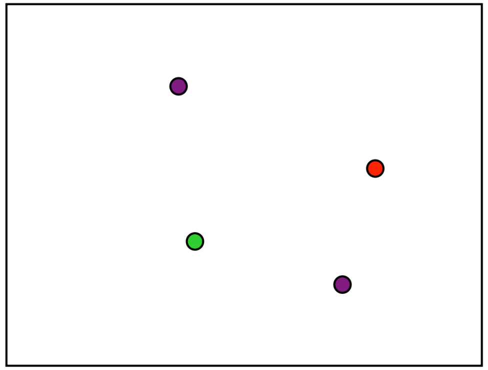
B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In Proceedings of the International Joint Conference on Artificial Intelligence, pp. 674–679, 1981.

Jonschkowski et al. 2020]

Feature tracking



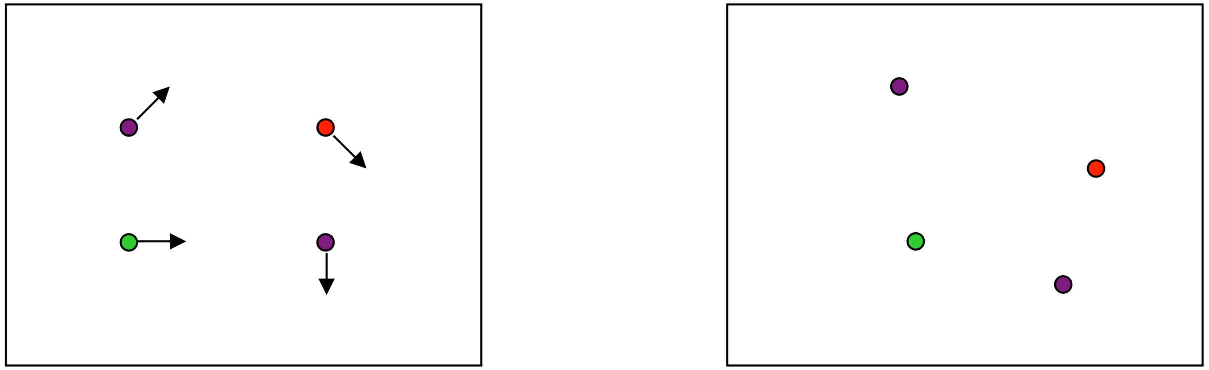
$I(x,y,t)$



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

- Given two subsequent frames, estimate the point translation

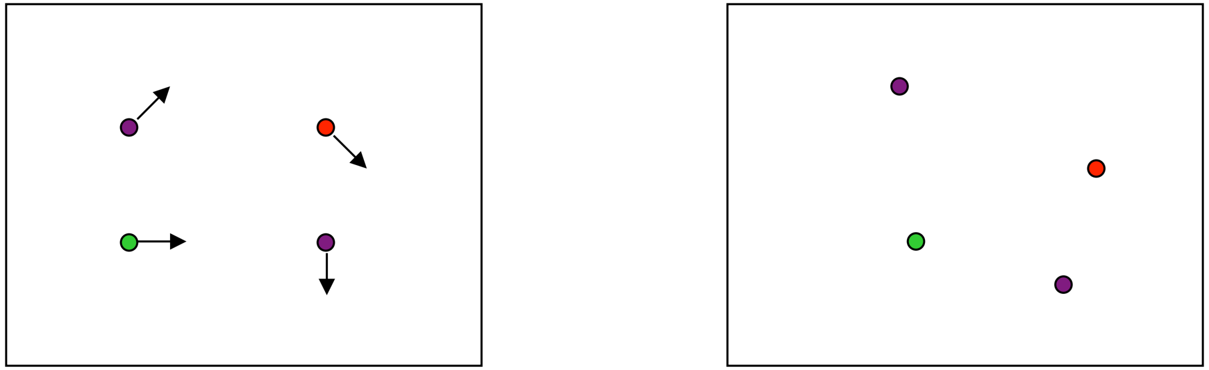
Feature tracking



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

- Given two subsequent frames, estimate the point translation
- Key assumptions:
  - 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

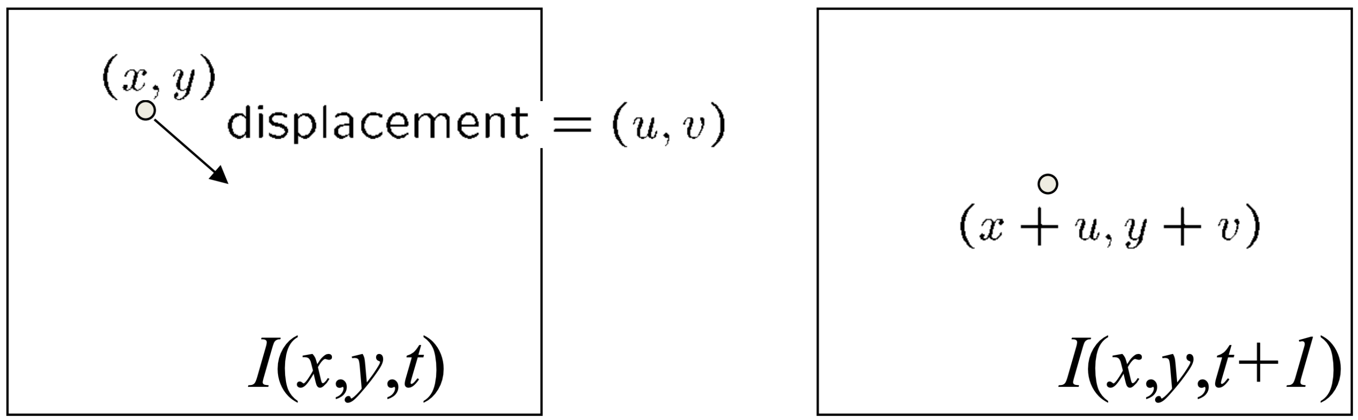
Feature tracking



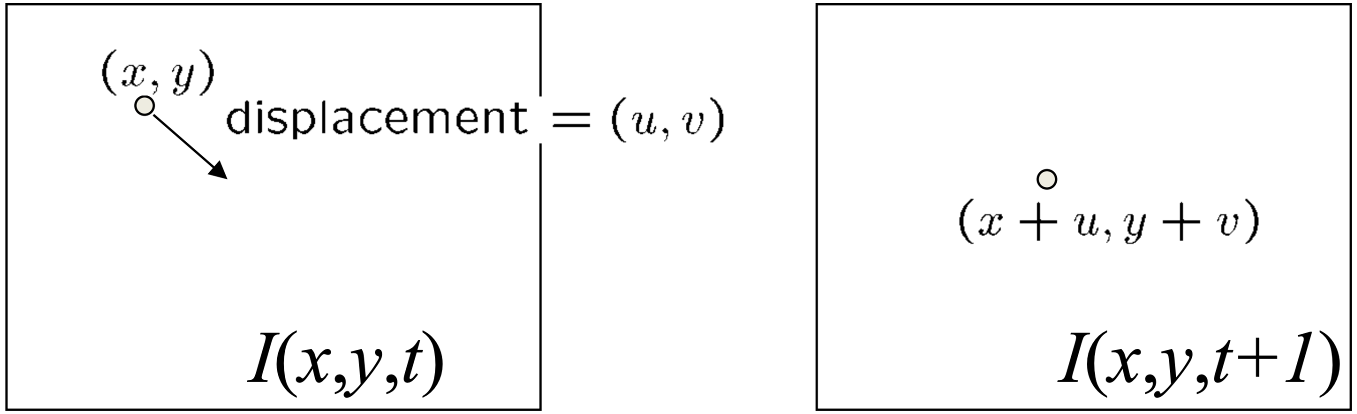
$I(x,y,t)$ $\quad$ $I(x,y,t+1)$

- Given two subsequent frames, estimate the point translation
- Key assumptions:
  - **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

