Lecture 6

Keypoints and Corners

Administrative

A1 due date extension ******

- You can use up to 2 late days

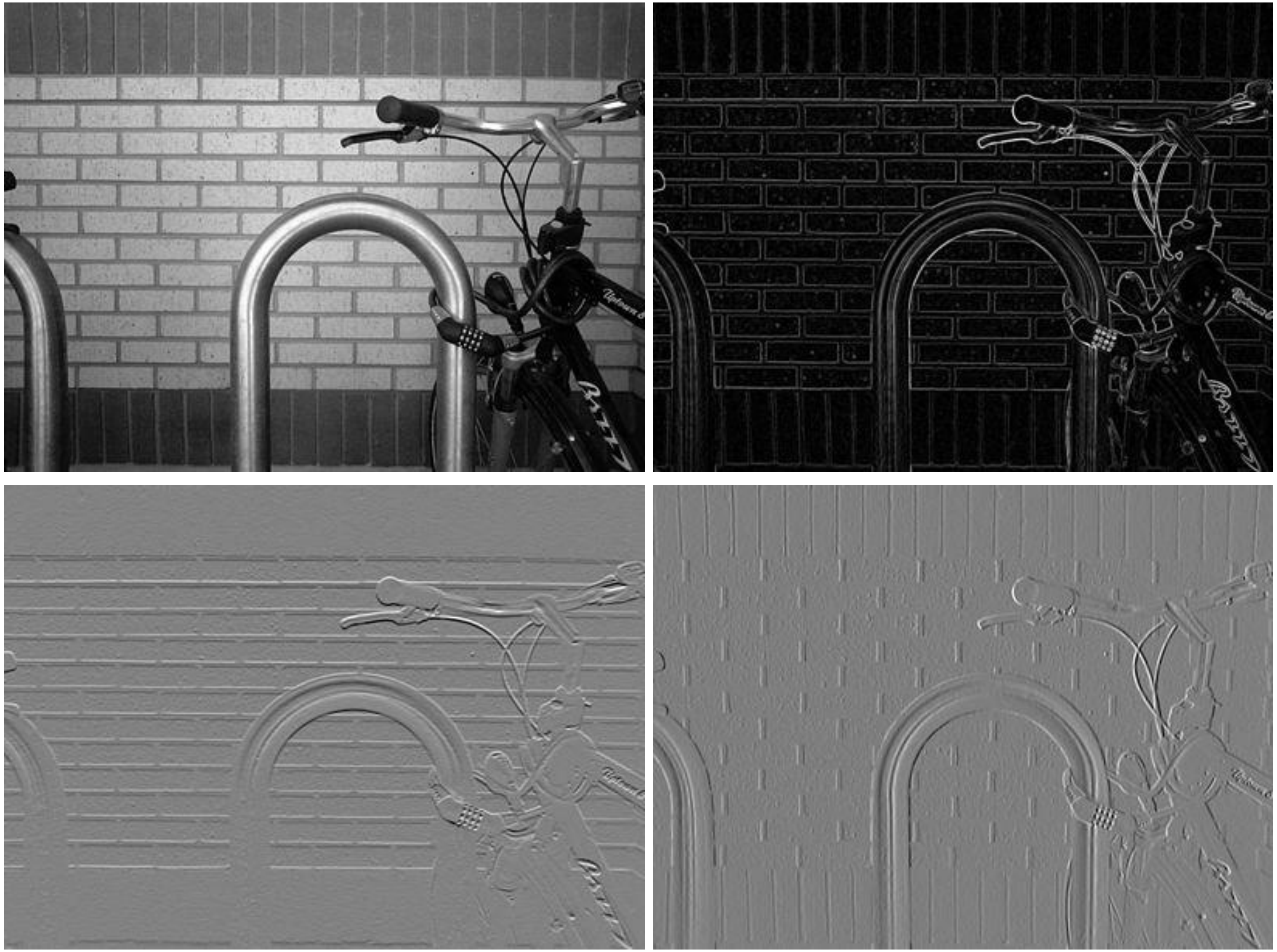
Administrative

- Recitation this Friday
- Geometric transformations

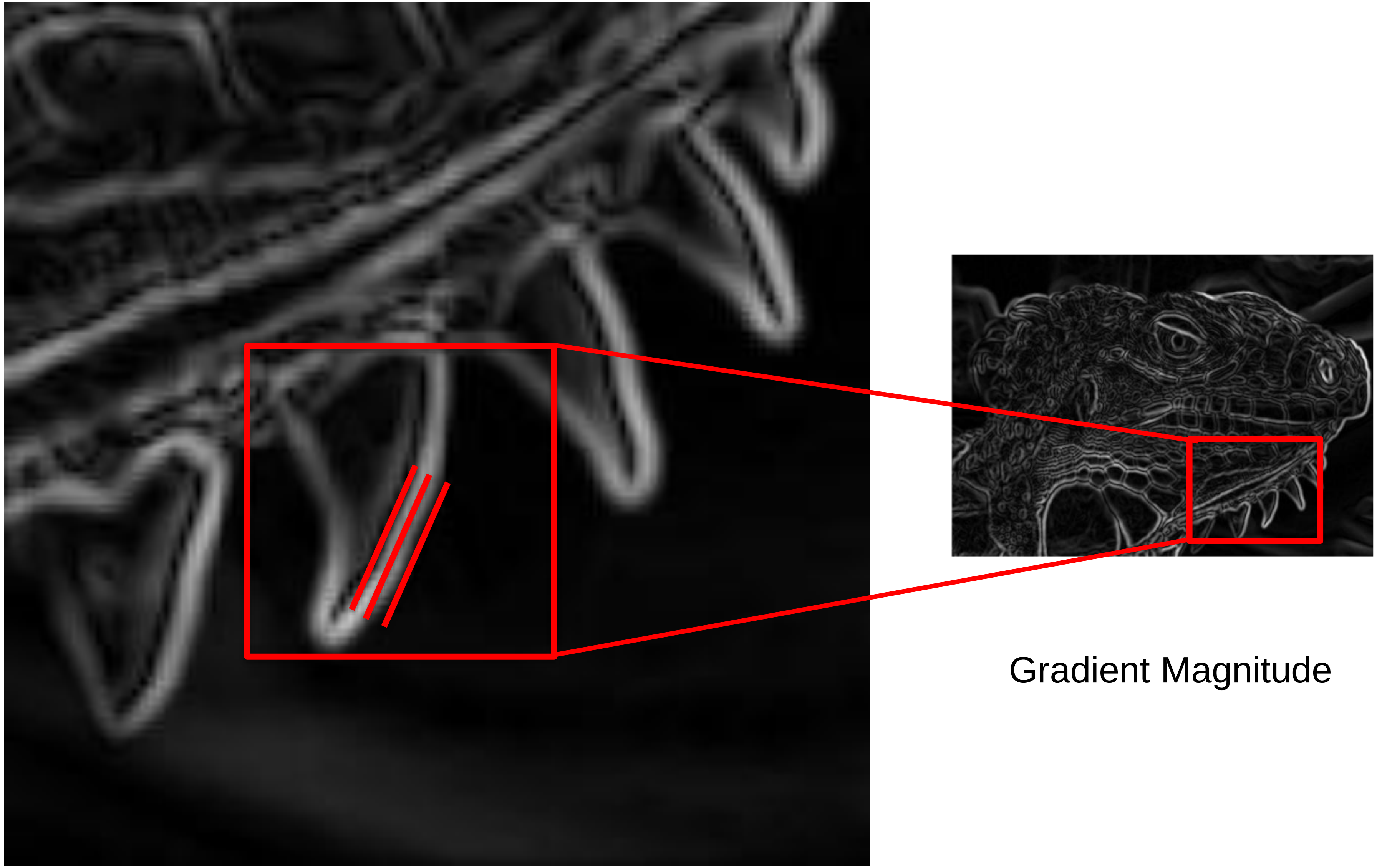
So far: Sobel Filter

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

Step 2: Threshold the values to generate a binary image

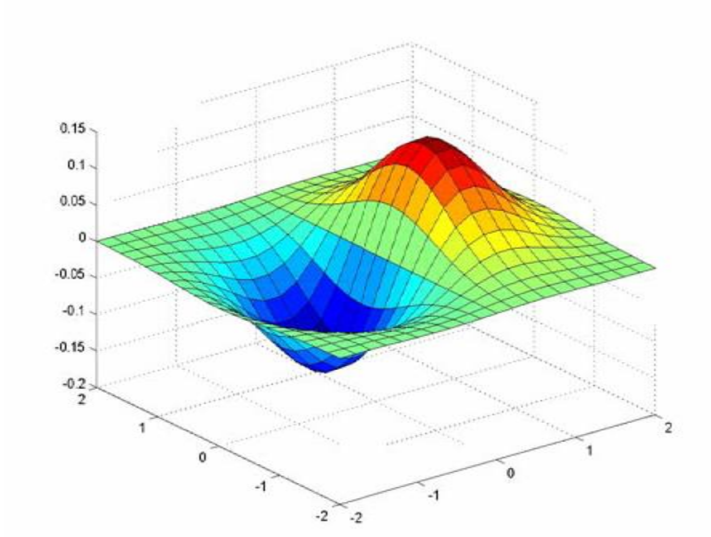


So far: challenges multiple disconnected edges

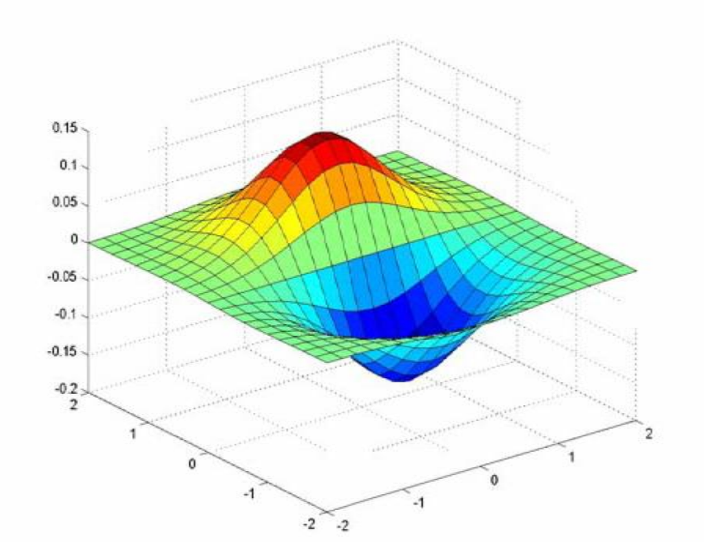


Gradient Magnitude

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

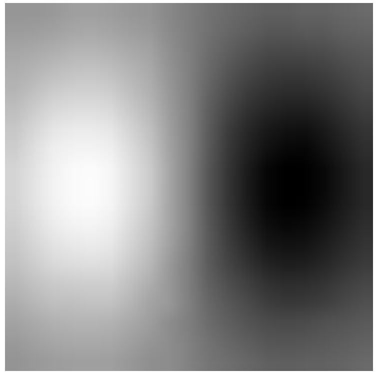


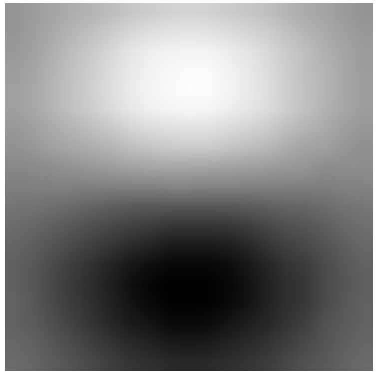
x-direction



y-direction

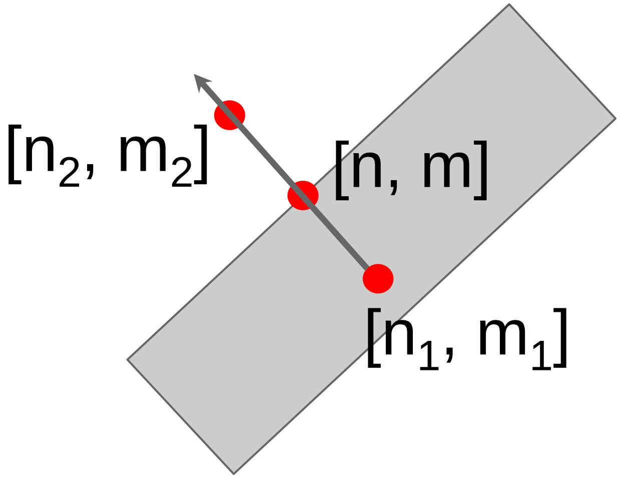
$$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$



