

Lecture 6

Lines and Corners

Administrative

A1 due Fri, Jan 24!!!

- You can use up to 2 late days

A2 is out

- Due Feb 7th

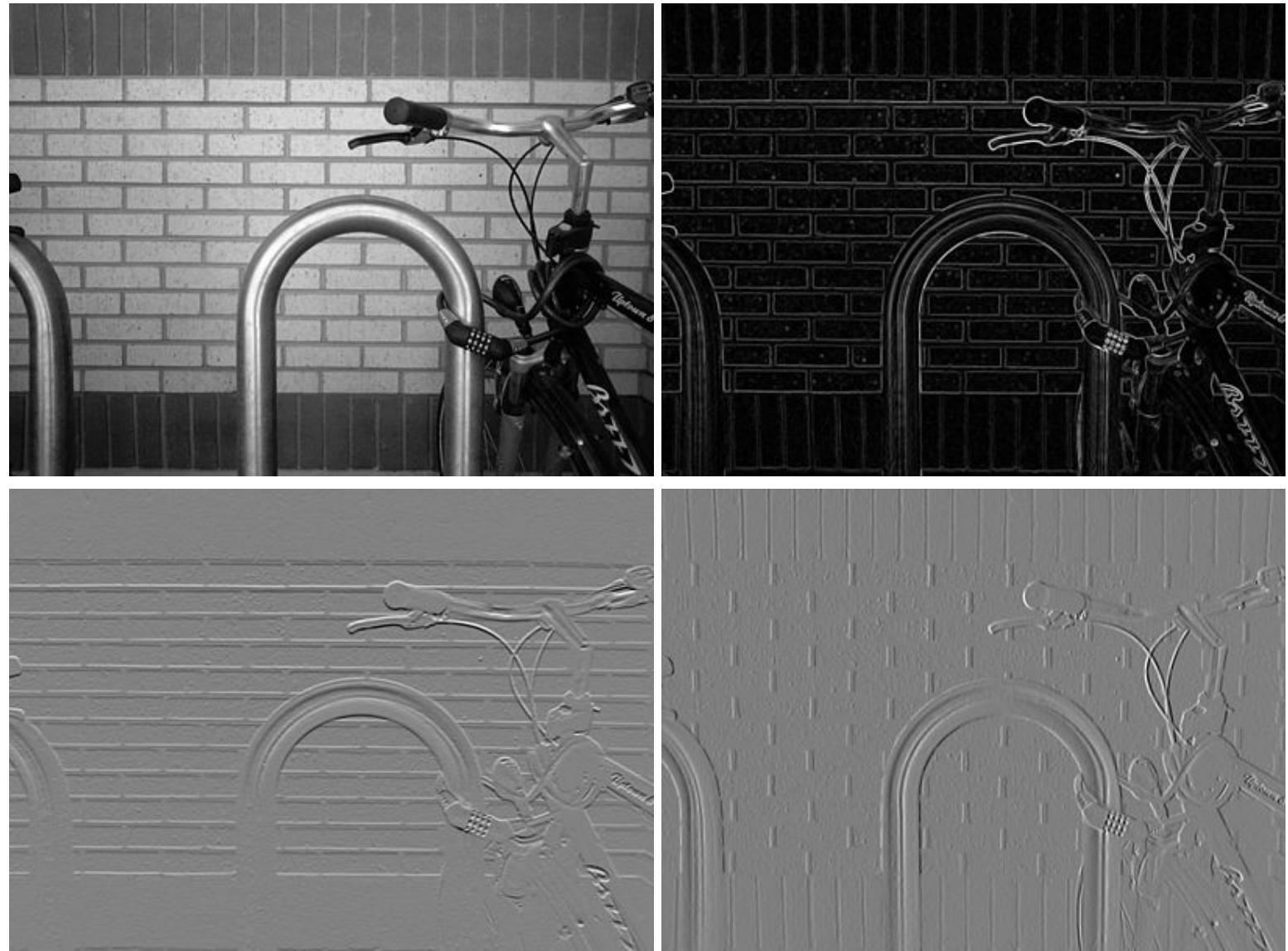
Administrative

- Recitation this Friday
- Geometric transformations

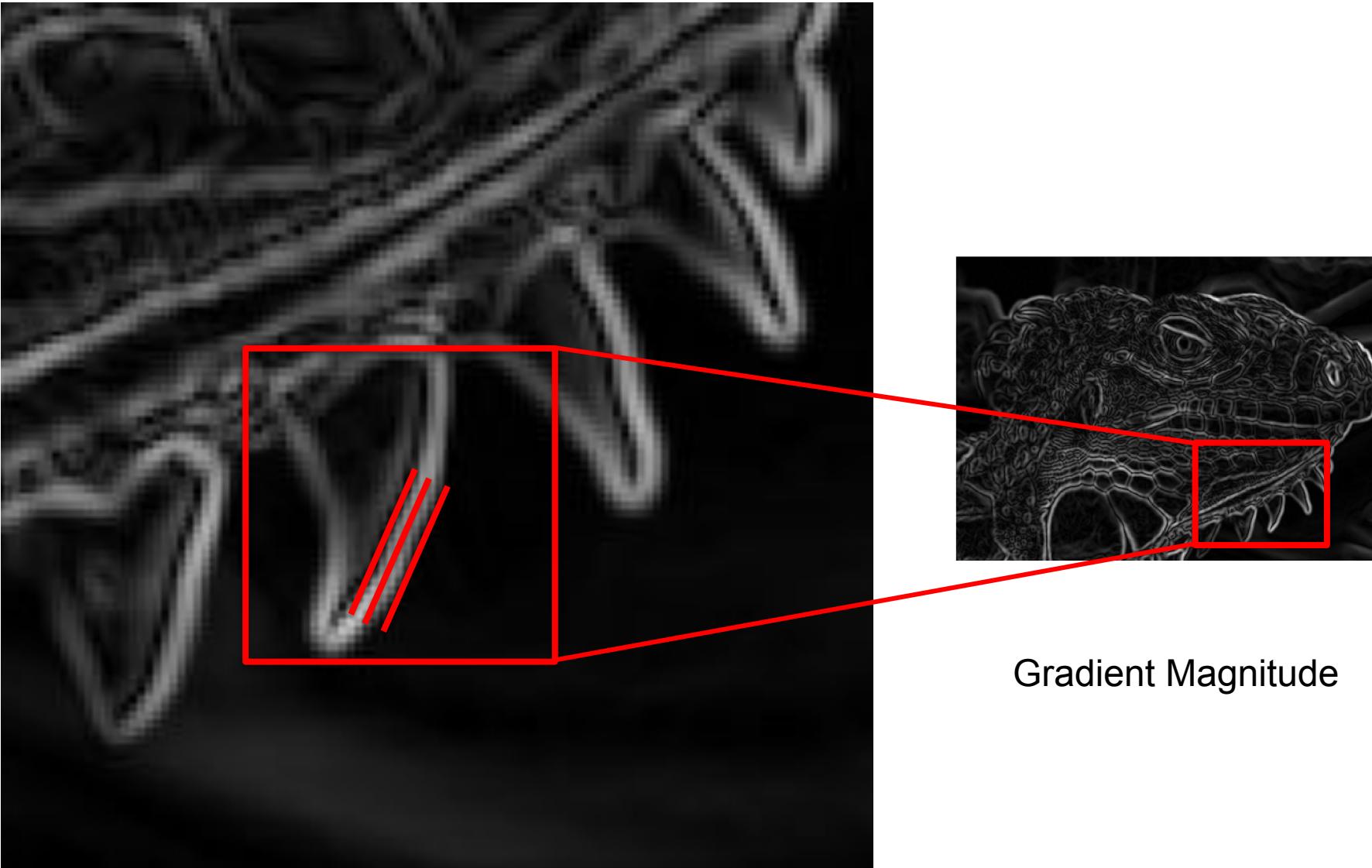
So far: Sobel Filter

Step 1: Calculate the gradient magnitude at every pixel location.

Step 2: Threshold the values to generate a binary image

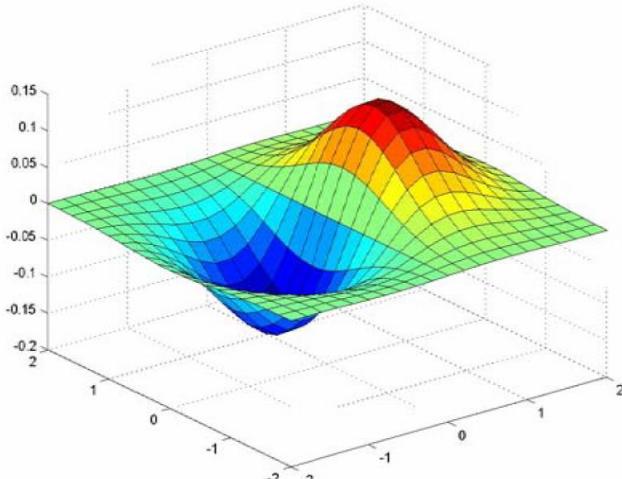


So far: challenges multiple disconnected edges

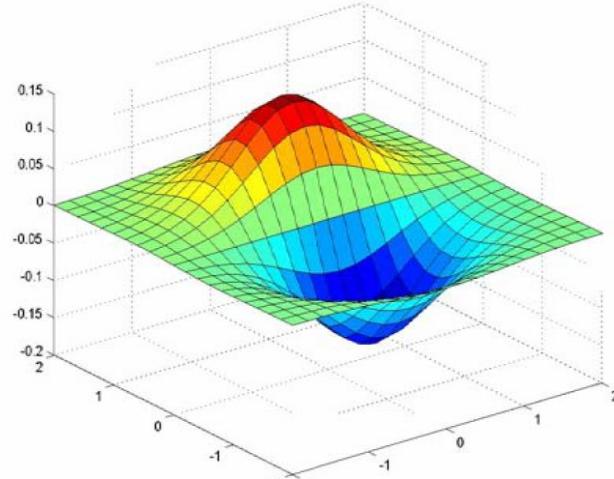


Gradient Magnitude

So far: Canny edge detector
Use Sobel filters to find line estimates

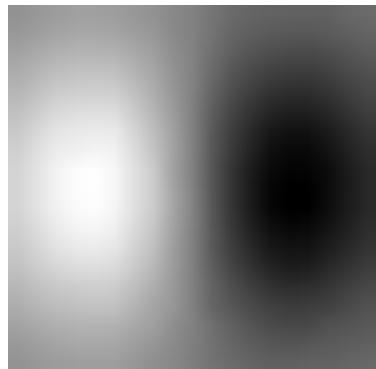


x-direction

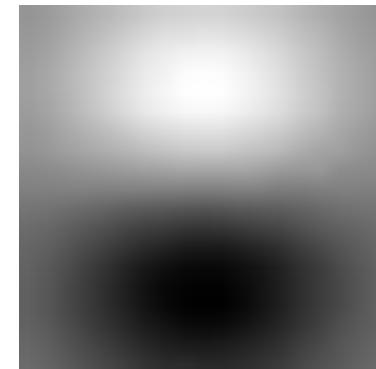


y-direction

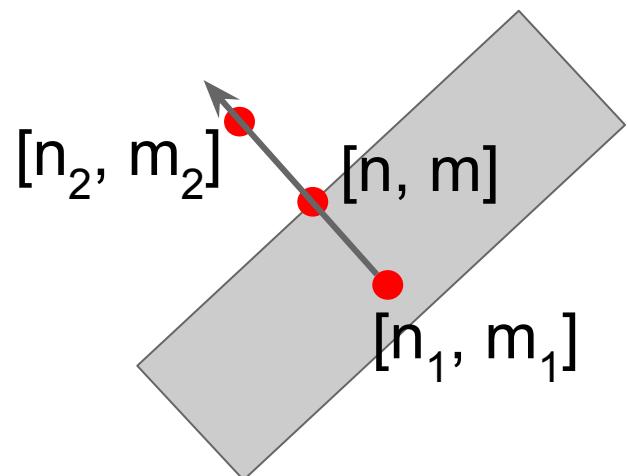
$$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$