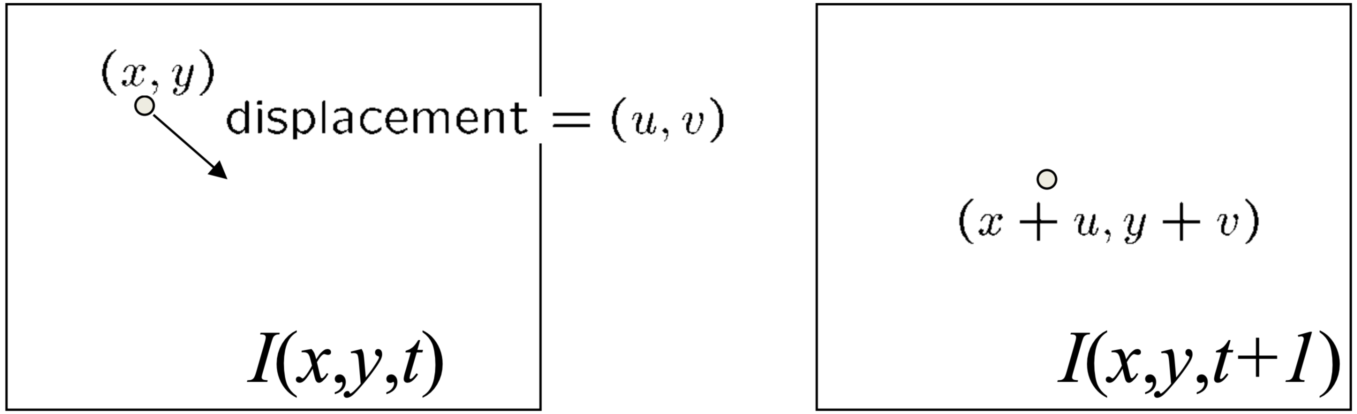


The brightness constancy constraint



Brightness Constancy Equation: $I(x, y, t) = I(x + u\ , y + v, t + 1)$

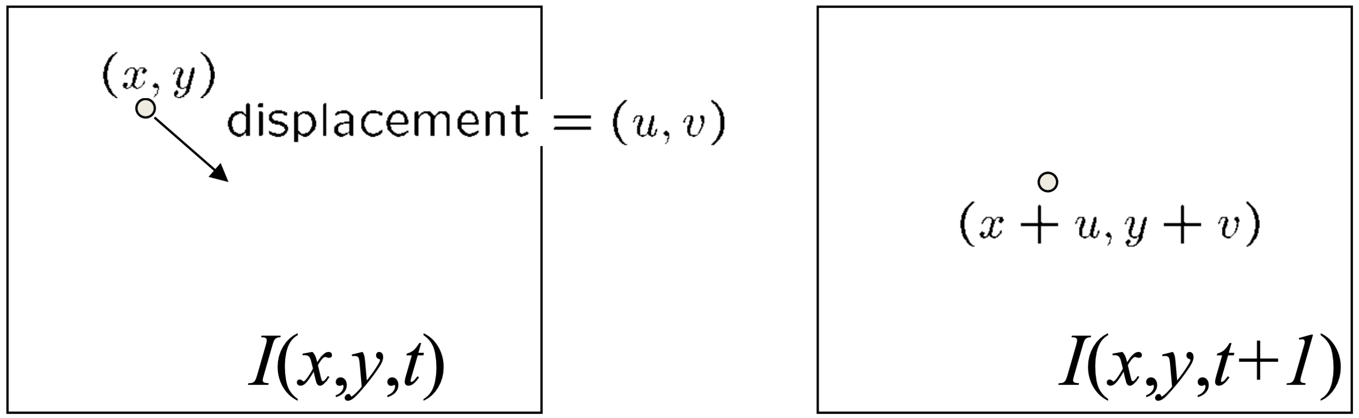
The brightness constancy constraint



Brightness Constancy Equation: $I(x, y, t) = I(x + u\ , y + v, t + 1)$

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

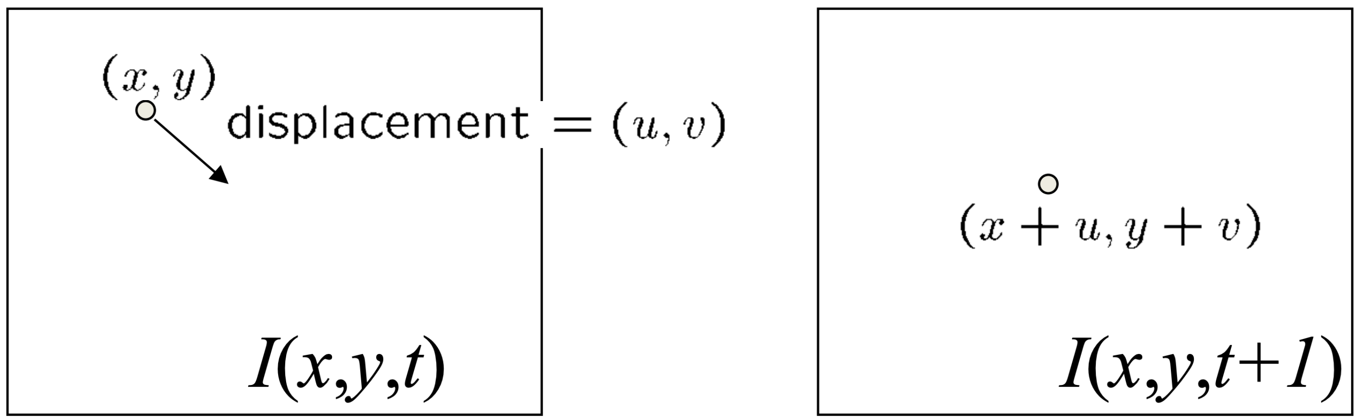
The brightness constancy constraint



Brightness Constancy Equation: $I(x, y, t) = I(x + u\ , y + v, t + 1)$

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

The brightness constancy constraint

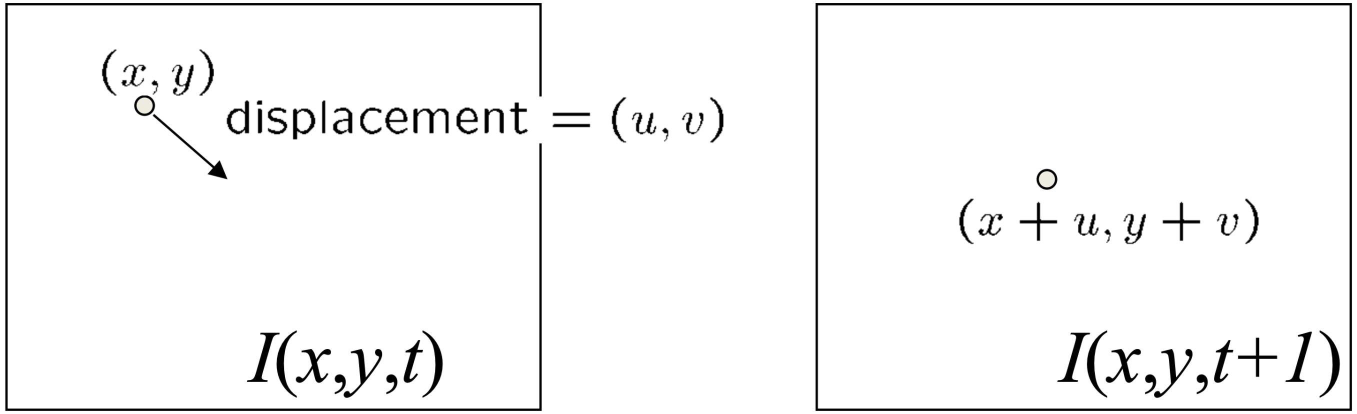


Brightness Constancy Equation: $I(x, y, t) = I(x + u\,, y + v, t + 1)$

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

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

The brightness constancy constraint



Brightness Constancy Equation: $I(x, y, t) = I(x + u\ , y + v, t + 1)$

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

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

$$\nabla I \begin{bmatrix} u \\ v \end{bmatrix} + I_t = 0$$

The brightness constancy constraint

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

$$\nabla I \,.\begin{bmatrix} u \\ v \end{bmatrix} + I_t = 0$$

- 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?