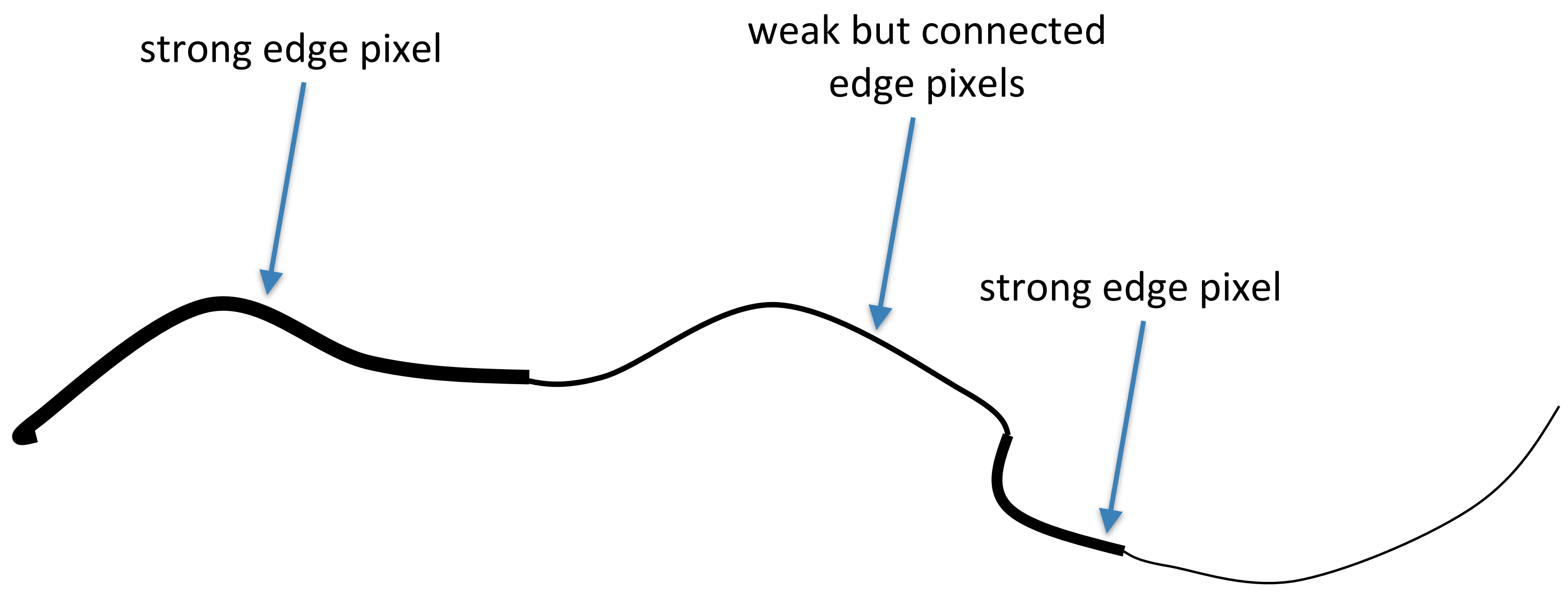


$$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

So far: Non-maximum suppression



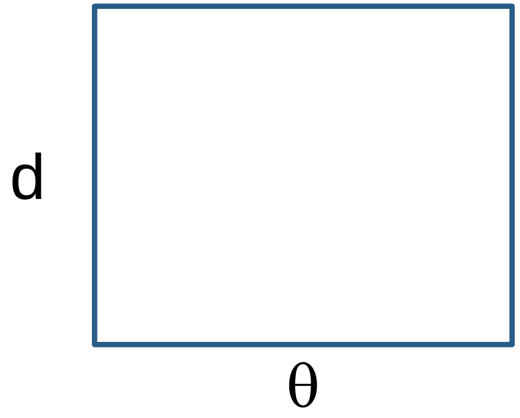
So far: Hysteresis thresholding
Strong and weak edges



Source: S. Seitz

So far: The Hough transform

- So: one point (x_i,y_i) gives a line in (d,θ) space.
- Or, we can use multiple θs to generate several possible lines through that point
- Iterate over θs to vote for buckets in (d,θ)-space



Today's agenda

- RANSAC
- Local Invariant Features
- Harris Corner Detector

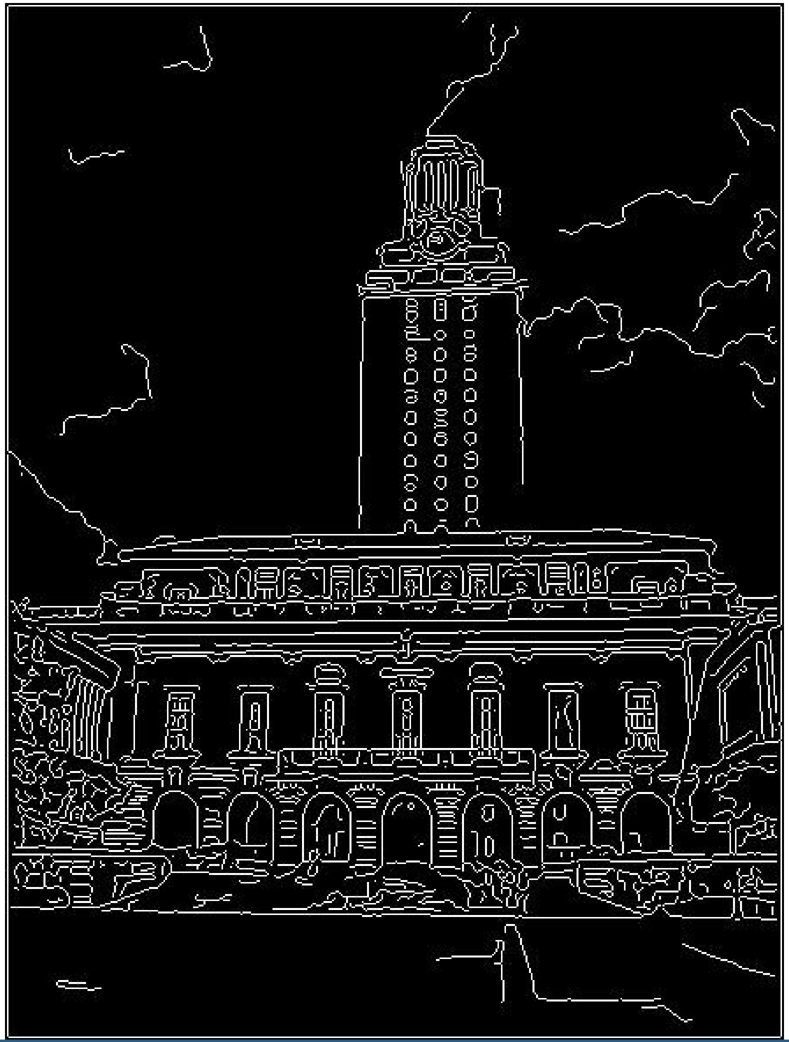
Today’s agenda

- RANSAC
- Local Invariant Features
- Harris Corner Detector

Is Hough transform inefficient?

- The version taught in the previous course checked ALL pairs of edge points. It's not feasible to check all pairs of points to calculate possible lines. That Hough Transform algorithm runs in $O(N^2)$.
- **Our** Hough Transform runs in O(N*number of thetas)
- 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.