$$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

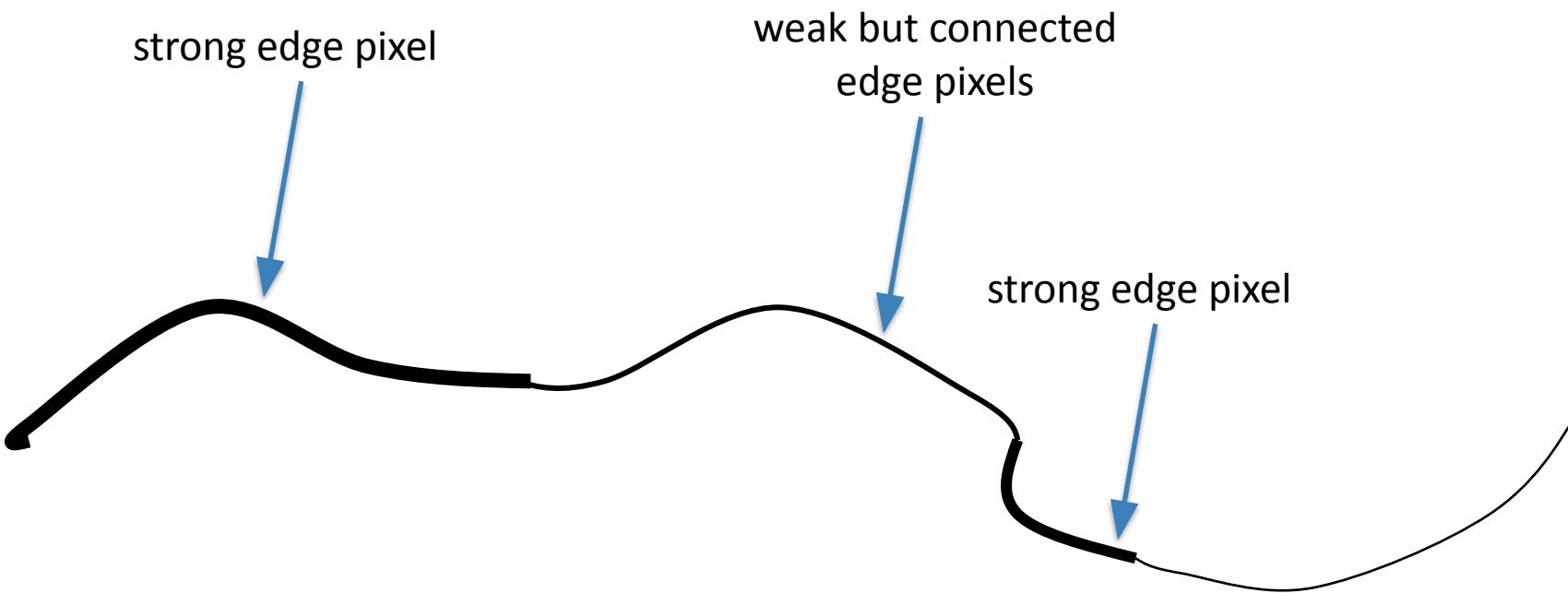


So far: Non-maximum suppression



So far: Hysteresis thresholding

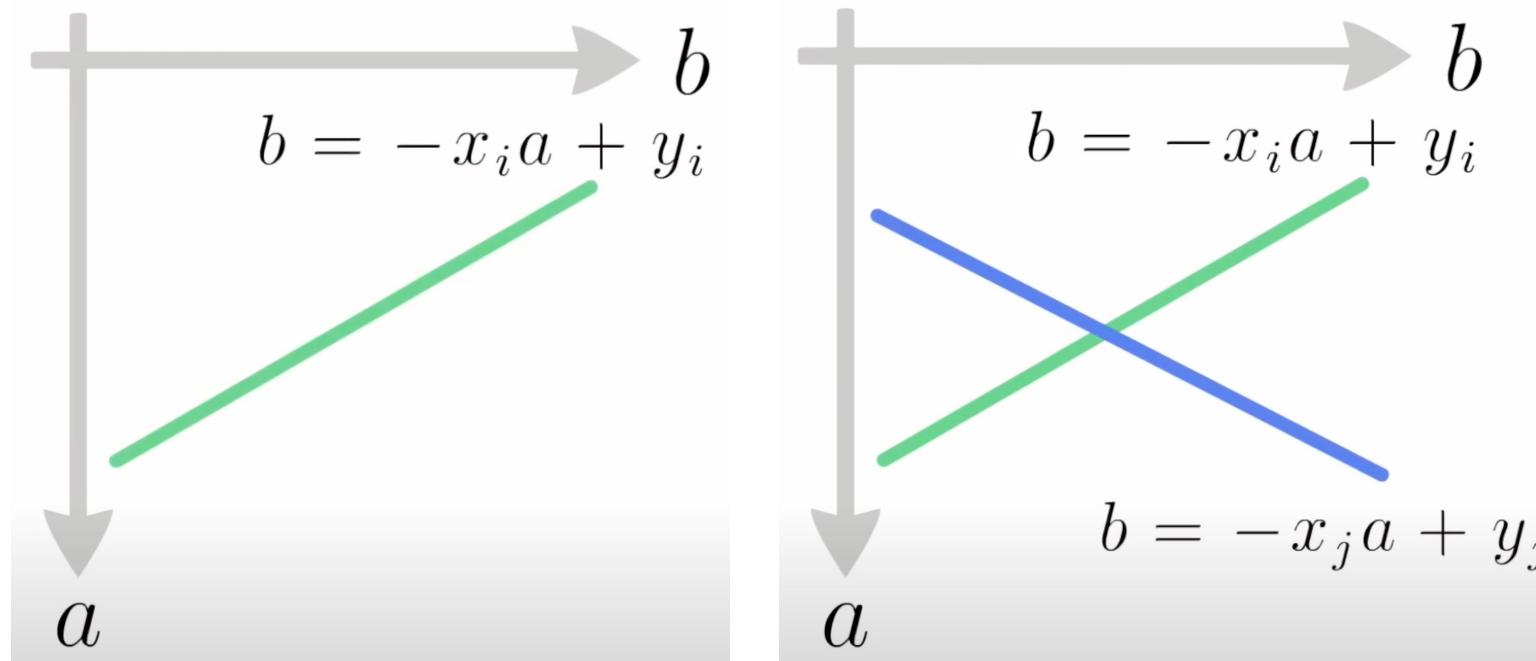
Strong and weak edges



Source: S. Seitz

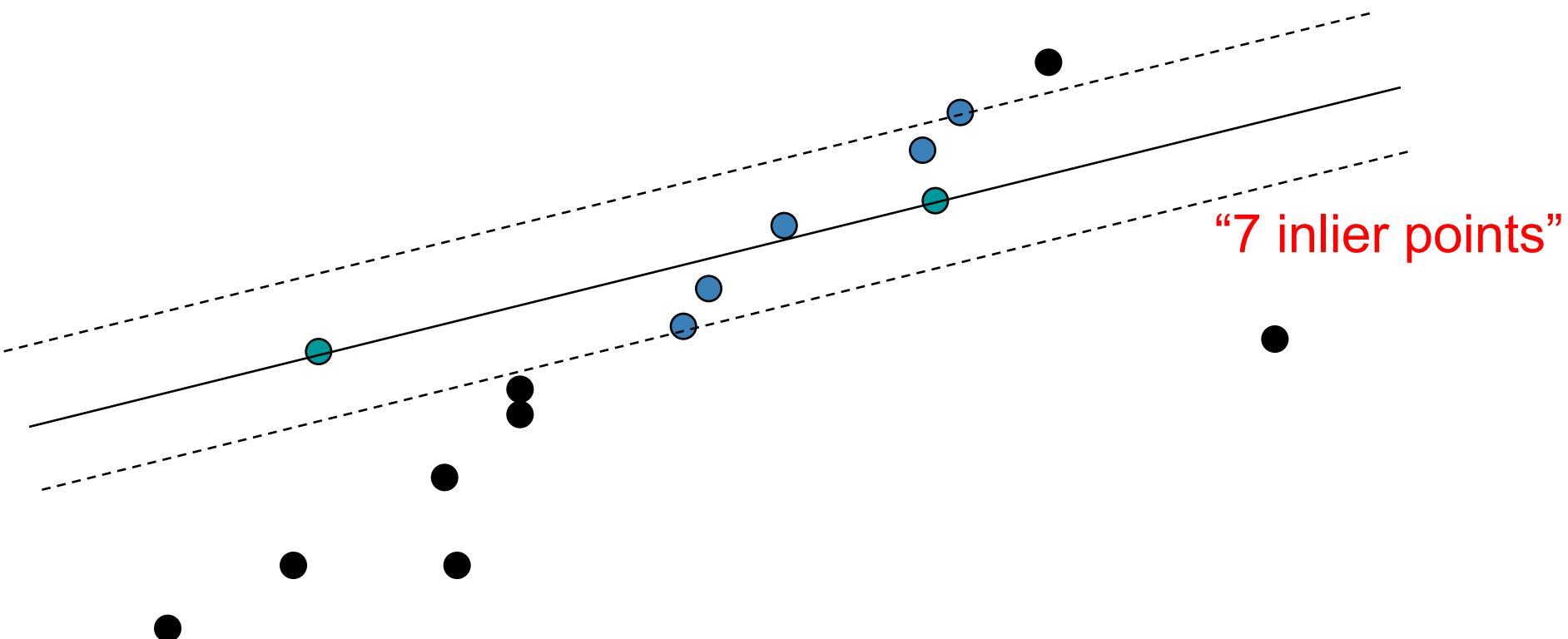
So far: The Hough transform

- So: one point (x_i, y_i) gives a line in (a, b) space.
- Another point (x_j, y_j) will give rise to another line in (a, b) -space.
- Iterate over pairs of points, to vote for buckets of intersection in (a, b) -space



So far: RANSAC

- Sample seed points, calculate line, count # of inliers, repeat



Today's agenda

- RANSAC
- Local Invariant Features
- Harris Corner Detector

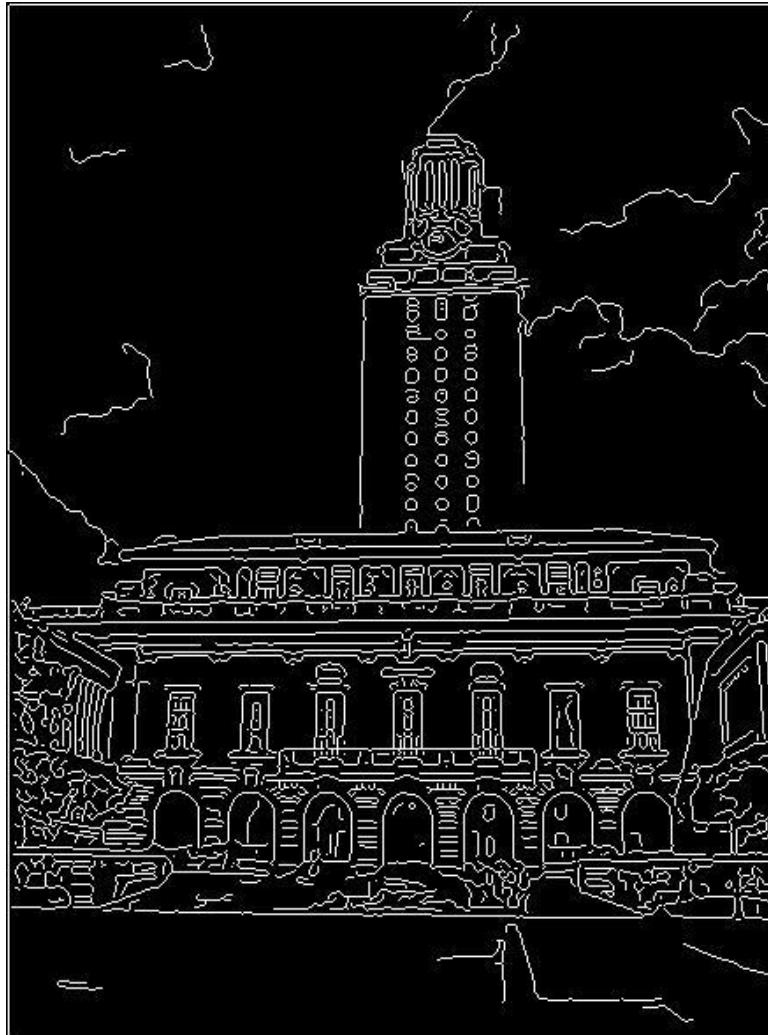
Today's agenda

- RANSAC
- Local Invariant Features
- Harris Corner Detector

Why is Hough transform inefficient?

- It's not feasible to check all pairs of points to calculate possible lines. For example, **Hough Transform algorithm runs in $O(N^2)$.**
- **Voting** is a general technique where we let the **each point vote** for all models that are compatible with it.
 - Iterate through features, cast votes for parameters.
 - Filter parameters that receive a lot of votes.
- **Problem:** Noisy points will cast votes too, *but* typically their votes should be inconsistent with the majority of “good” edge points.

Difficulty of voting for lines



- Noisy edge points, cast inconsistent votes:
 - Can we identify them without iterating over all pairs?
- Only some parts of each line detected, and some parts are missing:
 - How do we find a line that bridges missing evidence?
- Noise in measured edge points, orientations:
 - How to detect true underlying parameters?

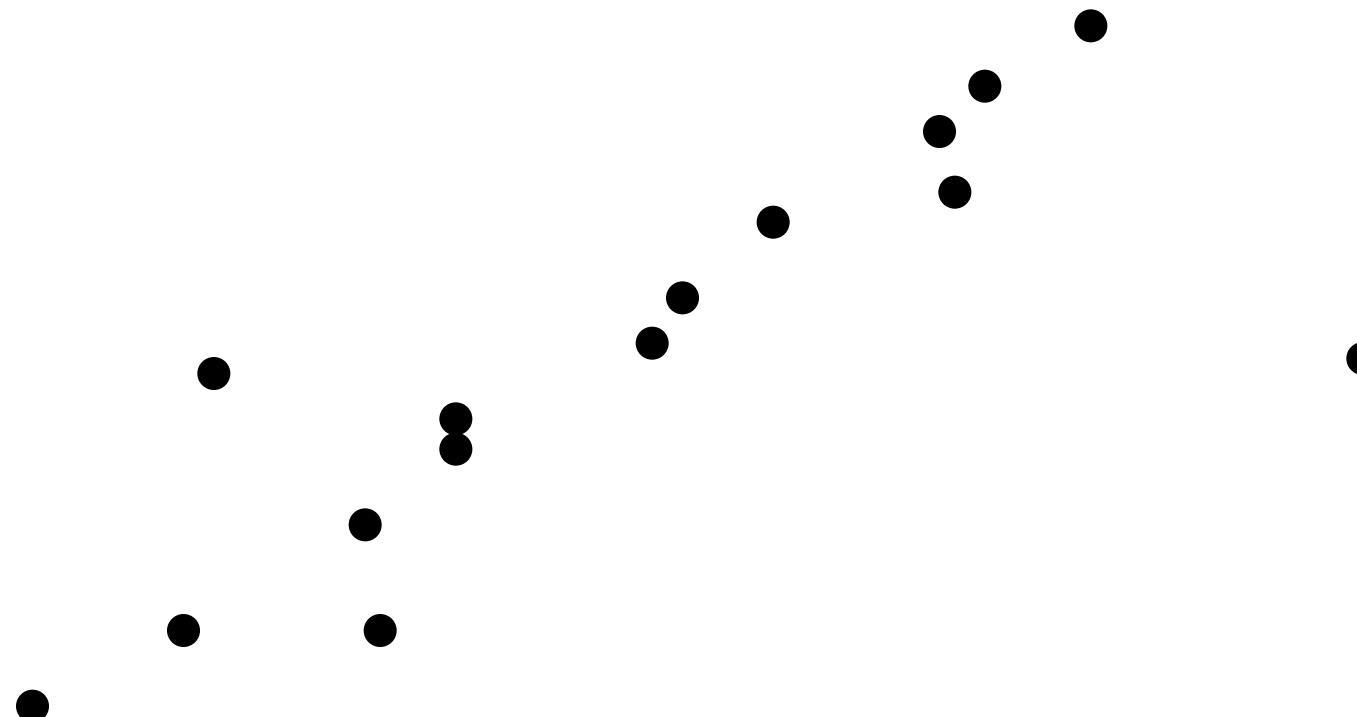


RANSAC [Fischler & Bolles 1981]

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

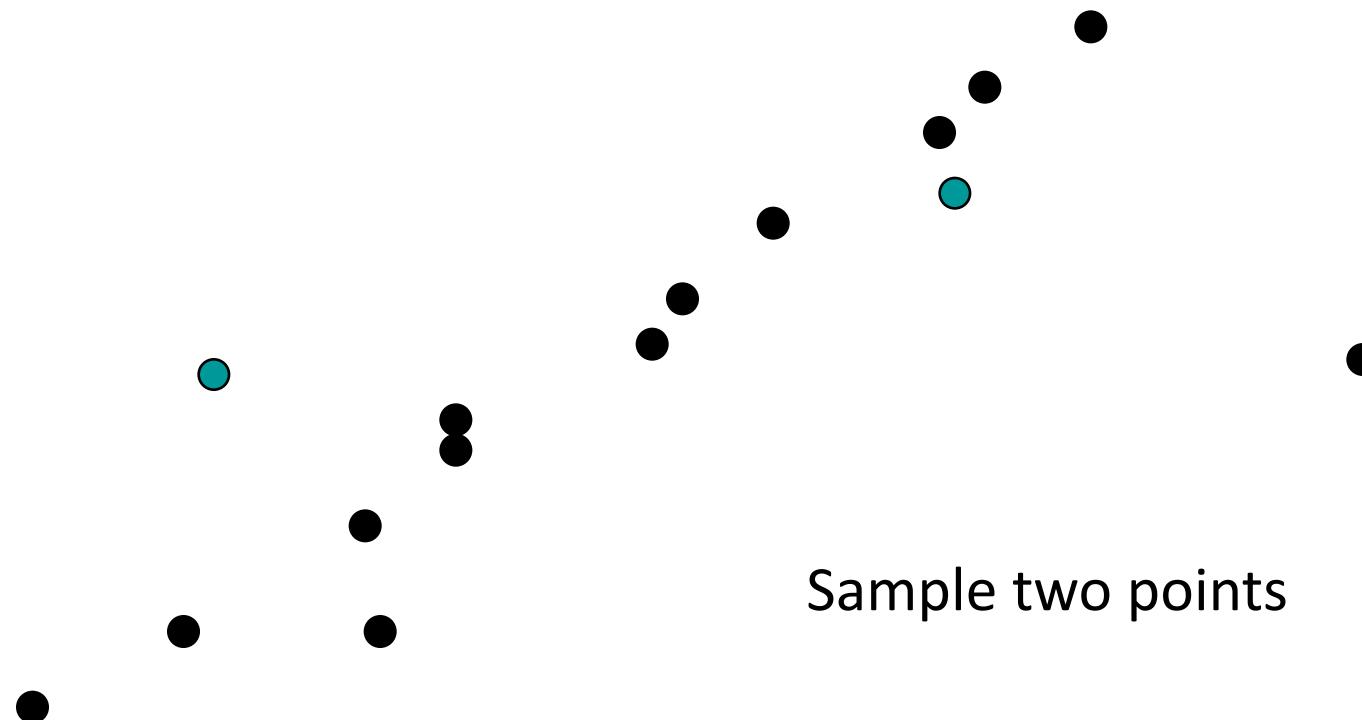
RANSAC Line Fitting Example

- Task: Estimate the best line
 - *Let's randomly select a subset of points and calculate a line*