$$\nabla I \, . \begin{bmatrix} u \\ v \end{bmatrix} + I_t = 0$$

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

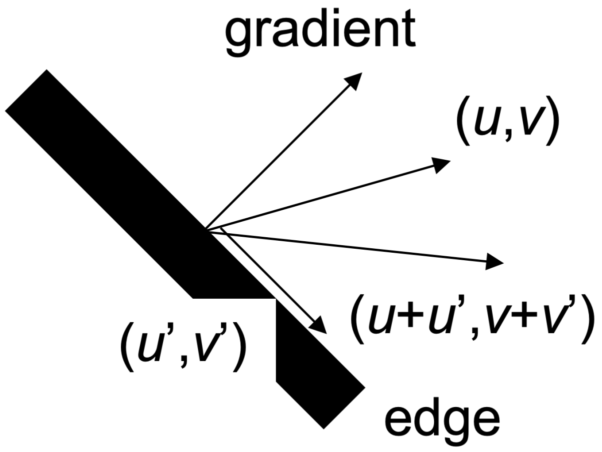
The brightness constancy constraint

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

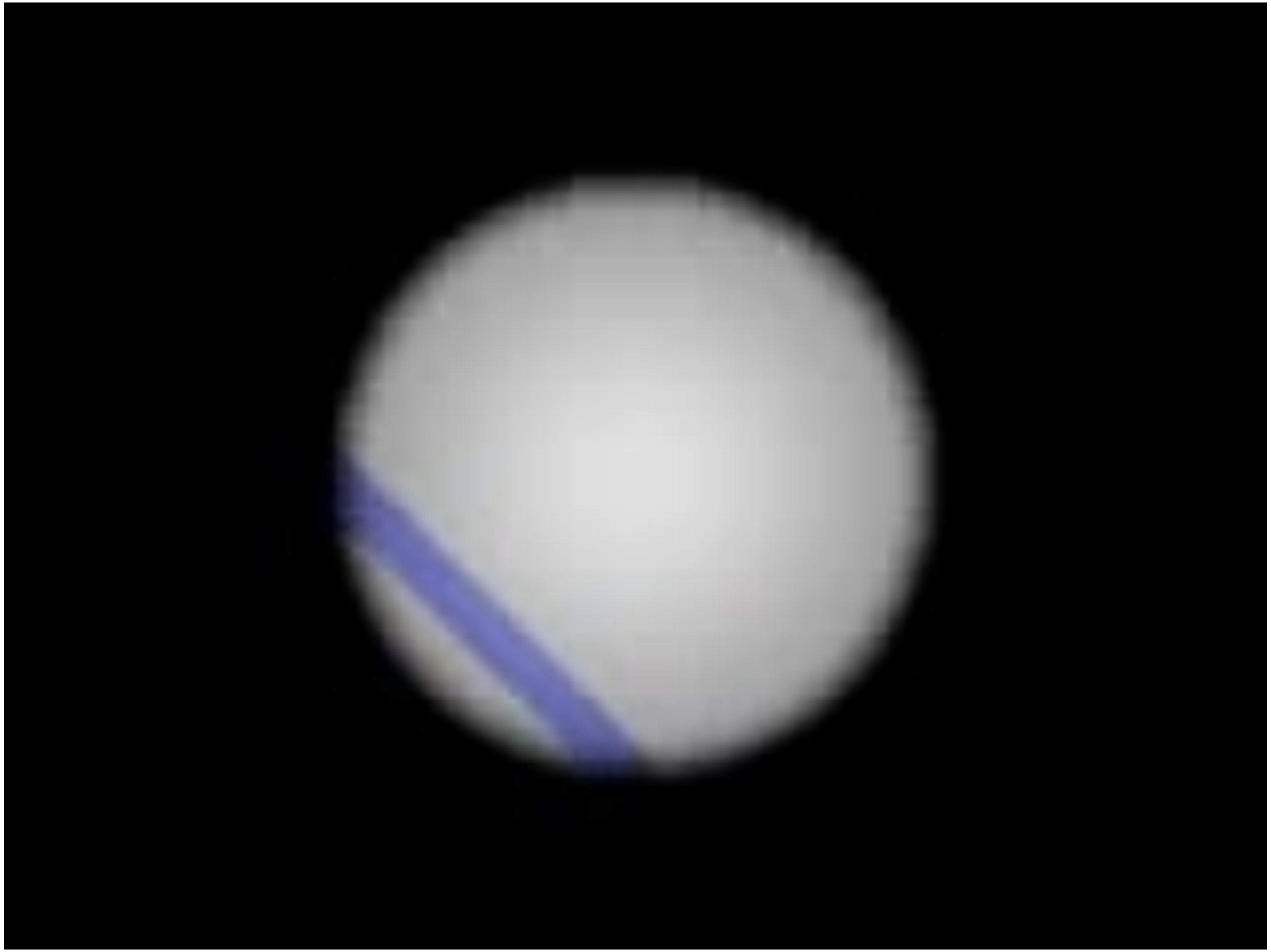
The brightness constancy constraint

- 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

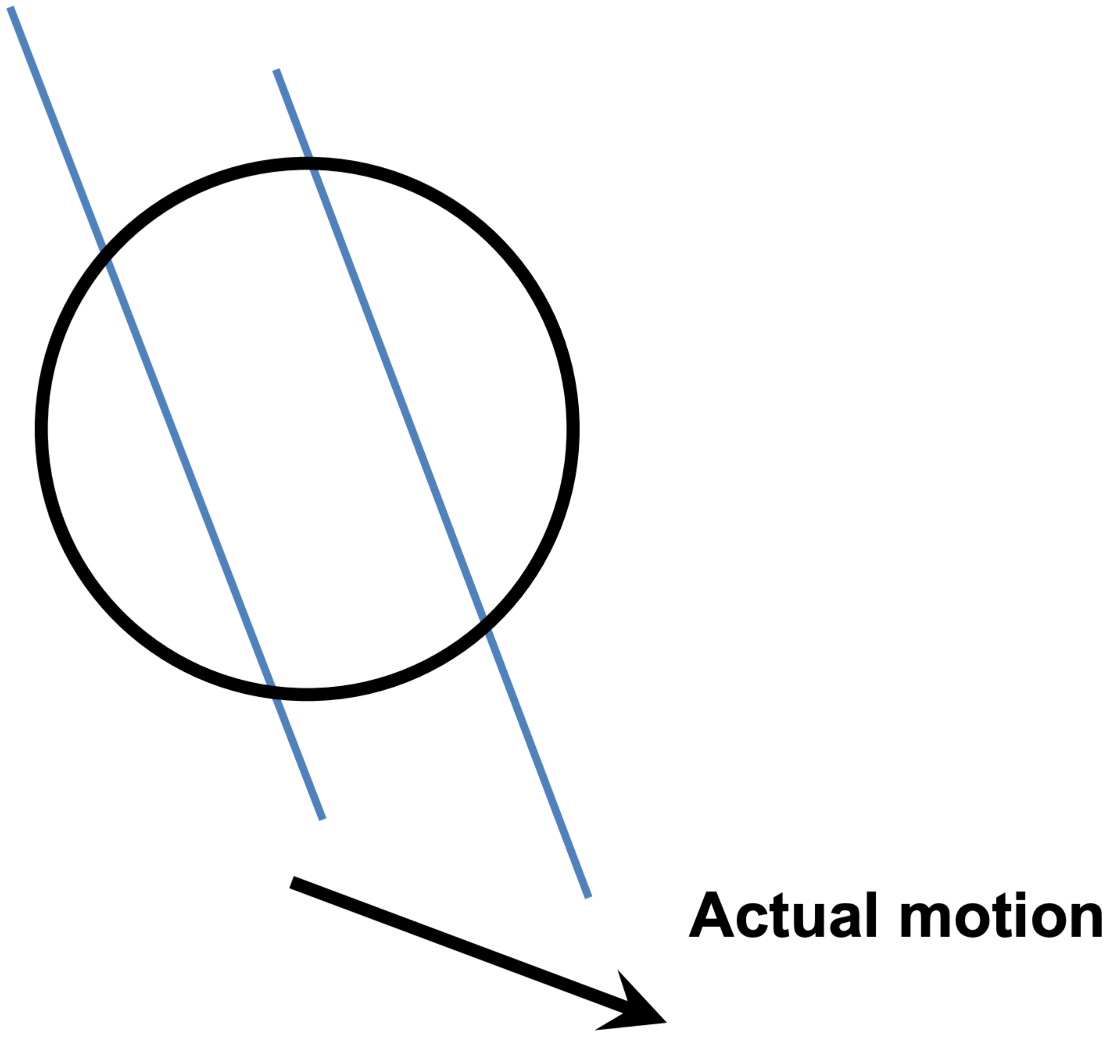
$$\nabla I \cdot \begin{bmatrix} u' \\ v' \end{bmatrix} = 0$$



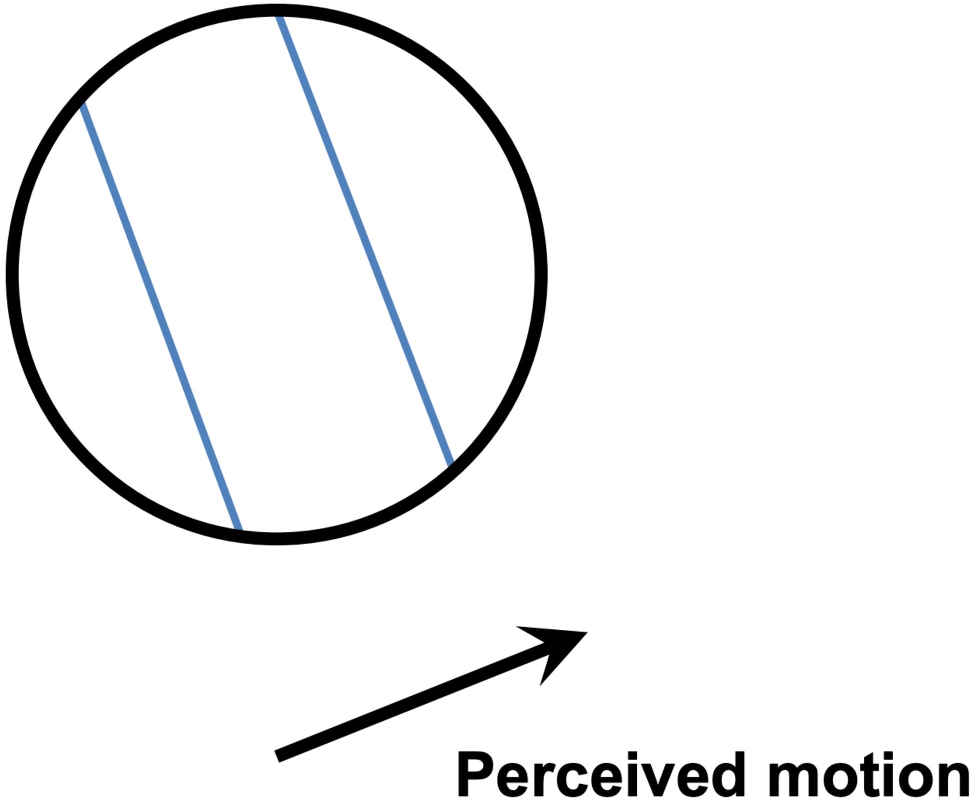
The aperture problem



The aperture problem



The aperture problem



The barber pole illusion

Solving the ambiguity...

- How to get more equations for a pixel?

Solving the ambiguity...

- How to get more equations for a pixel?
- what if the motion is smooth over a local region?

Solving the ambiguity...

- How to get more equations for a pixel?
- what if the motion is smooth over a local region?

- Assume the pixel's neighbors have the same (u, v)
  - If we use a 5x5 window, that gives us 25 equations per pixel

  - For $\forall_{p_i} : \nabla I(p_i) . \begin{bmatrix} u \\ v \end{bmatrix} + I_t(p_i) = 0$

Solving the ambiguity...