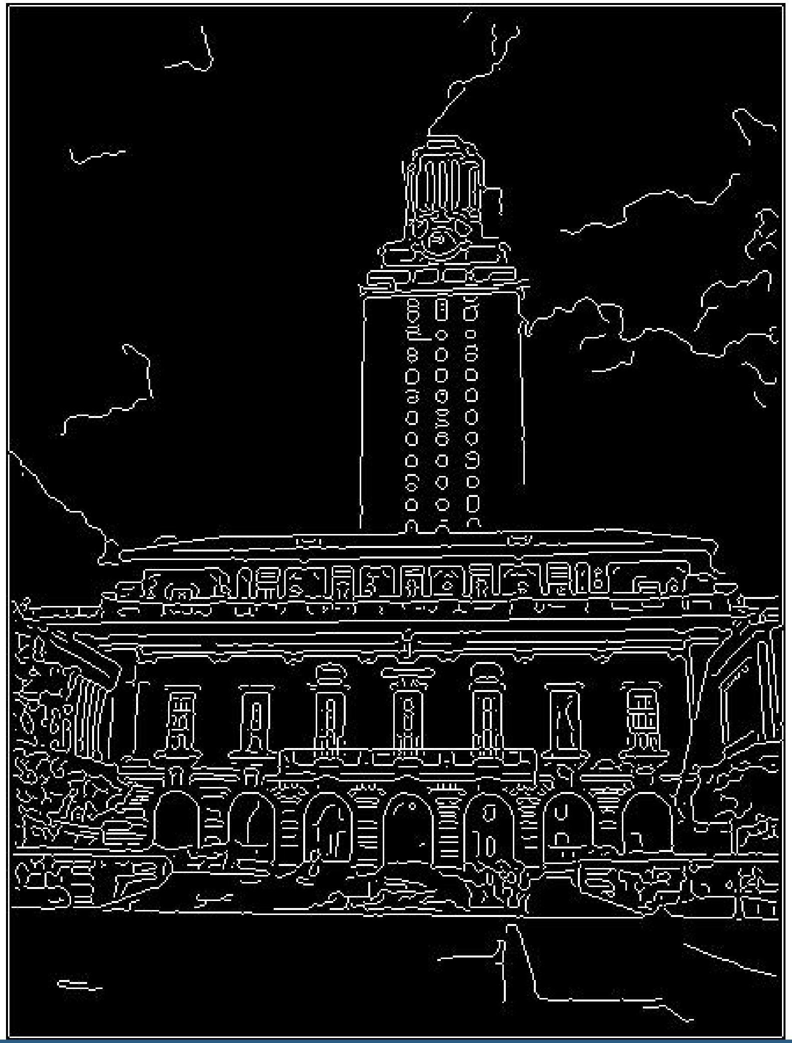
Difficulty of voting for lines

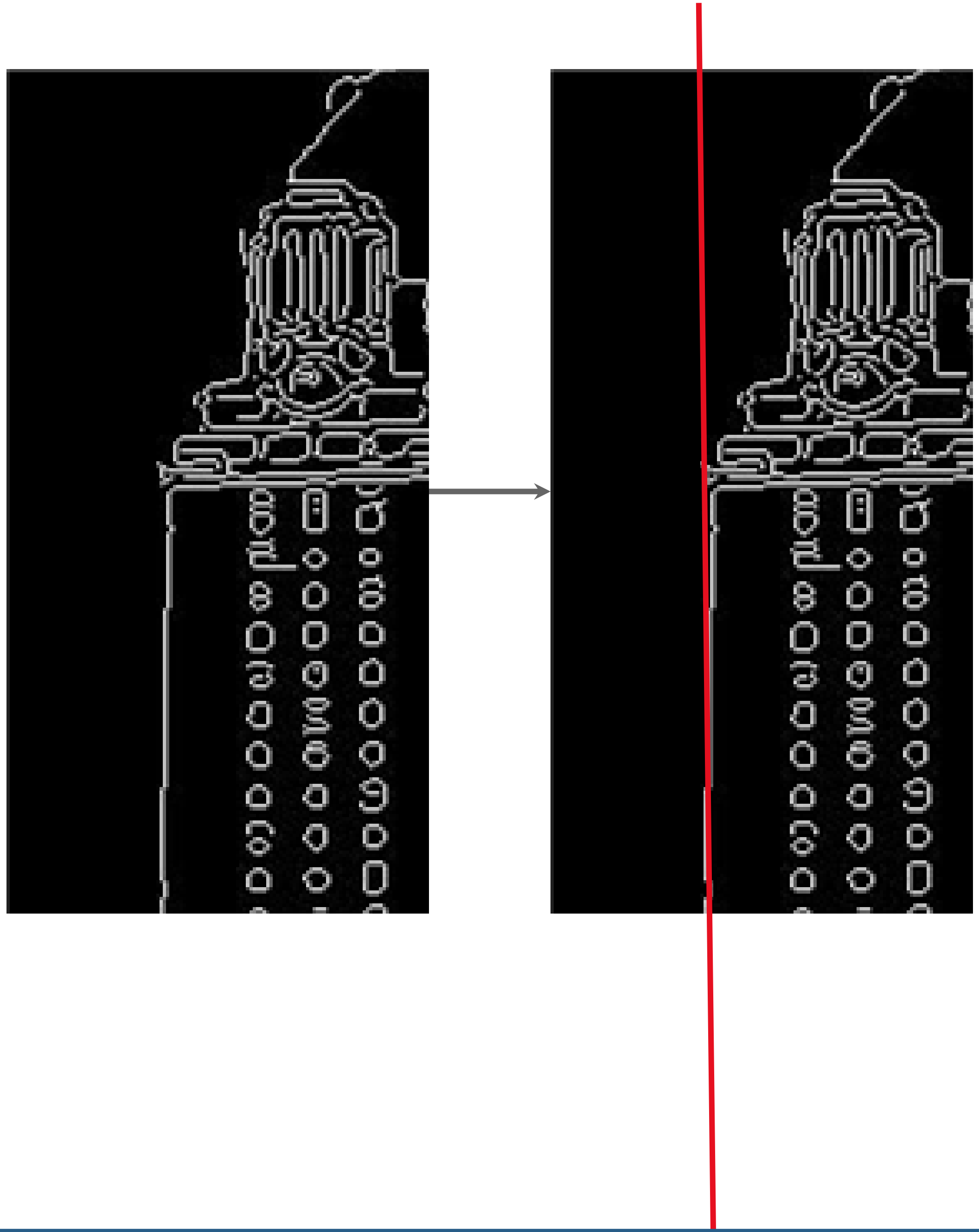


- Noisy edge pixels cast inconsistent votes:
  - Can we identify false edge pixels without iterating?

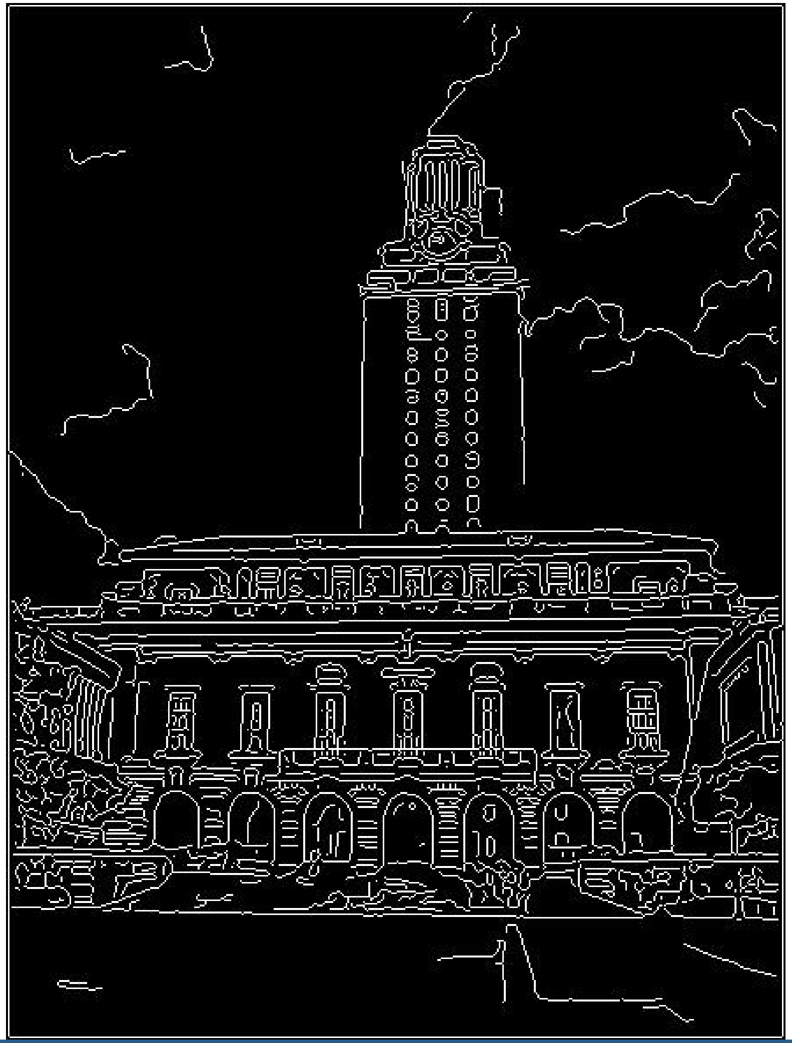
- Canny can predict false positive edge points:
  - Can we eliminate them?

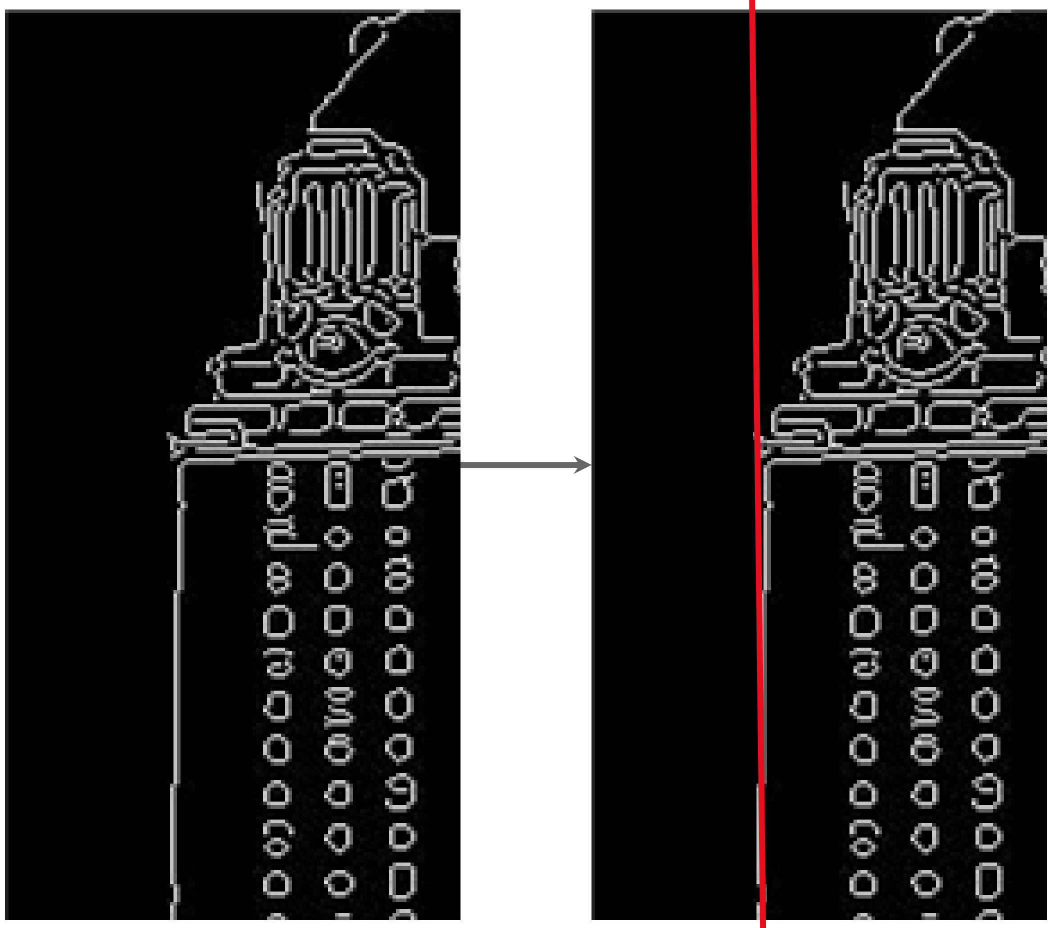
Intuition

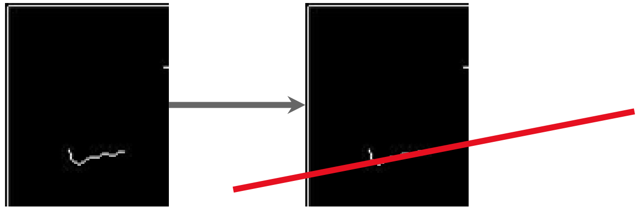




Intuition







RANSAC [Fischler & Bolles 1981]