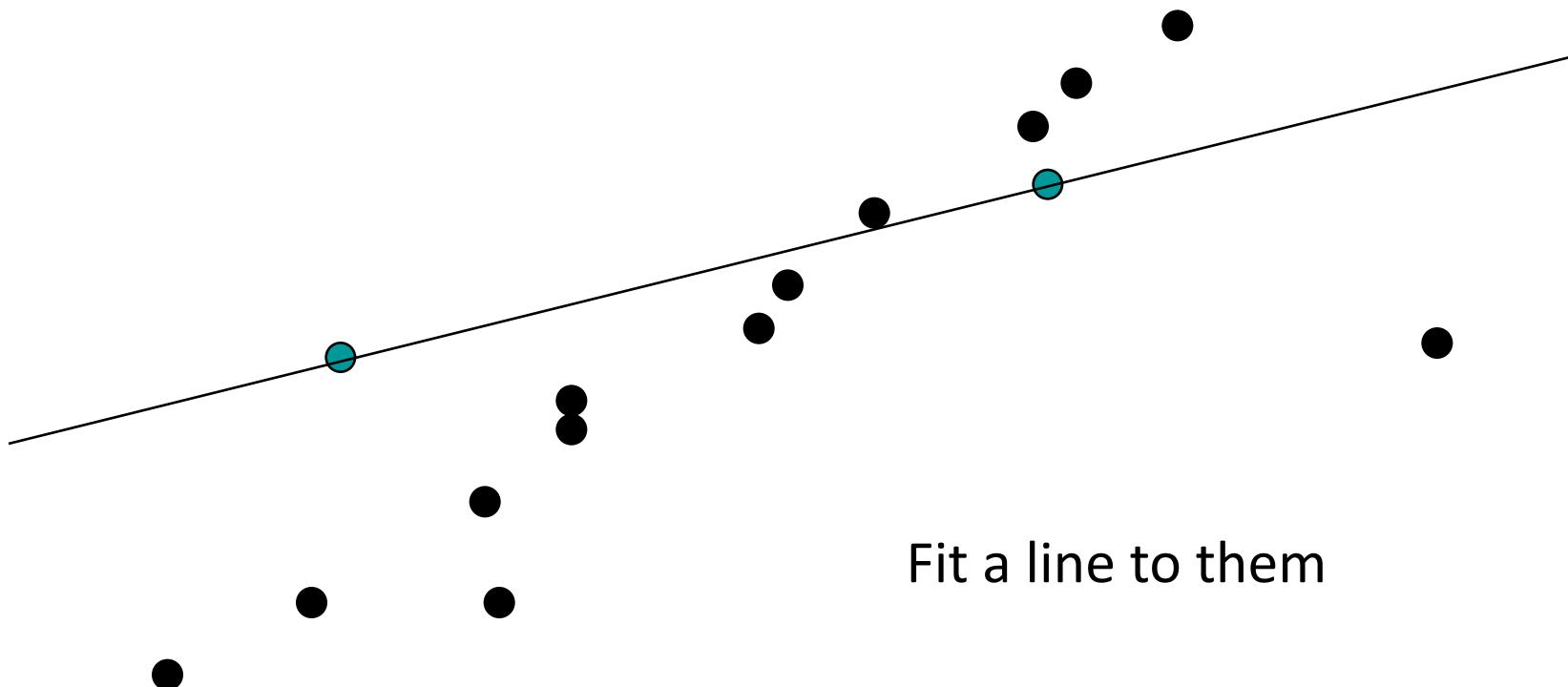
RANSAC Line Fitting Example

- Task: Estimate the best line
 - Let's select only 2 points as an example



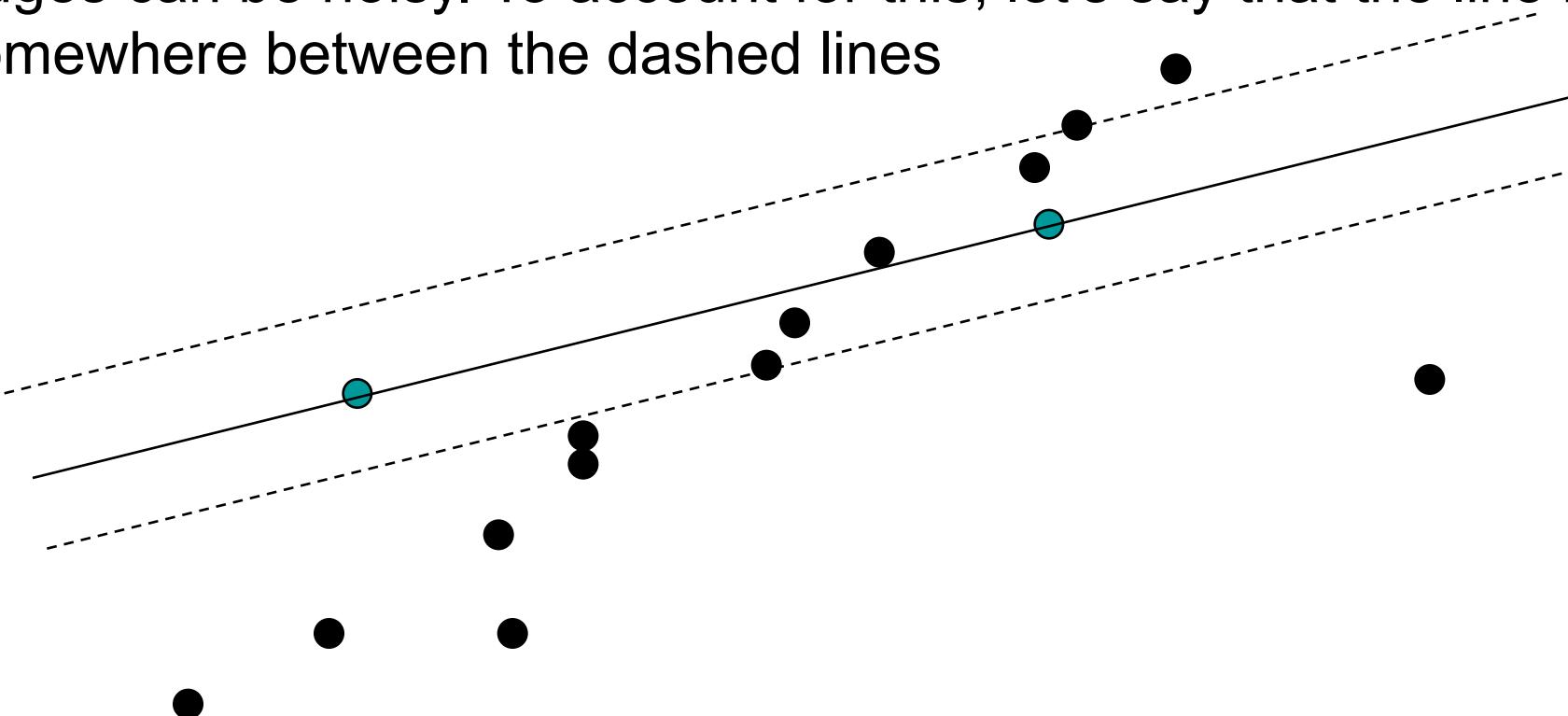
RANSAC Line Fitting Example

- Task: Estimate the best line
 - Calculate the line parameters



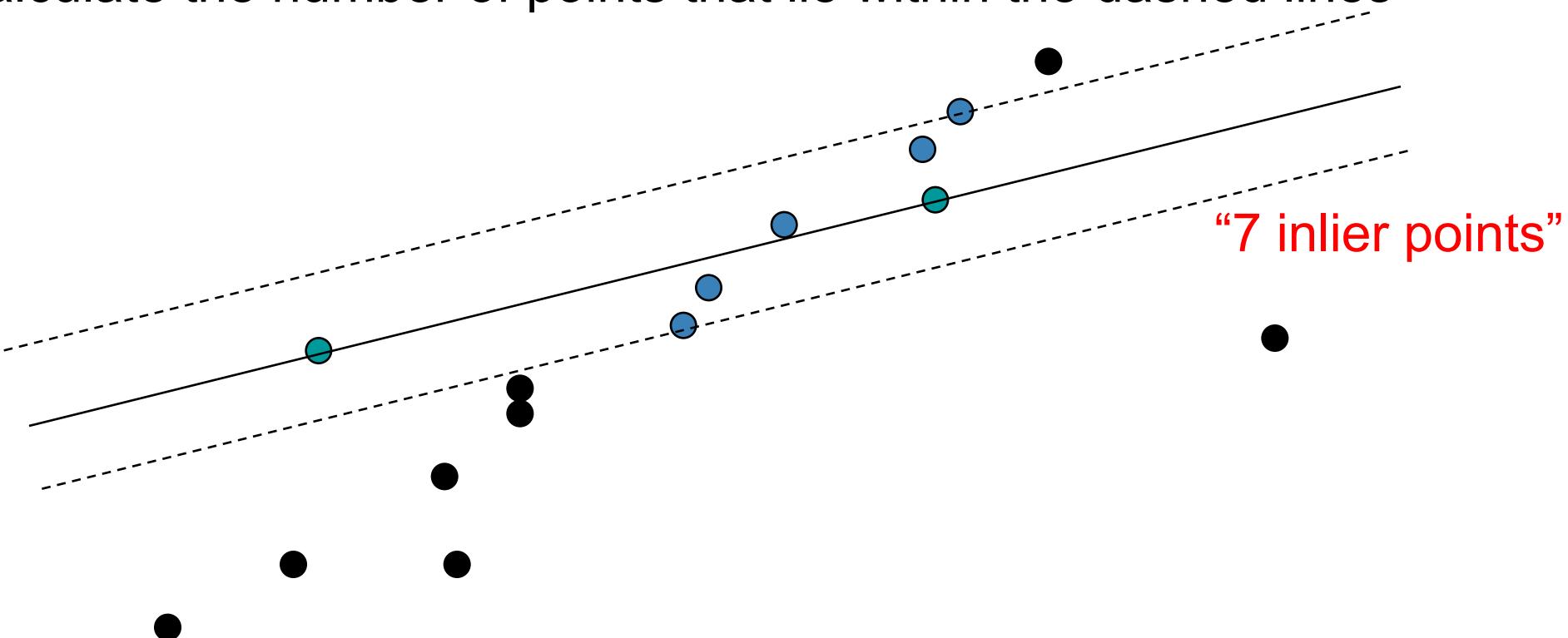
RANSAC Line Fitting Example

- Task: Estimate the best line
 - Edges can be noisy. To account for this, let's say that the line is somewhere between the dashed lines



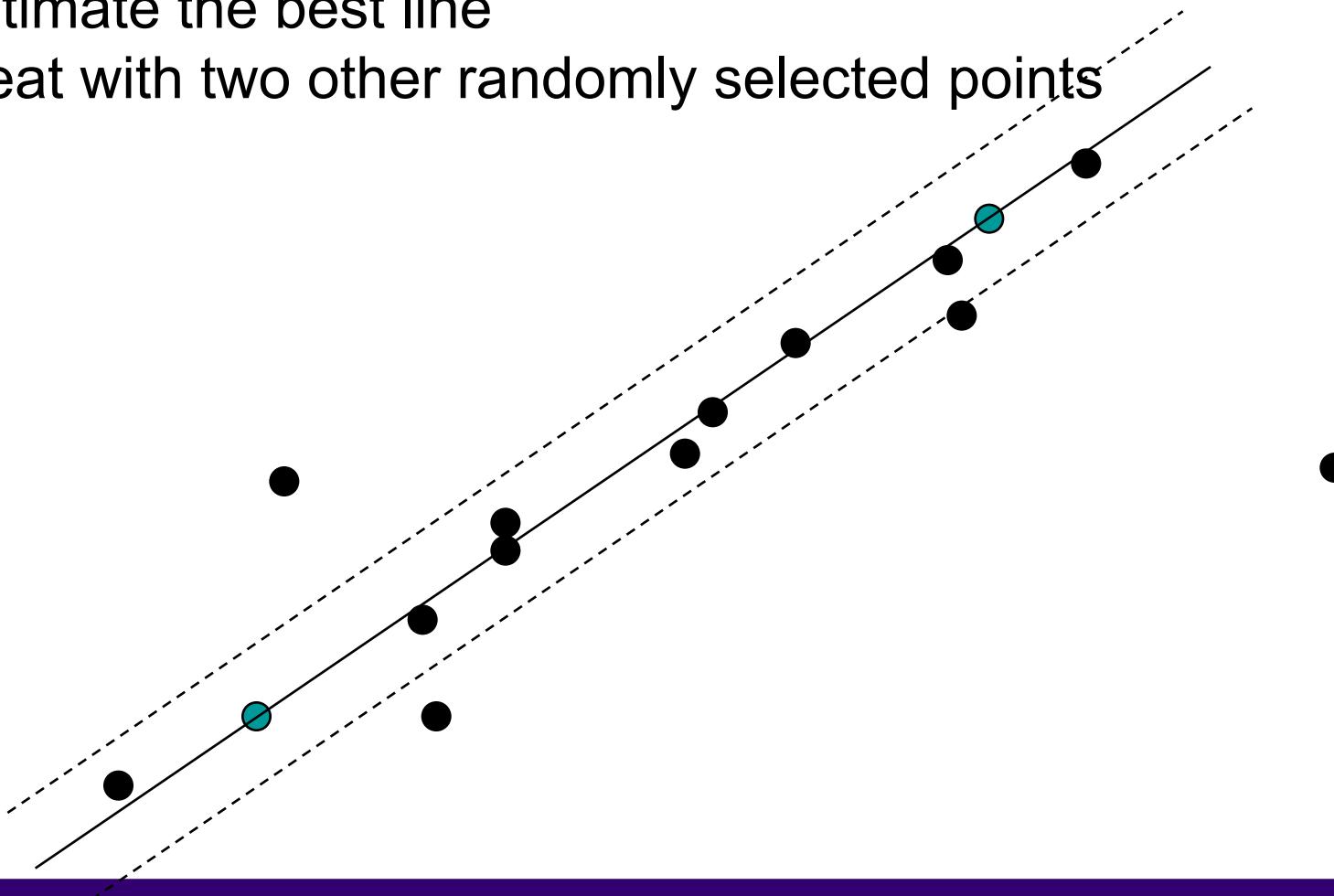
RANSAC Line Fitting Example

- Task: Estimate the best line
 - Calculate the number of points that lie within the dashed lines



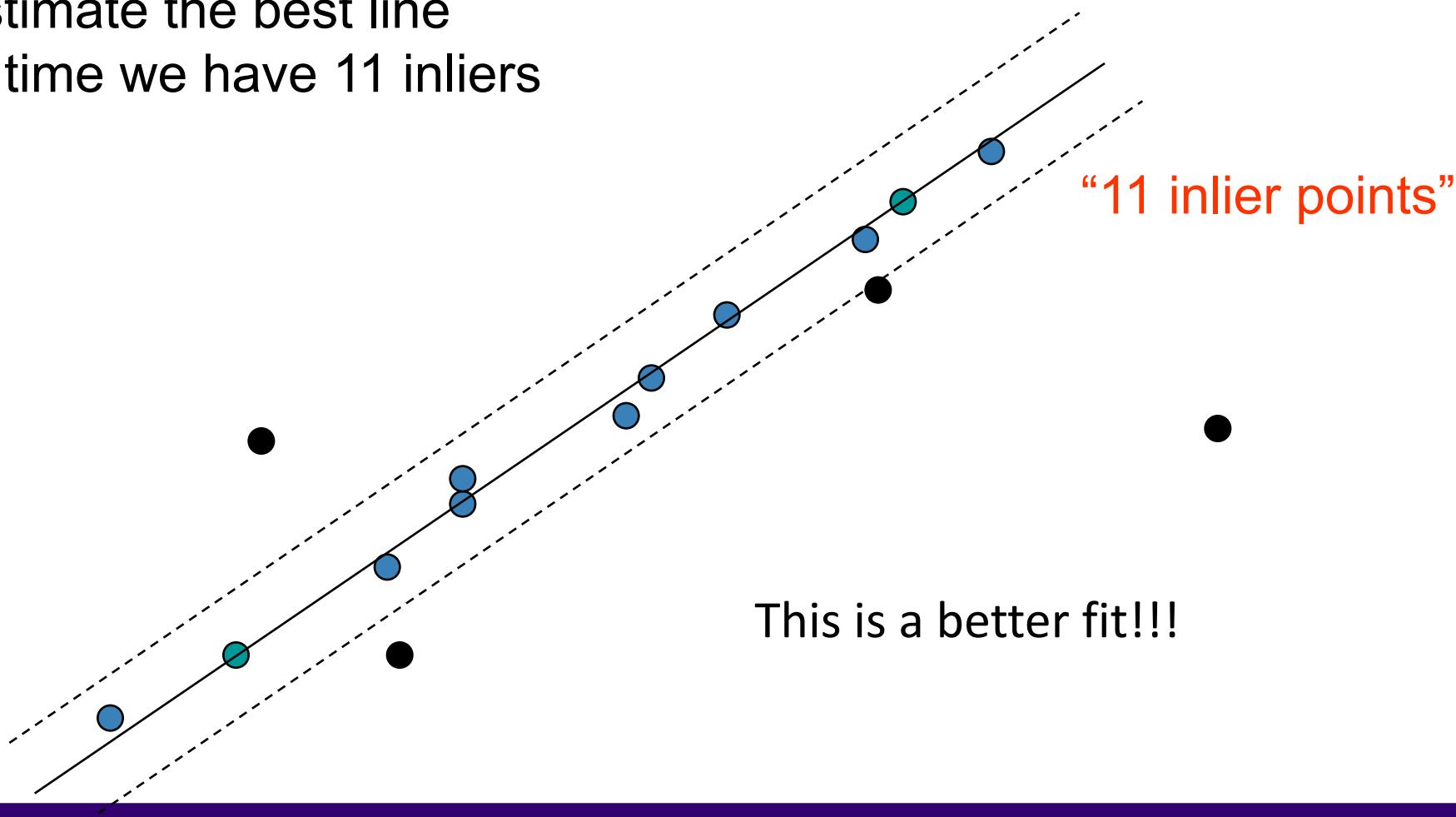
RANSAC Line Fitting Example

- Task: Estimate the best line
 - Repeat with two other randomly selected points



RANSAC Line Fitting Example

- Task: Estimate the best line
 - This time we have 11 inliers



The RANSAC algorithm [Fischler & Bolles 1981]

RANSAC loop:

Repeat for k iterations:

1. Randomly select a **seed** subset of points on which to perform a model estimate (e.g., a group of edge points)

The RANSAC algorithm [Fischler & Bolles 1981]

RANSAC loop:

Repeat for k iterations:

1. Randomly select a **seed** subset of points on which to perform a model estimate (e.g., a group of edge points)
2. Compute parameters from seed group

The RANSAC algorithm [Fischler & Bolles 1981]

RANSAC loop:

Repeat for k iterations:

1. Randomly select a **seed** subset of points on which to perform a model estimate (e.g., a group of edge points)
2. Compute parameters from seed group
3. Find **inliers** for these parameters

The RANSAC algorithm [Fischler & Bolles 1981]

RANSAC loop:

Repeat for k iterations:

1. Randomly select a **seed** subset of points on which to perform a model estimate (e.g., a group of edge points)
2. Compute parameters from seed group
3. Find **inliers** for these parameters
4. If the number of inliers is larger than the best so far, save these parameters and the inliers

The RANSAC algorithm [Fischler & Bolles 1981]

RANSAC loop:

Repeat for k iterations:

1. Randomly select a **seed** subset of points on which to perform a model estimate (e.g., a group of edge points)
2. Compute parameters from seed group
3. Find **inliers** for these parameters
4. If the number of inliers is larger than the best so far, save these parameters and the inliers

If number of inliers in the best line is $< m$, return no line

The RANSAC algorithm [Fischler & Bolles 1981]

RANSAC loop:

Repeat for k iterations:

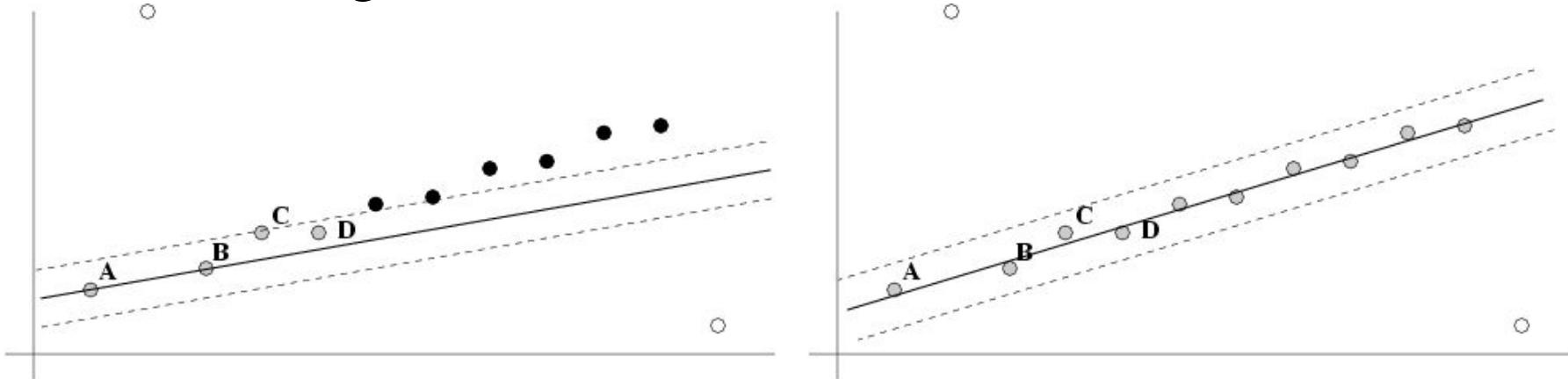
1. Randomly select a **seed** subset of points on which to perform a model estimate (e.g., a group of edge points)
2. Compute parameters from seed group
3. Find **inliers** for these parameters
4. If the number of inliers is larger than the best so far, save these parameters and the inliers

If number of inliers in the best line is $< m$, return no line

Else re-calculate the final parameters with all the inliers

Final step: Refining the parameters

- The best parameters were computed using a seed set of n points.
- We use these points to find the inliers.
- We can improve the parameters by estimating over all inliers (e.g. with standard least-squares minimization).
- But this may change the inliers, so repeat this last step until there is no change in inliers.