$$\begin{pmatrix} I_x(p_1) & I_y(p_1) \\ \vdots & \vdots \\ I_x(p_{25}) & I_y(p_{25}) \end{pmatrix} \begin{bmatrix} u \\ v \end{bmatrix} + \begin{pmatrix} I_t(p_1) \\ \vdots \\ I_t(p_{25}) \end{pmatrix} = 0$$

Solving the ambiguity...

$$\begin{pmatrix} I_x(p_1) & I_y(p_1) \\ \vdots & \vdots \\ I_x(p_{25}) & I_y(p_{25}) \end{pmatrix} \begin{bmatrix} u \\ v \end{bmatrix} + \begin{pmatrix} I_t(p_1) \\ \vdots \\ I_t(p_{25}) \end{pmatrix} = 0$$

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

$$A\,d = b$$

Solving the ambiguity...

$$\begin{pmatrix} I_x(p_1) & I_y(p_1) \\ \vdots & \vdots \\ I_x(p_{25}) & I_y(p_{25}) \end{pmatrix} \begin{bmatrix} u \\ v \end{bmatrix} + \begin{pmatrix} I_t(p_1) \\ \vdots \\ I_t(p_{25}) \end{pmatrix} = 0$$

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

$$A\,d = b$$

how do we solve this?

Solving the ambiguity...

- Least squares solution for d given by

$$A^T A d = A^T b$$

Solving the ambiguity...

- Least squares solution for d given by

$$A^T A d = A^T b$$

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

Solving the ambiguity...

- Least squares solution for d given by

$$A^T A d = A^T b$$

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

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

does this look familiar?

Solving the ambiguity...