Bob Bolles at SRI
RANSAC has 37,314 citations

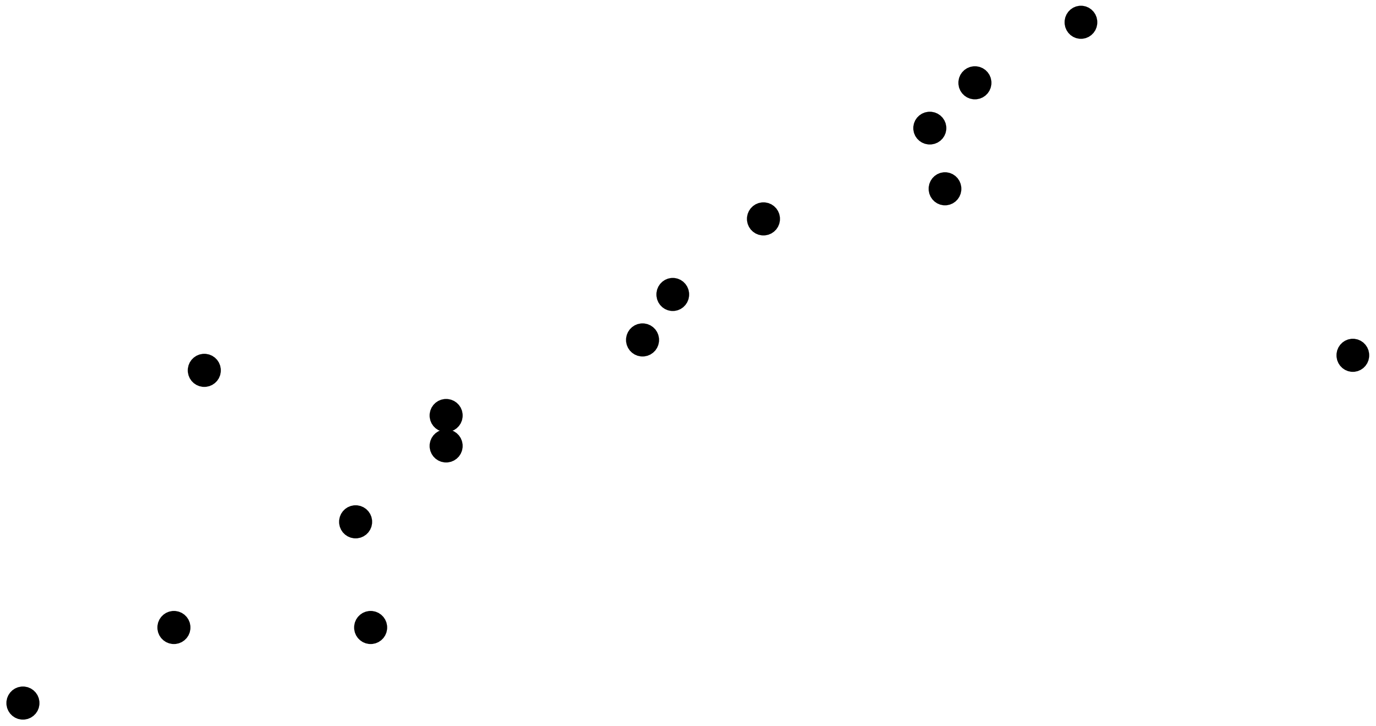
- RANdom SAmple Consensus

- **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 (a,b) of a line, then the resulting line won't have much support from rest of the points.

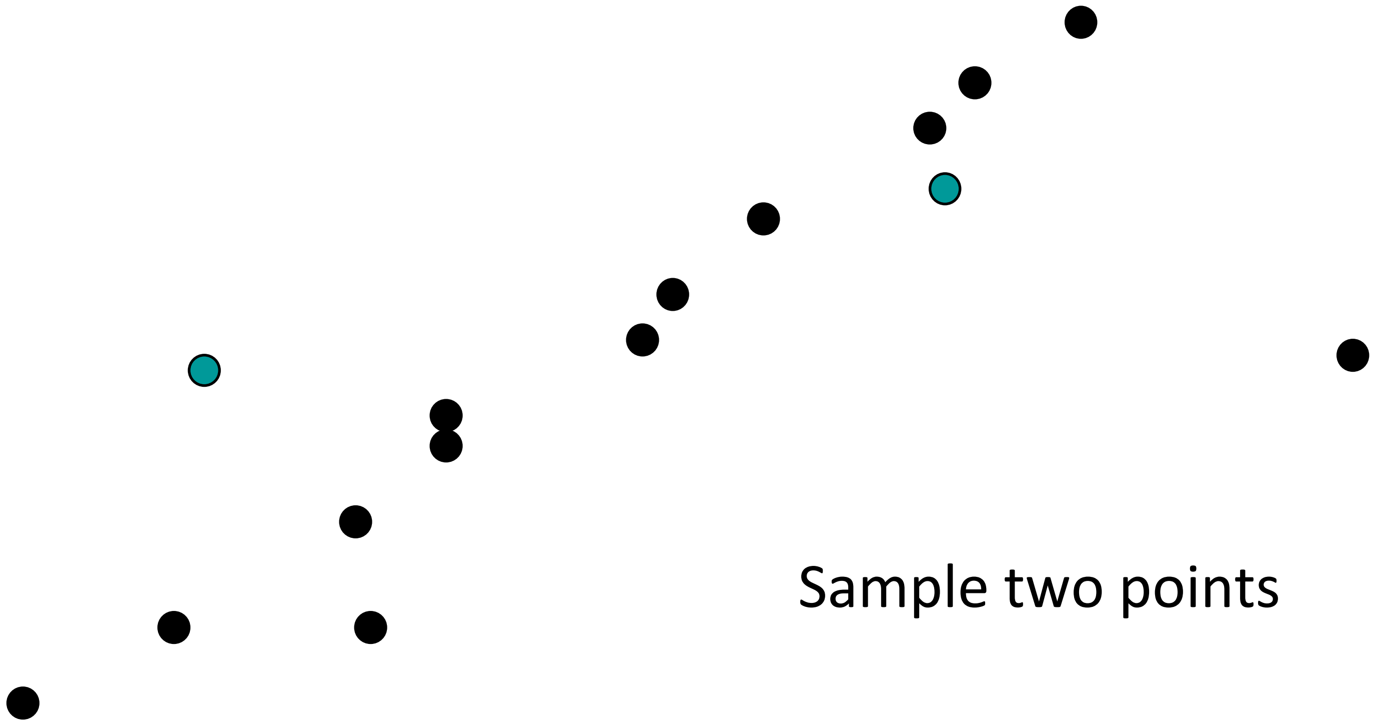
RANSAC Line Fitting Example

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



RANSAC Line Fitting Example

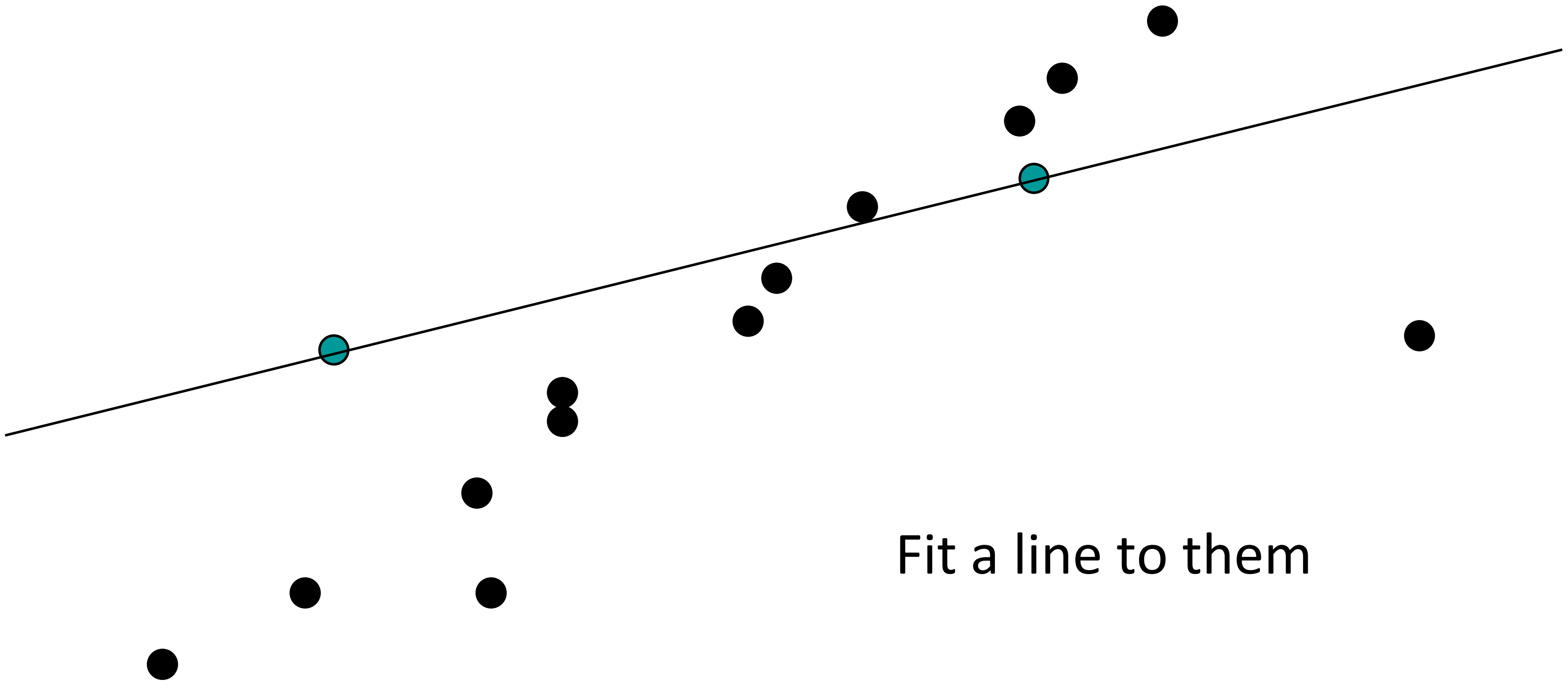
- Task: Estimate the best line
  - Let's select only 2 points as an example



Sample two points

RANSAC Line Fitting Example

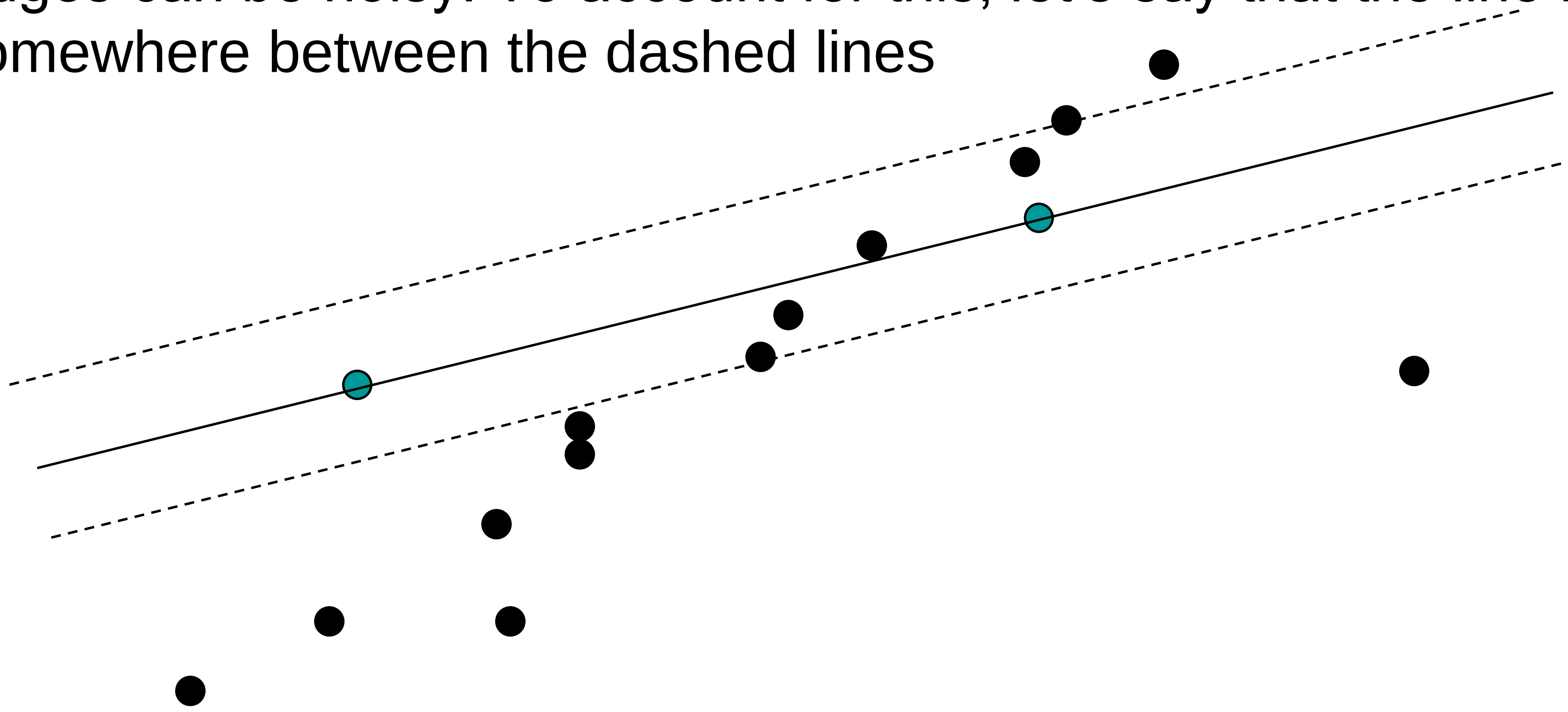
- Task: Estimate the best line
  - Calculate the line parameters



