



# Lecture 4. **RANSAC and feature detectors**

**Pattern Recognition and Computer Vision**

Guanbin Li,

School of Data and Computer Science, Sun Yat-Sen University

# 扫码签到

---



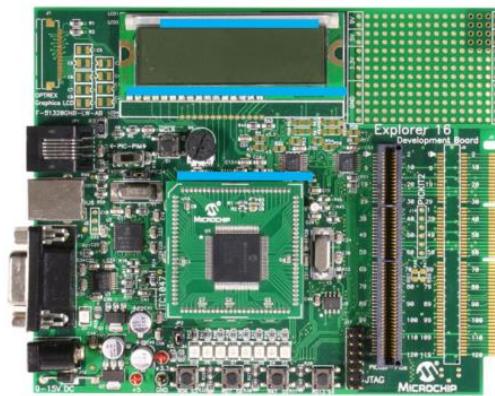
# What we will learn today?

---

- A model fitting method for line detection
  - RANSAC
- Local invariant features
  - Motivation
  - Requirements, invariances
- Keypoint localization
  - Harris corner detector

# Example: Line Fitting

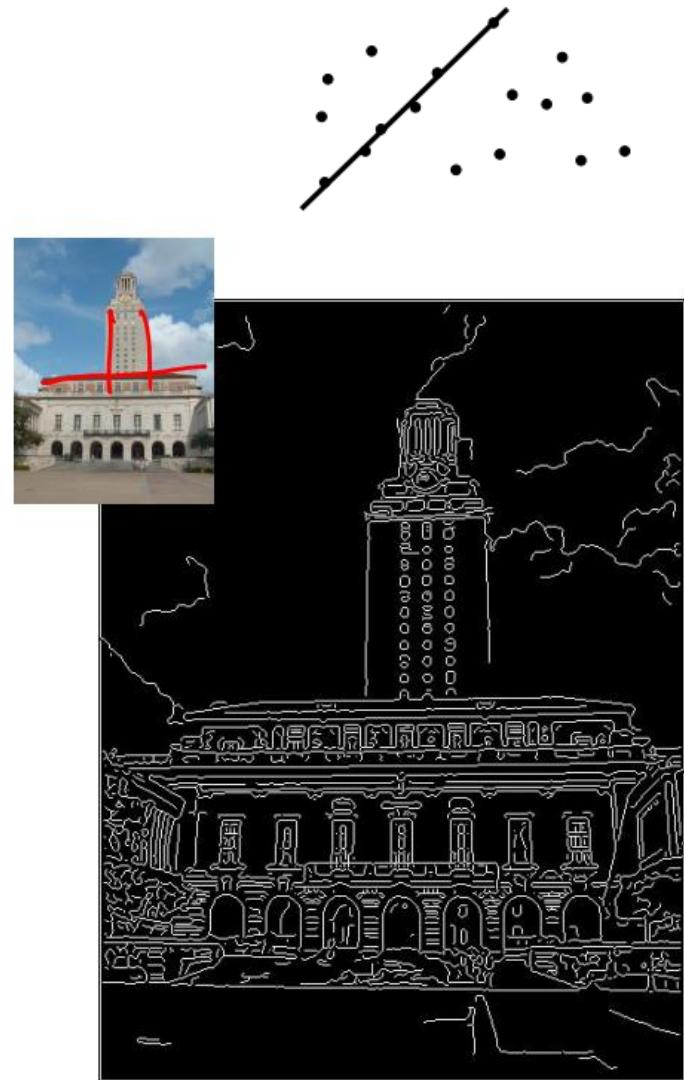
- Why fit lines?
  - Many objects characterized by presence of straight lines



- Wait, why aren't we done just by running edge detection?

# Difficulty of Line Fitting

- Extra edge points (clutter), multiple models:
  - Which points go with which line, if any?
- Only some parts of each line detected, and some parts are missing:
  - How to find a line that bridges missing evidence?
- Noise in measured edge points orientations:
  - How to detect true underlying parameters?



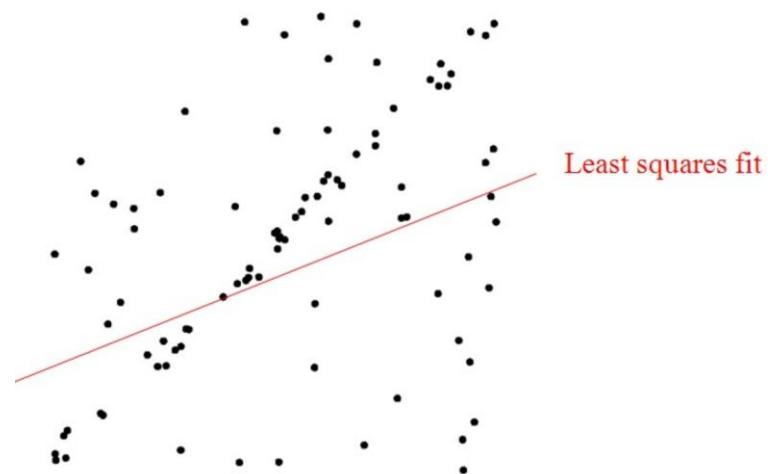
# Voting

---

- It's not feasible to check all combinations of features by fitting a model to each possible subset.
- **Voting** is a general technique where we let the features vote for all models that are compatible with it.
  - Cycle through features, cast votes for model parameters.
  - Look for model parameters that receive a lot of votes.
- Noise & clutter features will cast votes too, but typically their votes should be inconsistent with the majority of “good” features.
- Ok if some features not observed, as model can span multiple fragments.

# RANSAC [Fischler & Bolles 1981]

- RANdom SAmple Consensus
- Approach:
  - we want to avoid the impact of outliers, so let's look for “inliers”, and use only those.
- Intuition:
  - if an outlier is chosen to compute the current fit, then the resulting line won't have much support from rest of the points.



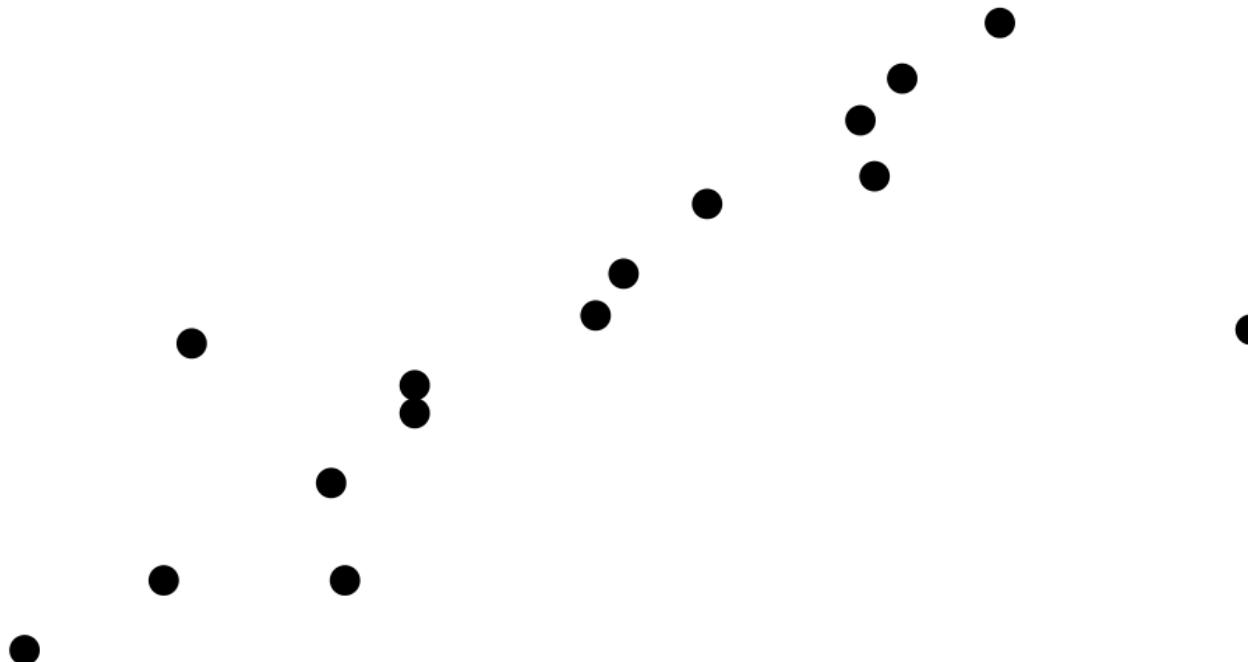
# RANSAC [Fischler & Bolles 1981]

---

- RANSAC loop:
- 1. Randomly select a seed group of points on which to base transformation estimate (e.g., a group of matches)
- 2. Compute transformation from seed group
- 3. Find inliers to this transformation
- 4. If the number of inliers is sufficiently large, re-compute least-squares estimate of transformation on all of the inliers
- Keep the transformation with the largest number of inliers

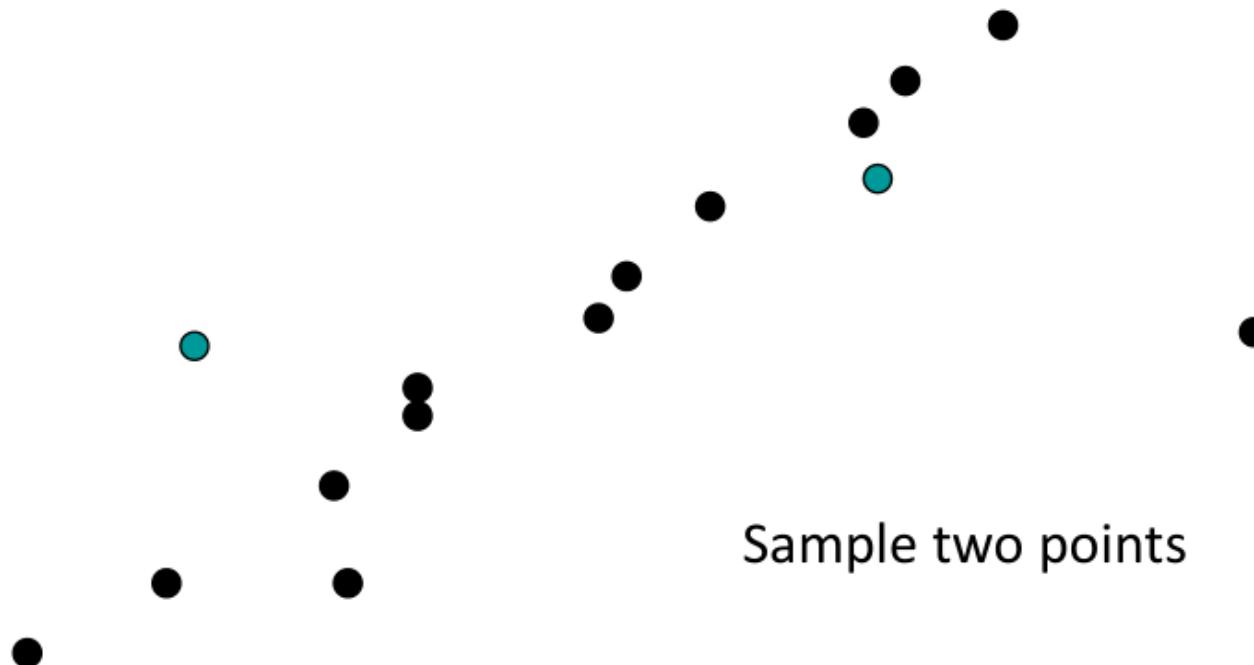
# RANSAC Line Fitting Example

- Task: Estimate the best line
  - How many points do we need to estimate the line?*