- Least squares solution for d given by

$$A^T A d = A^T b$$

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

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

does this look familiar?

$$M = \sum_x \sum_y w(x,y) \begin{bmatrix} I_x^2 & I_x \cdot I_y \\ I_x \cdot I_y & I_y^2 \end{bmatrix}$$

Conditions for solvability

- Optimal (u, v) satisfies Lucas-Kanade equation

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

Conditions for solvability

- Optimal (u, v) satisfies Lucas-Kanade equation

- 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

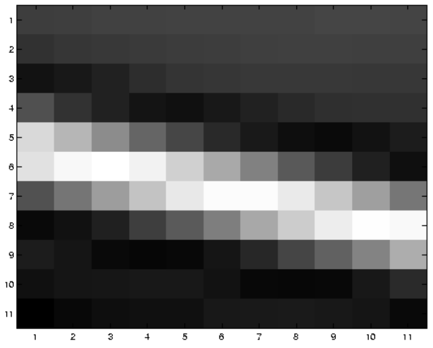
- When is this solvable? I.e., what are good points to track?
  - A^TA should be invertible
  - A^TA should not be too small due to noise
    - eigenvalues λ_1 and λ_2 of A^TA should not be too small

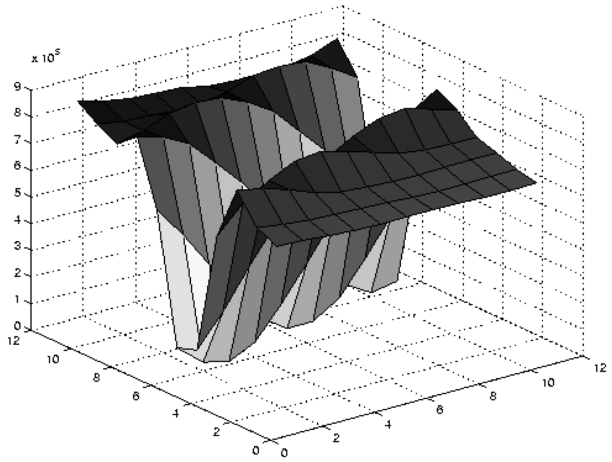
Conditions for solvability

- Optimal (u, v) satisfies Lucas-Kanade equation

- 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)

Edges cause problems

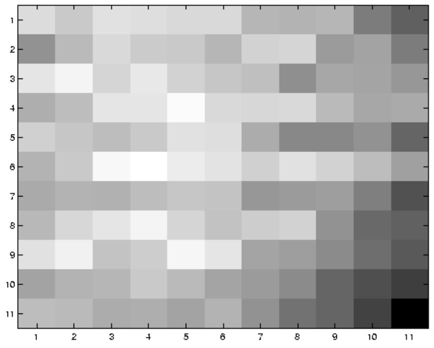


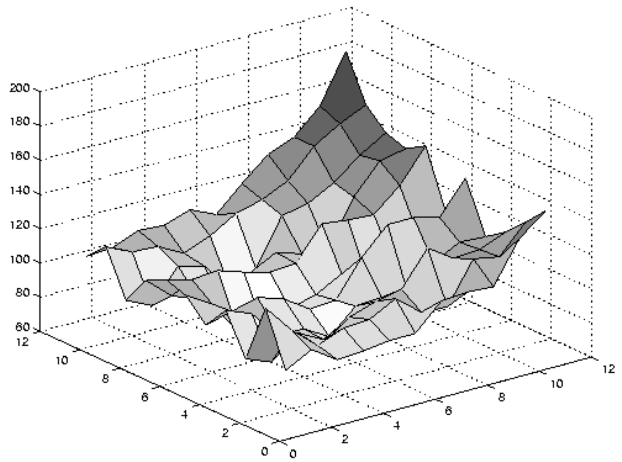


$$\sum \nabla I (\nabla I)^T$$

– large gradients, all the same
– large λ_1, small λ_2

Low texture regions don't work



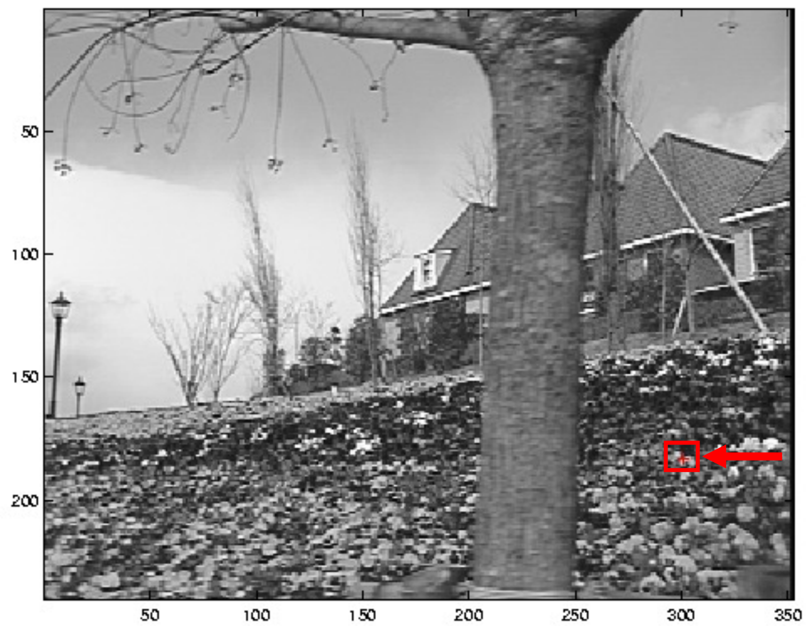


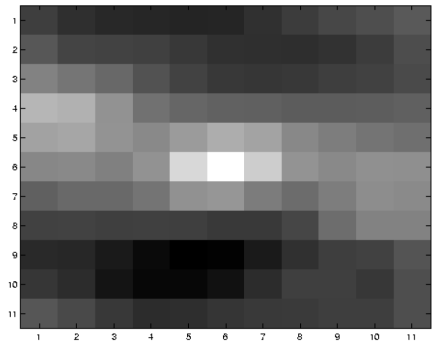
$$\sum \nabla I (\nabla I)^T$$

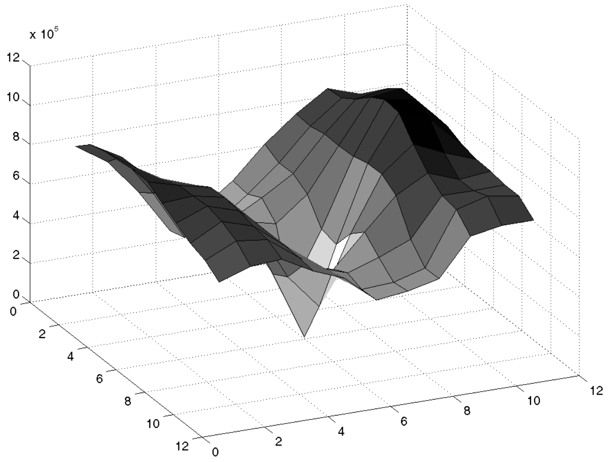
– gradients have small magnitude

– small λ_1, small λ_2

High textured region work best







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

– gradients are different, large magnitudes

– large λ_1, large λ_2

Errors in Lukas-Kanade

- What are the potential causes of errors in this procedure?
  - Suppose A^TA is easily invertible
  - Suppose there is not much noise in the image

Errors in Lukas-Kanade

- What are the potential causes of errors in this procedure?
  - Suppose $A^T A$ is easily invertible
  - Suppose there is not much noise in the image
- When our assumptions are violated

Errors in Lukas-Kanade

- What are the potential causes of errors in this procedure?
  - Suppose A^TA is easily invertible
  - Suppose there is not much noise in the image
- When our assumptions are violated
  - Brightness constancy is not satisfied
  - The motion is not small
  - A point does not move like its neighbors
    - window size is too large
    - what is the ideal window size?