Fit a line to them

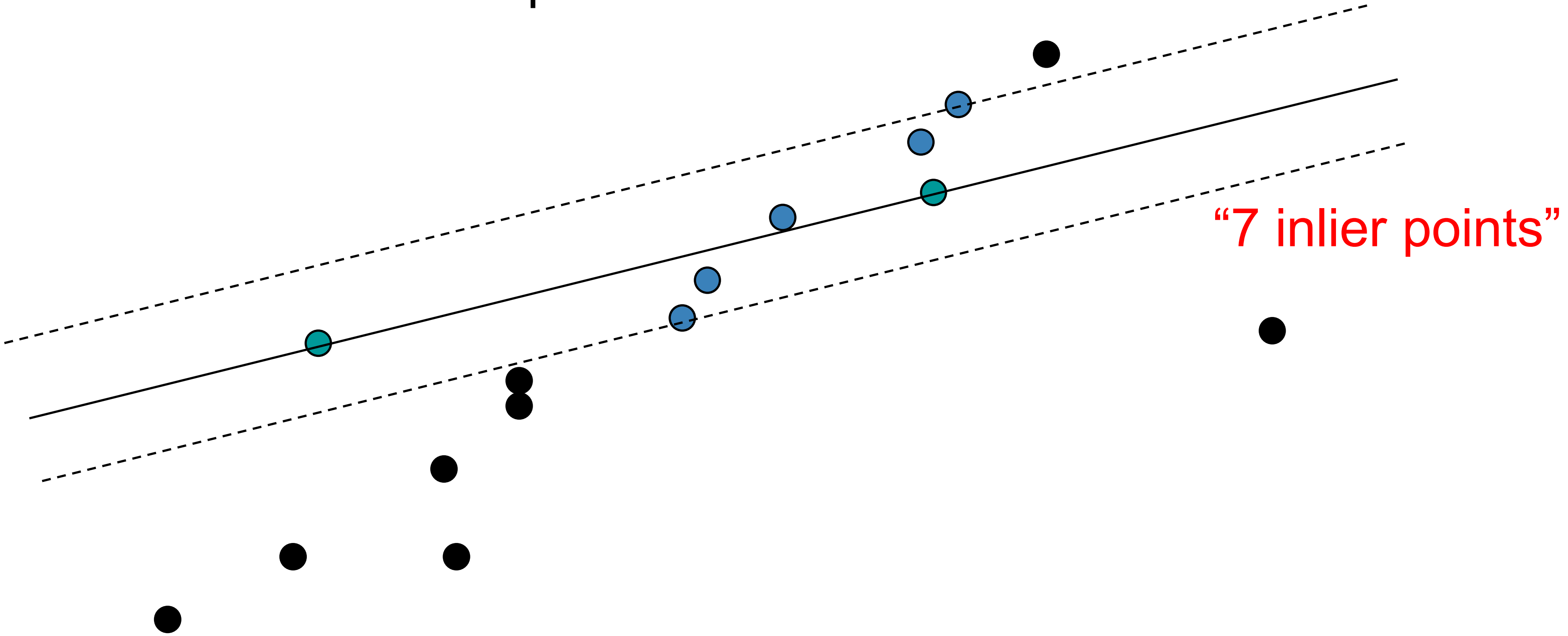
RANSAC Line Fitting Example

- 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

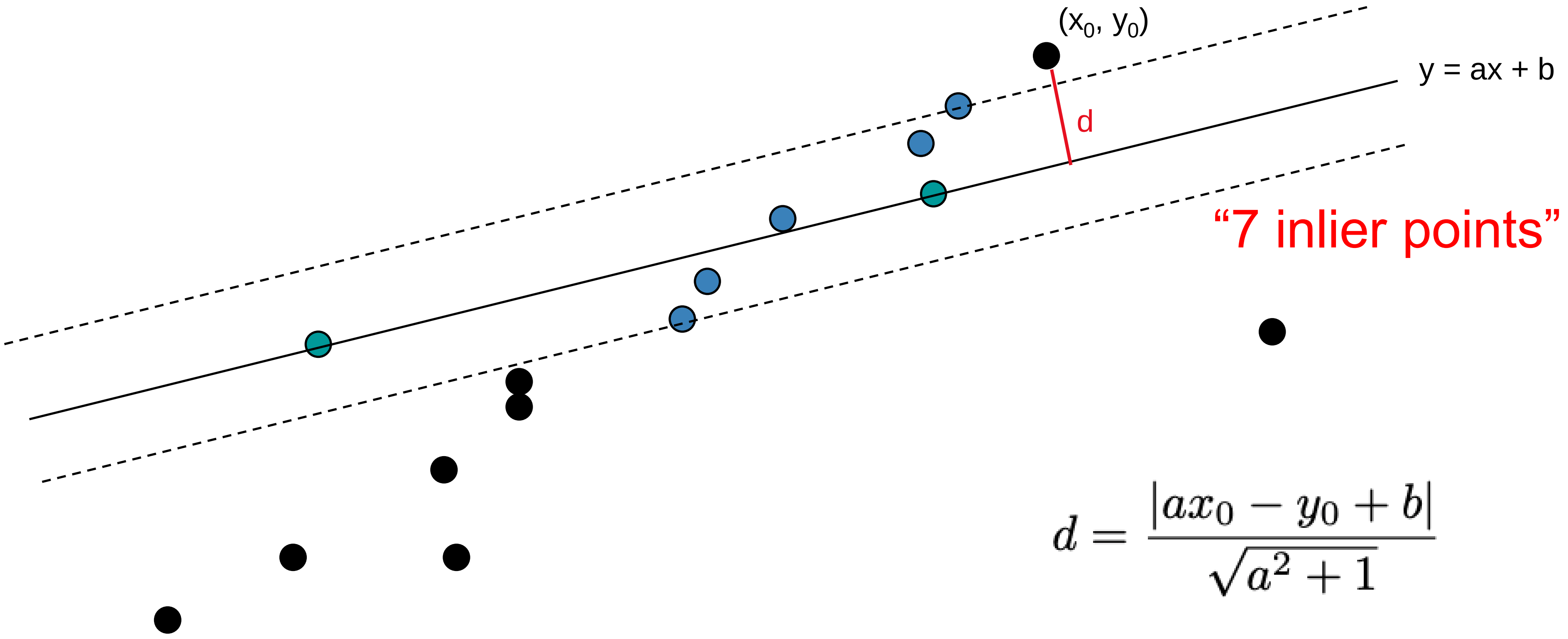


RANSAC Line Fitting Example

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



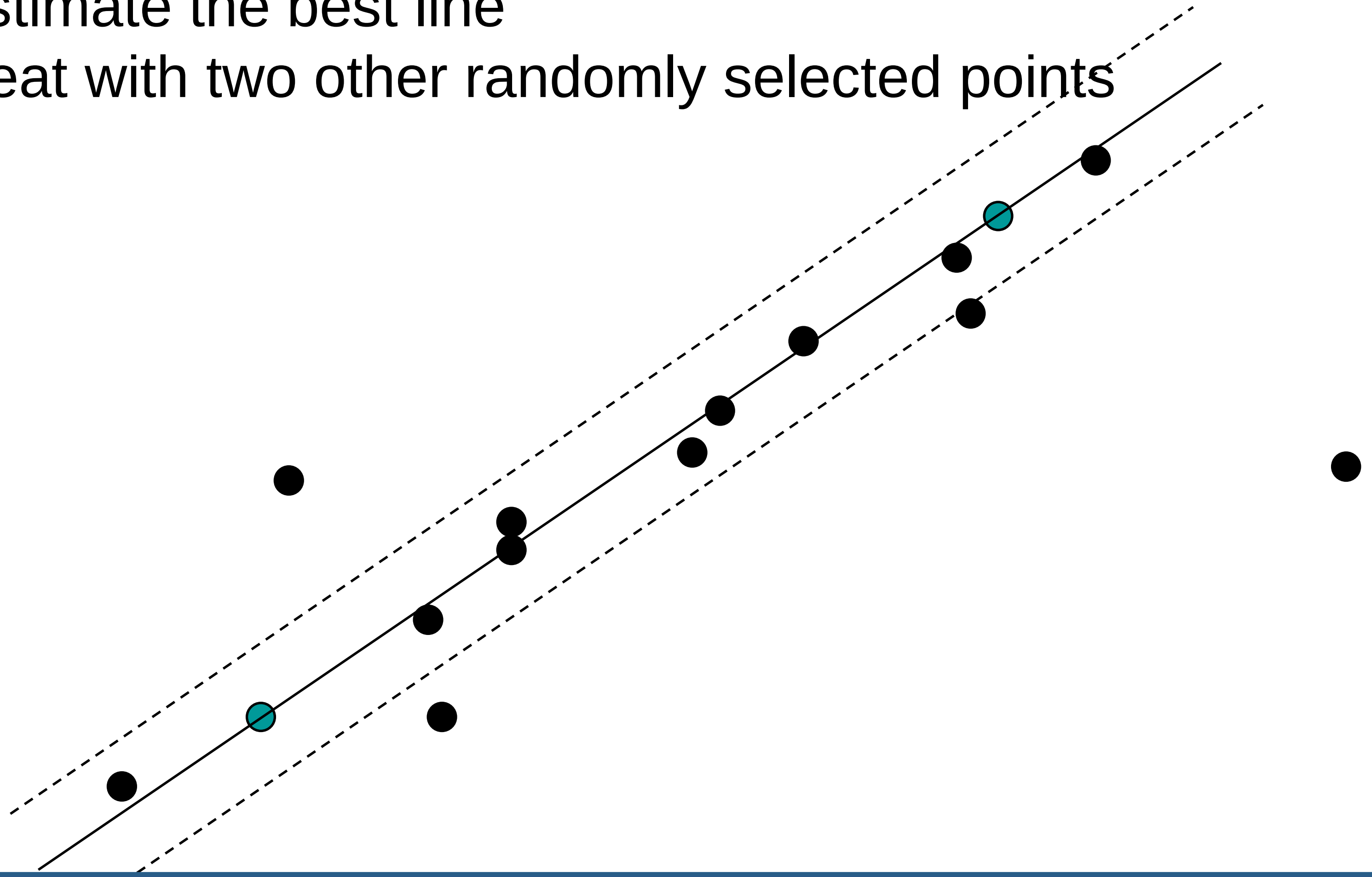
How do we calculate the inliers?
We use the distance from the point to the line