The RANSAC algorithm [Fischler & Bolles 1981]

RANSAC loop:

Repeat for k iterations:

1. Randomly select a **seed** subset of points on which to perform a model estimate (e.g., a group of edge points)
2. Compute parameters from seed group
3. Find **inliers** for these parameters
4. If the number of inliers is larger than the best so far, save these parameters and the inliers

If number of inliers in the best line is $< m$, return no line

Else re-calculate the final parameters with all the inliers

The hyperparameters

1. How many points to sample in the seed set?
 - a. We used 2 in the example above

The hyperparameters

1. How many points to sample in the seed set?
 - a. We used 2 in the example above
2. How many times should we repeat?
 - a. More repetitions increase computation but increase chances of finding best line

The hyperparameters

1. How many points to sample in the seed set?
 - a. We used 2 in the example above
2. How many times should we repeat?
 - a. More repetitions increase computation but increase chances of finding best line
3. The threshold for the dashed lines
 - a. Larger the gap between dashed lines, the more false positive inliers
 - b. Smaller the gap, the more false negatives outliers

The hyperparameters

1. How many points to sample in the seed set?
 - a. We used 2 in the example above
2. How many times should we repeat?
 - a. More repetitions increase computation but increase chances of finding best line
3. The threshold for the dashed lines
 - a. Larger the gap between dashed lines, the more false positive inliers
 - b. Smaller the gap, the more false negatives outliers
4. The minimum number of inliers to confidently claim there is a line
 - a. Smaller the number, the more false negative lines
 - b. Larger the number, the fewer lines we will find

RANSAC: Computed k ($p=0.99$)

Sample size n	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	

RANSAC: How many iterations “ k ”?

- 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 a single sample of n points fails: $1 - w^n$
- Prob. that all k samples fail is: $(1 - w^n)^k$
- Prob. that at least one of the k samples is correct: $1 - (1 - w^n)^k$

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

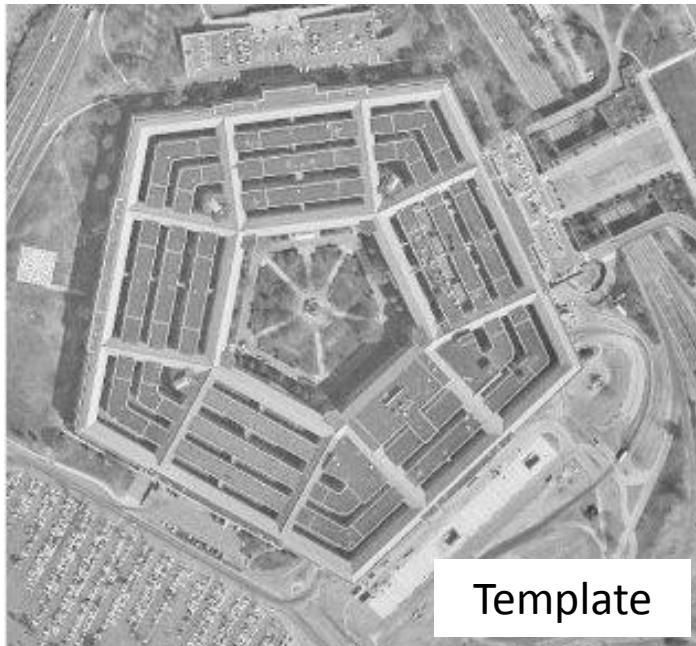
RANSAC: Pros and Cons

- **Pros:**
 - General method suited for a wide range of parameter 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

Today's agenda

- RANSAC
- Local Invariant Features
- Harris Corner Detector

Image matching: a challenging problem

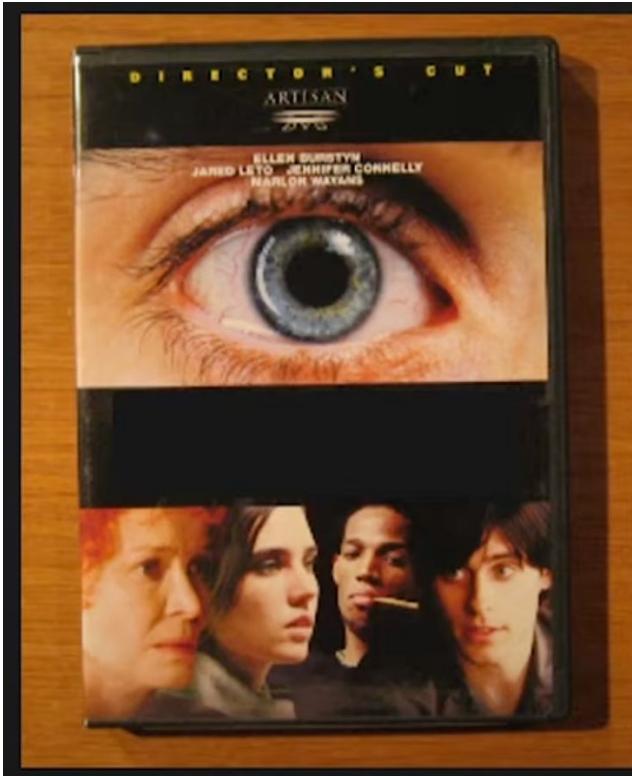


Q1. Will cross-correlation work?

Q2. Can we use match the lines?



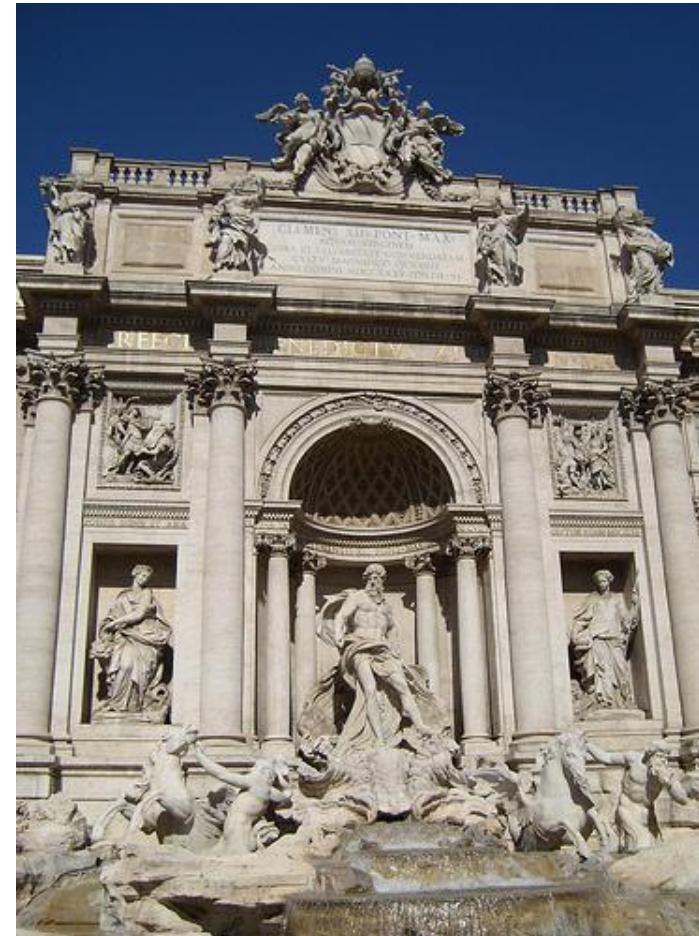
Q. How would you build a system that can detect this movie in the pile?



Challenge: Perspective / viewpoint changes



by Diva Sian



by swashford

Challenge: partial observability

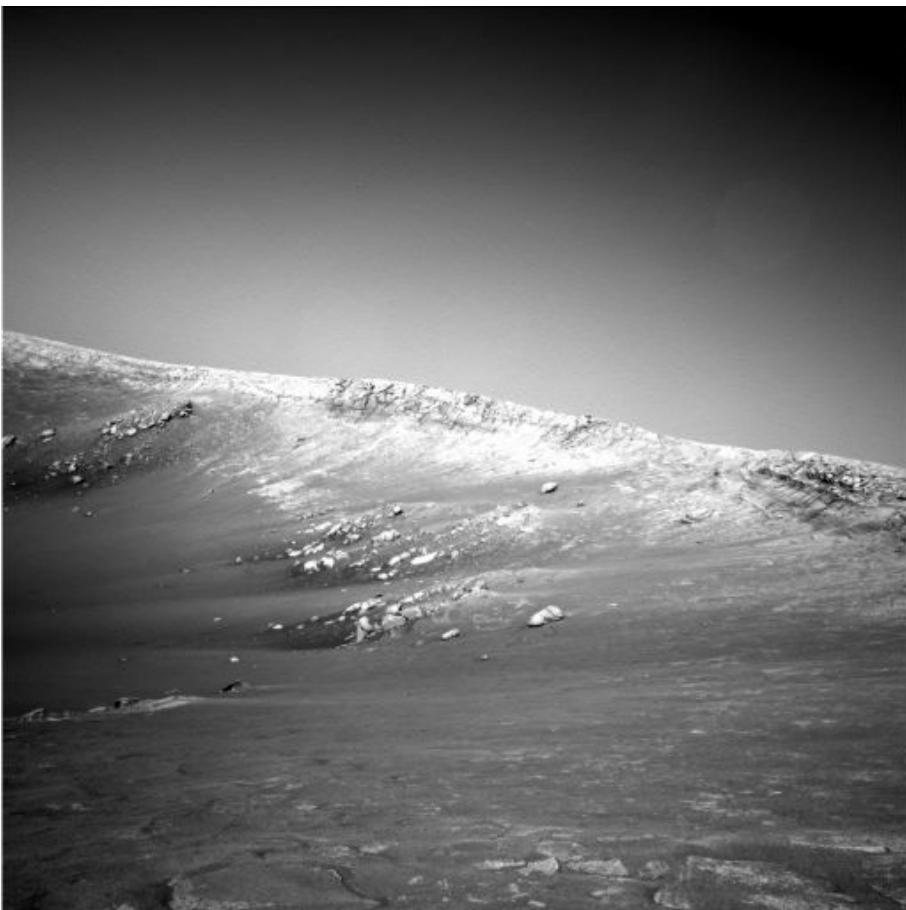


by Diva Sian



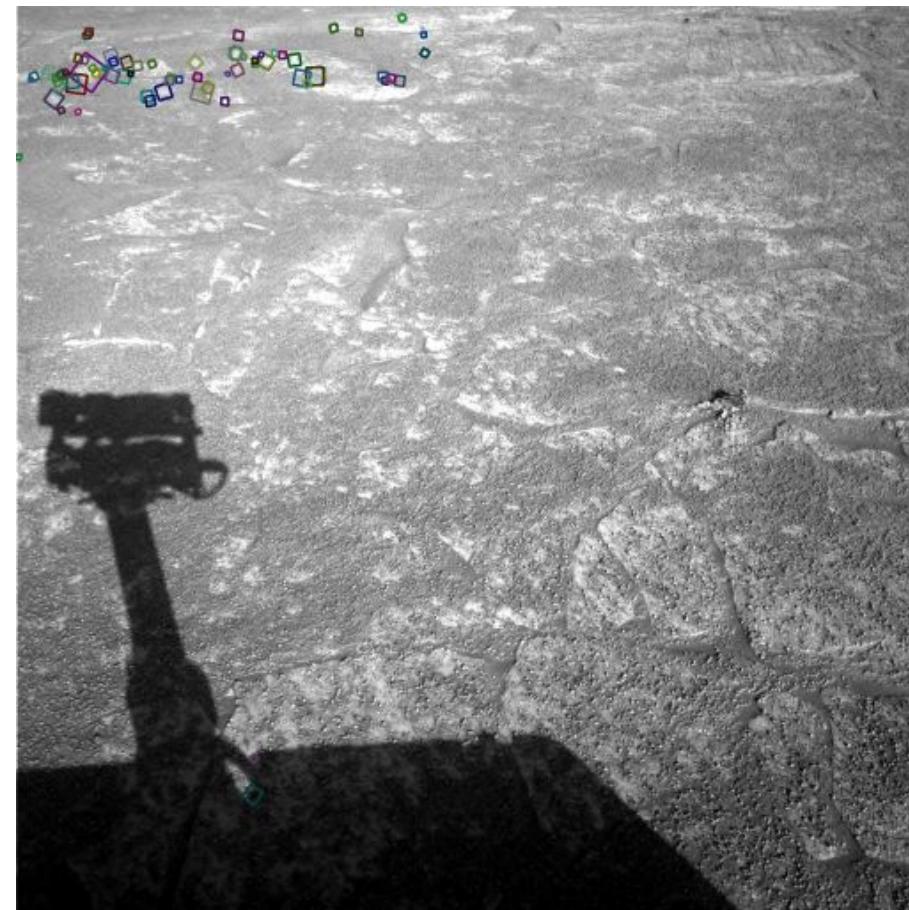
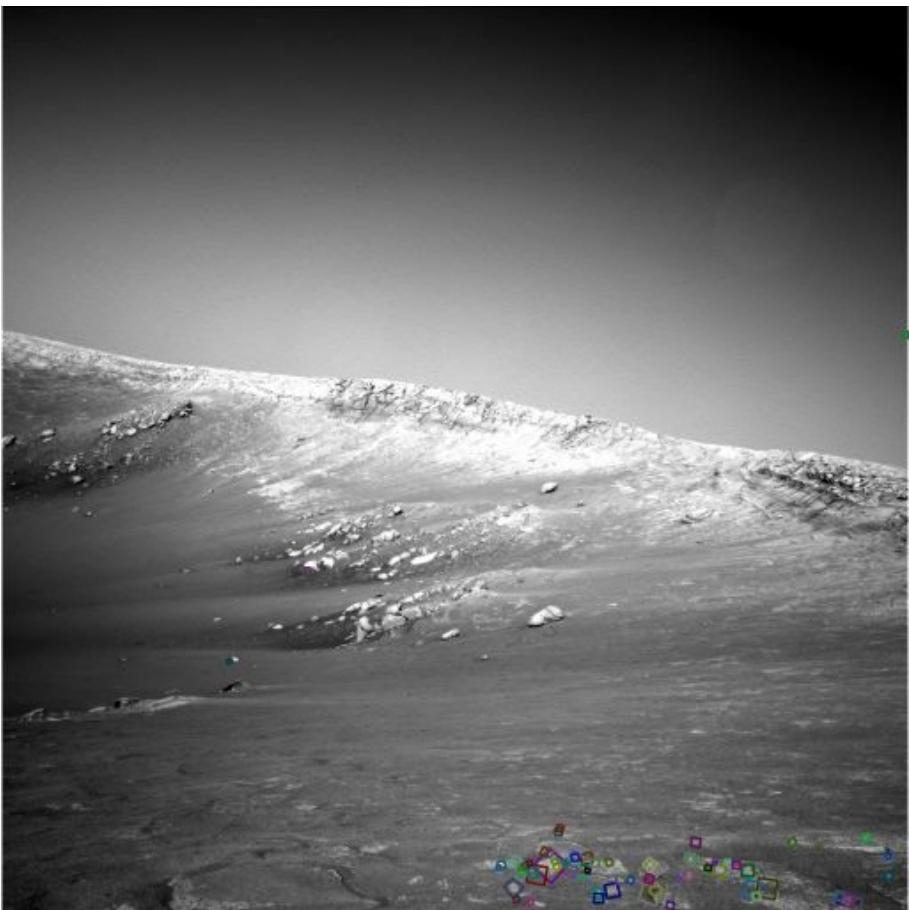
by scgbt

Challenge even for us



NASA Mars Rover images

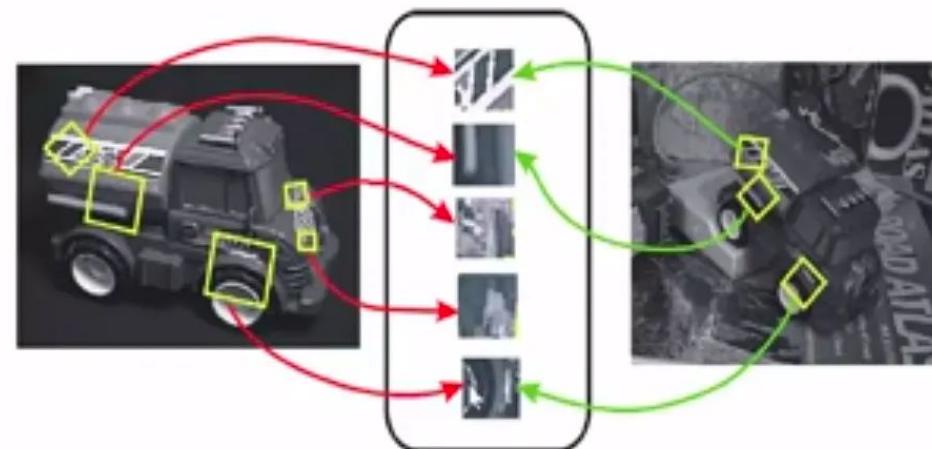
Answer Below (Look for tiny colored squares)



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

Intuition behind how to match images

- Find matching patches
- Check to make sure enough patches



Intuition behind how to match images

- Find matching patches
- Check to make sure enough patches

What do we need?