Errors in Lukas-Kanade

- What are the potential causes of errors in this procedure?
  - Suppose A^TA is easily invertible
  - Suppose there is not much noise in the image
- When our assumptions are violated
  - Brightness constancy is not satisfied
  - **The motion is not small**
  - A point does not move like its neighbors
    - window size is too large
    - what is the ideal window size?

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

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

Revisiting the small motion assumption

- Is this motion small enough?

Revisiting the small motion assumption

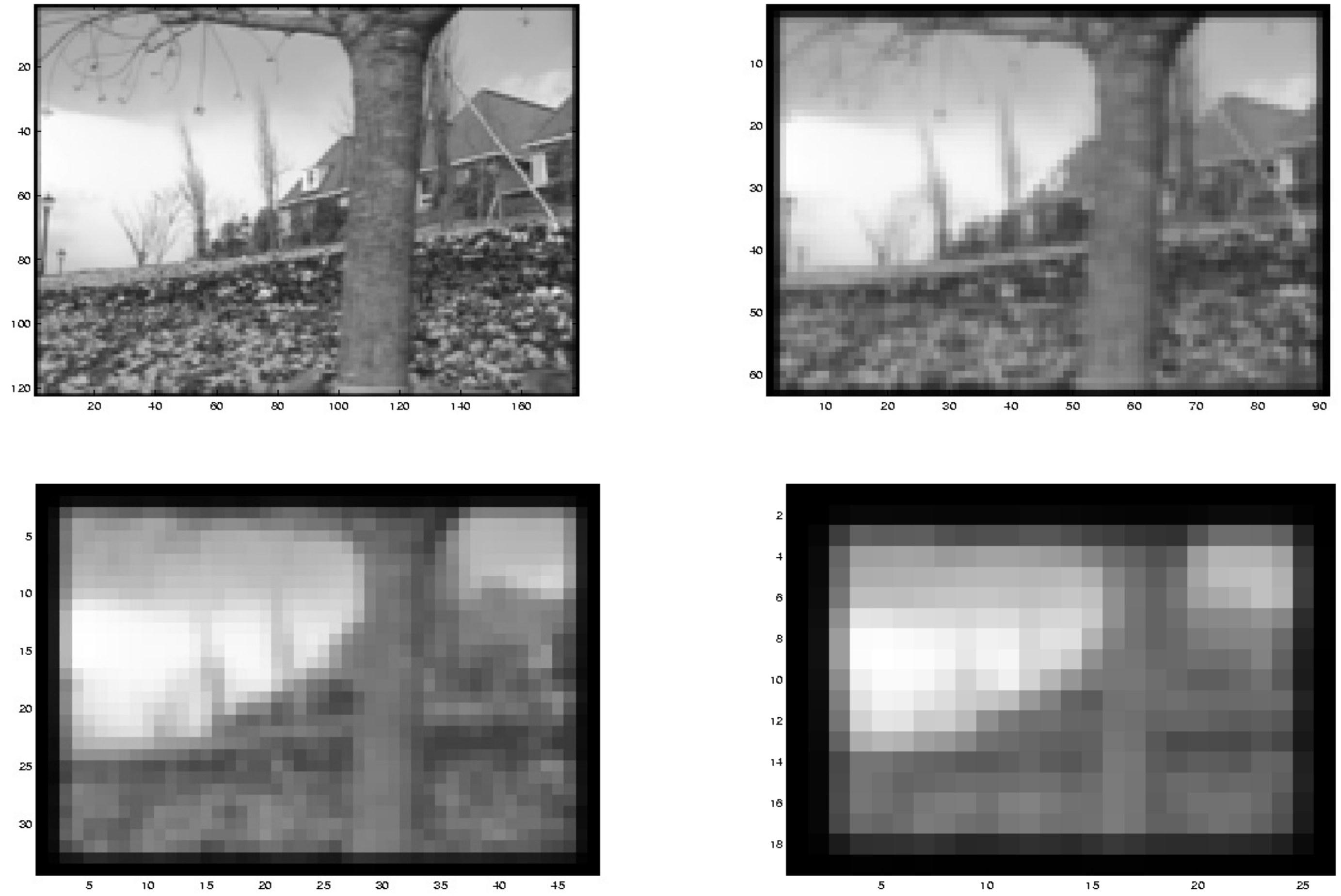
- Is this motion small enough?
    - Probably not—it's much larger than one pixel (2^{nd} order terms dominate)

Revisiting the small motion assumption

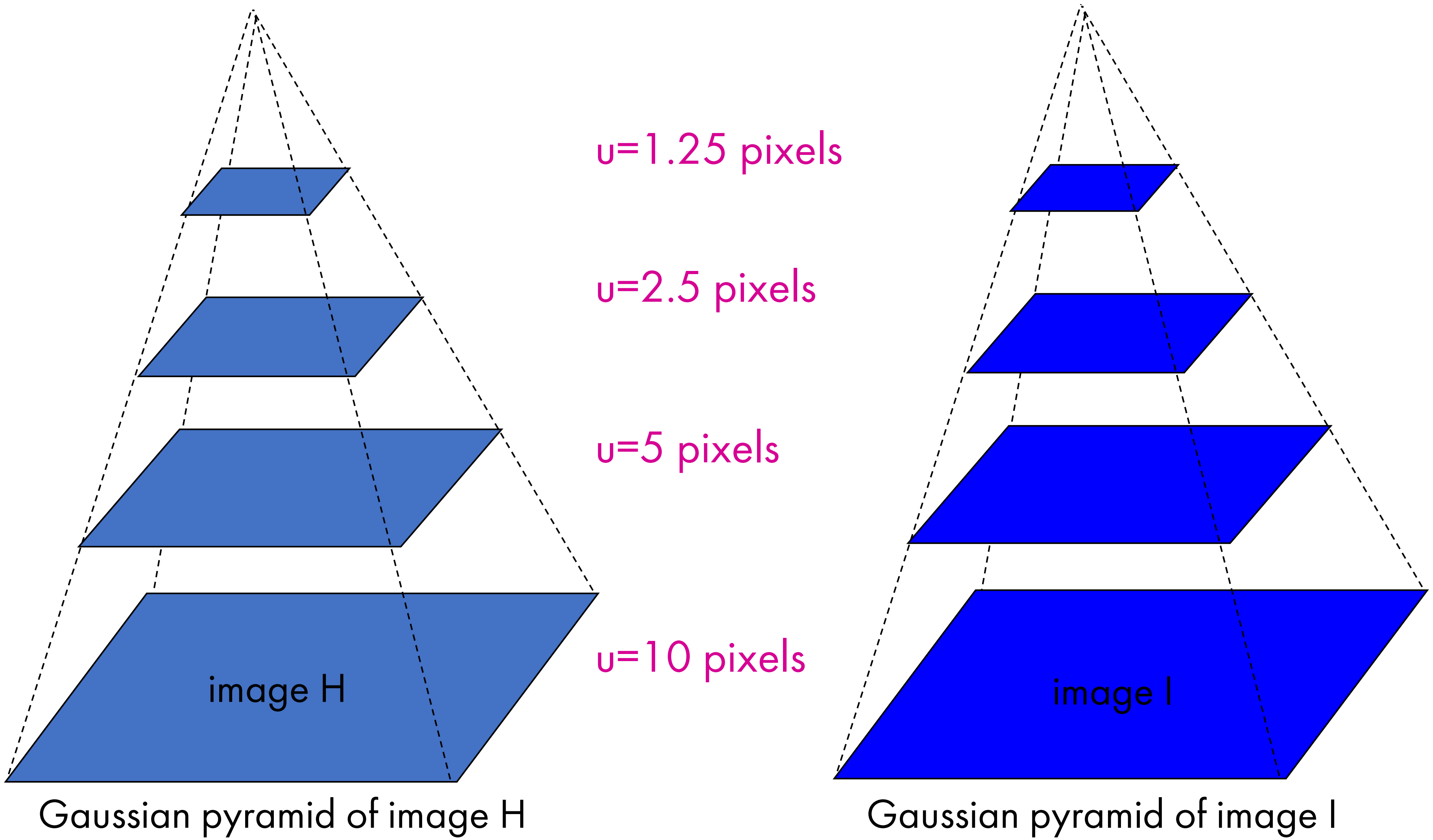
How might we solve this problem?