$$d = \frac{|ax_0 - y_0 + b|}{\sqrt{a^2 + 1}}$$

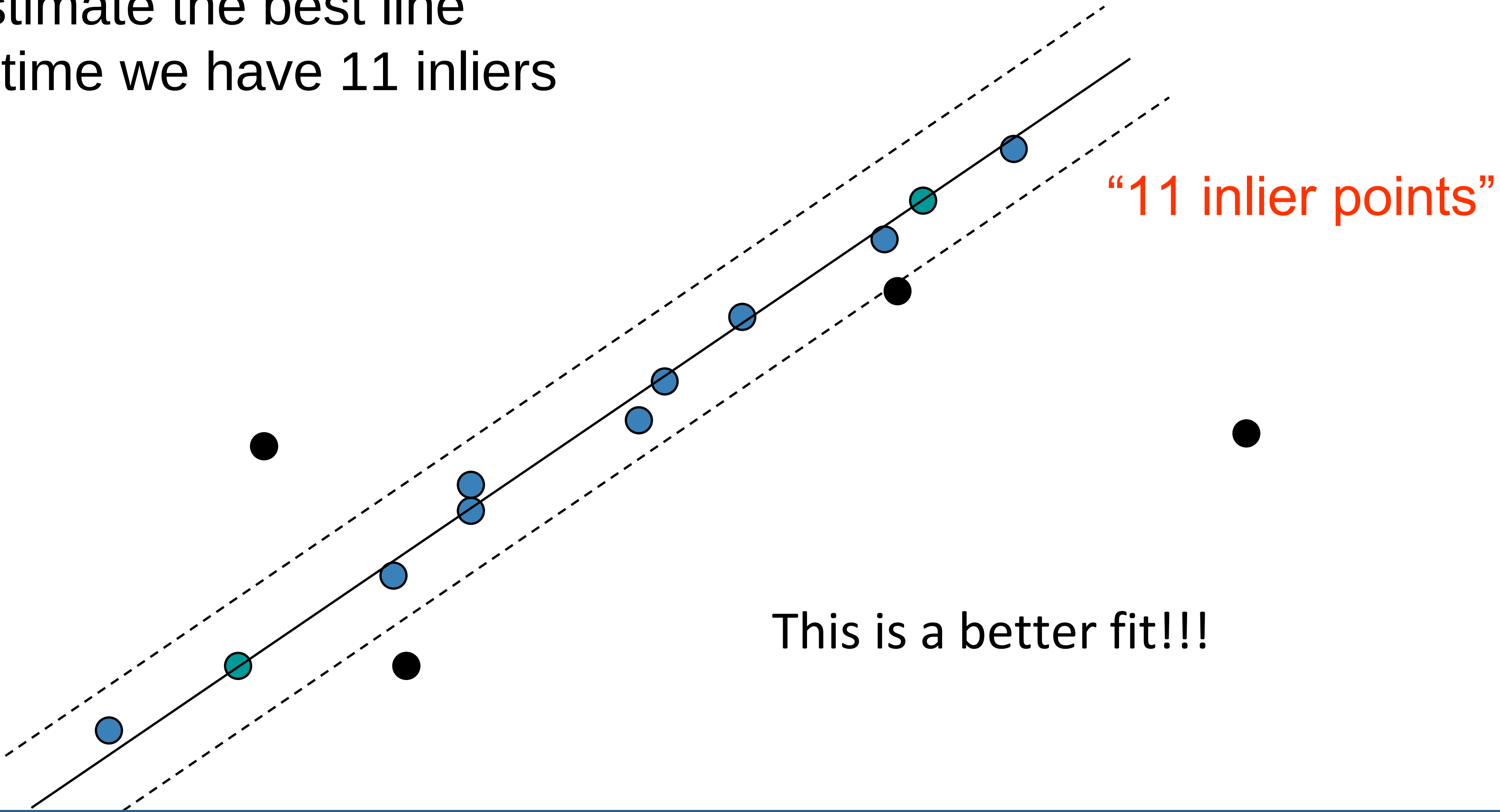
RANSAC Line Fitting Example

- 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 (a, b) 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 (a, b) 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 (a, b) 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]

<u>RANSAC loop</u>:

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 (a, b) 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:

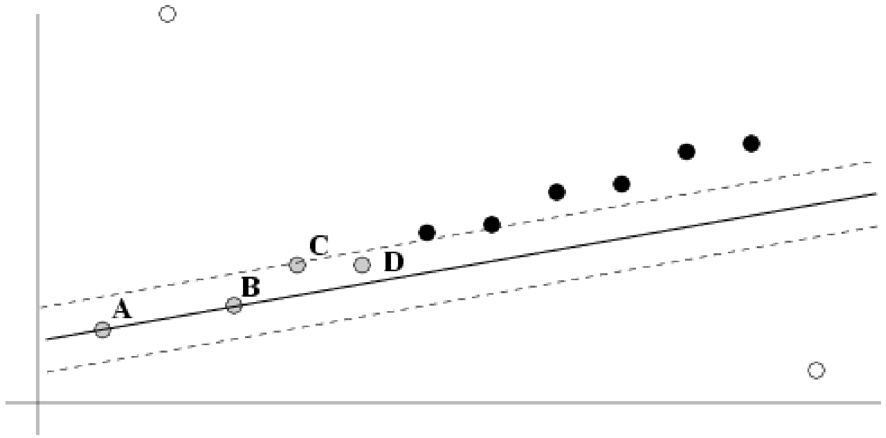
1. Randomly select a *seed* subset of points on which to perform a model estimate (e.g., a group of edge points)
2. Compute parameters (a, b) from seed group
3. Find inliers for these parameters
4. If the number of inliers is larger than the best so far, save these parameters and the inliers

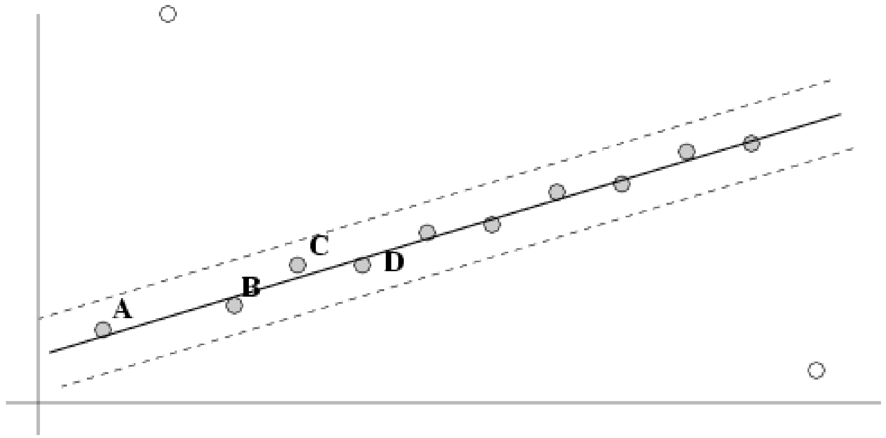
If number of inliers in the best line is < m, return no line

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.





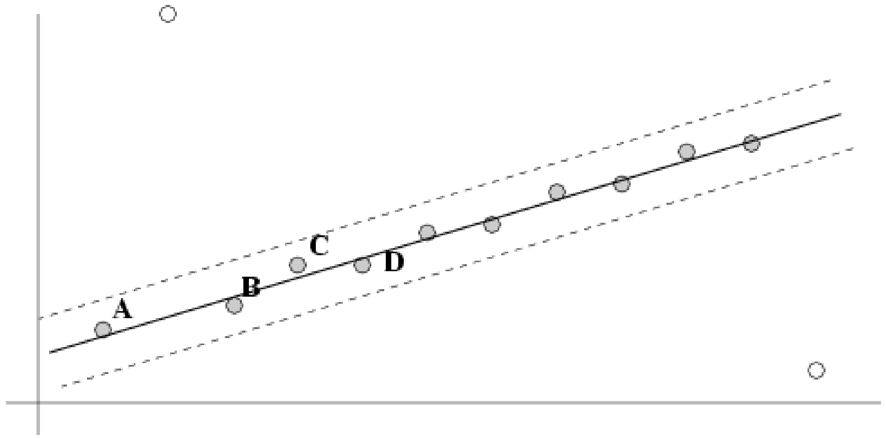
How do you calculate the line from many points?

(x_i, y_i) is a set of points we are going to use to estimate (a, b)

Least squares method:

$$a = \frac{(\sum_i x_i - \bar{x})(\sum_i y_i - \bar{y})}{(\sum_i x_i - \bar{x})^2}$$

$$b = y_i - a x_i$$



The RANSAC algorithm [Fischler & Bolles 1981]

<u>RANSAC loop</u>:

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 (a, b) 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