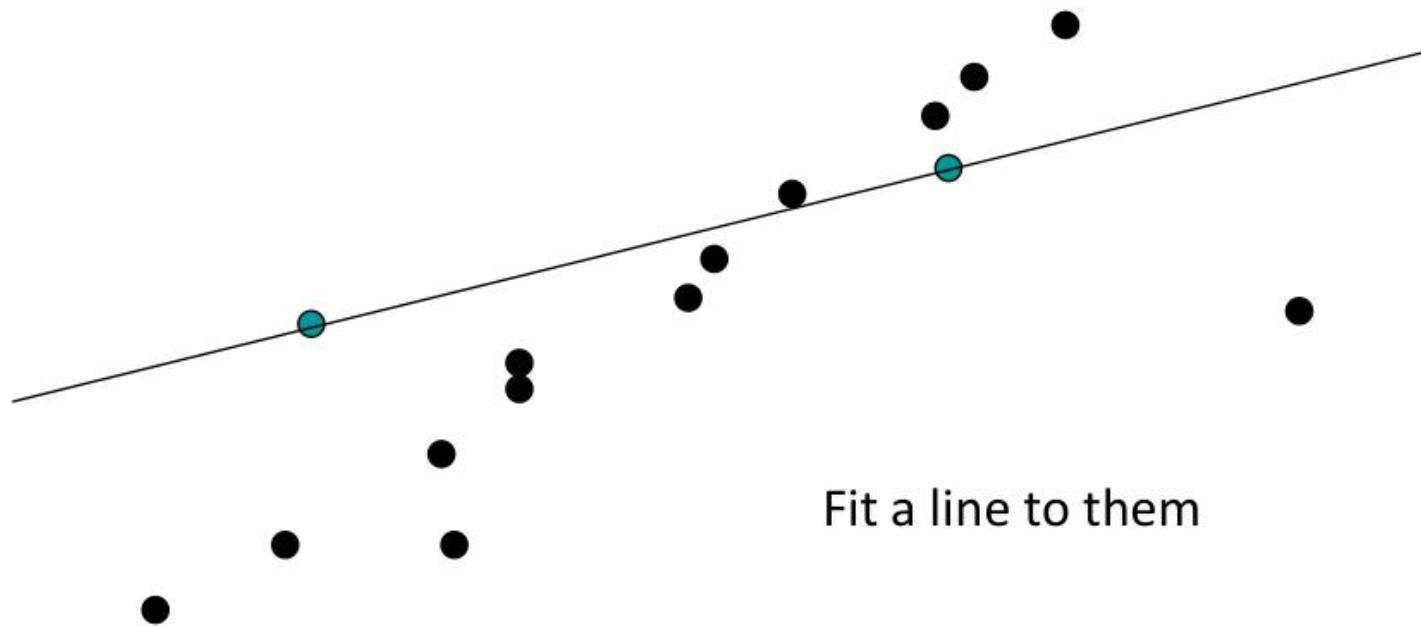
# RANSAC Line Fitting Example

- Task: Estimate the best line



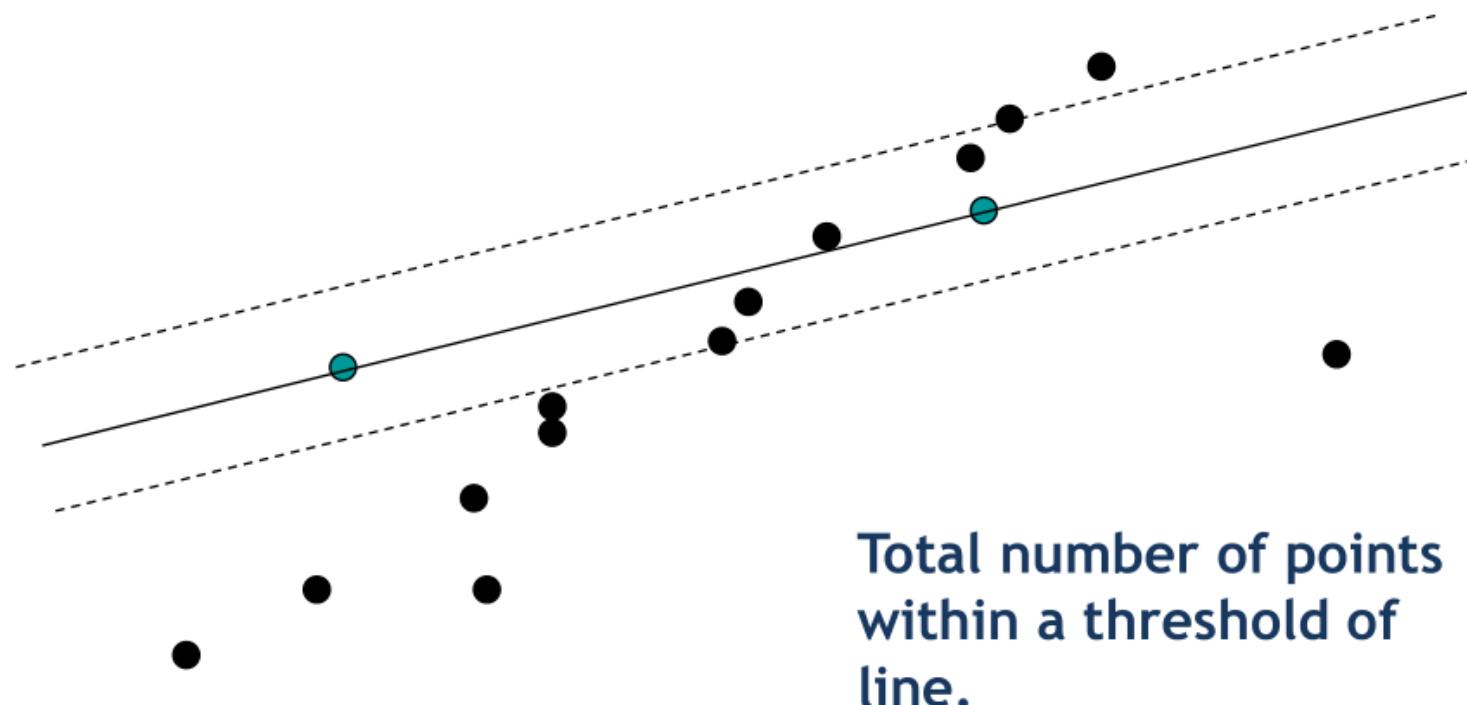
# RANSAC Line Fitting Example

- Task: Estimate the best line



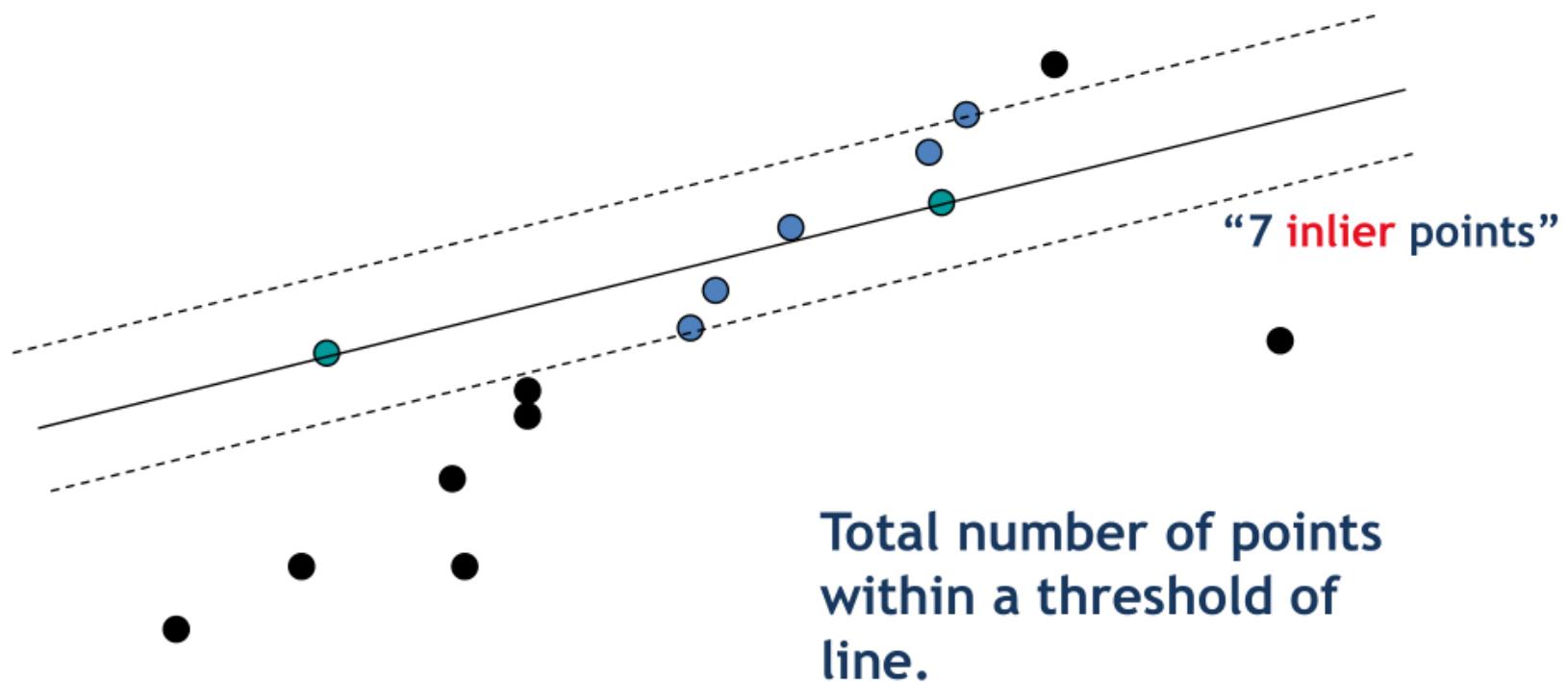
# RANSAC Line Fitting Example

- Task: Estimate the best line



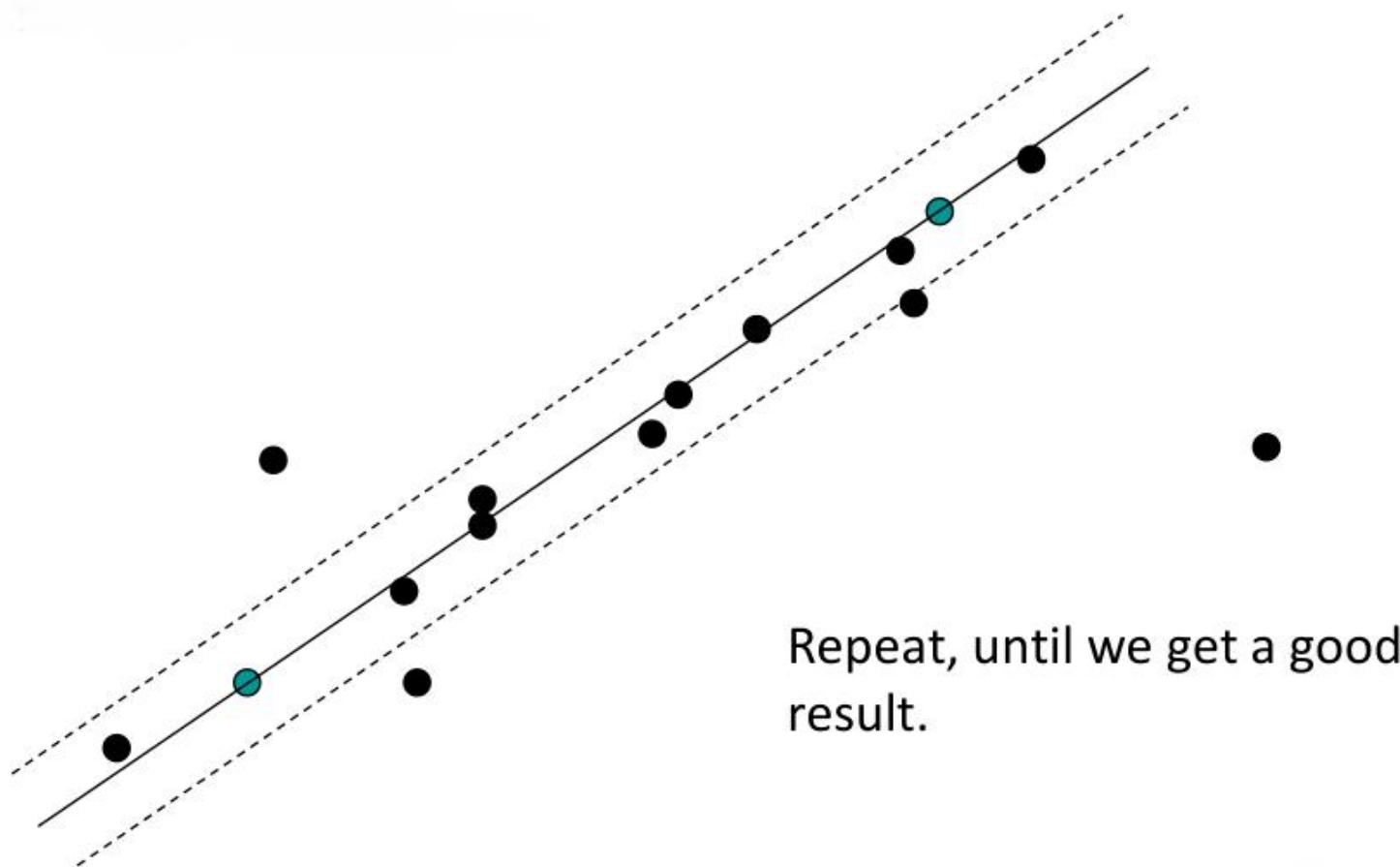
# RANSAC Line Fitting Example

- Task: Estimate the best line



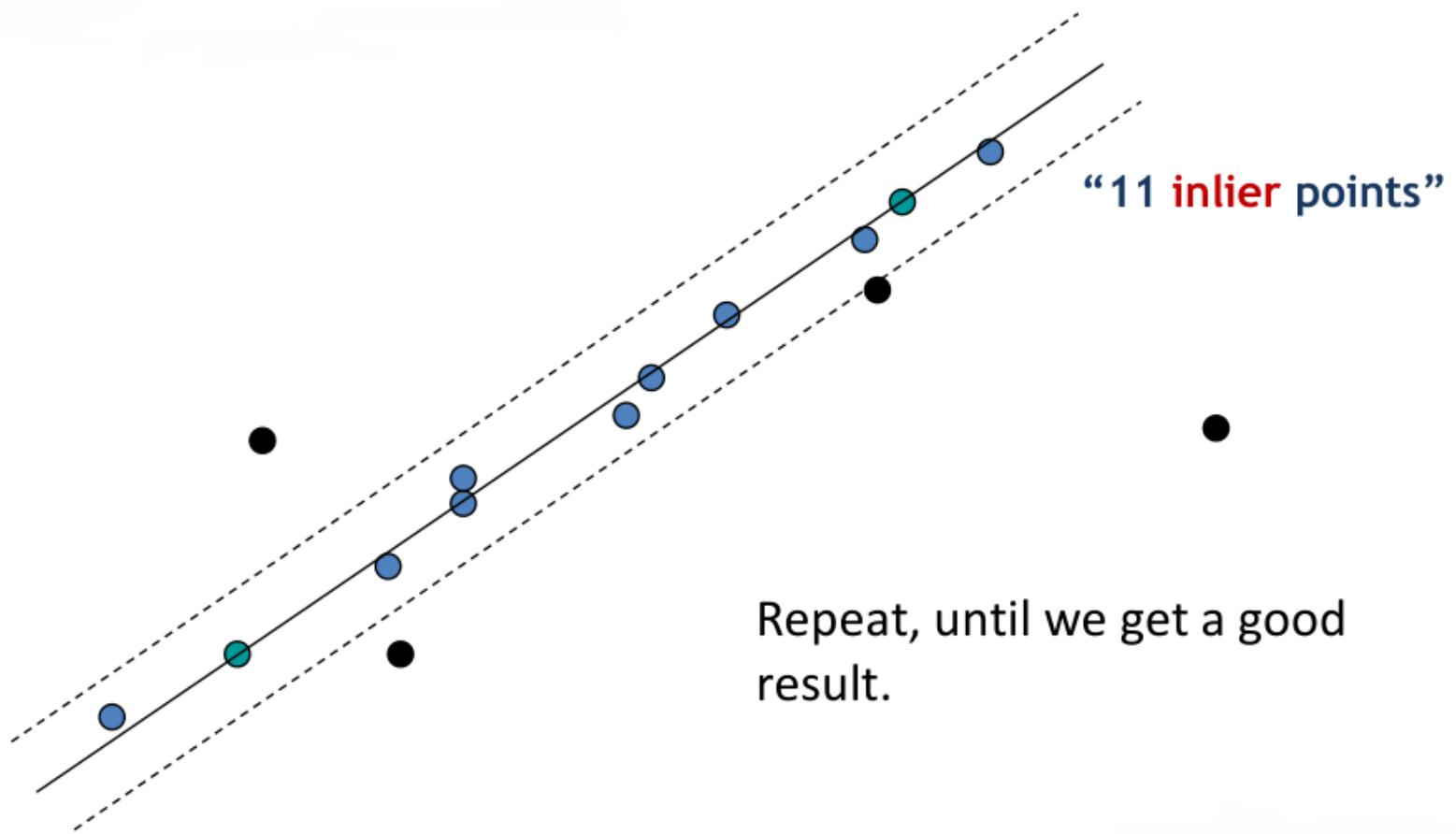
# RANSAC Line Fitting Example

- Task: Estimate the best line



# RANSAC Line Fitting Example

- Task: Estimate the best line



# RANSAC: Algorithm

## Algorithm 15.4: RANSAC: fitting lines using random sample consensus

Determine:

- $n$  — the smallest number of points required
- $k$  — the number of iterations required
- $t$  — the threshold used to identify a point that fits well
- $d$  — the number of nearby points required
  - to assert a model fits well

Until  $k$  iterations have occurred

    Draw a sample of  $n$  points from the data  
        uniformly and at random

    Fit to that set of  $n$  points

    For each data point outside the sample

        Test the distance from the point to the line  
            against  $t$ ; if the distance from the point to the line  
            is less than  $t$ , the point is close

    end

    If there are  $d$  or more points close to the line  
        then there is a good fit. Refit the line using all  
        these points.

end

Use the best fit from this collection, using the  
fitting error as a criterion

# RANSAC: How many samples?

---

- How many samples are needed?
  - Suppose  $w$  is fraction of inliers (points from line).
  - $n$  points needed to define hypothesis (2 for lines)
  - $k$  samples chosen.
- Prob. that a single sample of  $n$  points is correct:  $w^n$
- Prob. that all  $k$  samples fail is:  $(1 - w^n)^k$

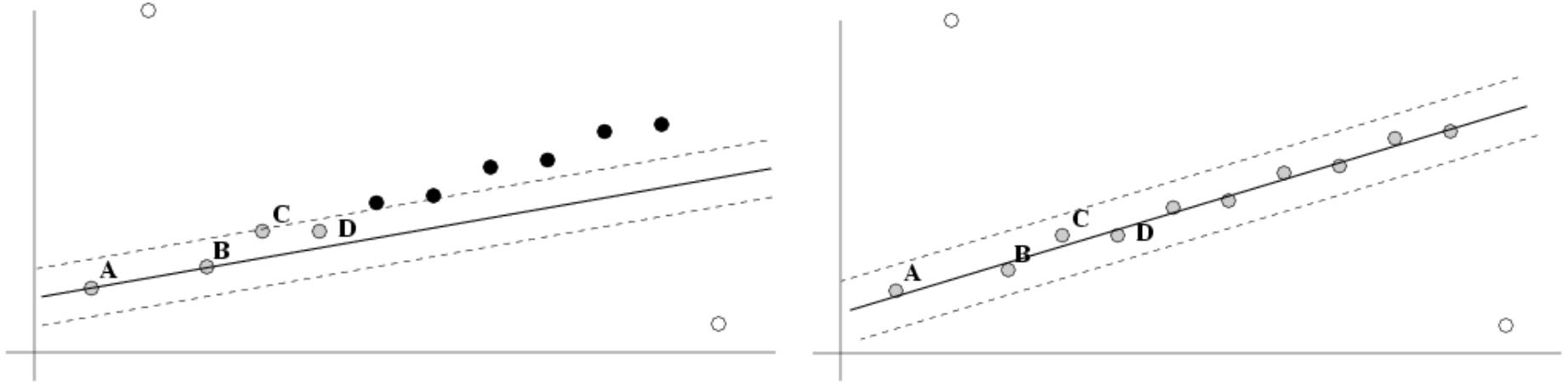
→ Choose  $k$  high enough to keep this below desired failure rate.

# RANSAC: Computed k (p=0.99)

Sample size <b>n</b>	Proportion of outliers						
	5%	10%	20%	25%	30%	40%	50%
2	2	3	5	6	7	11	17
3	3	4	7	9	11	19	35
4	3	5	9	13	17	34	72
5	4	6	12	17	26	57	146
6	4	7	16	24	37	97	293
7	4	8	20	33	54	163	588
8	5	9	26	44	78	272	1177

# After RANSAC

- RANSAC divides data into inliers and outliers and yields estimate computed from minimal set of inliers.
- Improve this initial estimate with estimation over all inliers (e.g. with standard least-squares minimization).
- But this may change inliers, so alternate fitting with re-classification as inlier/outlier.



# RANSAC: Pros and Cons

---

- **Pros:**
  - General method suited for a **wide range** of model fitting problems
  - Easy to **implement** and easy to **calculate** its failure rate
- **Cons:**
  - Only handles a **moderate** percentage of outliers without cost blowing up
  - Many real problems have **high rate** of outliers (but sometimes selective choice of random subsets can help)
- A voting strategy, The Hough transform, can handle high percentage of outliers

# What we will learn today?

---

- A model fitting method for line detection
    - RANSAC
  - Local invariant features
    - Motivation
    - Requirements, invariances
  - Keypoint localization
    - Harris corner detector
- Some background reading:  
Rick Szeliski, Chapter 4.1.1; David Lowe, IJCV 2004