- Is this motion small enough?
  - Probably not—it's much larger than one pixel (2^{nd} order terms dominate)

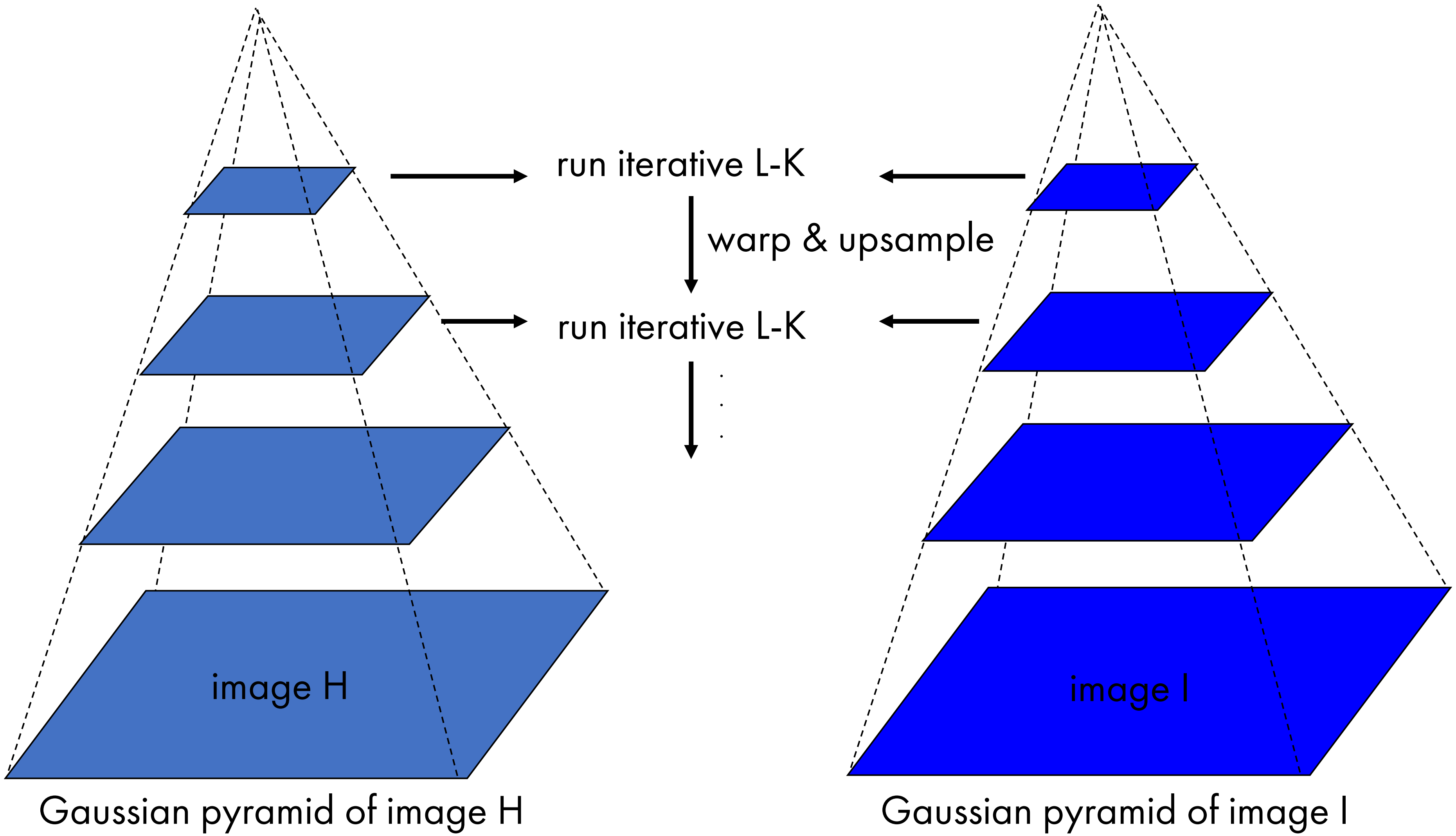
Reduce the resolution!



Coarse-to-fine optical flow estimation



Coarse-to-fine optical flow estimation



A Few Details

- Top Level
  - Apply L-K to get a flow field representing the flow from the first frame to the second frame.
  - Apply this flow field to warp the first frame toward the second frame.

A Few Details

- Top Level
  - Apply L-K to get a flow field representing the flow from the first frame to the second frame.
  - Apply this flow field to warp the first frame toward the second frame.
  - Rerun L-K on the new warped image to get a flow field from it to the second frame.
  - Repeat till convergence.

A Few Details

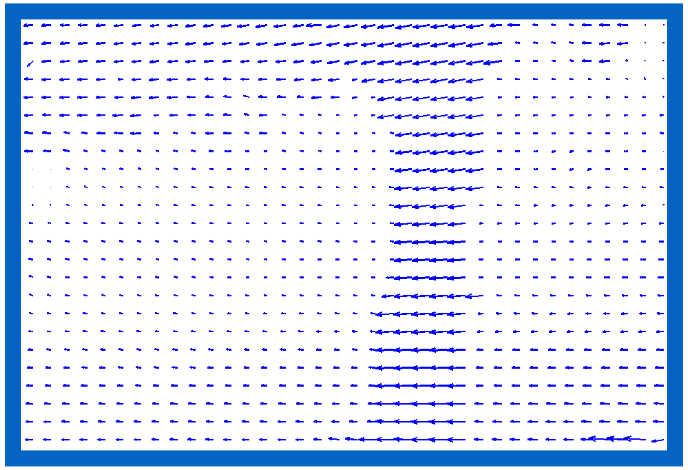
- Top Level
  - Apply L-K to get a flow field representing the flow from the first frame to the second frame.
  - Apply this flow field to warp the first frame toward the second frame.
  - Rerun L-K on the new warped image to get a flow field from it to the second frame.
  - Repeat till convergence.
- Next Level
  - Upsample the flow field to the next level as the first guess of the flow at that level.