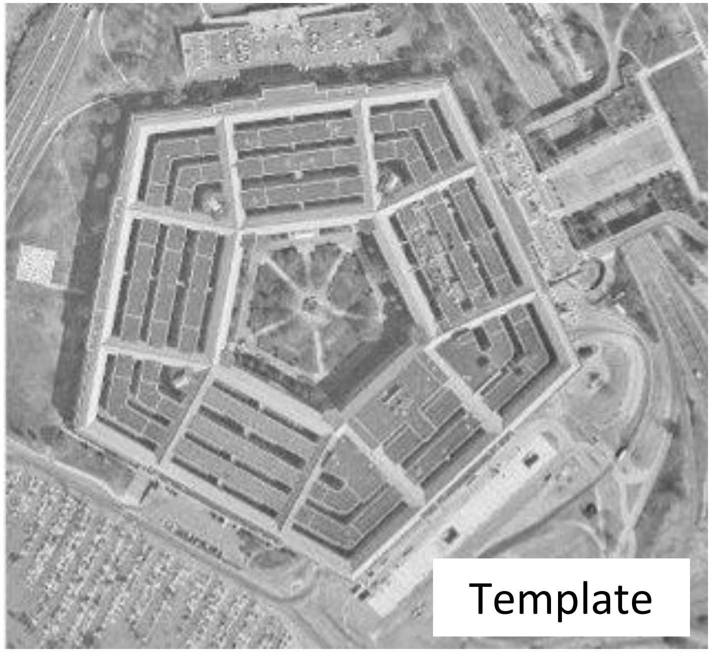
RANSAC: Pros and Cons

- **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**

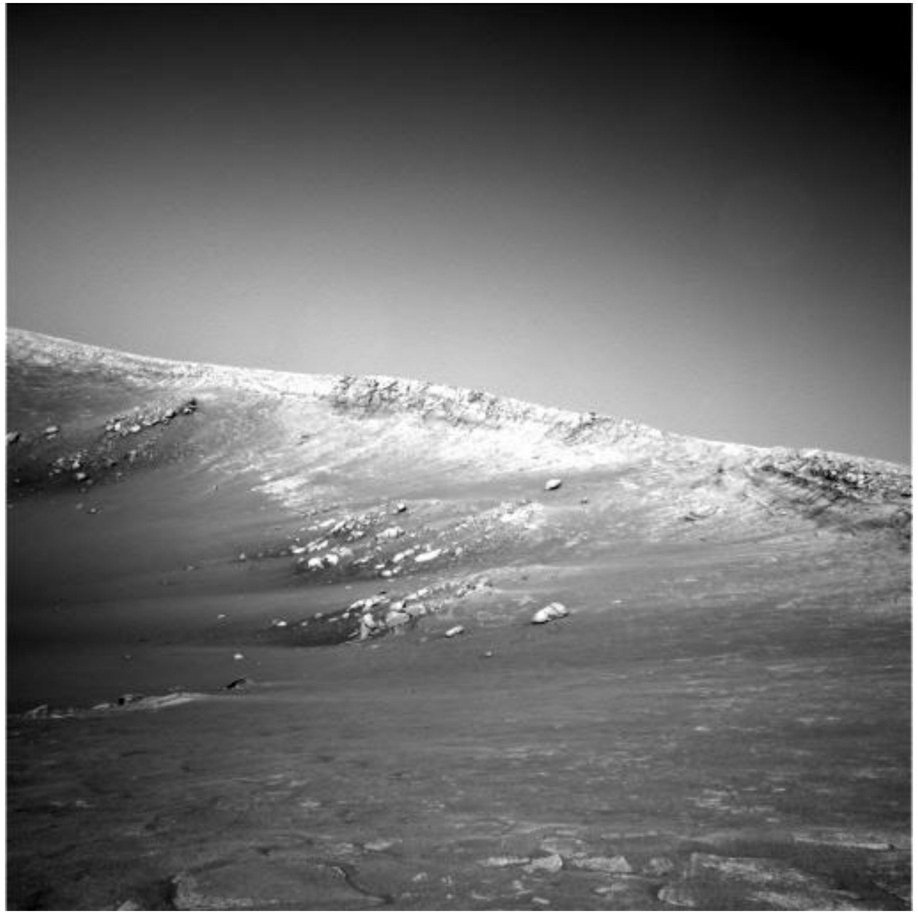
by **swashford**

Challenge: partial observability

by Diva Sian

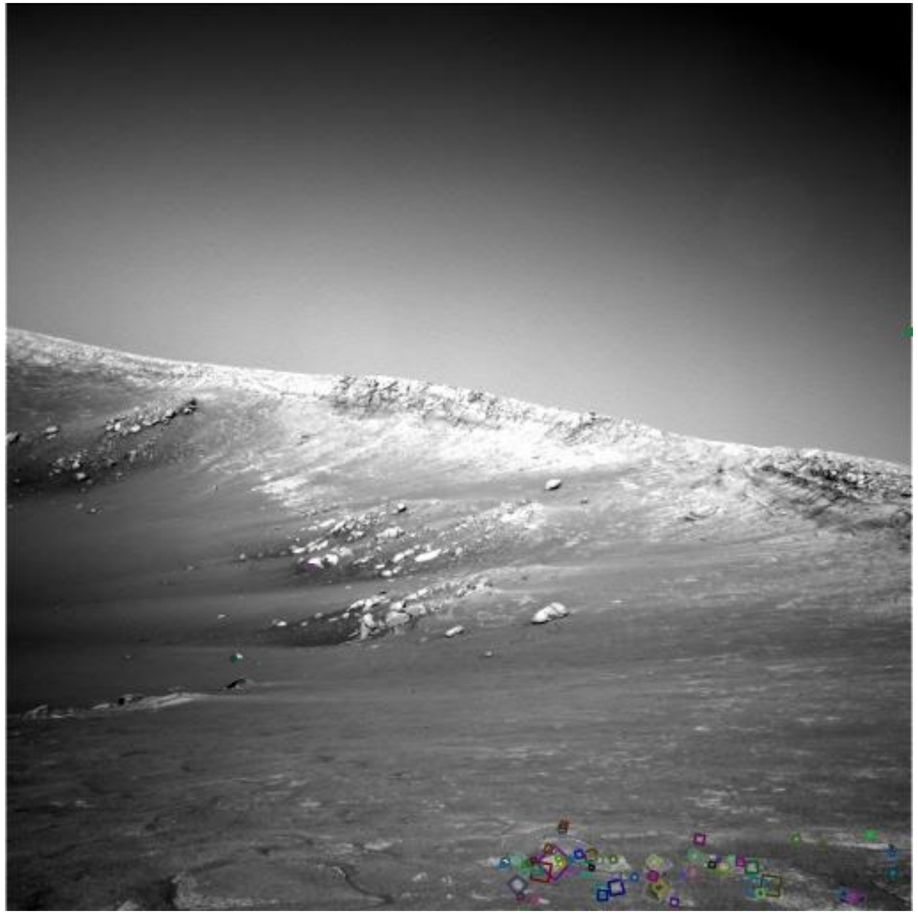
by scgbt

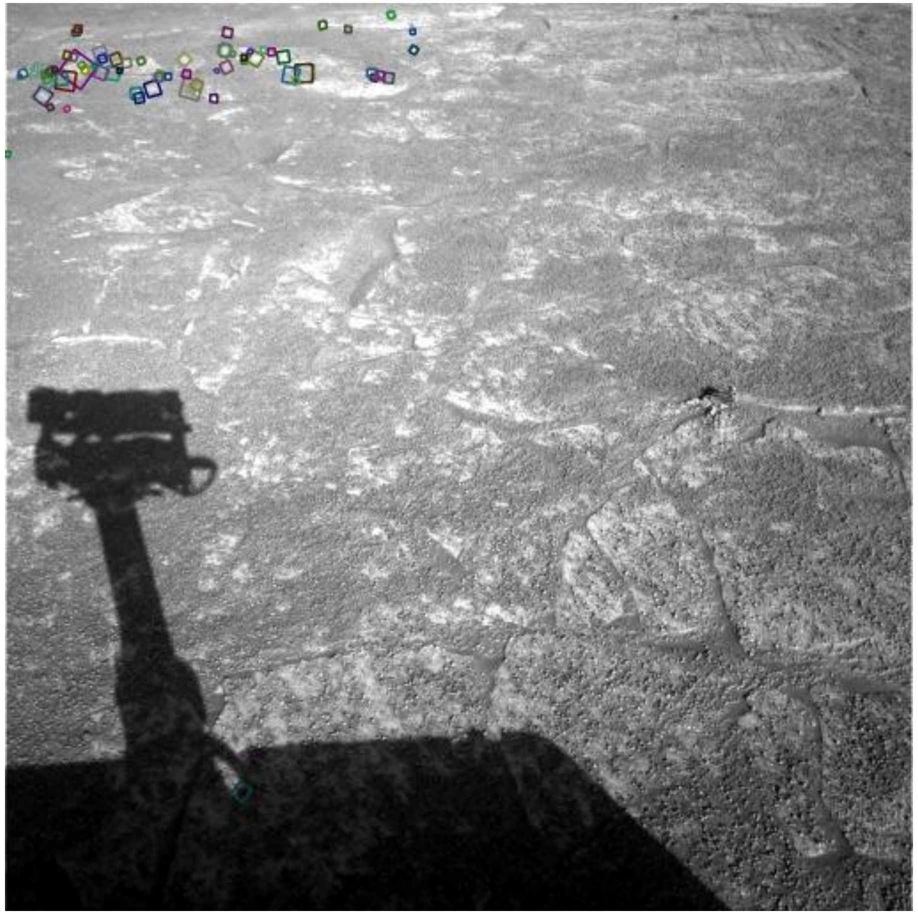
Challenge even for us



NASA Mars Rover images

Answer Below (Look for tiny colored squares)

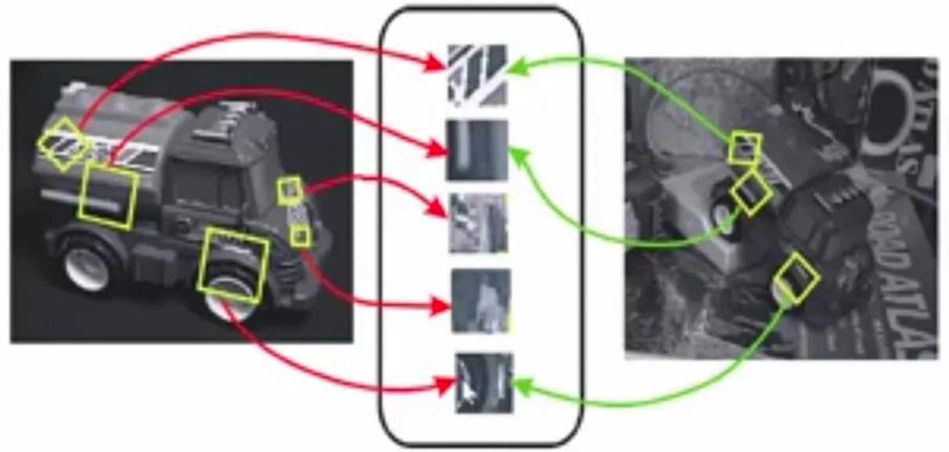




NASA Mars Rover images with SIFT feature matches
(Figure by Noah Snavely)

Intuition behind how to match images

- Find matching patches
- Check to make sure enough patches



Intuition behind how to match images

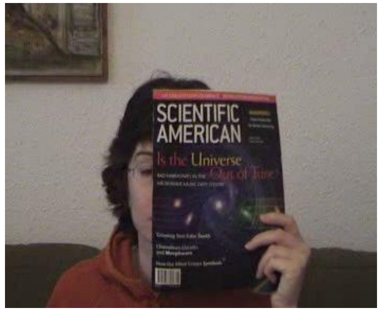
- Find matching patches
- Check to make sure enough patches

What do we need?

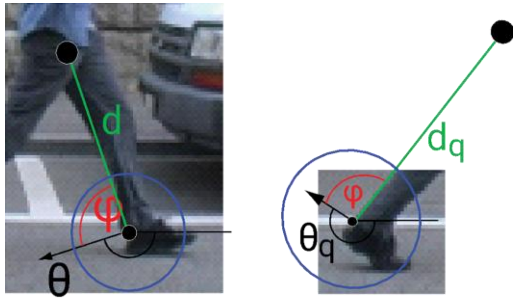
- We need to identify patches
- We need to learn to a way to describe each patch
- We need an algorithm to match the description between two patches

Motivation for using local features

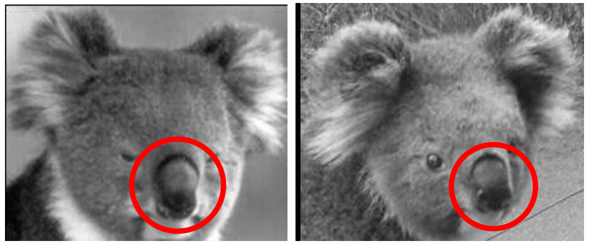
- Matching large patches have major challenges (mentioned in previous slides)
- Instead, let’s describe and match only local image patches
- Smaller, local patches are more likely to find an object even if it is partially occluded (covered)