# Image matching: a challenging problem



# Image matching: a challenging problem

---



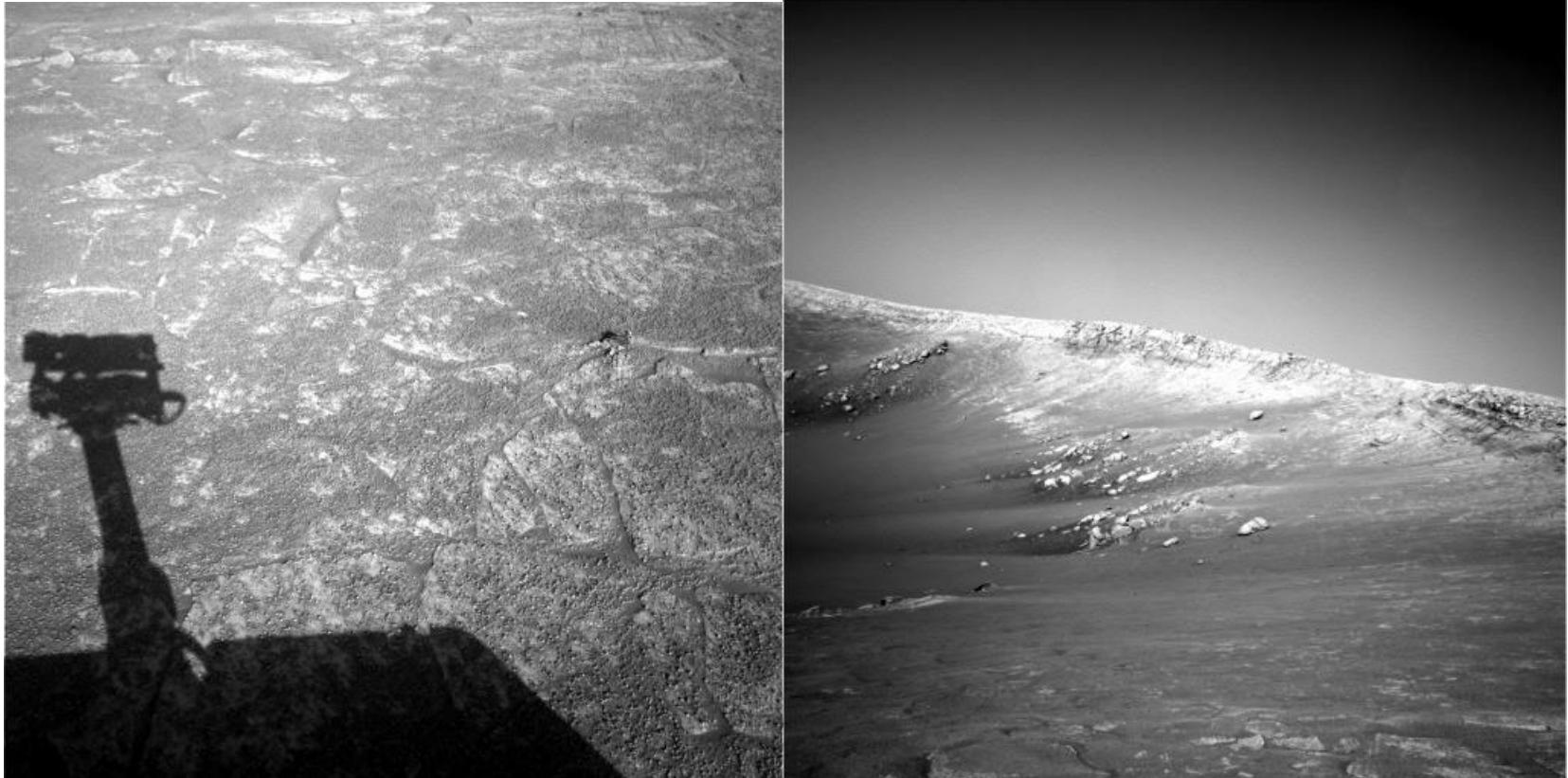
# Harder Case

---



# Harder Still?

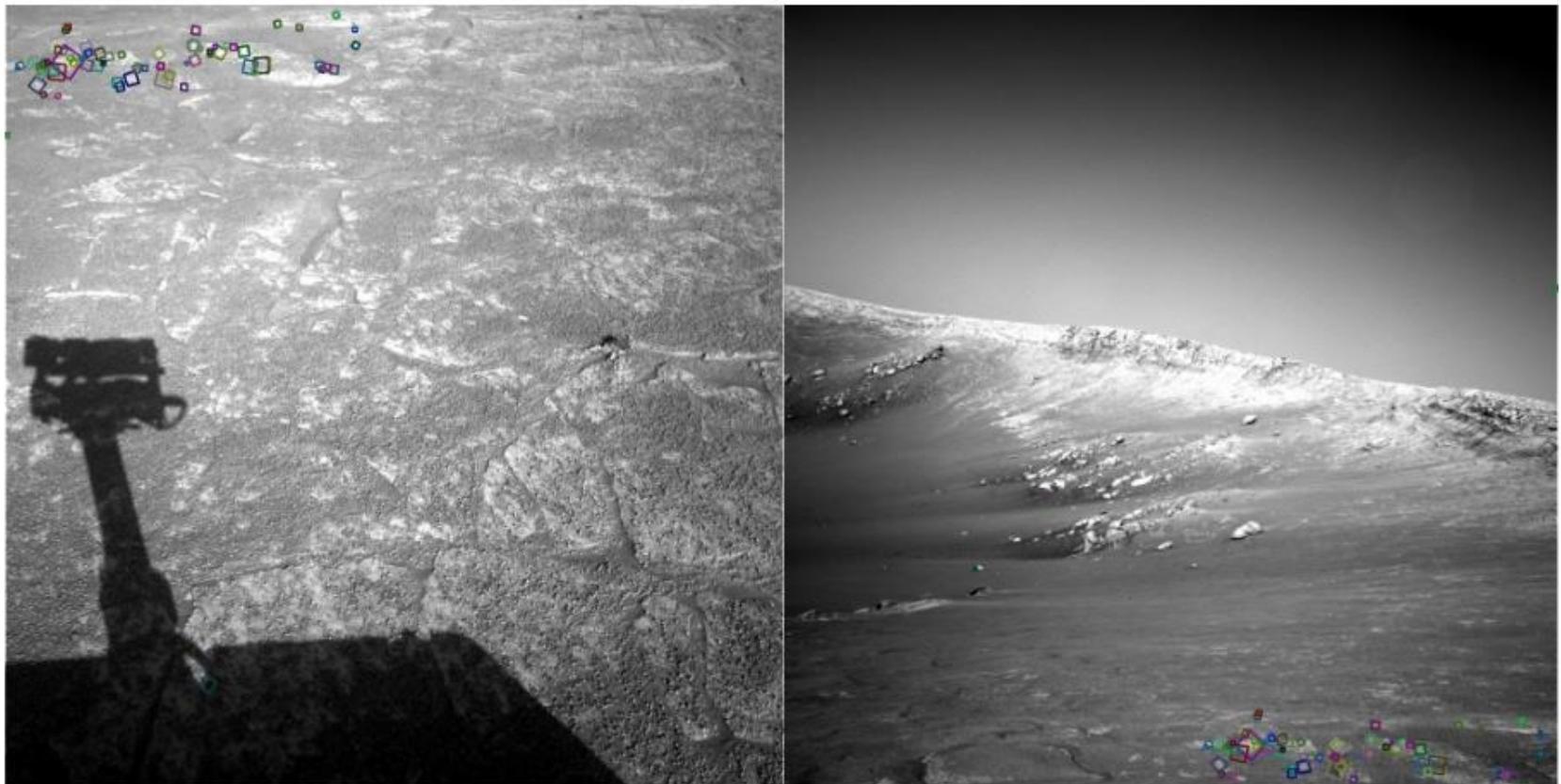
---



NASA Mars Rover images

# Answer Below (Look for tiny colored squares)

---



NASA Mars Rover images with SIFT feature matches  
(Figure by Noah Snavely)

# Applications

---

Comparison Between 2 or more images

- Image alignment
- Image Stitching
- 3D reconstruction
- Object recognition ( matching )
- Indexing and database retrieval
- Object tracking
- Robot navigation

The following slides are borrowed from Course ME5286 (Saad J Bedros)

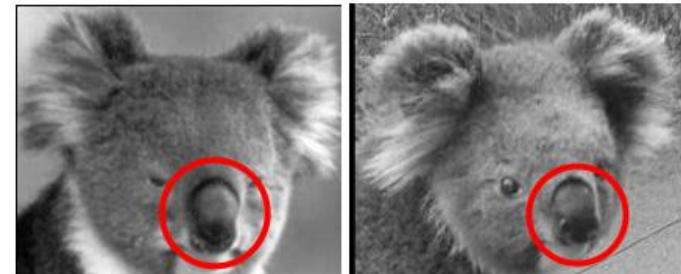
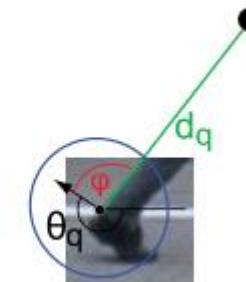
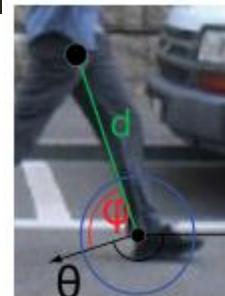
# Motivation for using local features

- Global representations have major limitations
- Instead, describe and match only local regions
- Increased robustness to

- Occlusions

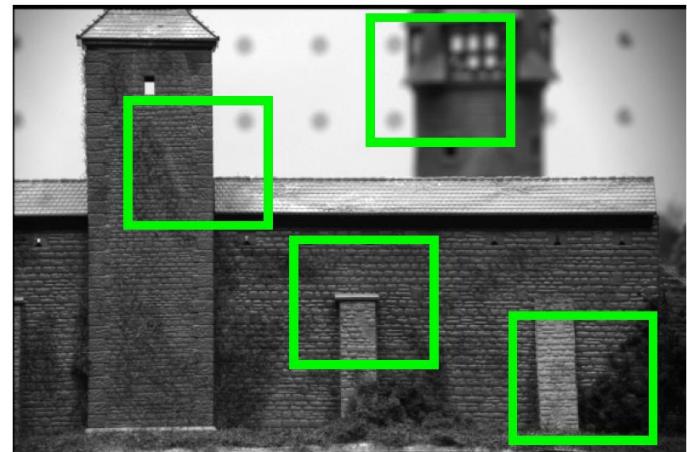
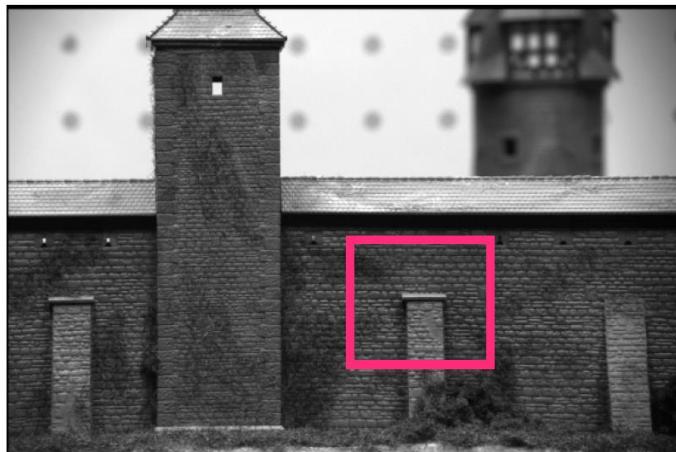


- Articulation
- Intra-category variations



# Motivation: Patch Matching

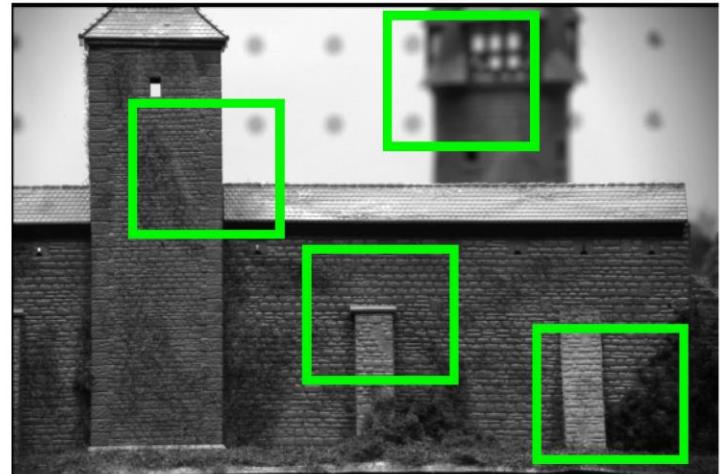
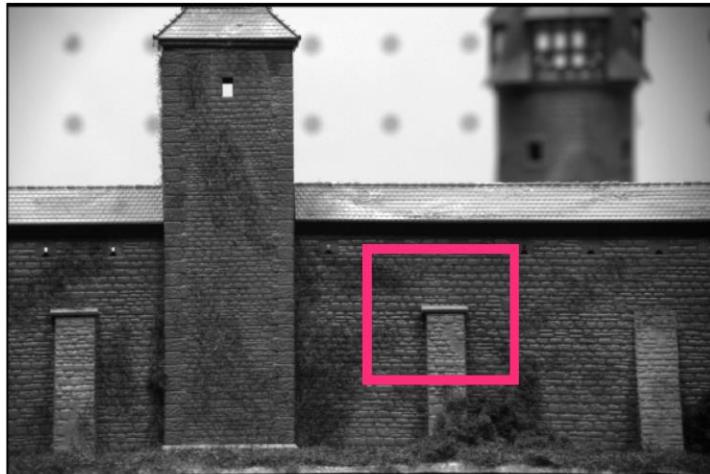
Elements to be matched are image patches of fixed size



Task: find the best (most similar) patch in a second image



# Not all Patches are Created Equal!



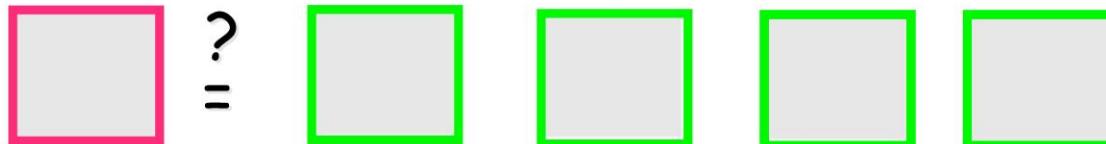
Intuition: this would be a good patch for matching, since it is very distinctive (there is only one patch in the second frame that looks similar).



# Not all Patches are Created Equal!



Intuition: this would be a BAD patch for matching, since it is not very distinctive (there are many similar patches in the second frame)



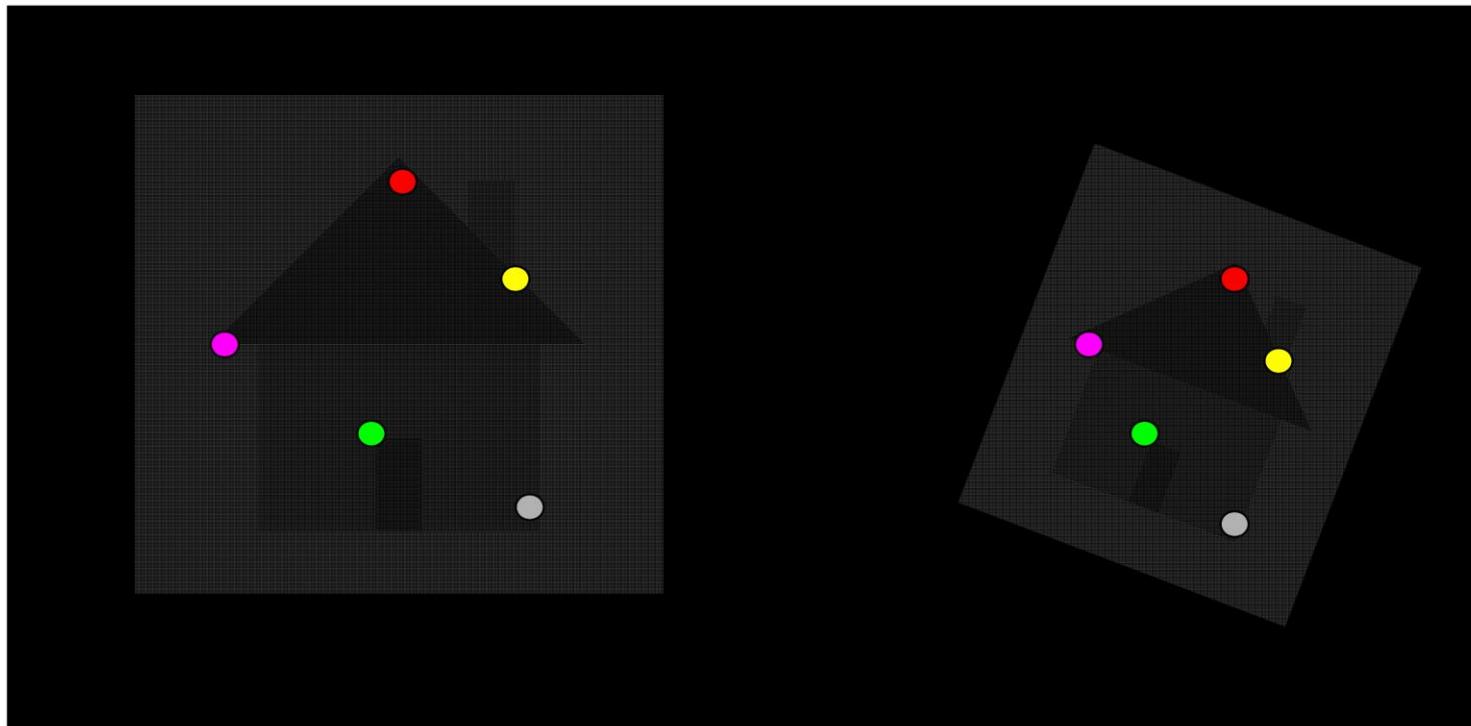
# Interest Points (Key Points)

- Local features associated with a significant change of an image property of several properties simultaneously (e.g., intensity, color, texture).

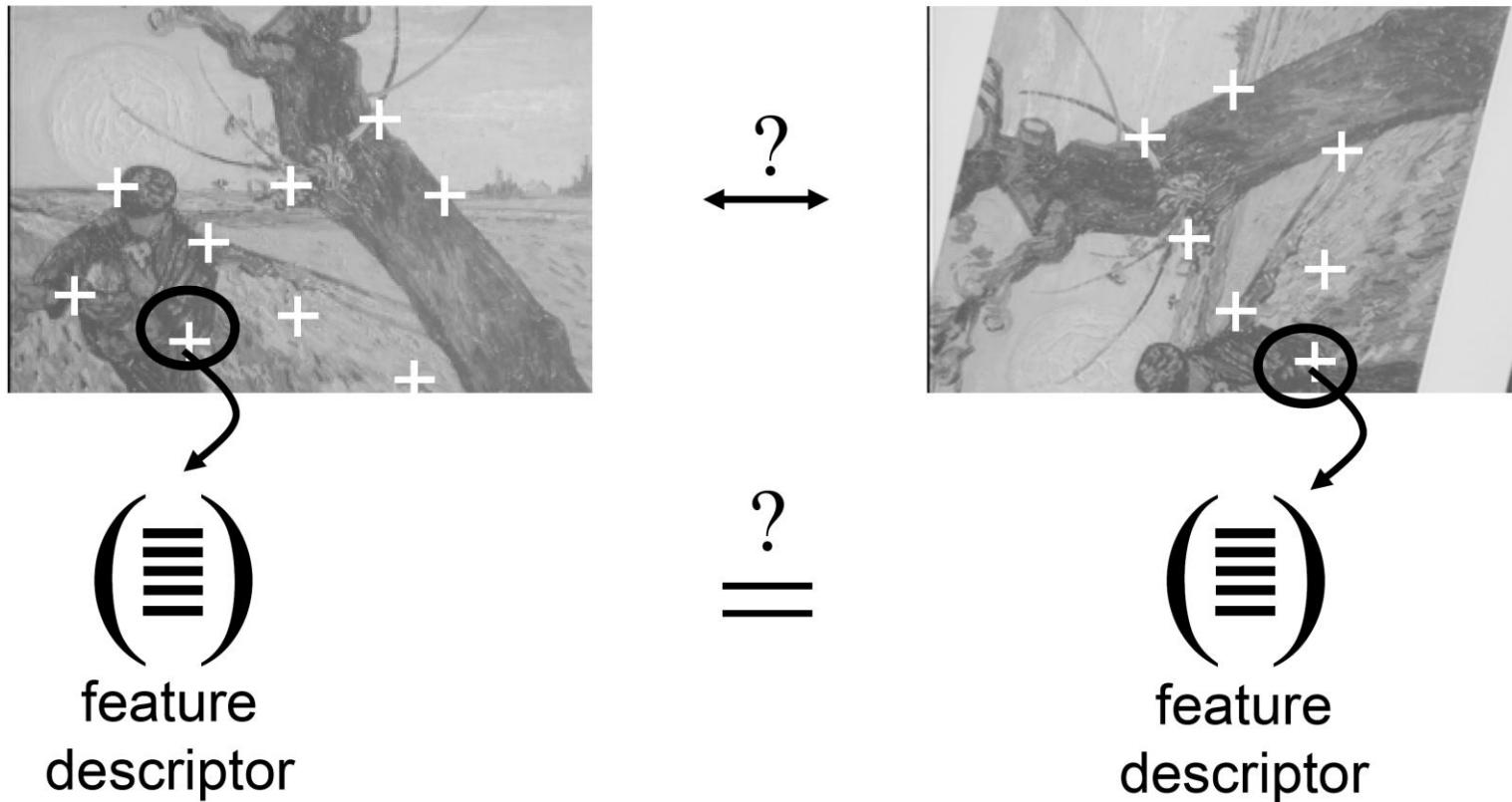


# Why Extract Interest Points?

- Corresponding points (or features) between images enable the estimation of parameters describing geometric transforms between the images.