A Few Details

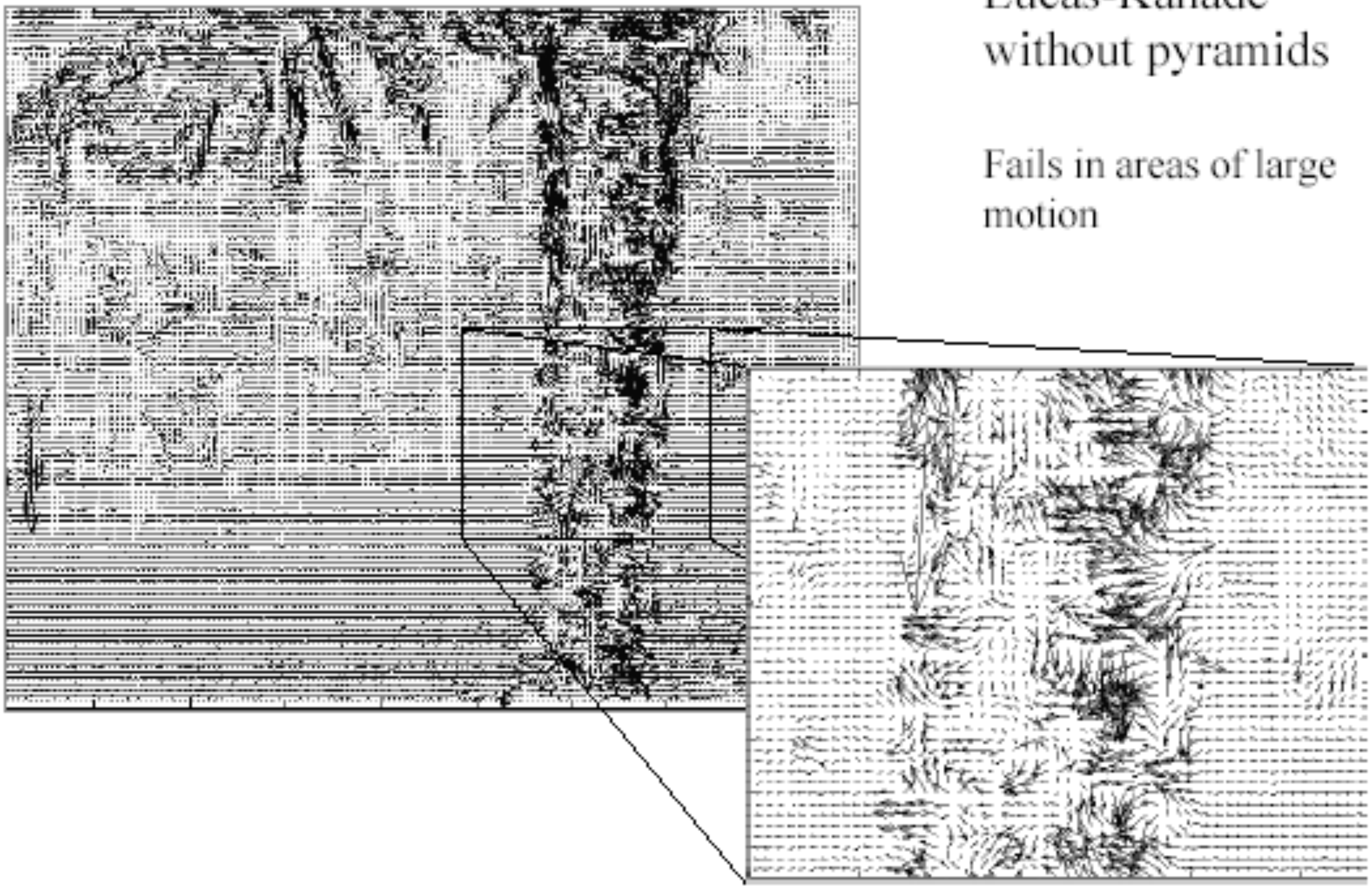
- Top Level
  - Apply L-K to get a flow field representing the flow from the first frame to the second frame.
  - Apply this flow field to warp the first frame toward the second frame.
  - Rerun L-K on the new warped image to get a flow field from it to the second frame.
  - Repeat till convergence.
- Next Level
  - Upsample the flow field to the next level as the first guess of the flow at that level.
  - Apply this flow field to warp the first frame toward the second frame.
  - Rerun L-K and warping till convergence as above.
- Etc.

The Flower Garden Video

- What should the
- optical flow be?



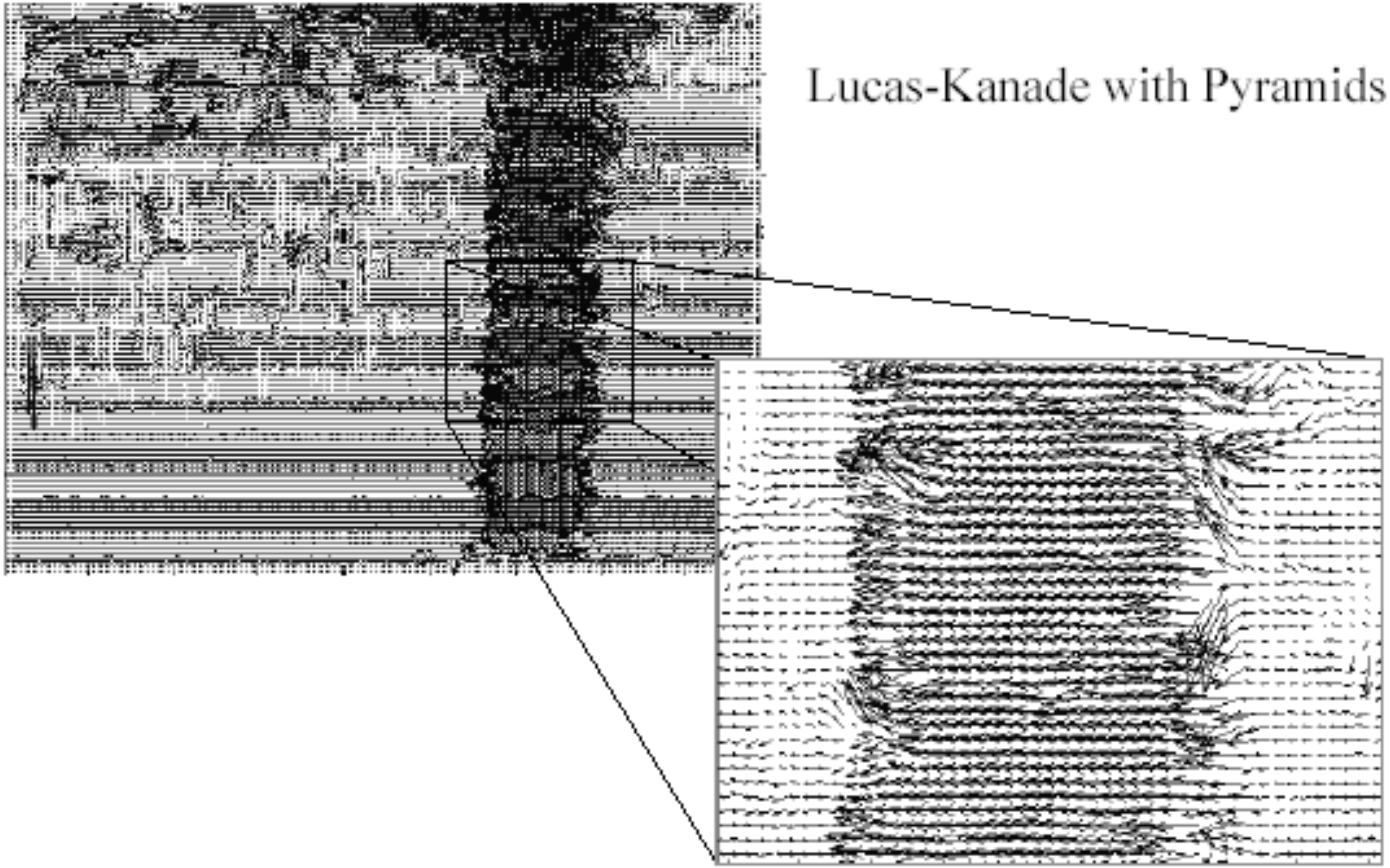
Optical Flow Results



Lucas-Kanade
without pyramids

Fails in areas of large
motion

Optical Flow Results



Lucas-Kanade with Pyramids

Next Time

- Can we also define keypoints that are shift, rotation, and scale invariant/covariant?
- What should be our description around keypoint?