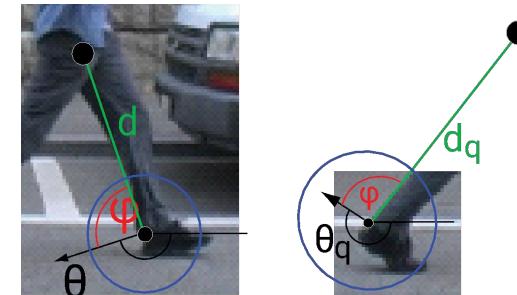
- We need to identify patches
- We need to learn to a way to describe each patch
- We need an algorithm to match the description between two patches

Motivation for using local features

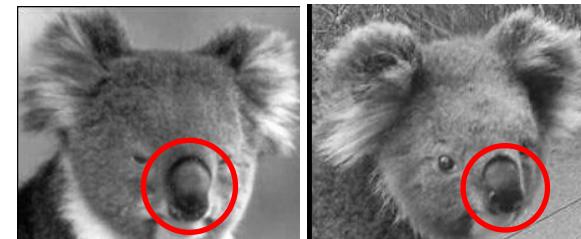
- Global representations have major limitations
- Instead, describe and match only local regions
- Increased robustness to
 - Occlusions



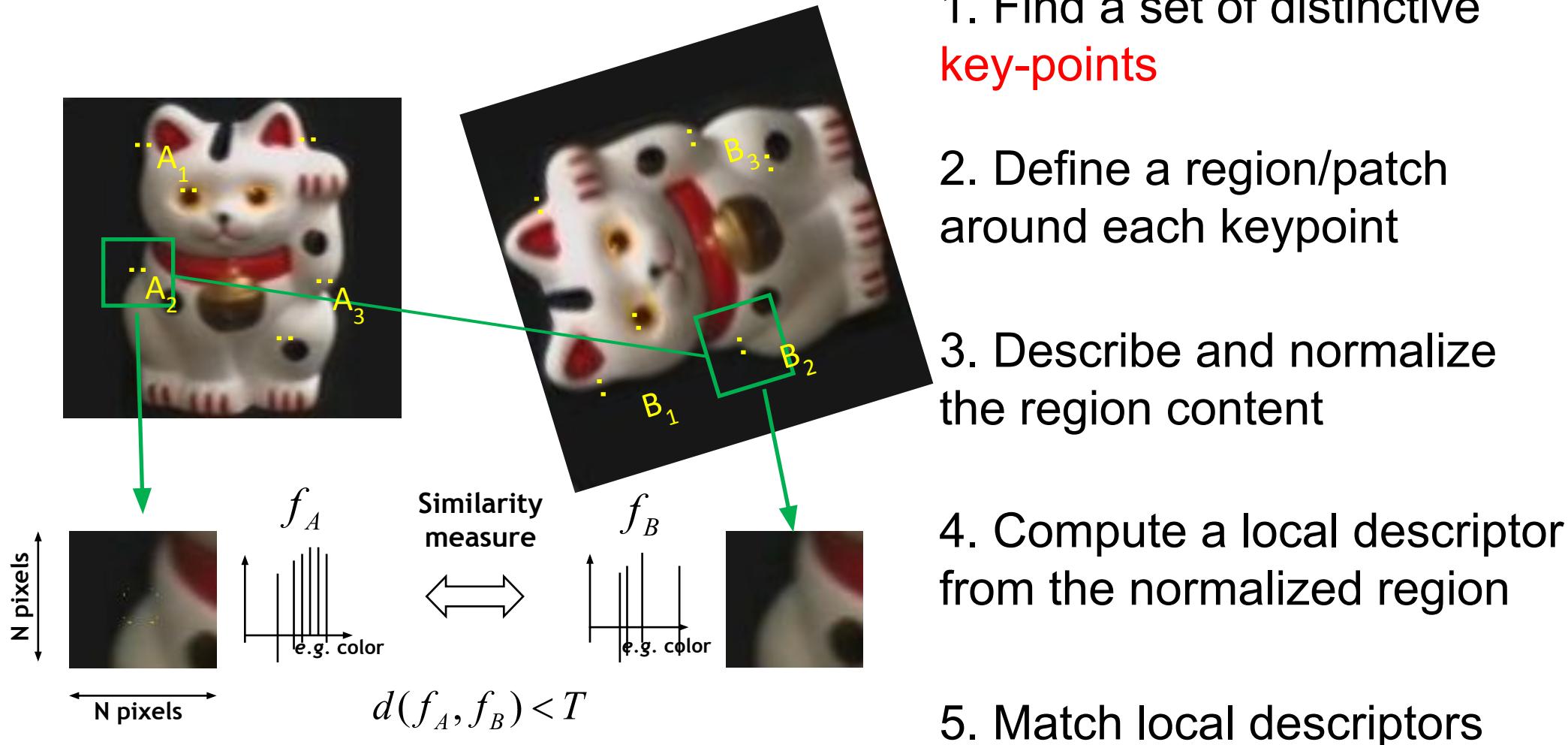
- Articulation



- Intra-category variations



General Approach



Common Requirements

- Problem 1: How should we choose the key-points?
 - We want to detect the same points **independently** in both images

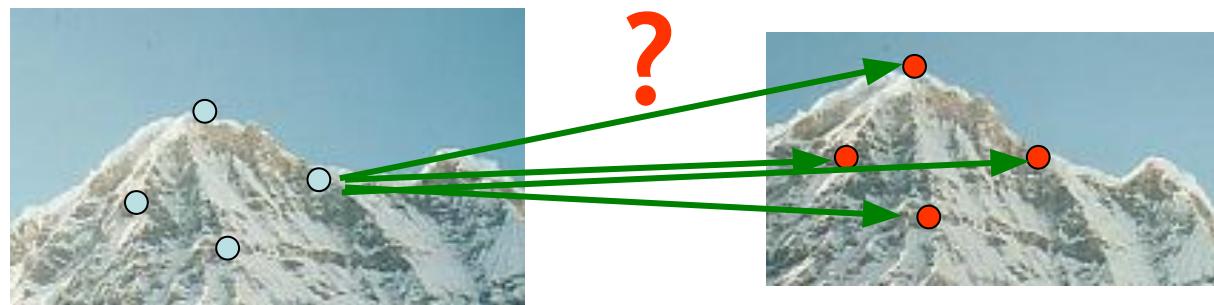


No chance to match if the key-points
aren't the same

We need a repeatable detector!

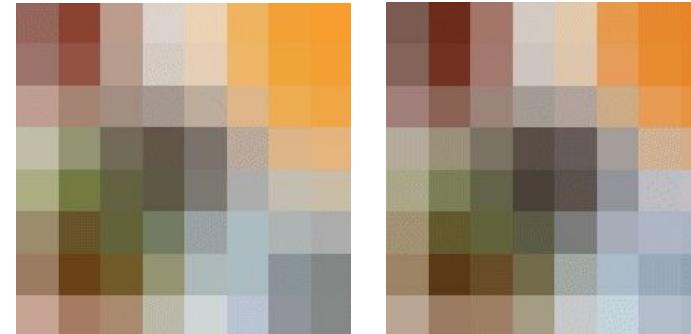
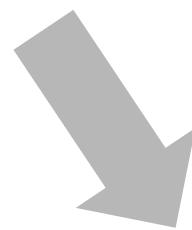
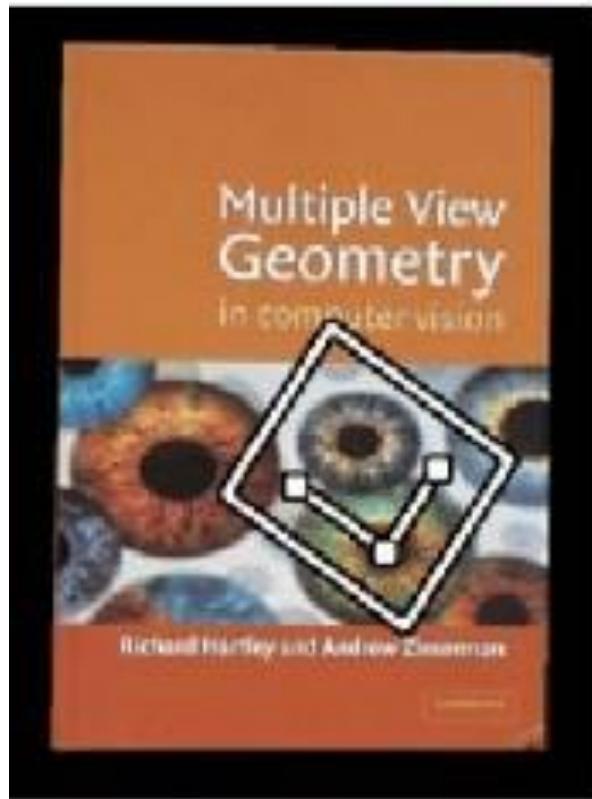
Common Requirements

- Problem 1: How should we choose the key-points?
 - Detect the same point **independently** in both images
- Problem 2: How should we describe each patch?
 - For each point correctly recognize the corresponding one

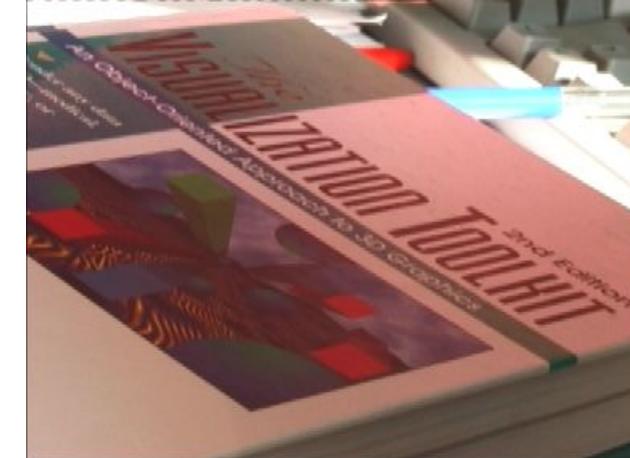
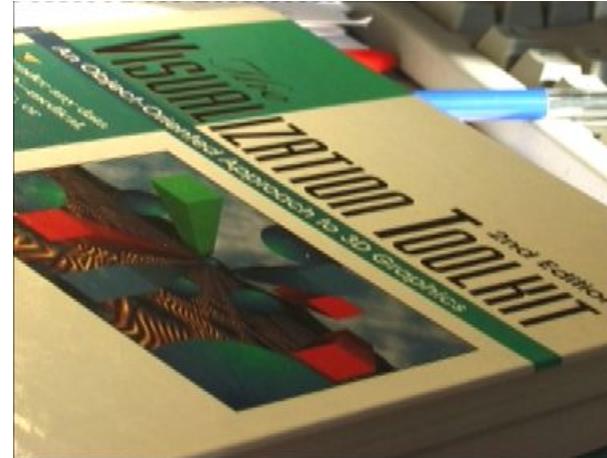
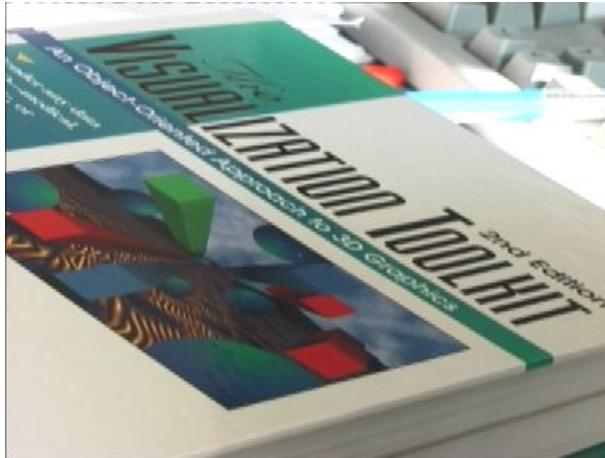


We need a reliable and distinctive descriptor!

Descriptions should be invariant to rotation and translation



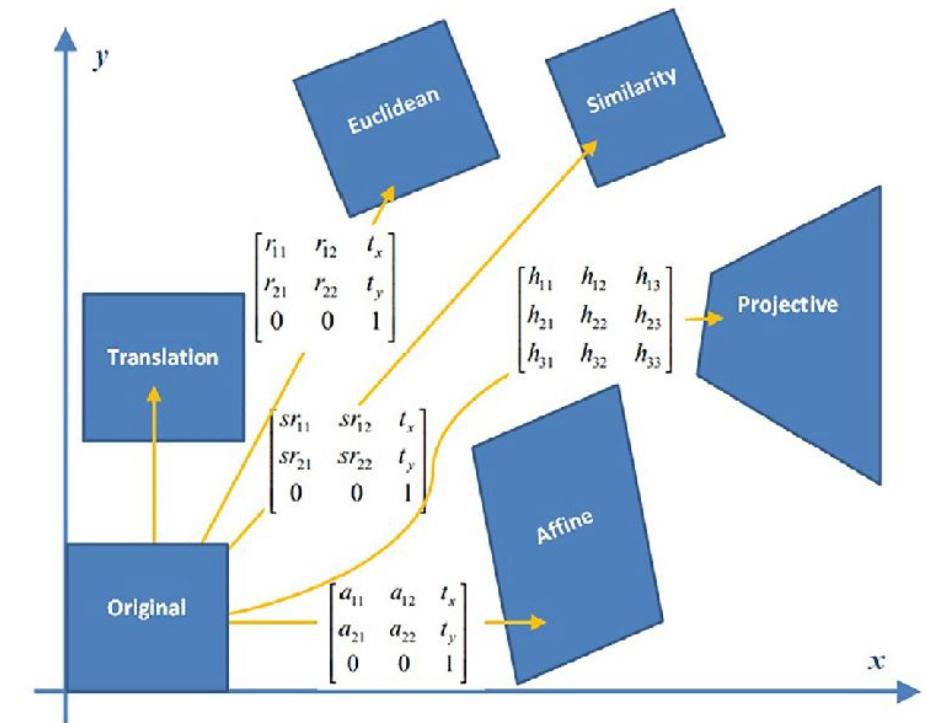
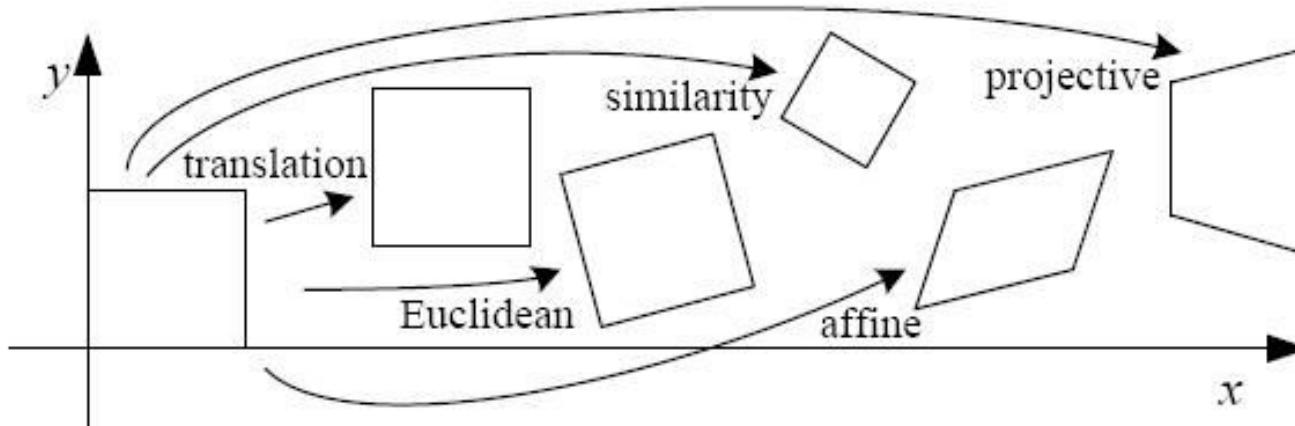
Descriptions should be invariant to photometric transformations



- Often modeled as a linear transformation:
 - Scaling + Offset

Slide credit: Tinne Tuytelaars

Levels of geometric transformations



Requirements for Local Features

- Patch selection needs to be **repeatable** and **accurate**
 - **Invariant** to translation, rotation, scale changes
 - **Robust** to out-of-plane (\approx affine) transformations
 - **Robust** to lighting variations, noise, blur, quantization
- **Locality**: Features are local, therefore robust to occlusion and clutter.
- **Quantity**: We need a sufficient number of regions to cover the object.
- **Distinctiveness**: The regions should contain “unique” structure.
- **Efficiency**: Close to real-time performance.

Many existing feature detectors available

- Hessian & **Harris** [Beaudet '78], [Harris '88]
- **Laplacian, DoG** [Lindeberg '98], [Lowe '99]
- Harris-/Hessian-Laplace [Mikolajczyk & Schmid '01]
- Harris-/Hessian-Affine [Mikolajczyk & Schmid '04]
- EBR and IBR [Tuytelaars & Van Gool '04]
- MSER [Matas '02]
- Salient Regions [Kadir & Brady '01]
- **Neural networks** [Krizhevsky '12]
- *Those detectors have become a basic building block for many applications in Computer Vision.*

Today's agenda

- Local Invariant Features
- Harris Corner Detector

Keypoint Localization



- **Goals:**
 - Repeatable detection
 - Precise localization
 - Interesting content

intuition \Rightarrow *Look for 2D signal changes (LSI systems strike again)*

What are good patches?