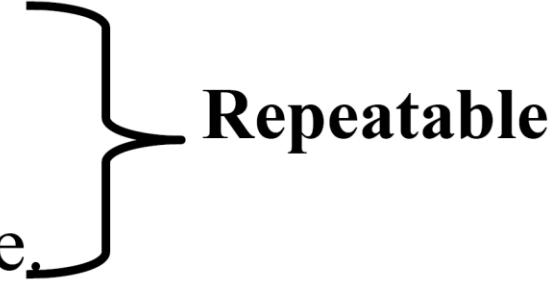
# What if we don't know the correspondences?



- Need to compare *feature descriptors* of local patches surrounding interest points

# Properties of good features

---

- **Local:** features are local, robust to occlusion and clutter (no prior segmentation!).
  - **Accurate:** precise localization.
  - **Invariant / Covariant**
  - **Robust:** noise, blur, compression, etc. do not have a big impact on the feature.
  - **Distinctive:** individual features can be matched to a large database of objects.
  - **Efficient:** close to real-time performance.
- 
- Repeatable**

# Invariant / Covariant

---

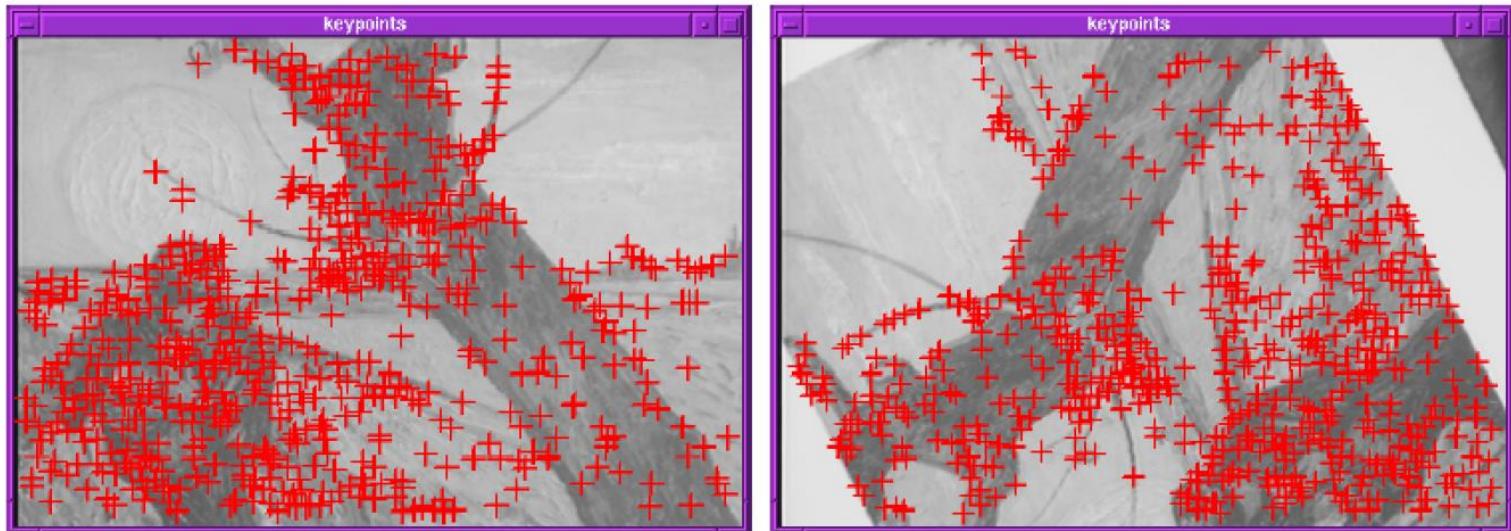
- A function  $f()$  is **invariant** under some transformation  $T()$  if its value does not change when the transformation is applied to its argument:  
$$\text{if } f(x) = y \text{ then } f(T(x))=y$$

- A function  $f()$  is **covariant** when it commutes with the transformation  $T()$ :

$$\text{if } f(x) = y \text{ then } f(T(x))=T(f(x))=T(y)$$

# Invariance

- Features should be detected despite geometric or photometric changes in the image.
- Given two transformed versions of the same image, features should be detected in corresponding locations.



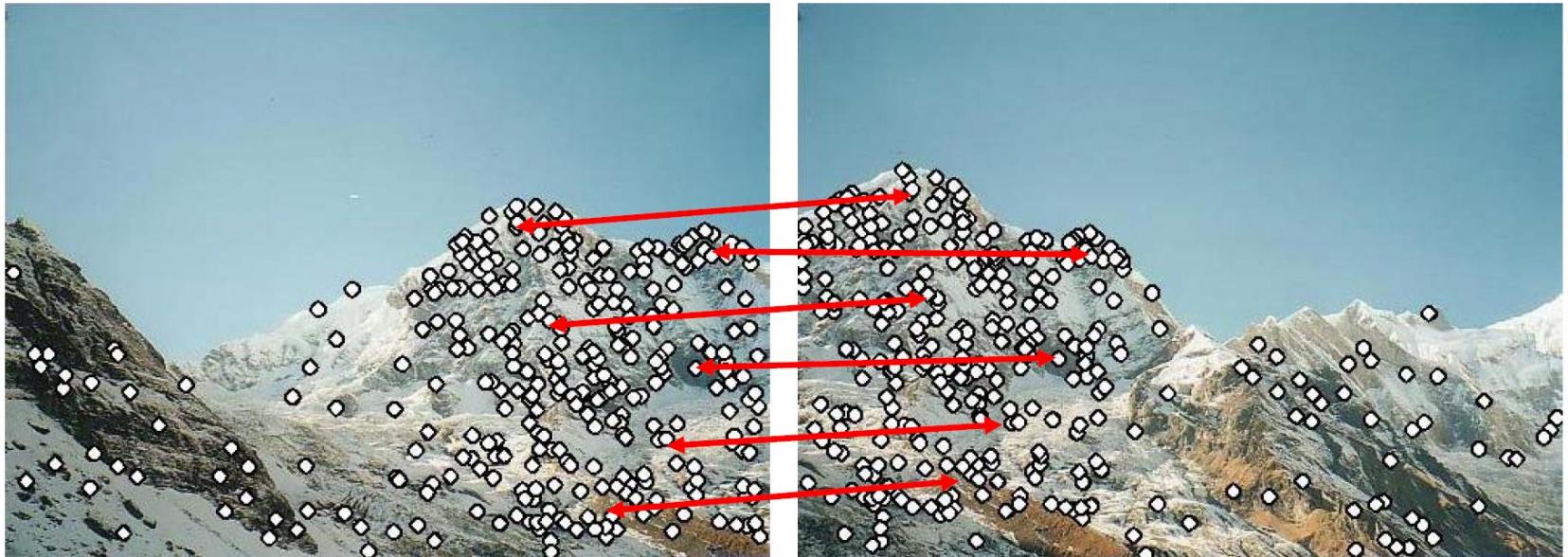
# Example: Panorama Stitching

---

- How do we combine these two images?



# Example: Panorama Stitching



Step 1: extract features

Step 2: match features

# Example: Panorama Stitching

---

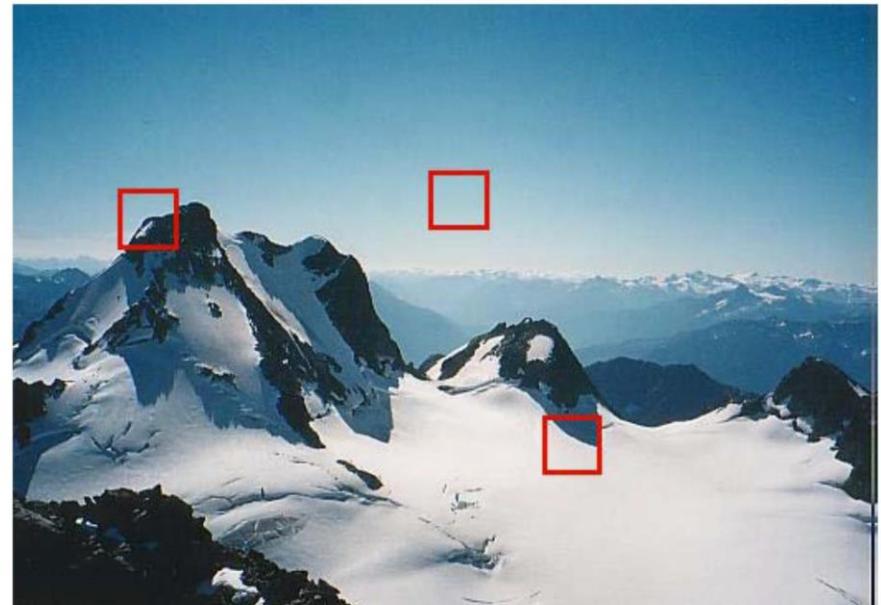
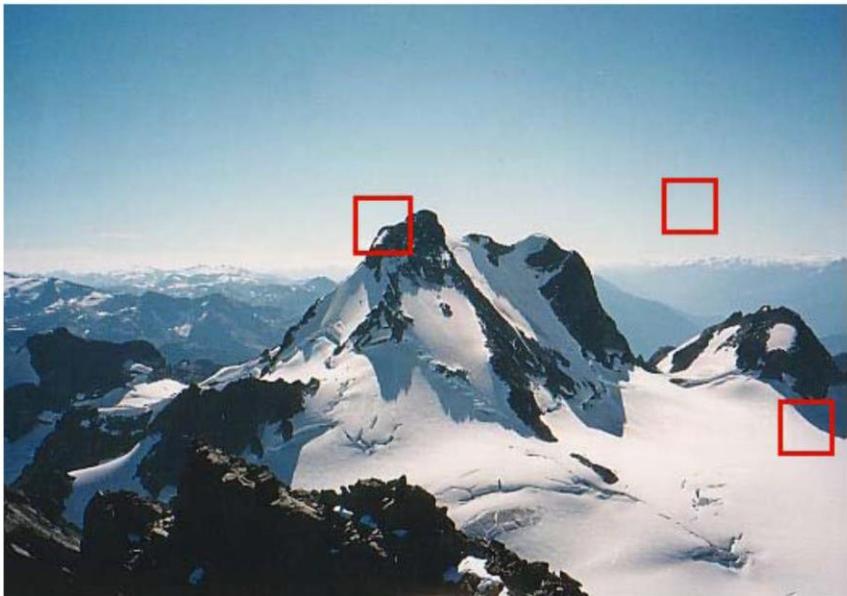


Step 1: extract features

Step 2: match features

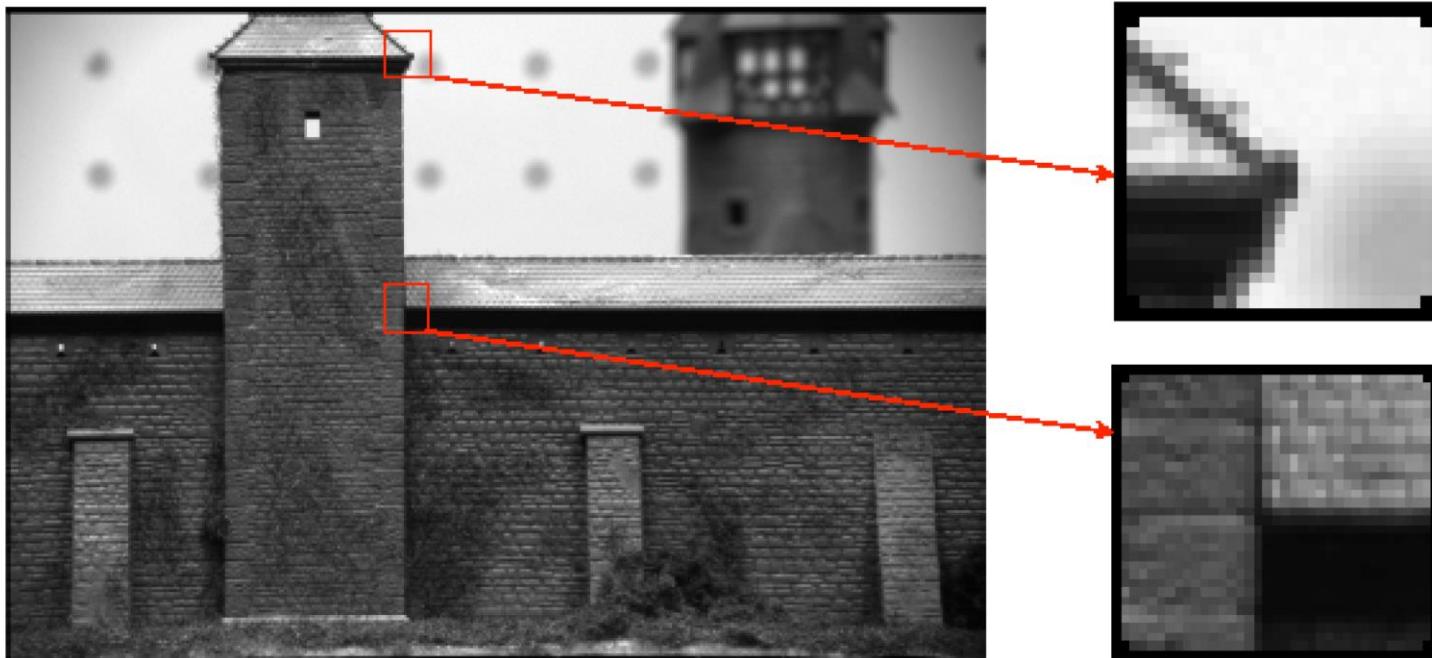
Step 3: align images

# What features should we use?



Use features with gradients in at least two (significantly) different orientations → patches ? Corners ?

# What are Corners?

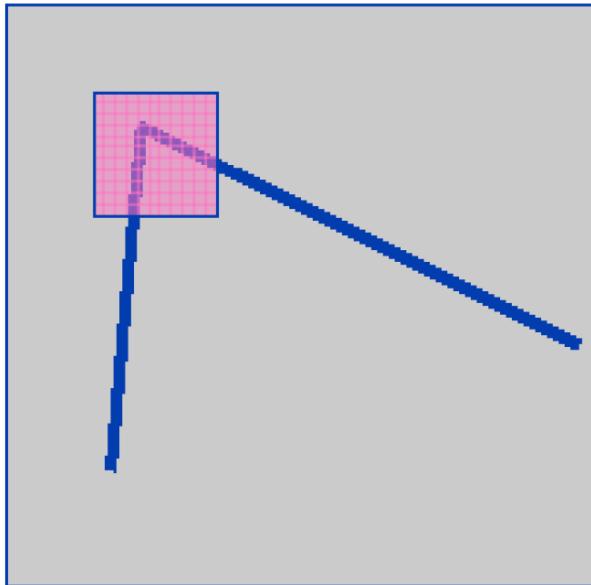


- Intuitively, junctions of contours.
- Generally more stable features over changes of viewpoint
- Intuitively, large variations in the neighborhood of the point in all directions
- They are good features to match!

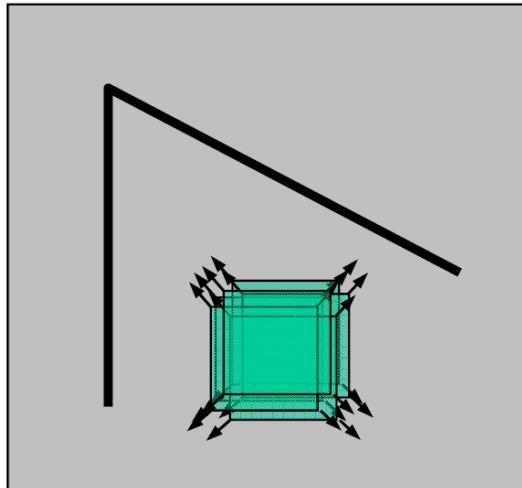
# Corner Points: Basic Idea

---

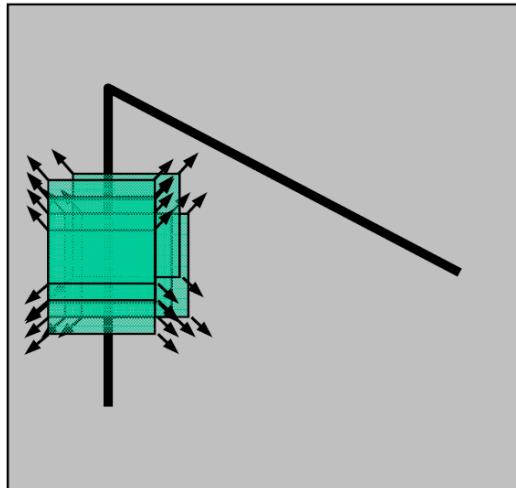
- We should easily recognize the point by looking at intensity values within a small window
- Shifting the window in *any direction* should yield a *large change* in appearance.



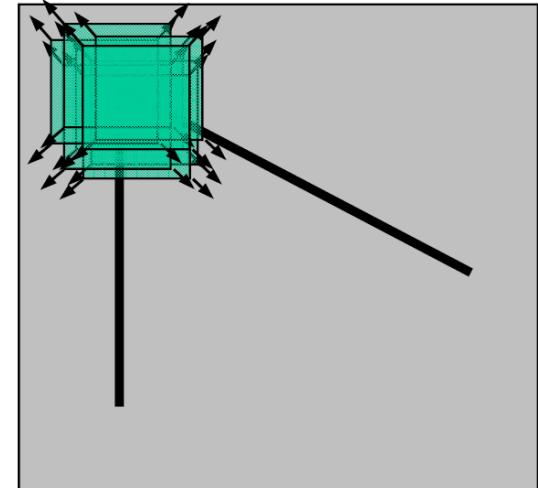
# Corner Detector: Basic Idea



“flat” region:  
no change in  
all directions



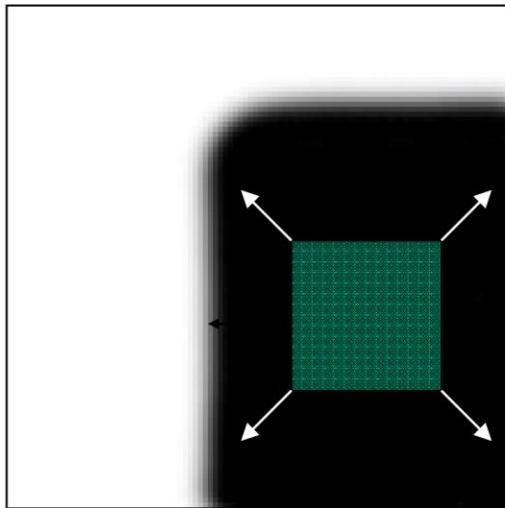
“edge”:  
no change along  
the edge direction



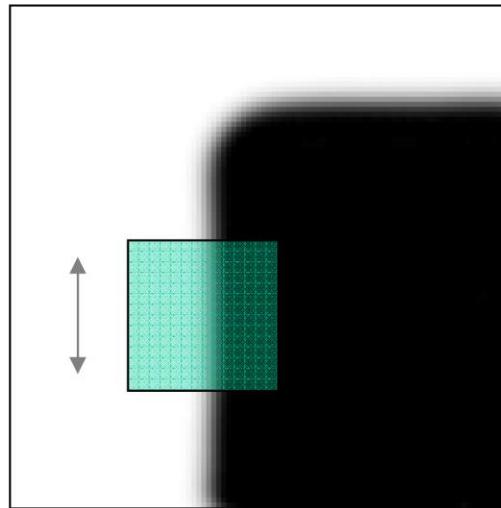
“corner”:  
significant change  
in all directions

# Corner Detection Using Intensity: Basic Idea

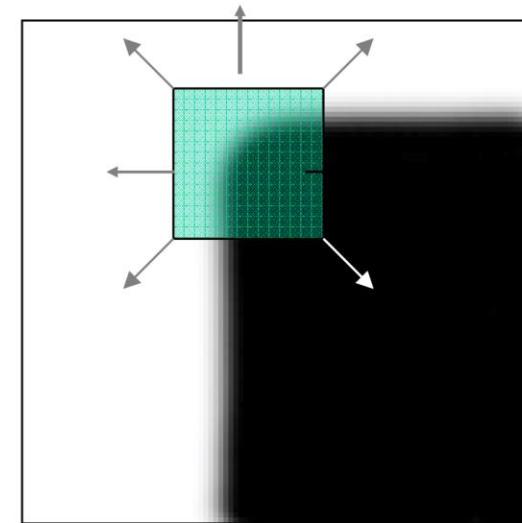
- Image gradient has two or more dominant directions near a corner.
- Shifting a window in *any direction* should give *a large change* in intensity.



“flat” region:  
no change in all  
directions



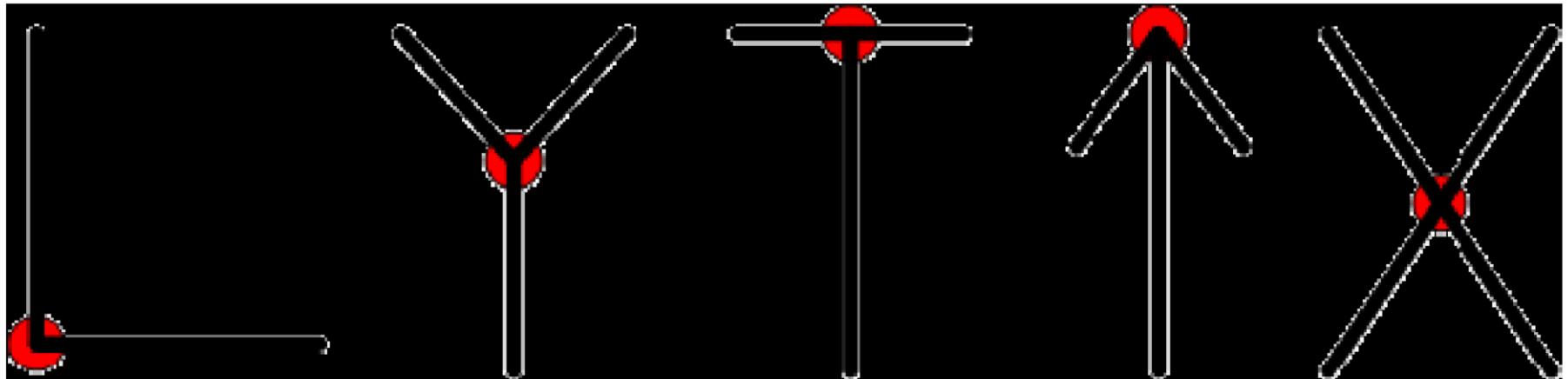
“edge”: no change  
along the edge  
direction



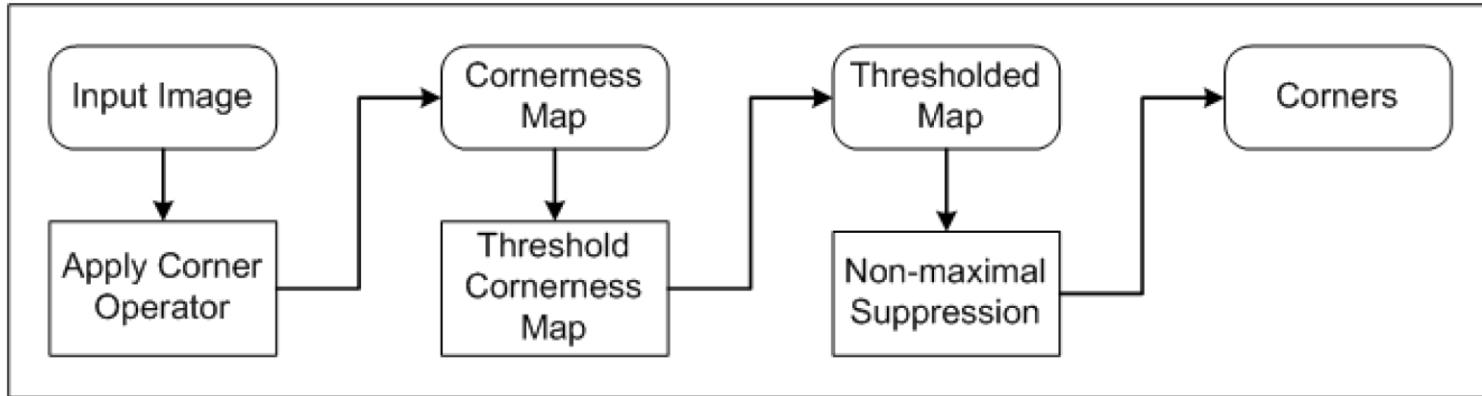
“corner”: significant  
change in all  
directions

# Corner Types

Example of L-junction, Y-junction, T-junction, Arrow-junction, and X-junction corner types

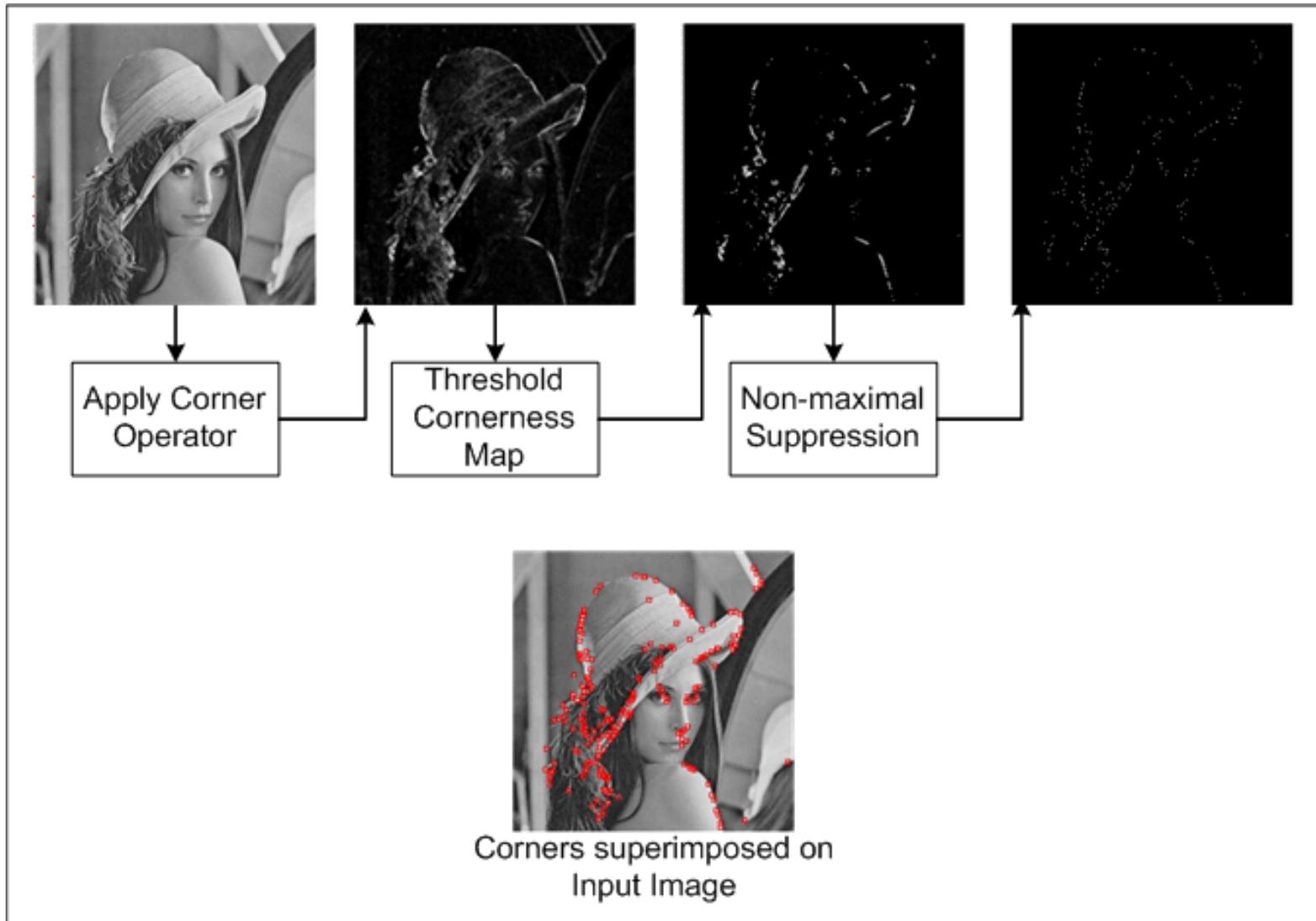


# Mains Steps in Corner Detection



1. For each pixel in the input image, the corner operator is applied to obtain a *corneress* measure for this pixel.
2. Threshold corneress map to eliminate weak corners.
3. Apply non-maximal suppression to eliminate points whose corneress measure is not larger than the corneress values of all points within a certain distance.

# Mains Steps in Corner Detection



# Moravec Detector (1977)

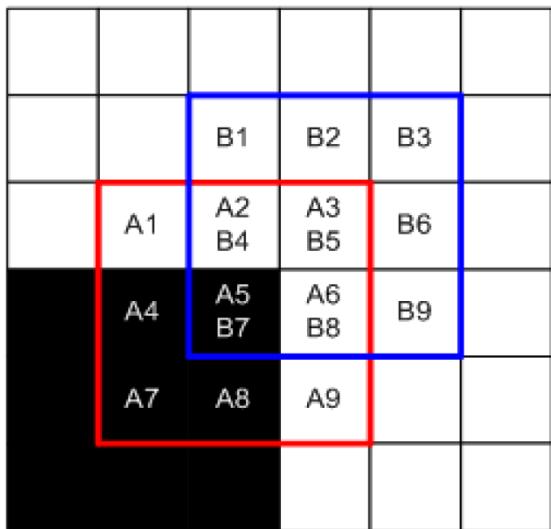
---

Measure intensity variation at  $(x,y)$  by shifting a small window ( $3 \times 3$  or  $5 \times 5$ ) by one pixel in each of the **eight** principle directions (horizontally, vertically, and four diagonals).



# Moravec Detector (1977)

Calculate intensity variation by taking the sum of squares of intensity differences of corresponding pixels in these two windows.



$$S_W(\Delta x, \Delta y) = \sum_{x_i \in W} \sum_{y_i \in W} (f(x_i, y_i) - f(x_i - \Delta x, y_i - \Delta y))^2.$$

8 directions

$\Delta x, \Delta y$  in  $\{-1, 0, 1\}$

$S_W(-1, -1), S_W(-1, 0), \dots, S_W(1, 1)$

# Moravec Detector (1977)

- The “cornerness” of a pixel is the minimum intensity variation found over the eight shift directions:

$$\text{Cornerness}(x,y) = \min\{S_W(-1,-1), S_W(-1,0), \dots, S_W(1,1)\}$$

x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
x	x	0	0	0	0	0	0	0	0	0	0	1	1	1	x	x	
x	x	0	0	0	0	0	0	1	1	0	0	1	2	1	x	x	
x	x	0	0	0	0	0	0	2	1	0	0	1	1	1	x	x	
x	x	0	0	0	0	0	0	0	0	0	0	0	0	0	x	x	
x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	
x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	

Cornerness  
Map  
(normalized)

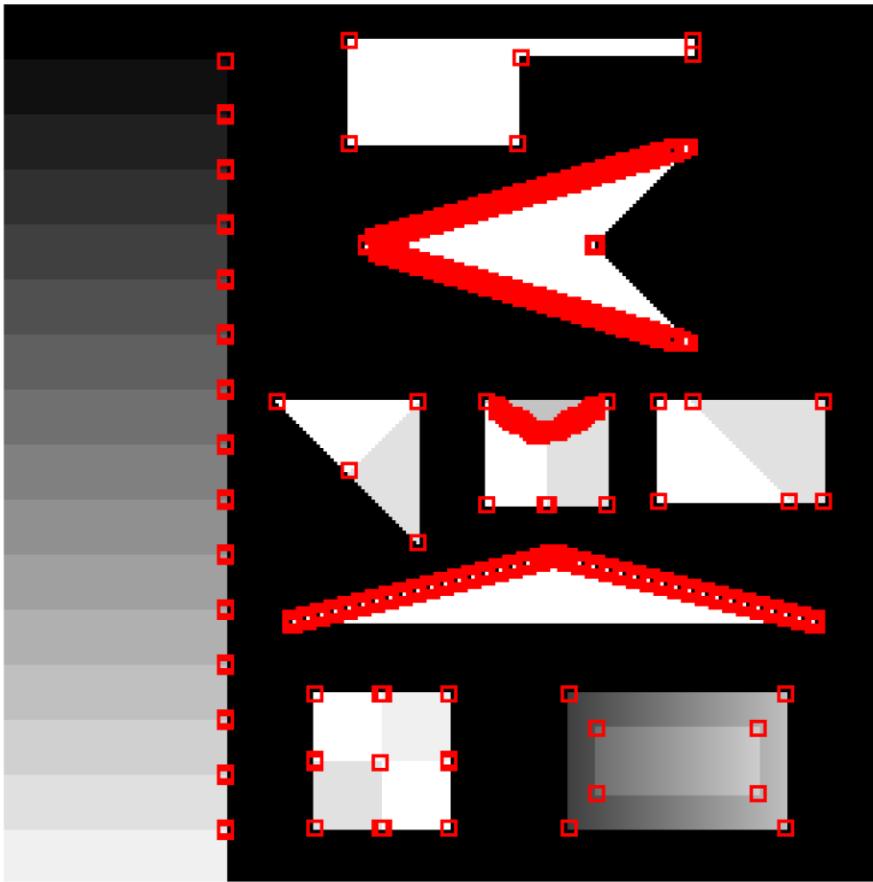
Note response to isolated points!

# Moravec Detector (1977)

- Use Non-maximal suppression will yield the final corners.  
Process of Zero out all pixels that are not the maximum along the direction of the gradient (look at 1 pixel on each side)

X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
X	X	0	0	0	0	0	0	0	0	0	1	1	1	1	X	X	X
X	X	0	0	0	0	0	1	1	0	0	1	2	1	X	X		
X	X	0	0	0	0	0	0	2	1	0	0	1	1	1	X	X	
X	X	0	0	0	0	0	0	0	0	0	0	0	0	0	X	X	
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

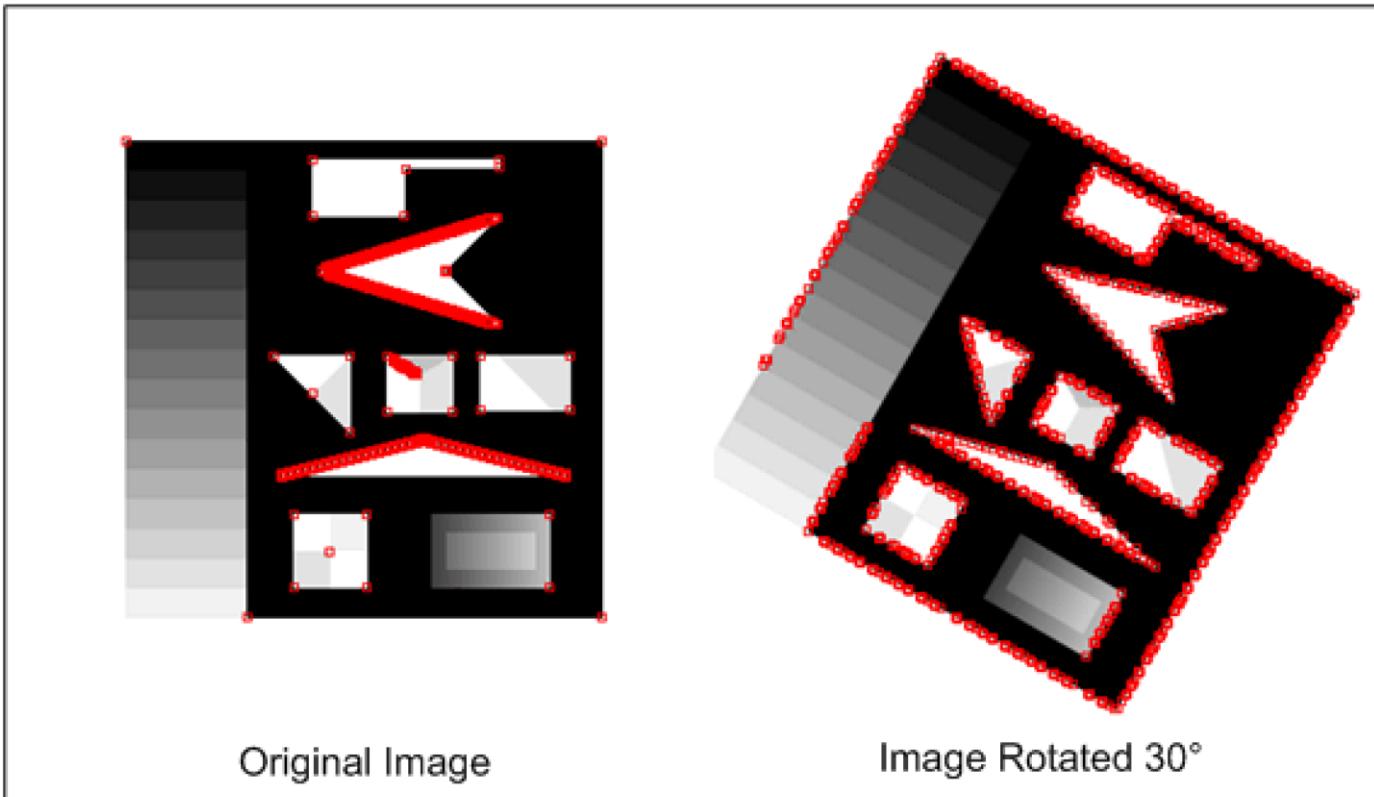
# Moravec Detector (1977)



- Does a reasonable job in finding the majority of true corners.
- Edge points not in one of the eight principle directions will be assigned a relatively large cornerness value.