Q. Is this a good patch for
image matching?



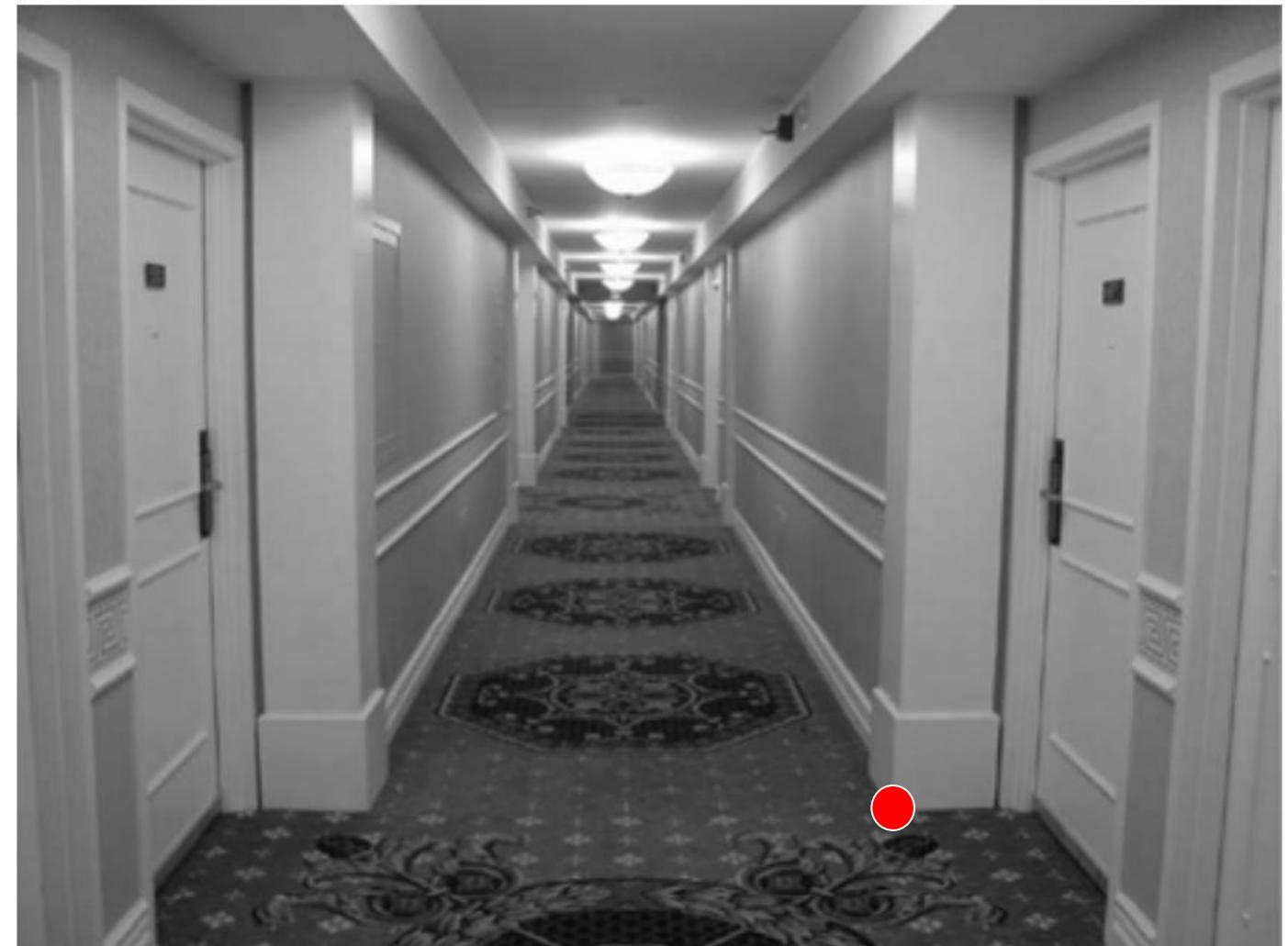
What are good patches?

Q. What about this one?

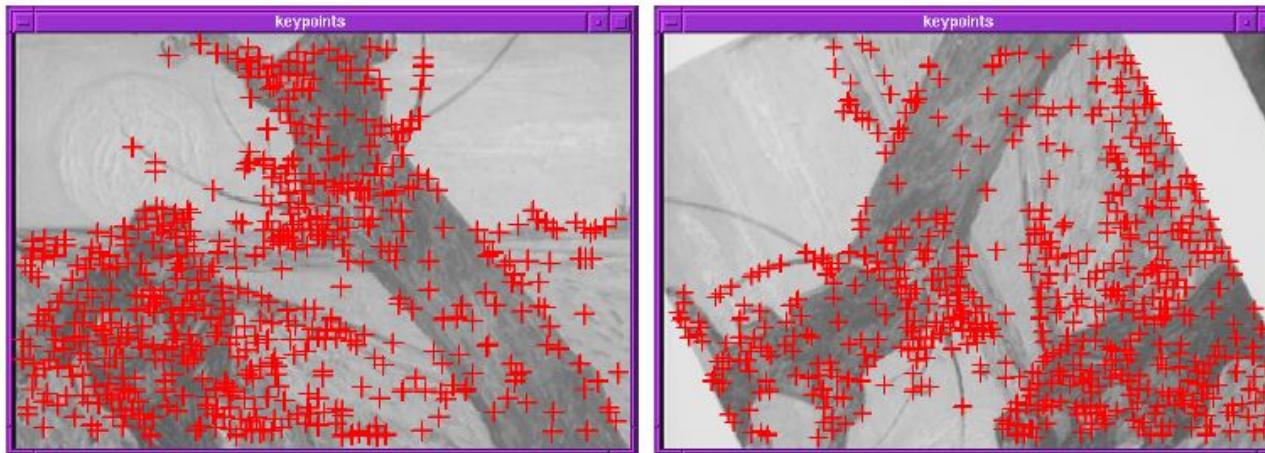


What are good patches?

Q. Let's try another one?



Finding Corners



How do we find corners using LSI systems?

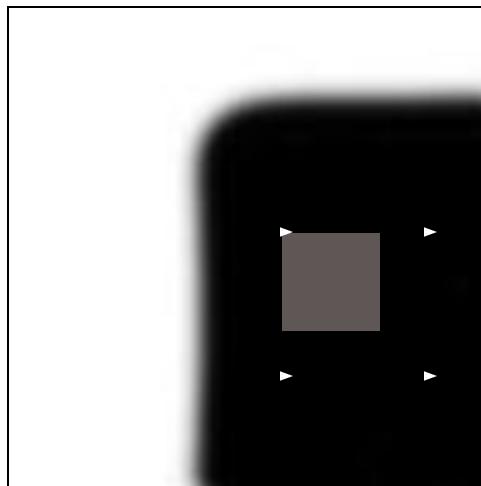
- The image gradient around a corner has two or more dominant directions

Corners are **repeatable** and **distinctive**

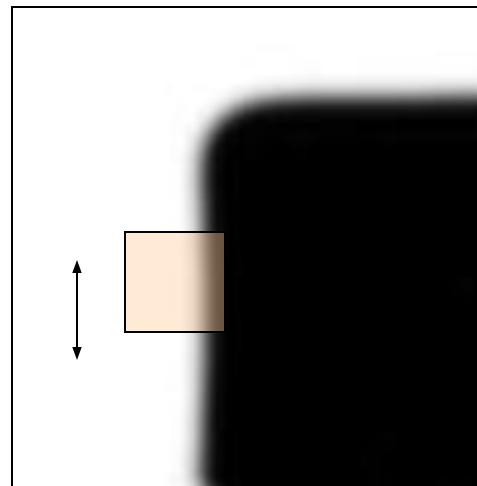
C.Harris and M.Stephens. "A Combined Corner and Edge Detector."
Proceedings of the 4th Alvey Vision Conference, 1988.

Corners are distinctive key-points

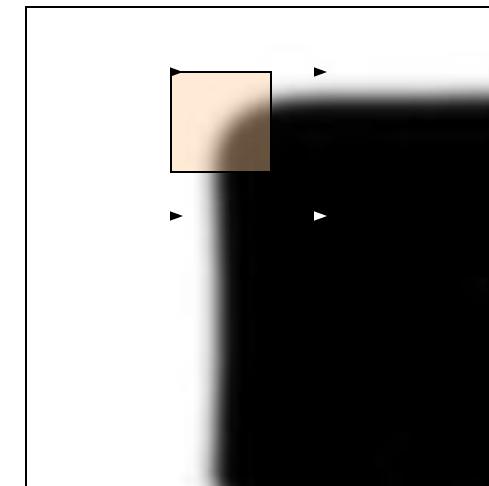
- We should easily recognize the corner point by looking through a small image patch (*locality*)
- Shifting the window in *any direction* should give a *large change* in intensity (*good localization*)



“flat” region:
no change in
all directions



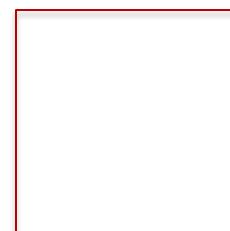
“edge”:
no change along
the edge direction



“corner”:
significant change
in all directions

Slide credit: Alyosha Efros

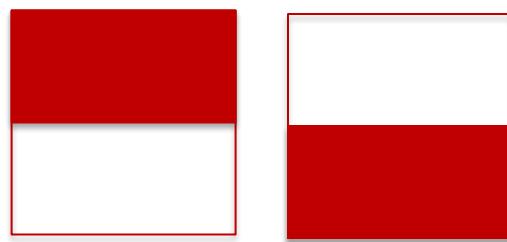
Flat patches have small image gradients



$$\sum I_x^2 \rightarrow \text{Small}$$
$$\sum I_y^2 \rightarrow \text{Small}$$

Flat

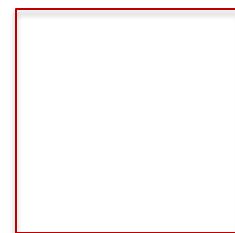
Edges have high gradient in one direction



$$\sum I_x^2 \rightarrow \text{Small}$$

$$\sum I_y^2 \rightarrow \text{Large}$$

Edge

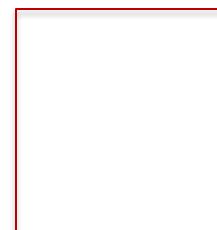
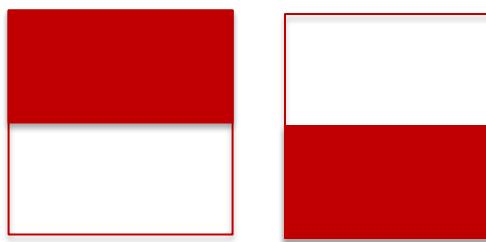
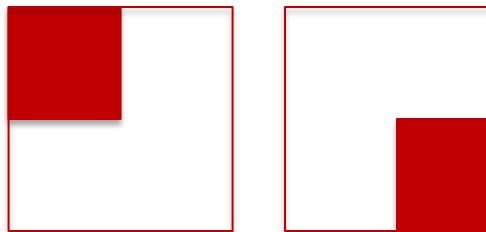
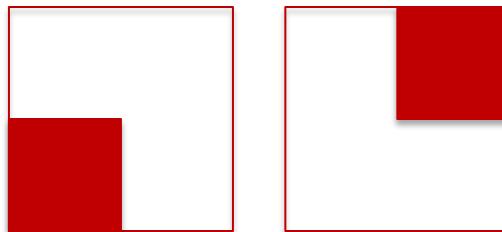


$$\sum I_x^2 \rightarrow \text{Small}$$

$$\sum I_y^2 \rightarrow \text{Small}$$

Flat

Corners versus edges



$$\sum I_x^2 \rightarrow \text{Large}$$

$$\sum I_y^2 \rightarrow \text{Large}$$

$$\sum I_x^2 \rightarrow \text{Small}$$

$$\sum I_y^2 \rightarrow \text{Large}$$

$$\sum I_x^2 \rightarrow \text{Small}$$

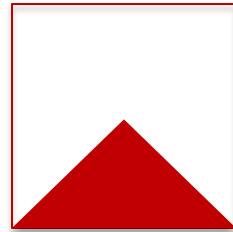
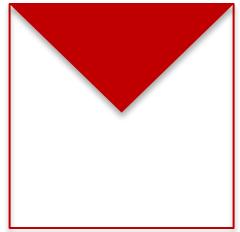
$$\sum I_y^2 \rightarrow \text{Small}$$

Corner

Edge

Flat

Generalizing to corners in any direction



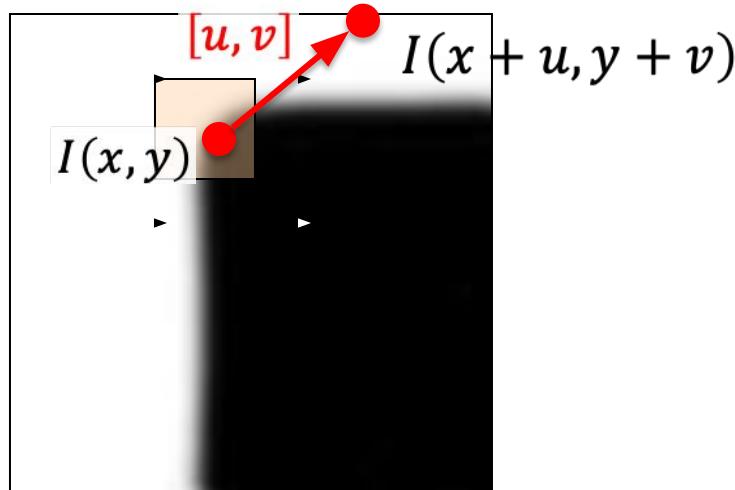
$$\sum I_x^2 \rightarrow ??$$

$$\sum I_y^2 \rightarrow ??$$

Corner

Harris Detector Formulation

- Find patches that result in large change of pixel values when shifted in *any direction*.
- When we shift by $[u, v]$, the intensity change at the center pixel is:



“corner”:
significant change
in all directions

- Measure change as intensity difference:
$$(I(x + u, y + v) - I(x, y))$$
- That's for a single point, but we have to accumulate over the patch or “small window” around that point...

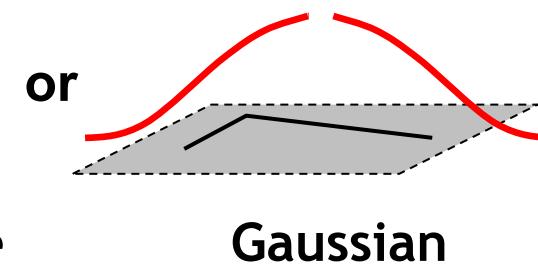
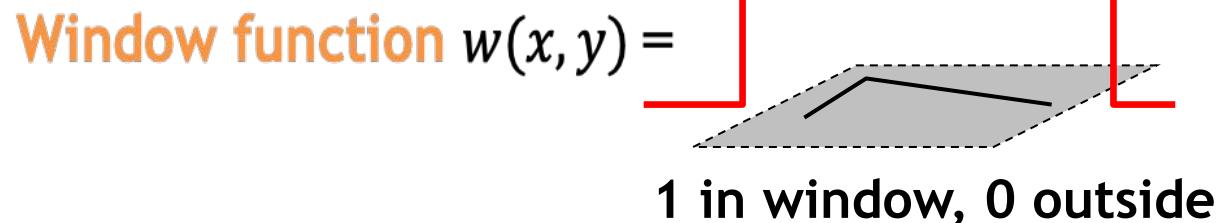
Harris Detector Formulation

- When we shift by $[u, v]$, the change in intensity for the “small window” is:

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

Diagram annotations:

- Sum over window**: Points to the summation symbol (\sum) in the equation.
- Window function**: Points to the term $w(x, y)$.
- Shifted intensity**: Points to the term $I(x + u, y + v)$.
- Intensity**: Points to the term $I(x, y)$.
- Intensity change**: Points to the squared difference term $[I(x + u, y + v) - I(x, y)]^2$.