# Moravec Detector (1977)

- The response is anisotropic (directionally sensitive) as the intensity variation is only calculated at a discrete set of shifts (i.e., not rotationally invariant)



# What we will learn today?

---

- A model fitting method for line detection
    - RANSAC
  - Local invariant features
    - Motivation
    - Requirements, invariances
  - Keypoint localization
    - Harris corner detector
- Some background reading:  
Rick Szeliski, Chapter 4.1.1; David Lowe, IJCV 2004

# Harris Detector: Mathematics

---

Change of intensity for the shift  $[u, v]$ :

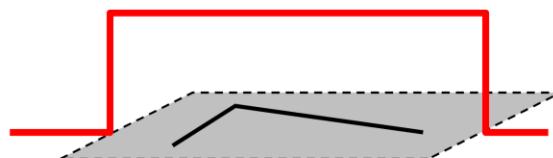
$$E(u, v) = \sum_{x, y} w(x, y) [I(x + u, y + v) - I(x, y)]^2$$

Window  
function

Shifted  
intensity

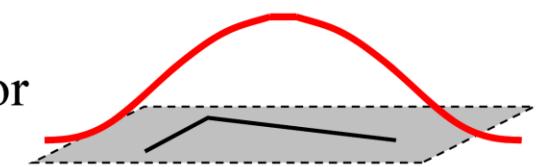
Intensity

Window function  $w(x, y) =$



1 in window, 0 outside

or



Gaussian

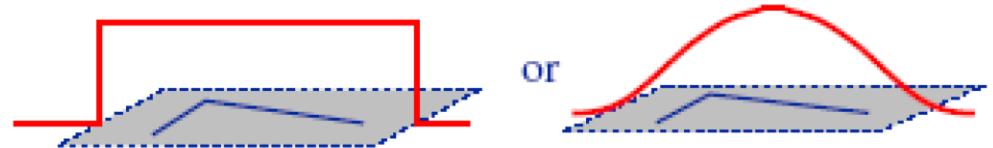
# Harris Detector

Change of intensity for the shift  $[u, v]$ :

$$E(u, v) = \sum_{x, y} w(x, y) [I(x + u, y + v) - I(x, y)]^2$$

Window function     
 Shifted intensity     
 Intensity

Window function  $w(x, y) =$



- Improves the Moravec operator by avoiding the use of discrete directions and discrete shifts.
- Uses a Gaussian window instead of a square window.

# Harris Detector: Intuition

Change of intensity for the shift  $[u, v]$ :

$$E(u, v) = \sum_{x, y} w(x, y) [I(x+u, y+v) - I(x, y)]^2$$

The diagram illustrates the components of the Harris detector equation. A green rectangular box contains the equation  $E(u, v) = \sum_{x, y} w(x, y) [I(x+u, y+v) - I(x, y)]^2$ . Three blue arrows point from three light-green ovals below the box to specific parts of the equation: the left arrow points to  $w(x, y)$  under the summation, the middle arrow points to  $I(x+u, y+v)$ , and the right arrow points to  $I(x, y)$ .

For nearly constant patches, this will be near 0.  
For very distinctive patches, this will be larger.  
Hence... we want patches where  $E(u, v)$  is LARGE.

# Taylor Series for 2D Functions

---

- 一元函数在点 $x_k$ 处的泰勒展开式为：

$$f(x) = f(x_k) + (x - x_k)f'(x_k) + \frac{1}{2!}(x - x_k)^2 f''(x_k) + o^n$$

- 二元函数在点 $(x_k, y_k)$ 处的泰勒展开式为：

$$\begin{aligned} f(x, y) = & f(x_k, y_k) + (x - x_k)f'_x(x_k, y_k) + (y - y_k)f'_y(x_k, y_k) \\ & + \frac{1}{2!}(x - x_k)^2 f''_{xx}(x_k, y_k) + \frac{1}{2!}(x - x_k)(y - y_k)f''_{xy}(x_k, y_k) \\ & + \frac{1}{2!}(x - x_k)(y - y_k)f''_{yx}(x_k, y_k) + \frac{1}{2!}(y - y_k)^2 f''_{yy}(x_k, y_k) \\ & + o^n \end{aligned}$$

# Taylor Series for 2D Functions

$$f(x+u, y+v) = f(x, y) + uf_x(x, y) + vf_y(x, y) +$$

**First partial derivatives**

$$\frac{1}{2!} [u^2 f_{xx}(x, y) + uv f_{xy}(x, y) + v^2 f_{yy}(x, y)] +$$

**Second partial derivatives**

$$\frac{1}{3!} [u^3 f_{xxx}(x, y) + u^2 v f_{xxy}(x, y) + u v^2 f_{xyy}(x, y) + v^3 f_{yyy}(x, y)]$$

**Third partial derivatives**

+ ... (Higher order terms)

First order approx

$$f(x+u, y+v) \approx f(x, y) + uf_x(x, y) + vf_y(x, y)$$

# Harris Corner Derivation

---

$$\sum [I(x+u, y+v) - I(x, y)]^2$$

$$\approx \sum [I(x, y) + uI_x + vI_y - I(x, y)]^2 \quad \text{First order approx}$$

$$= \sum u^2 I_x^2 + 2uvI_xI_y + v^2 I_y^2$$

$$= \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} \quad \text{Rewrite as matrix equation}$$

$$= \begin{bmatrix} u & v \end{bmatrix} \left( \sum \begin{bmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{bmatrix} \right) \begin{bmatrix} u \\ v \end{bmatrix}$$

# Harris Detector: Mathematics

---

For small shifts  $[u, v]$  we have a *bilinear* approximation:

$$E(u, v) \cong [u, v] \cdot M \begin{bmatrix} u \\ v \end{bmatrix}$$

where  $M$  is a  $2 \times 2$  matrix computed from image derivatives:

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

**Windowing function - computing a weighted sum (simplest case,  $w=1$ )**

**Note: these are just products of components of the gradient,  $I_x, I_y$**

# Intuitive Way to Understand Harris

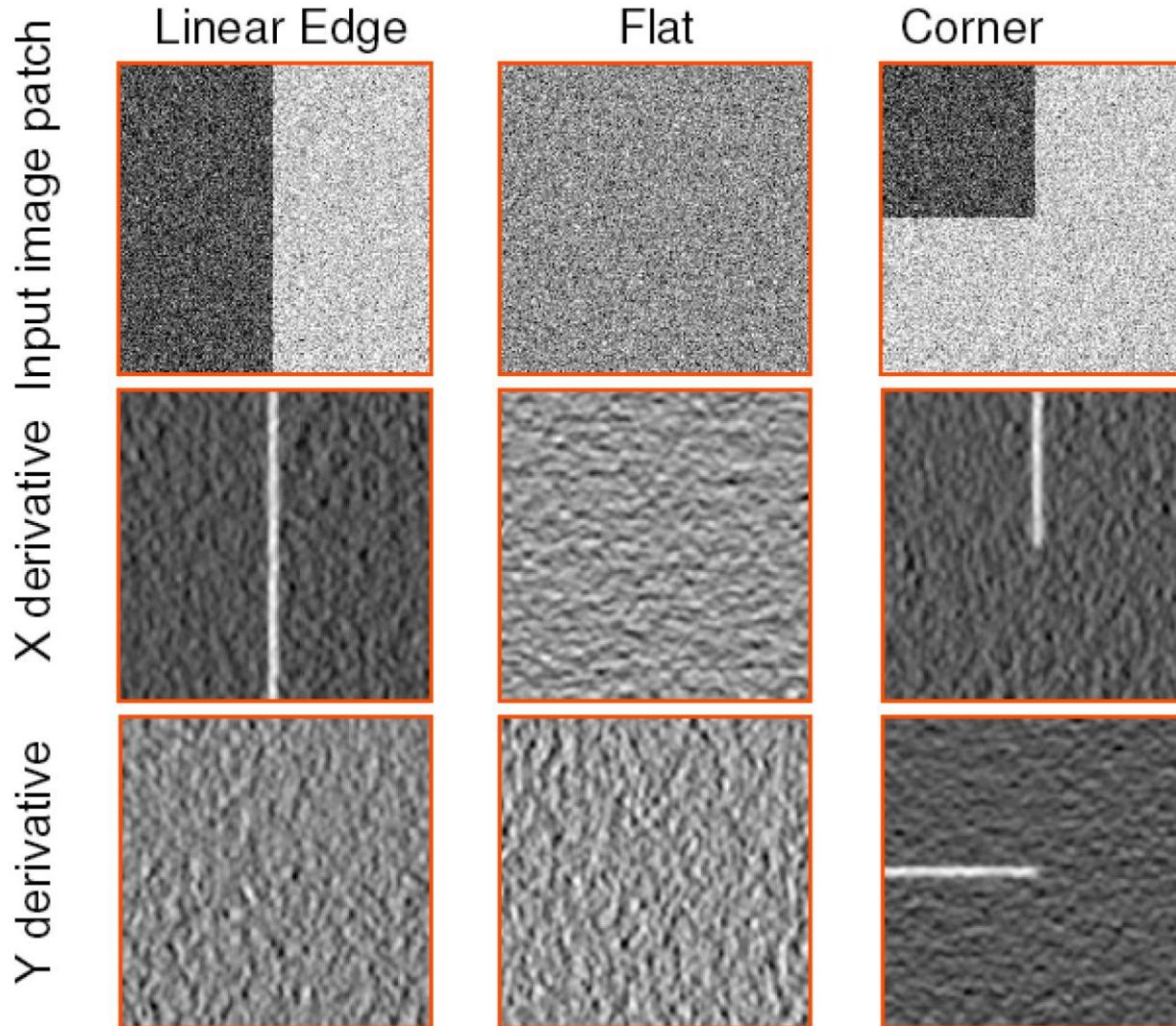
---

Treat gradient vectors as a set of  $(dx, dy)$  points with a center of mass defined as being at  $(0,0)$ .

Fit an ellipse to that set of points via scatter matrix

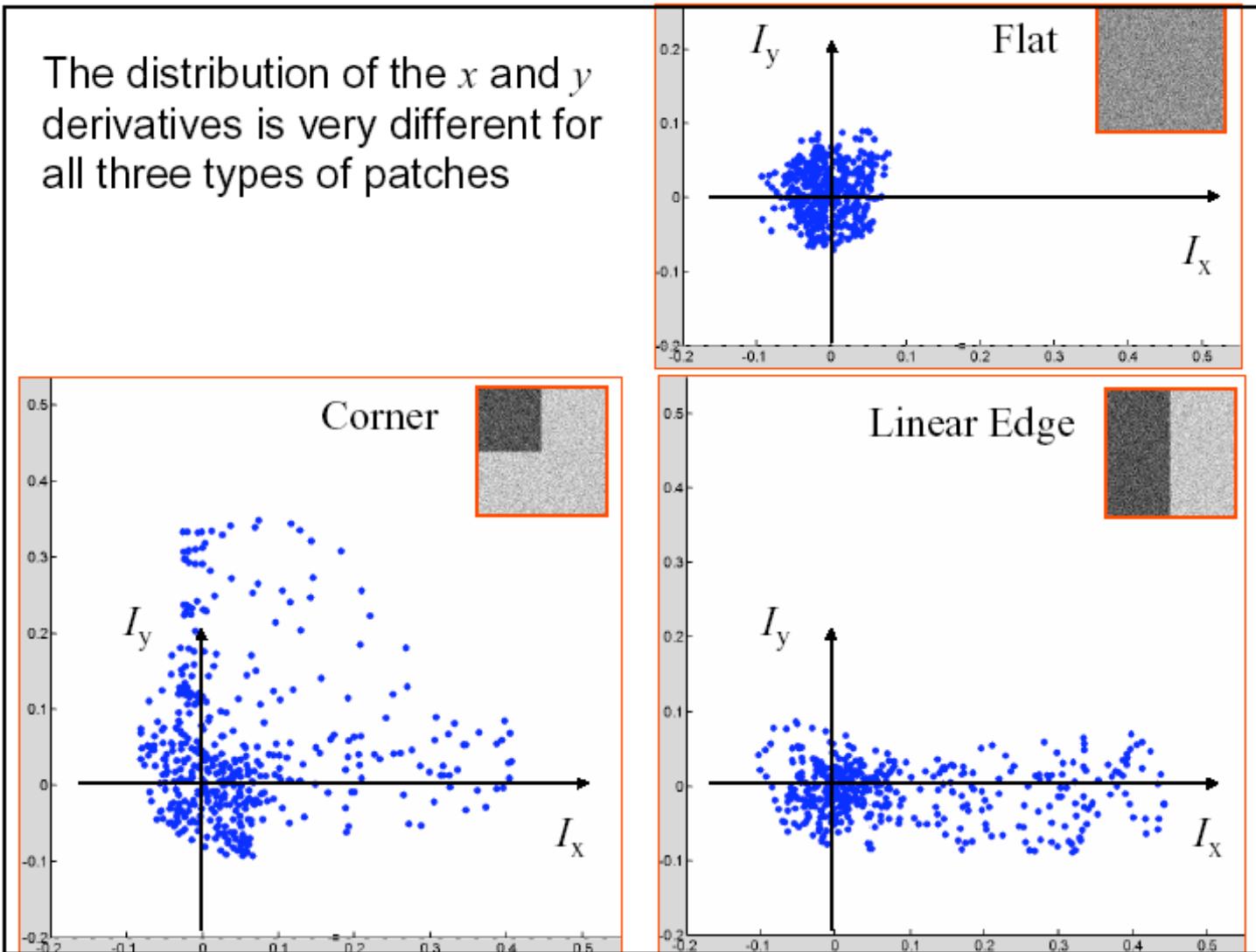
Analyze ellipse parameters for varying cases...

# Example: Cases and 2D Derivatives



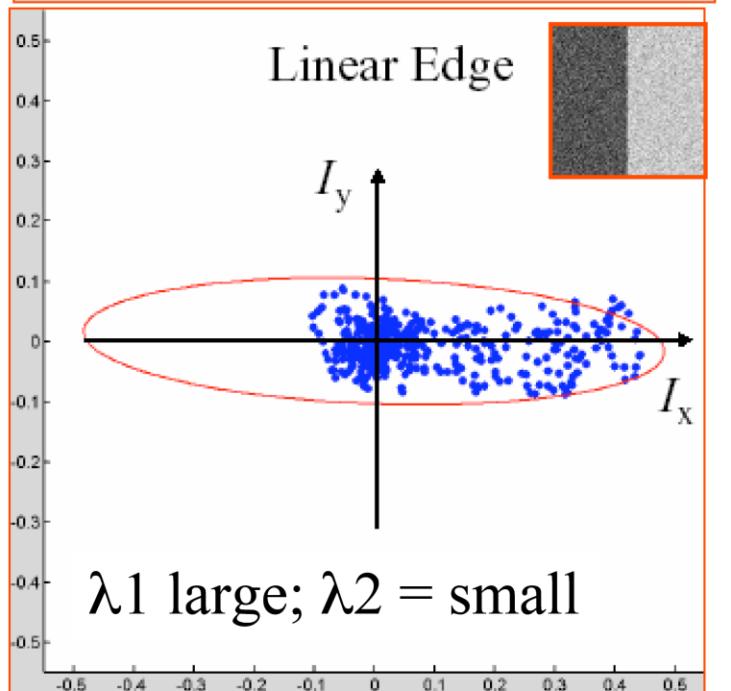
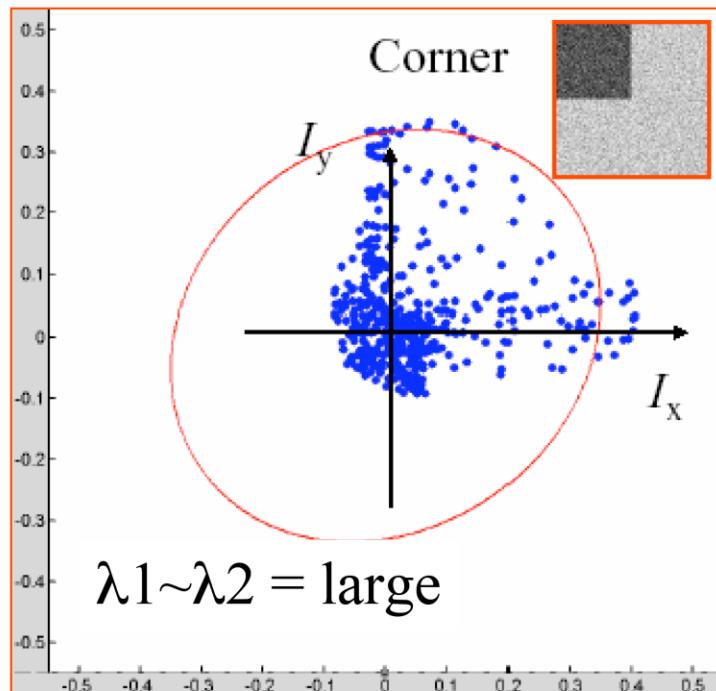
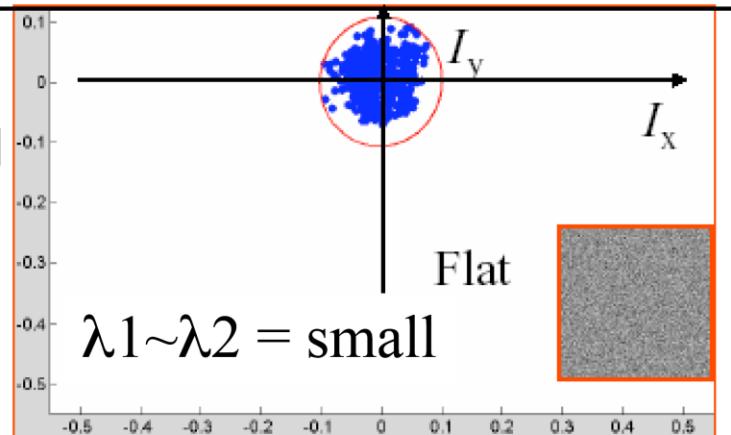
# Plotting Derivatives as 2D Points

The distribution of the  $x$  and  $y$  derivatives is very different for all three types of patches



# Fitting Ellipse to each Set of Points

The distribution of  $x$  and  $y$  derivatives can be characterized by the shape and size of the principal component ellipse

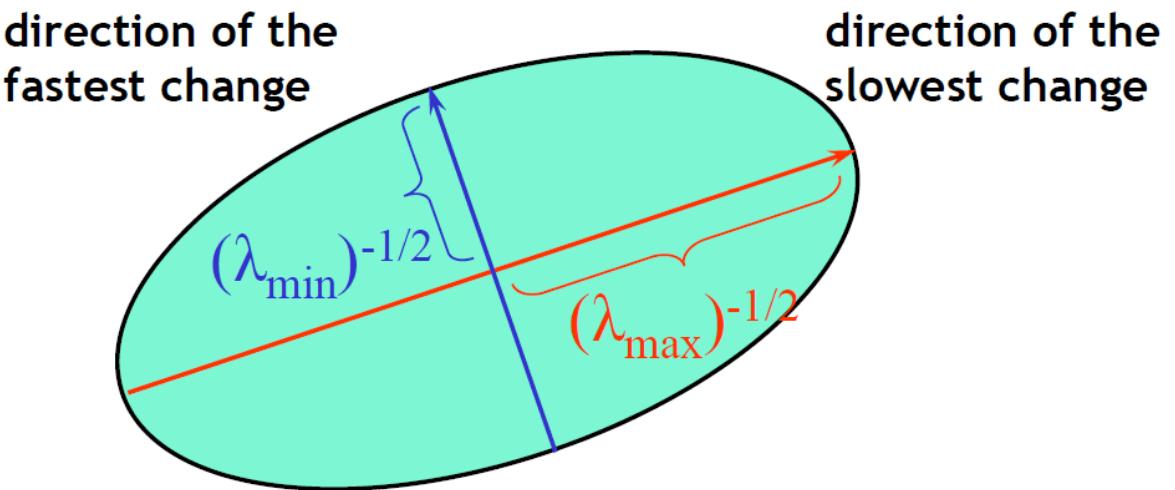


# Harris Detector

Since  $M$  is symmetric, we have:

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

We can visualize  $M$  as an ellipse with axis lengths determined by the eigenvalues and orientation determined by  $R$



# Classification via Eigenvalues

Classification of image points using eigenvalues of  $M$ :

$\lambda_1$  and  $\lambda_2$  are small;  
 $E$  is almost constant  
in all directions

$\lambda_2$

“Edge”

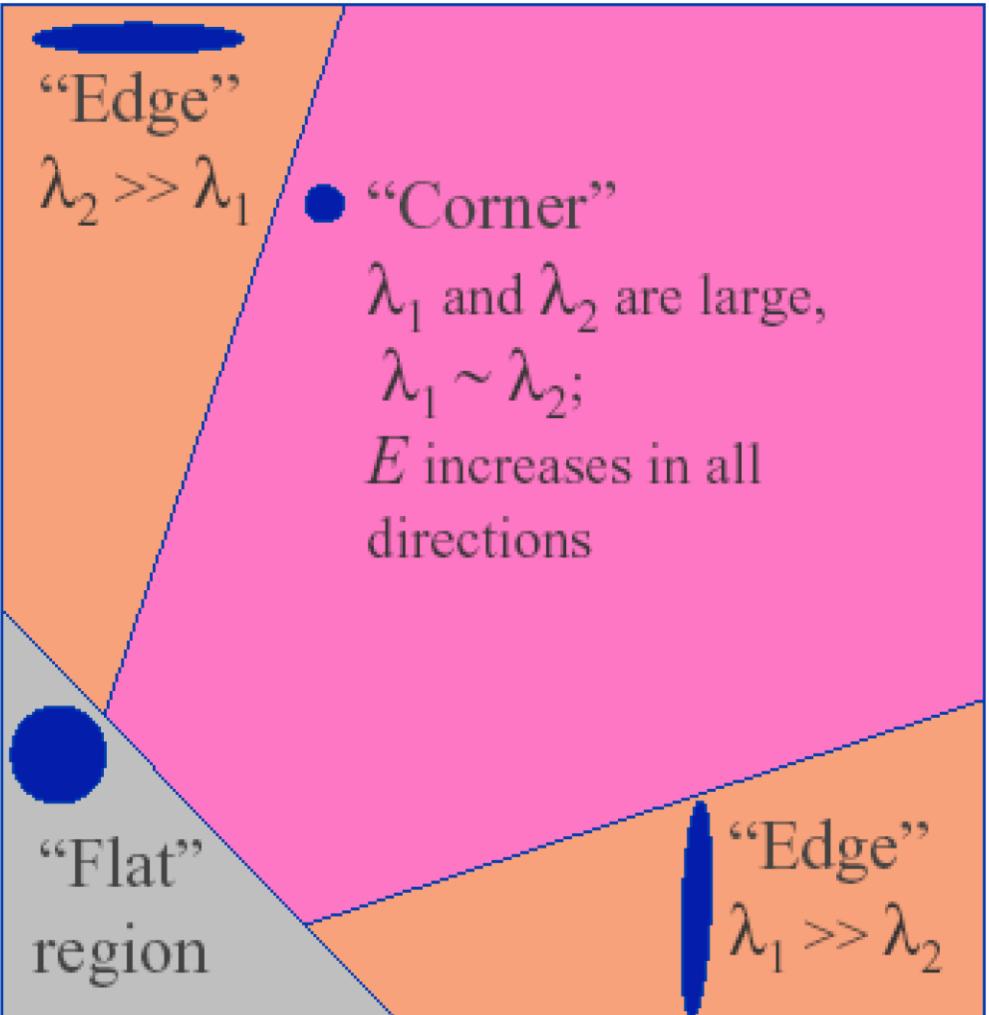
$\lambda_2 > \lambda_1$

● “Corner”

$\lambda_1$  and  $\lambda_2$  are large,  
 $\lambda_1 \sim \lambda_2$ ;  
 $E$  increases in all  
directions

“Flat”  
region

$\lambda_1$



# Corner Response Measure

---

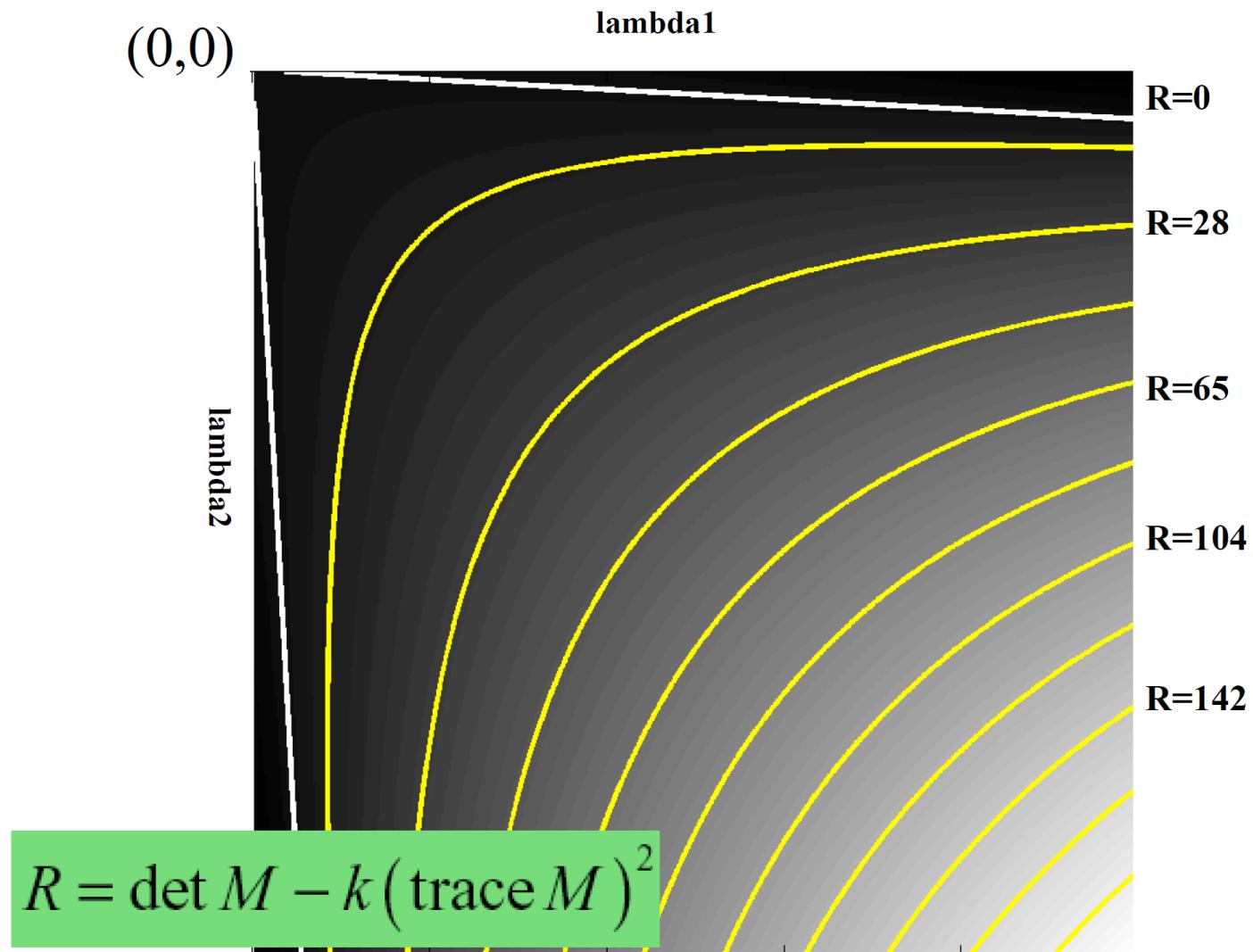
Measure of corner response:

$$R = \det M - k(\operatorname{trace} M)^2$$

$$\det M = \lambda_1 \lambda_2$$
$$\operatorname{trace} M = \lambda_1 + \lambda_2$$

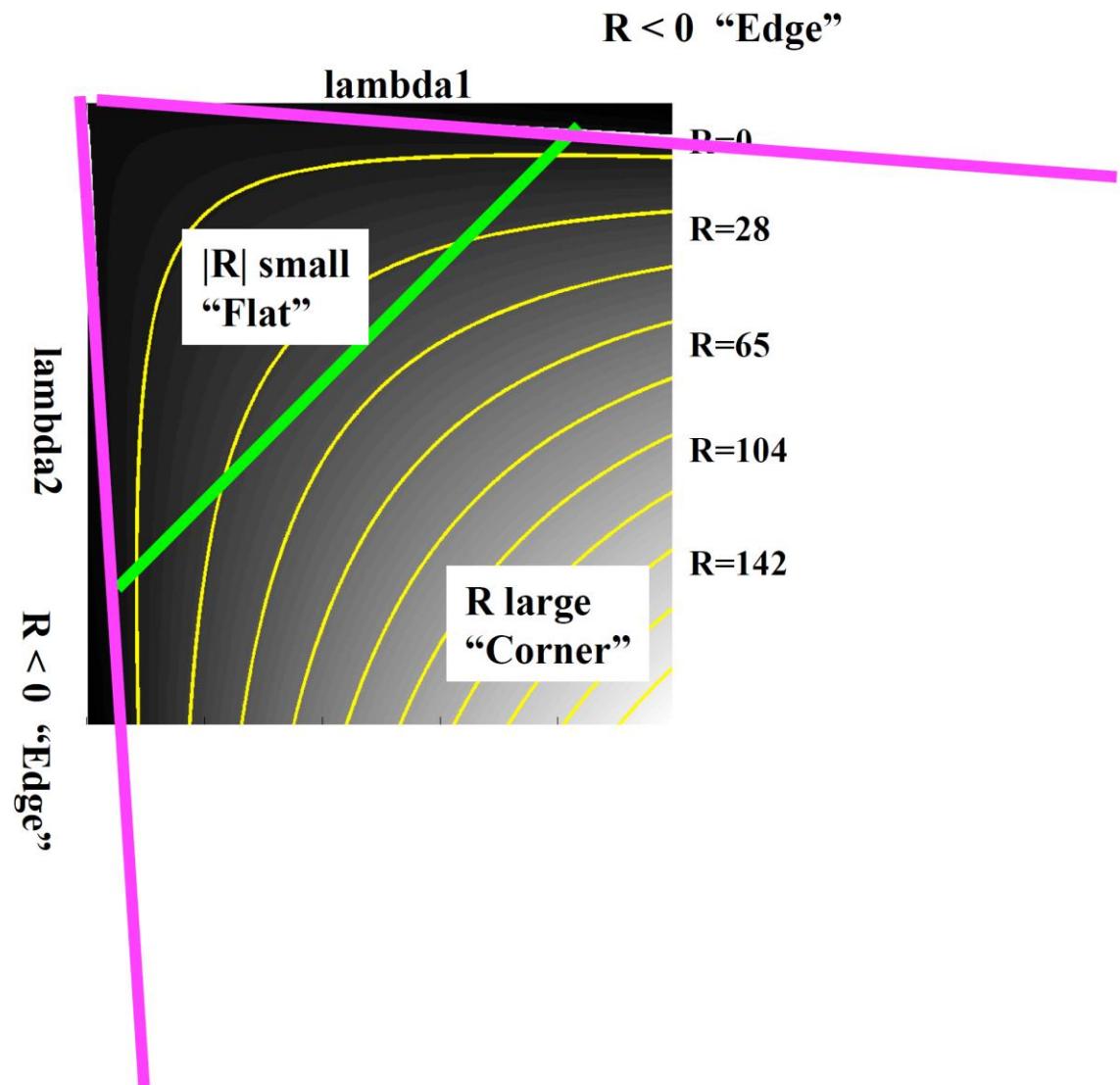
( $k$  is an empirically determined constant;  $k = 0.04 - 0.06$ )

# Corner Response Map

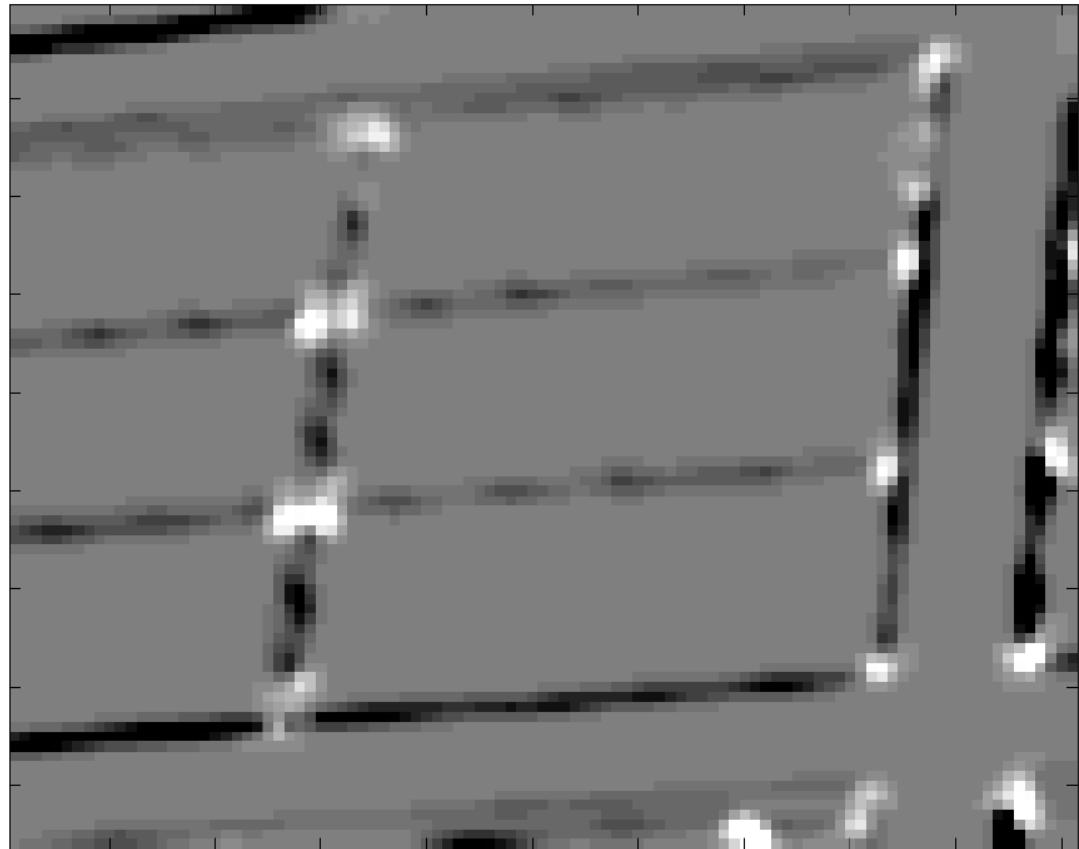
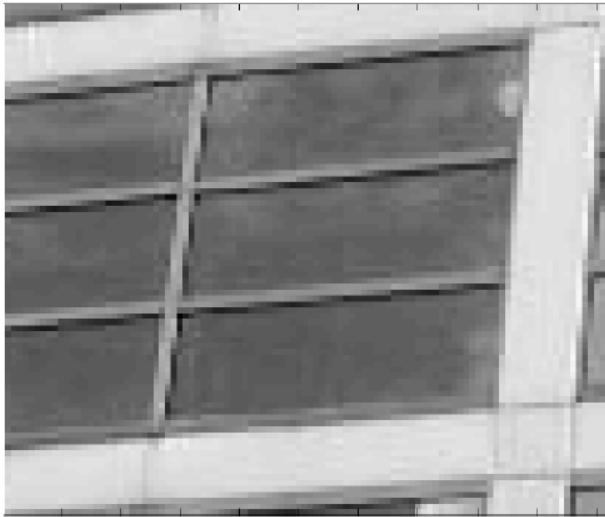


# Corner Response Map

- $R$  depends only on eigenvalues of  $M$
- $R$  is large for a corner
- $R$  is negative with large magnitude for an edge
- $|R|$  is small for a flat region



# Corner Response Example



Harris R score.