Change in intensity function

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

We can rewrite the shifted intensity using Taylor's expansion:

$$I(x + u, y + v) \approx I(x, y) + I_x u + I_y v$$

Substituting it back into $E(u, v)$:

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

Re-writing E:

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

Re-writing E:

$$\begin{aligned}E(u, v) &= \sum_{x,y} w(x, y) [I_x u + I_y v]^2 \\&= \sum_{x,y} w(x, y) \begin{bmatrix} I_x u & I_y v \end{bmatrix} \begin{bmatrix} I_x u \\ I_y v \end{bmatrix}\end{aligned}$$

Re-writing E:

$$\begin{aligned}E(u, v) &= \sum_{x,y} w(x, y) [I_x u + I_y v]^2 \\&= \sum_{x,y} w(x, y) \begin{bmatrix} I_x u & I_y v \end{bmatrix} \begin{bmatrix} I_x u \\ I_y v \end{bmatrix} \\&= \sum_{x,y} w(x, y) \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} I_x \\ I_y \end{bmatrix} \begin{bmatrix} I_x u \\ I_y v \end{bmatrix}\end{aligned}$$

Re-writing E:

$$\begin{aligned}E(u, v) &= \sum_{x,y} w(x, y) [I_x u + I_y v]^2 \\&= \sum_{x,y} w(x, y) \begin{bmatrix} I_x u & I_y v \end{bmatrix} \begin{bmatrix} I_x u \\ I_y v \end{bmatrix} \\&= \sum_{x,y} w(x, y) \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} I_x \\ I_y \end{bmatrix} \begin{bmatrix} I_x u \\ I_y v \end{bmatrix} \\&= \sum_{x,y} w(x, y) \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} I_x \\ I_y \end{bmatrix} \begin{bmatrix} I_x & I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}\end{aligned}$$

Re-writing E:

$$\begin{aligned}E(u, v) &= \sum_{x,y} w(x, y) [I_x u + I_y v]^2 \\&= \sum_{x,y} w(x, y) \begin{bmatrix} I_x u & I_y v \end{bmatrix} \begin{bmatrix} I_x u \\ I_y v \end{bmatrix} \\&= \sum_{x,y} w(x, y) \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} I_x \\ I_y \end{bmatrix} \begin{bmatrix} I_x u \\ I_y v \end{bmatrix} \\&= \sum_{x,y} w(x, y) \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} I_x \\ I_y \end{bmatrix} \begin{bmatrix} I_x & I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} \\&= \sum_{x,y} w(x, y) \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}\end{aligned}$$

Re-writing E:

$$\begin{aligned}E(u, v) &= \sum_{x,y} w(x, y) [I_x u + I_y v]^2 \\&= \sum_{x,y} w(x, y) \begin{bmatrix} I_x u & I_y v \end{bmatrix} \begin{bmatrix} I_x u \\ I_y v \end{bmatrix} \\&= \sum_{x,y} w(x, y) \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} I_x \\ I_y \end{bmatrix} \begin{bmatrix} I_x u \\ I_y v \end{bmatrix} \\&= \sum_{x,y} w(x, y) \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} I_x \\ I_y \end{bmatrix} \begin{bmatrix} I_x & I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} \\&= \sum_{x,y} w(x, y) \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} \\&= w(x, y) \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} \sum_{x,y} I_x^2 & \sum_{x,y} I_x I_y \\ \sum_{x,y} I_x I_y & \sum_{x,y} I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}\end{aligned}$$

Assuming $w(x, y) = 1$

Re-writing E:

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

Re-writing E:

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

$$= w(x, y) \begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix}$$

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

Re-writing E:

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

$$= w(x, y) \begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix}$$

Does anyone know what this part of the equation is?

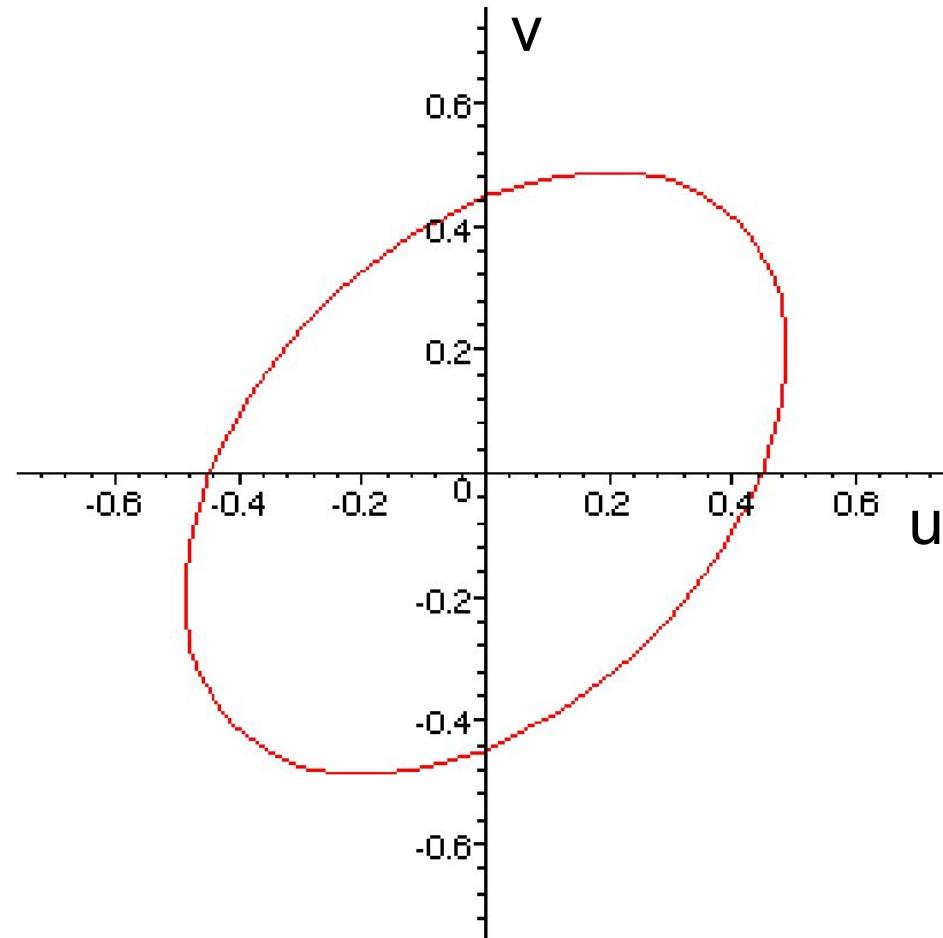
$$M = \begin{bmatrix} \sum_{x,y} I_x^2 & \sum_{x,y} I_x I_y \\ \sum_{x,y} I_x I_y & \sum_{x,y} I_y^2 \end{bmatrix}$$

It's the equation of an ellipse

$$5u^2 - 4uv + 5v^2 = 1$$

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

$$M = \begin{bmatrix} 5 & -2 \\ -2 & 5 \end{bmatrix}$$



Change in intensity in a patch

- So, using Taylor's expansion, the change in intensity in an image patch:

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

where M is a 2×2 matrix computed from image derivatives:

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

Sum over image region – the area we are
checking for corner

**Gradient with
respect to x ,
times gradient
with respect to y**

Harris Detector Formulation

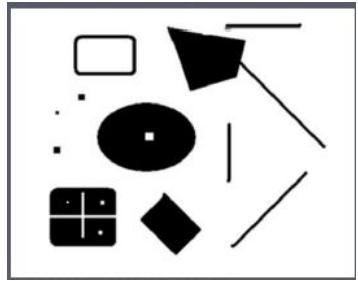
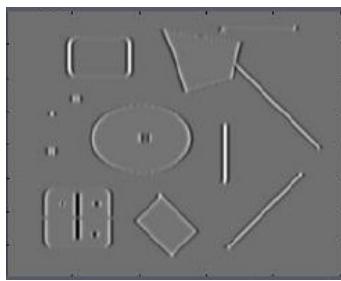
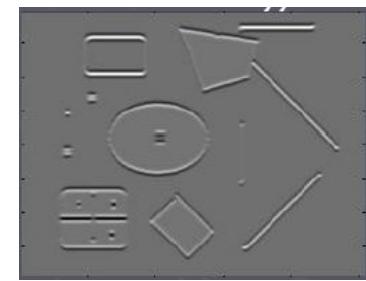


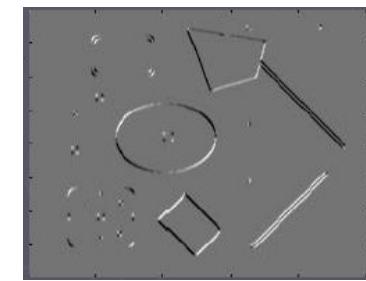
Image I



I_x



I_y



II_{xy}

where M is a 2×2 matrix computed from image derivatives:

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

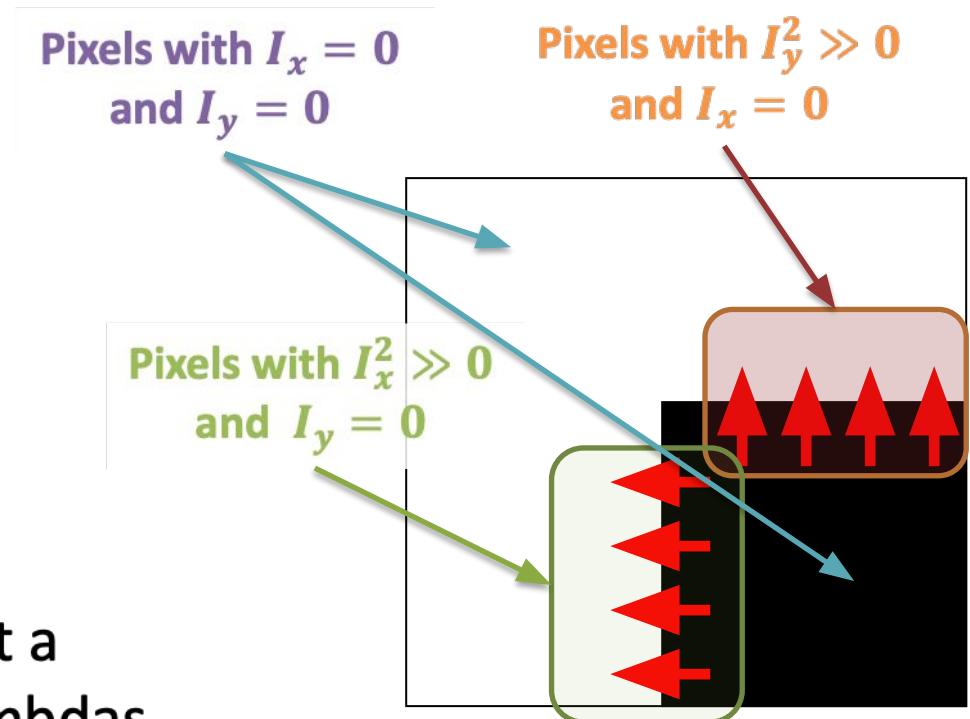
↑
Sum over image region – the area we are
checking for corner

**Gradient with
respect to x ,
times gradient
with respect to y**

$$M = \begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix}$$

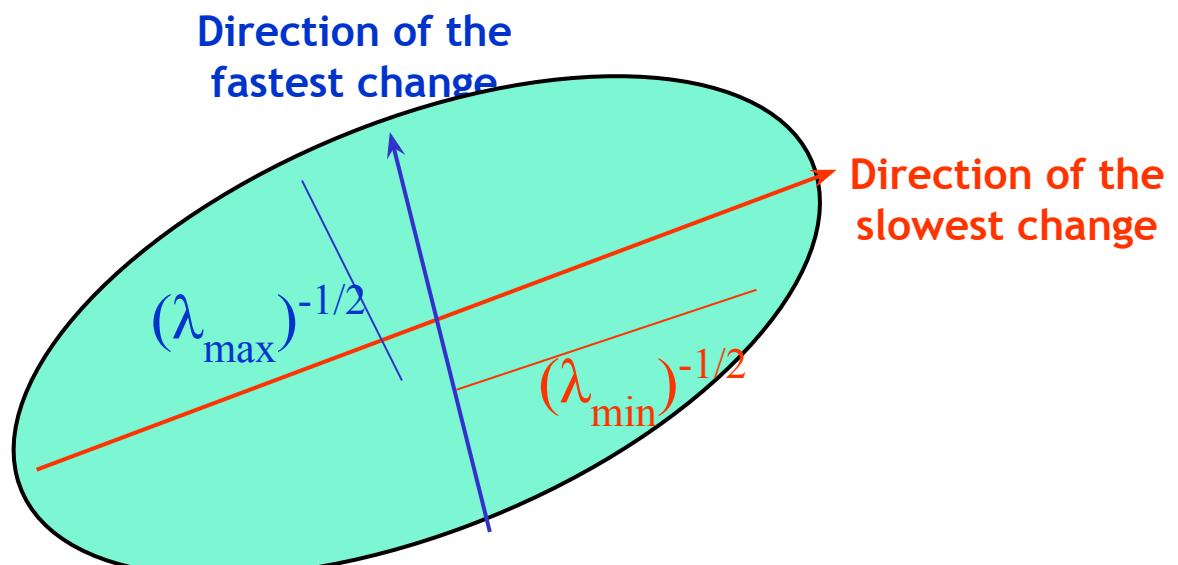
What Does This Matrix Reveal?

- First, let's consider an axis-aligned corner.
- In that case, the dominant gradient directions align with the x or the y axis
- $M = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$
- This means: if either λ is close to 0, then this is not a corner, so look for image windows where both lambdas are large.
- What if we have a corner that is not aligned with the image axes?



General Case

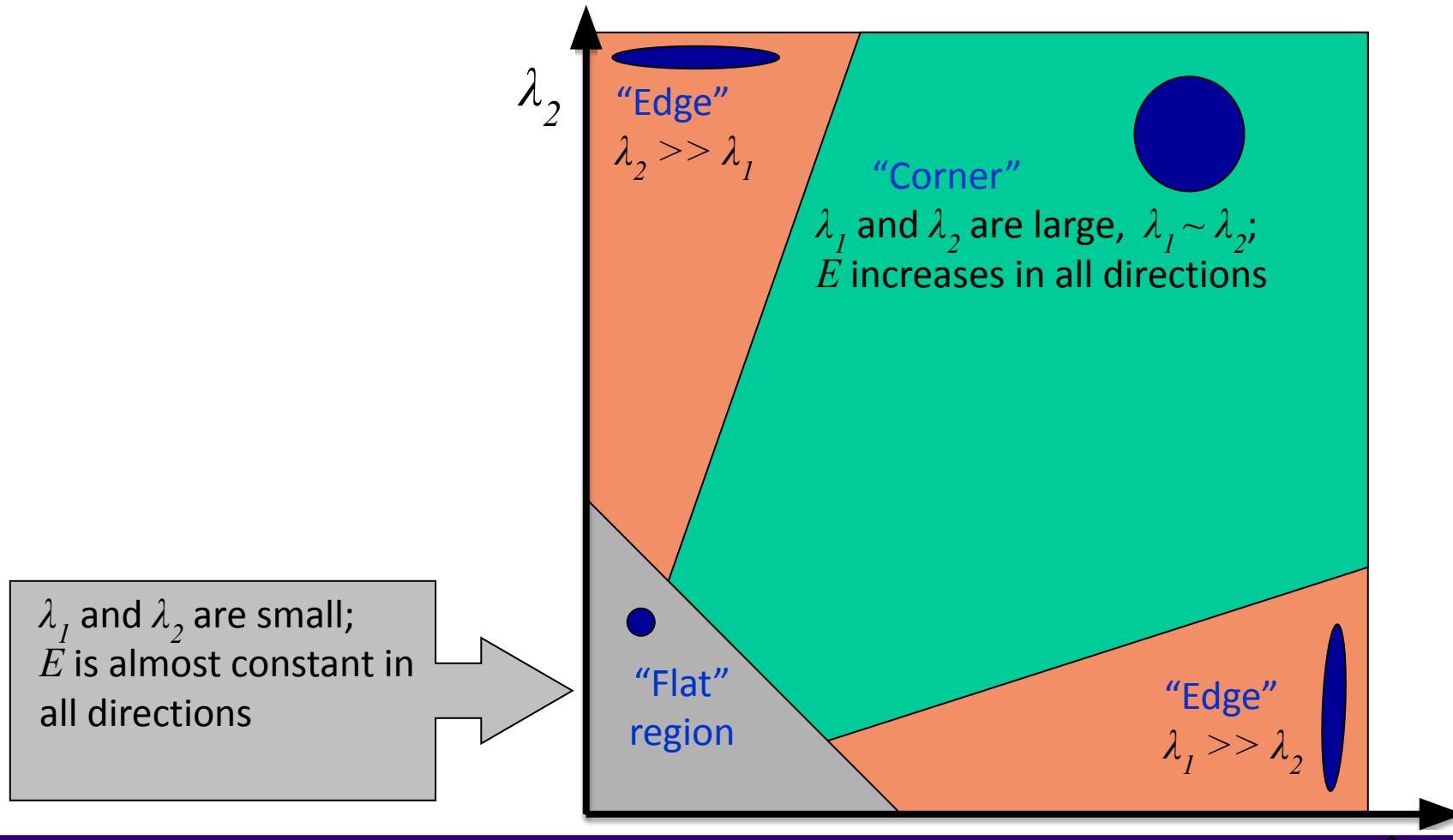
- Since $M = \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$ is symmetric, we can re-rewrite $M = R^{-1} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} R$
(Eigenvalue decomposition)
- We can think of M as an ellipse with its axis lengths determined by the eigenvalues λ_1 and λ_2 ; and its orientation determined by R



- A rotated corner would produce the same eigenvalues as its non-rotated version.