$I_x, I_y$  computed using Sobel operator

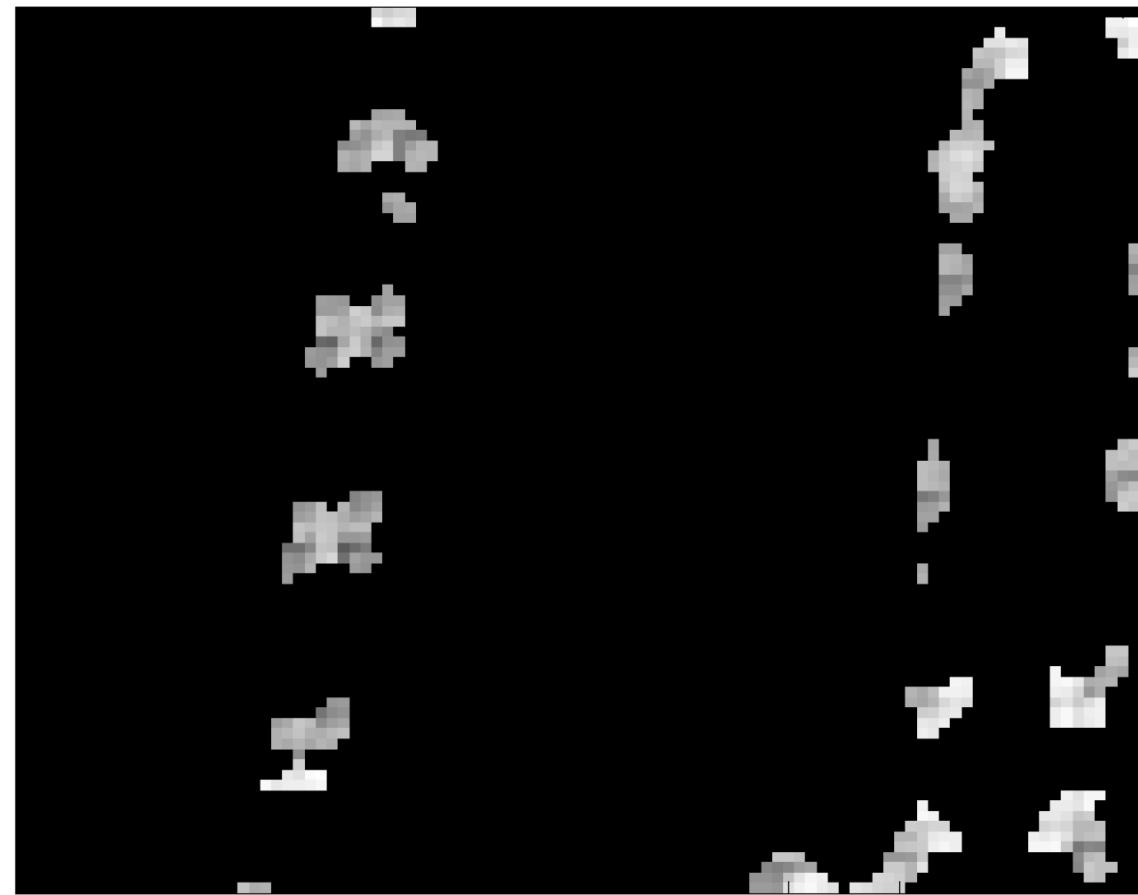
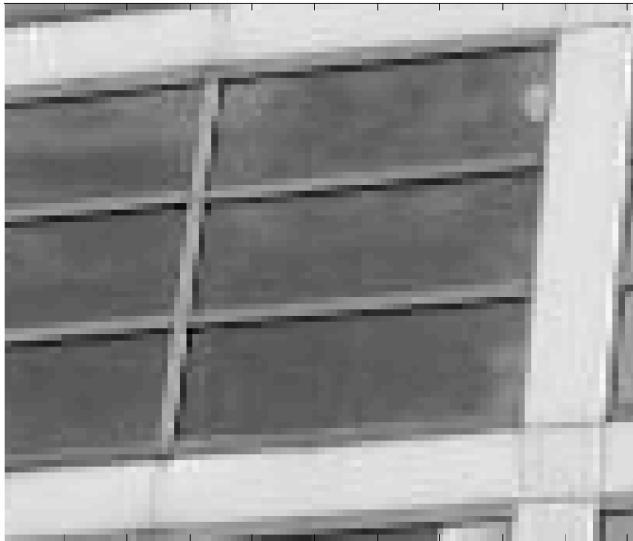
Windowing function  $w = \text{Gaussian}$ ,  $\sigma=1$

# Corner Response Example



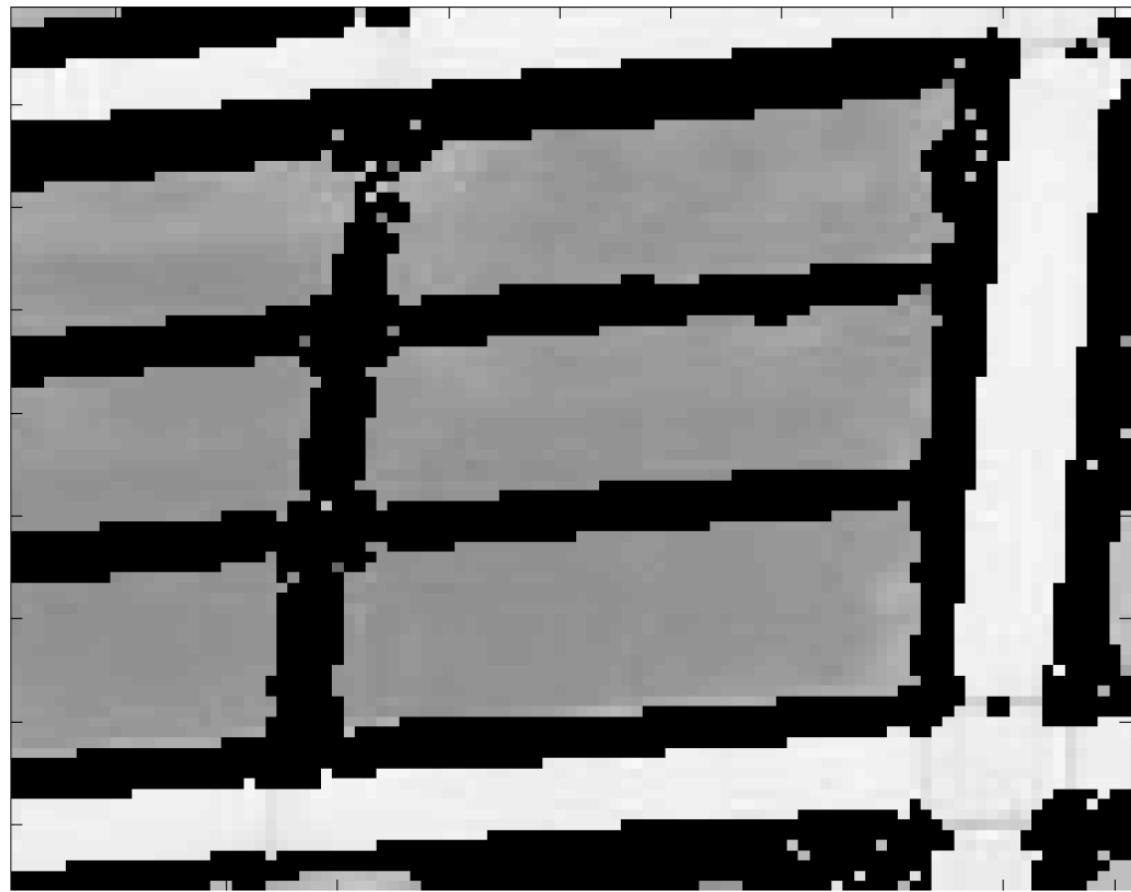
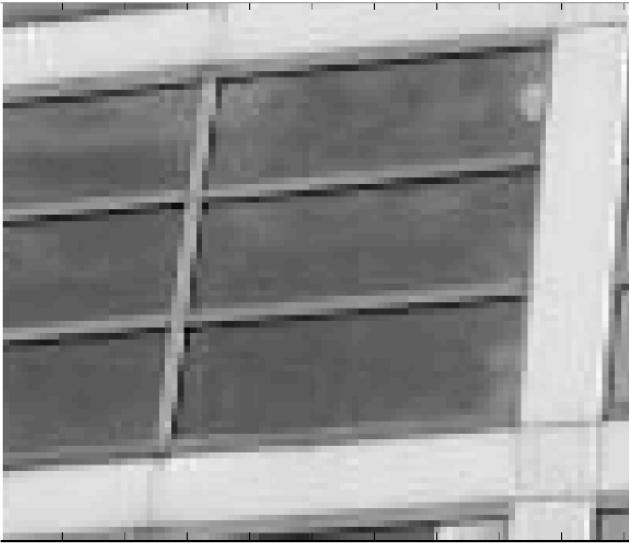
Threshold:  $R < -10000$   
(edges)

# Corner Response Example



Threshold:  $> 10000$   
(corners)

# Corner Response Example



Threshold:  $-10000 < R < 10000$   
(neither edges nor corners)

# Harris Corner Detection Algorithm

---

1. Compute  $x$  and  $y$  derivatives of image

$$I_x = G_\sigma^x * I \quad I_y = G_\sigma^y * I$$

2. Compute products of derivatives at every pixel

$$I_{x2} = I_x \cdot I_x \quad I_{y2} = I_y \cdot I_y \quad I_{xy} = I_x \cdot I_y$$

3. Compute the sums of the products of derivatives at each pixel

$$S_{x2} = G_{\sigma t} * I_{x2} \quad S_{y2} = G_{\sigma t} * I_{y2} \quad S_{xy} = G_{\sigma t} * I_{xy}$$

4. Define at each pixel  $(x, y)$  the matrix

$$H(x, y) = \begin{bmatrix} S_{x2}(x, y) & S_{xy}(x, y) \\ S_{xy}(x, y) & S_{y2}(x, y) \end{bmatrix}$$

5. Compute the response of the detector at each pixel

$$R = \text{Det}(H) - k(\text{Trace}(H))^2$$

6. Threshold on value of R. Compute nonmax suppression.

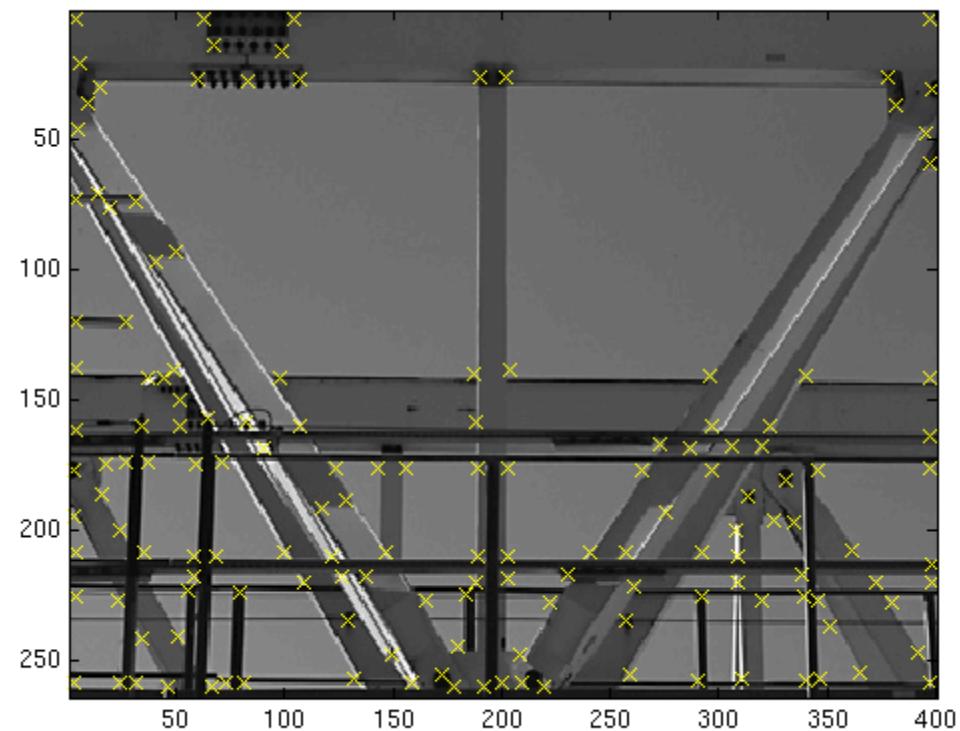
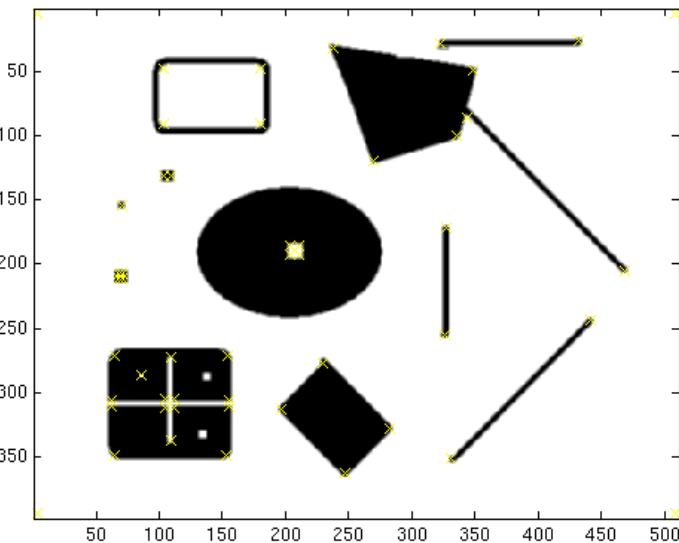
# Harris Detector: Workflow

- Resulting Harris points



# Harris Detector – Responses [Harris88]

- Effect: A very precise corner detector.



# Harris Detector – Responses [Harris88]



# Harris Detector – Responses [Harris88]

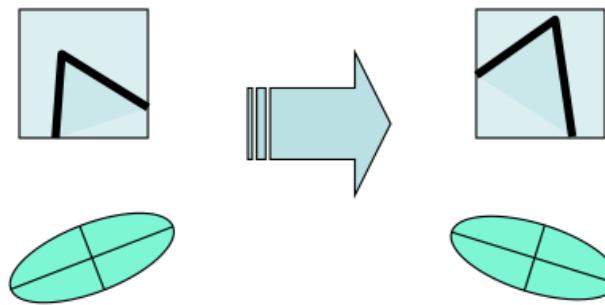
- Results are well suited for finding stereo correspondences



# Harris Detector: Properties

- Translation invariance

- Rotation invariance

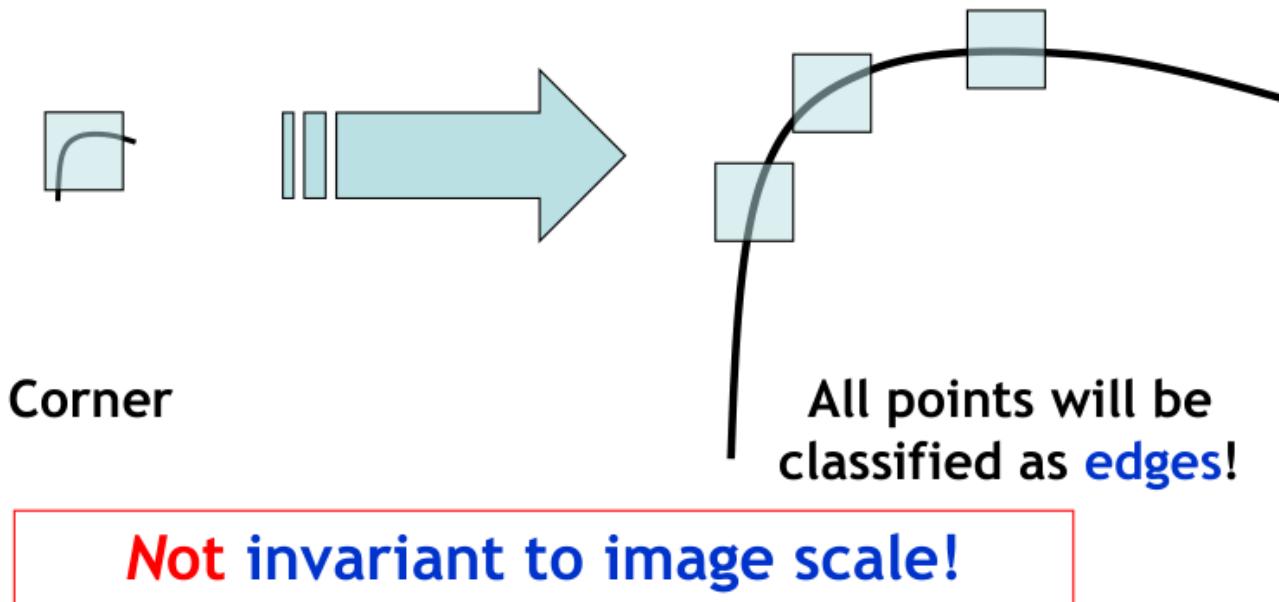


Ellipse rotates but its shape (i.e. eigenvalues) remains the same

*Corner response  $\theta$  is invariant to image rotation*

# Harris Detector: Properties

- Translation invariance
- Rotation invariance
- **Scale invariance**



# What we are learned today?

---

- A model fitting method for line detection
  - RANSAC
- Local invariant features
  - Motivation
  - Requirements, invariances
- Keypoint localization
  - Harris corner detector