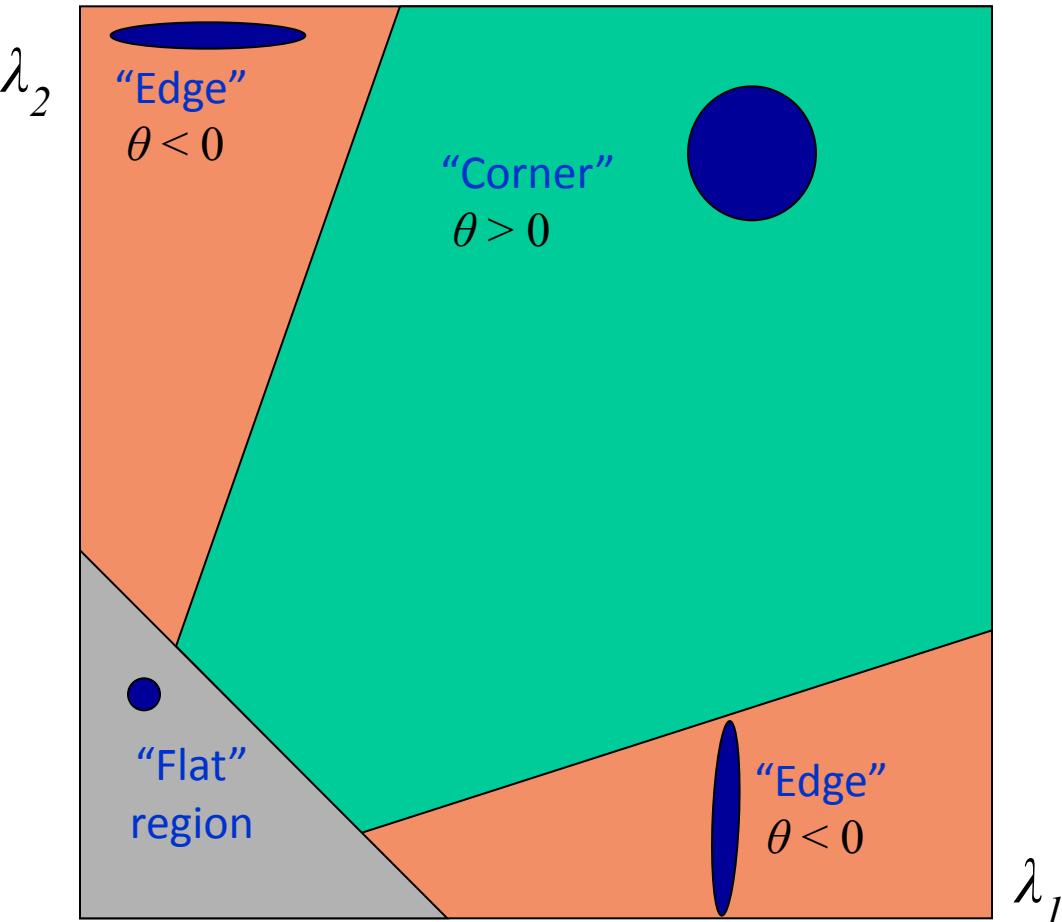
Interpreting the Eigenvalues

- Classification of image points using eigenvalues of M :



Corner Response Function

$$\theta = \det(M) - \alpha \operatorname{trace}(M)^2 = \lambda_1 \lambda_2 - \alpha(\lambda_1 + \lambda_2)^2$$



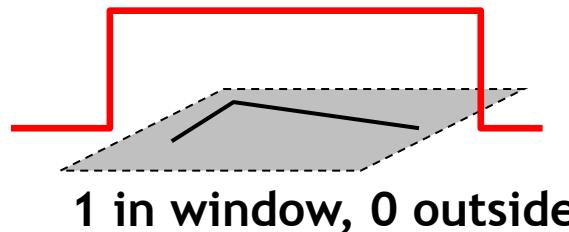
- Fast approximation
 - Avoid computing the eigenvalues
 - α : constant (0.04 to 0.06)

Window Function $w(x,y)$

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

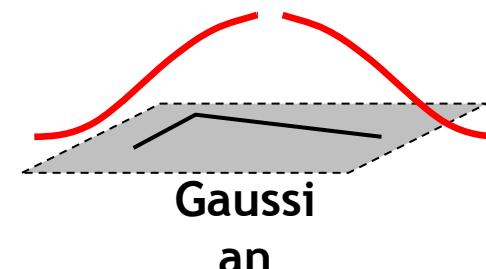
- Option 1: uniform window
 - Sum over square window
 - Problem: not rotation invariant

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



- Option 2: Smooth with Gaussian
 - Gaussian already performs weighted sum

$$M = g(\sigma) * \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$



- Result is rotation invariant

Summary: Harris Detector [Harris88]

- Compute second moment matrix (autocorrelation matrix)

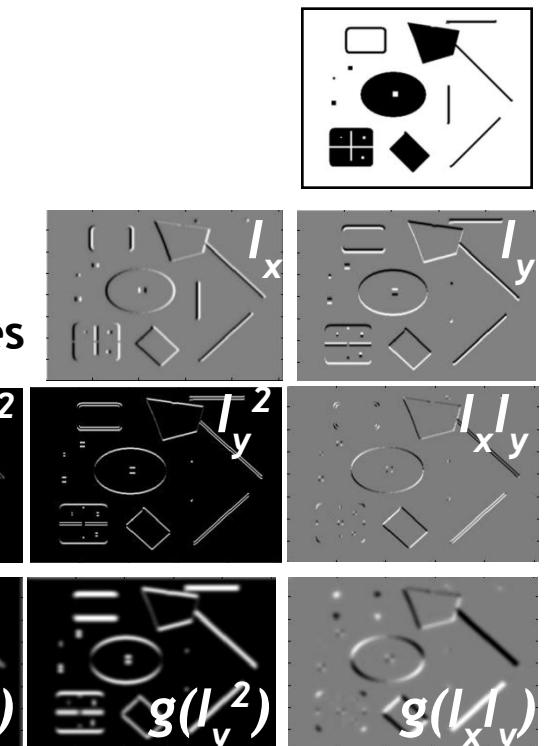
$$M(\sigma_I, \sigma_D) = g(\sigma_I) * \begin{bmatrix} I_x^2(\sigma_D) & I_x I_y(\sigma_D) \\ I_x I_y(\sigma_D) & I_y^2(\sigma_D) \end{bmatrix}$$

σ_D : for Gaussian in the derivative calculation
 σ_I : for Gaussian in the windowing function

2. Square of derivatives

1. Image derivatives

3. Gaussian filter $g(\sigma)$



4. Cornerness function - two strong eigenvalues

$$\begin{aligned}\theta &= \det[M(\sigma_I, \sigma_D)] - \alpha[\text{trace}(M(\sigma_I, \sigma_D))]^2 \\ &= g(I_x^2)g(I_y^2) - [g(I_x I_y)]^2 - \alpha[g(I_x^2) + g(I_y^2)]^2\end{aligned}$$

5. Perform non-maximum suppression



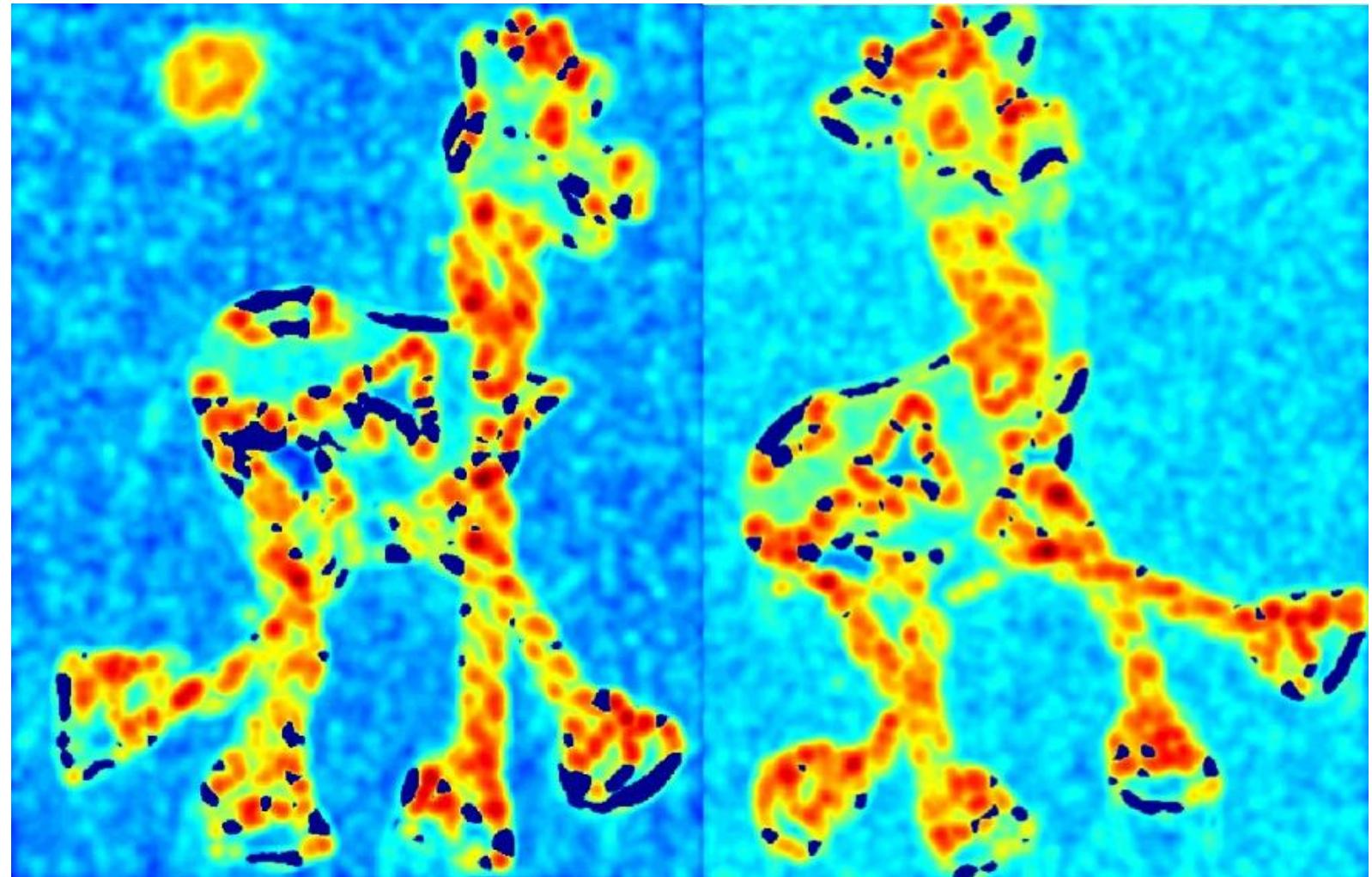
Harris Detector: Example

- Input Image



Harris Detector: Example

- Input Image
 - Compute corner response function
- θ



Harris Detector: Example

- Input Image
- Compute corner response function θ
- Take only the local maxima of θ , where $\theta >$ threshold

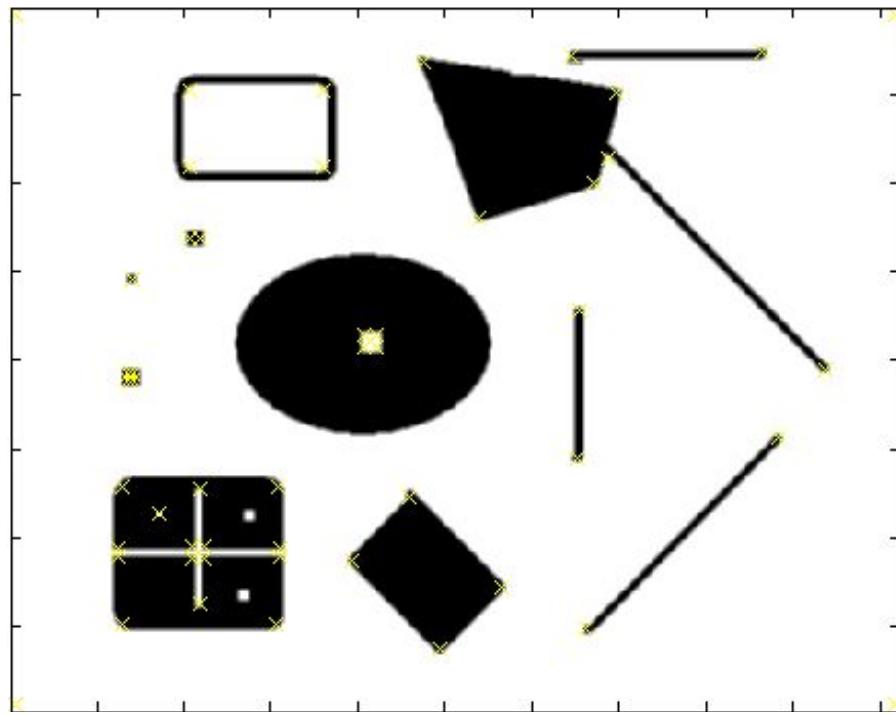


Harris Detector: Example

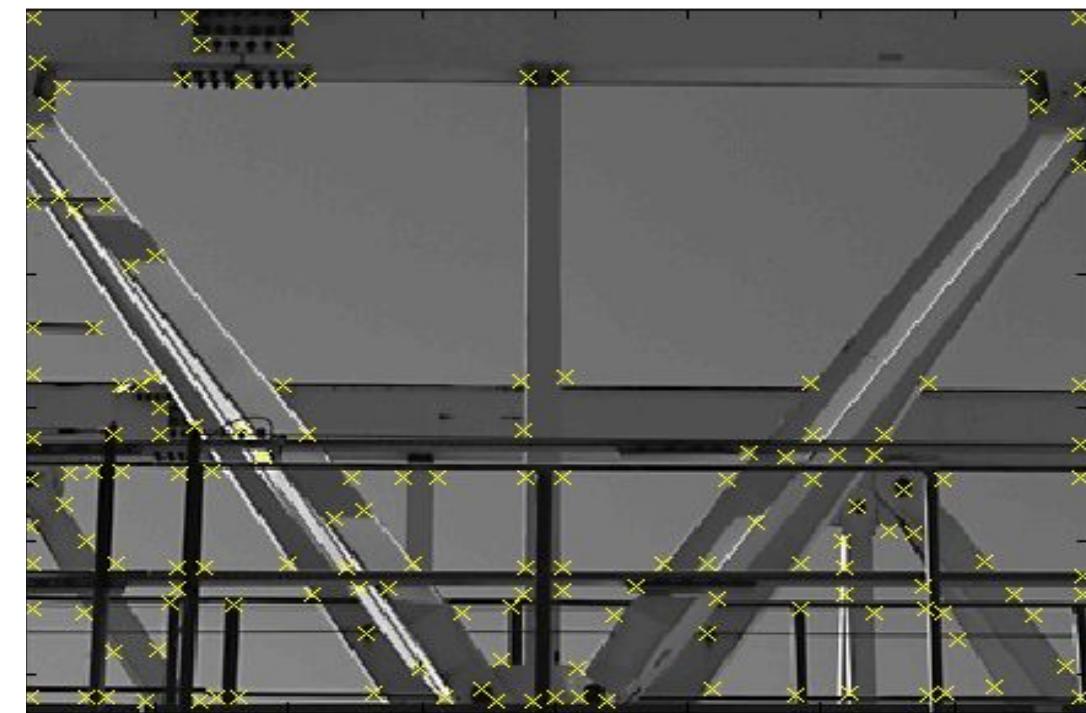
- Input Image
- Compute corner response function θ
- Take only the local maxima of θ , where $\theta >$ threshold



Harris Detector – Responses [Harris88]



Effect: A very precise corner detector.



Harris Detector – Responses [Harris88]



Harris Detector – Responses [Harris88]



- Results are great for finding correspondences matches between images

Summary

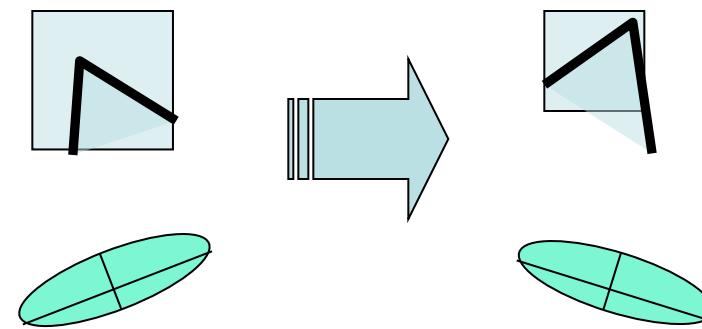
- Local Invariant Features
- Harris Corner Detector

Harris Detector: Properties

- Translation invariance?

Harris Detector: Properties

- Translation invariance
- Rotation invariance?

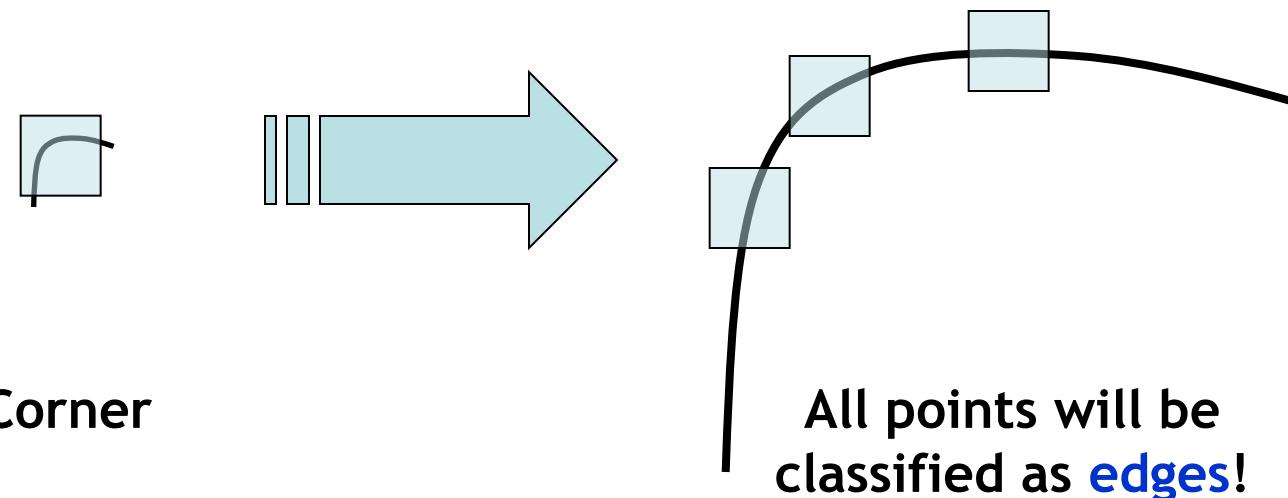


Ellipse rotates but its shape (i.e. eigenvalues) remains the same

Corner response θ is invariant to image rotation

Harris Detector: Properties

- Translation invariance
- Rotation invariance
- Scale invariance?



Not invariant to image scale!

Next time

Detectors and Descriptors