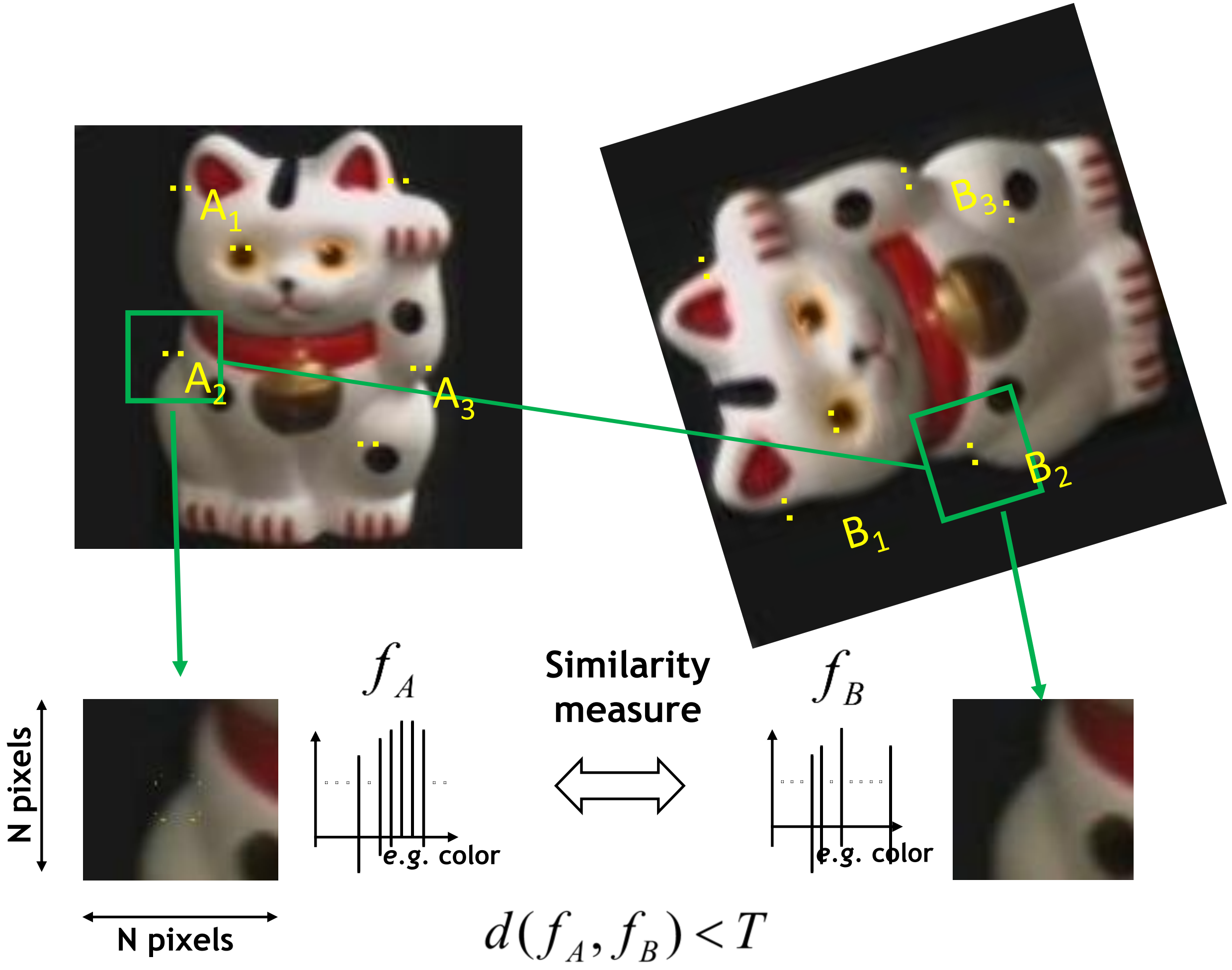
- Articulation



- Intra-category variations



General approach for search



1. Find a set of distinctive keypoints

2. Define a region/patch around each keypoint

3. Normalize the region content

4. Compute a local descriptor from the normalized region

5. Match local descriptors

Common Requirements

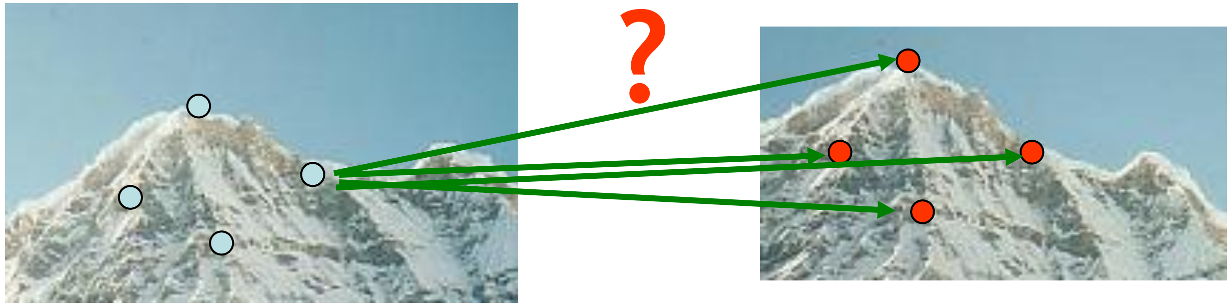
- Problem 1: How should we choose the keypoints?
  - We want to detect the same points independently in both images

No chance to match if the keypoints aren't the same

We need a repeatable detector!

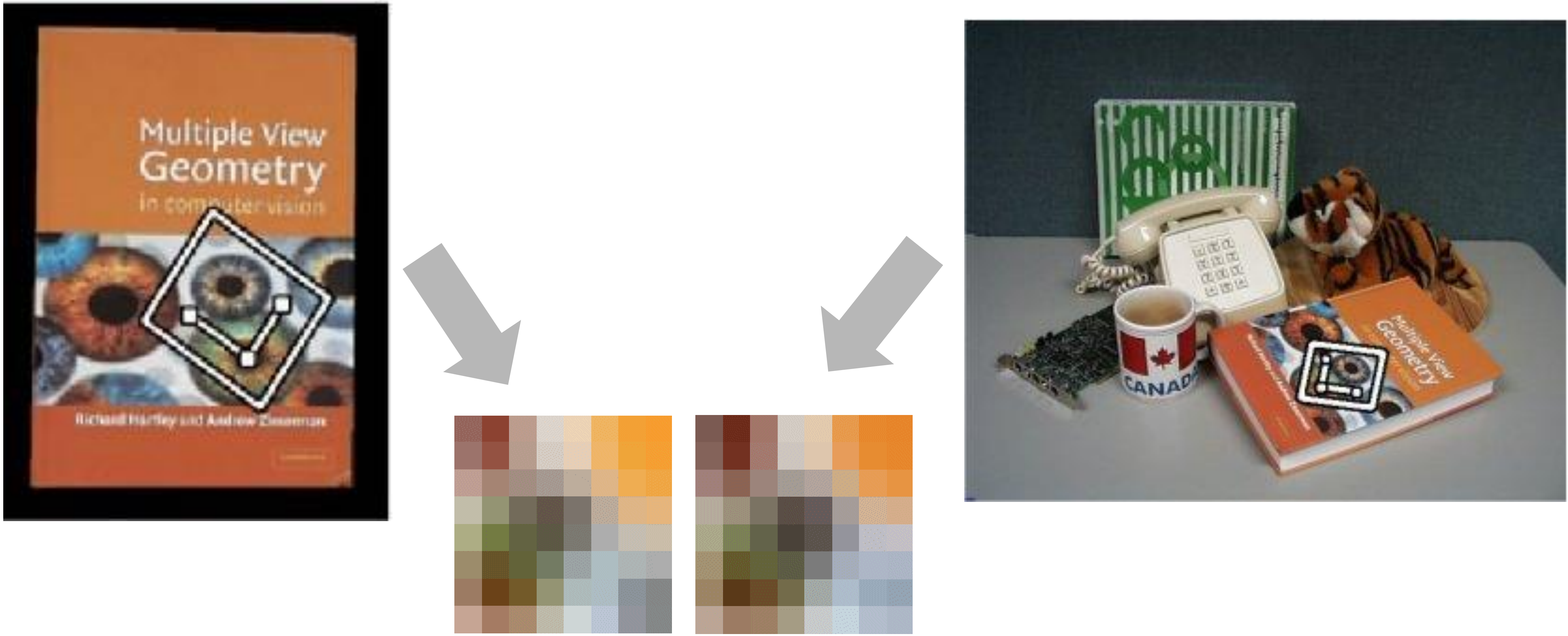
Common Requirements

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

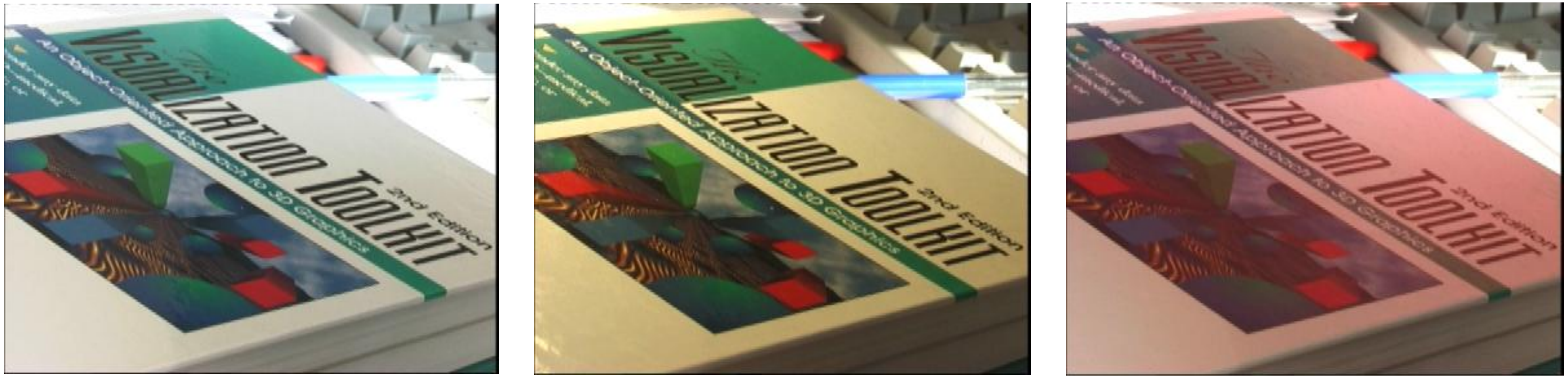


We need a reliable and distinctive descriptor!

Descriptions should be invariant to rotation and translation



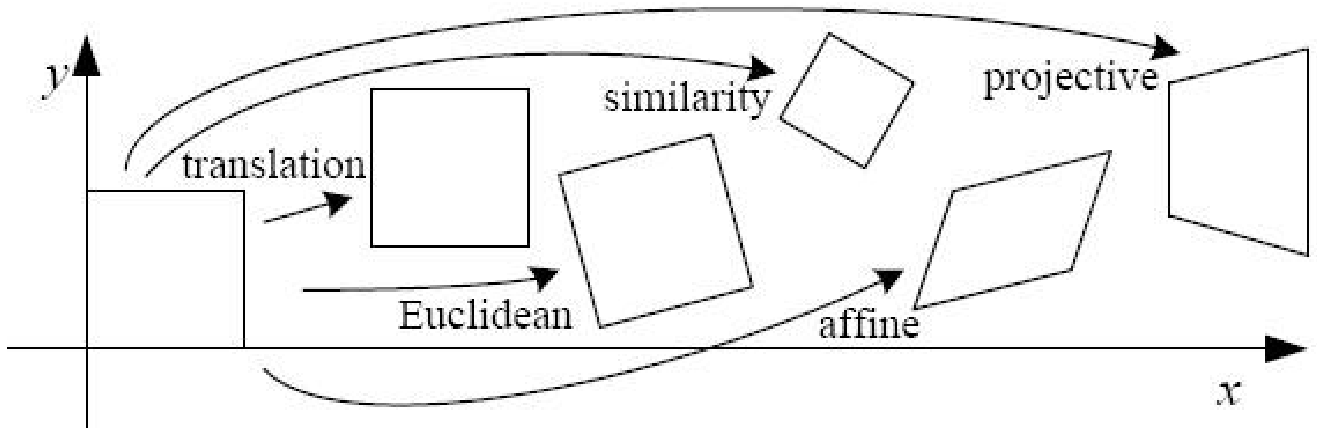
Descriptions should be invariant to photometric transformations

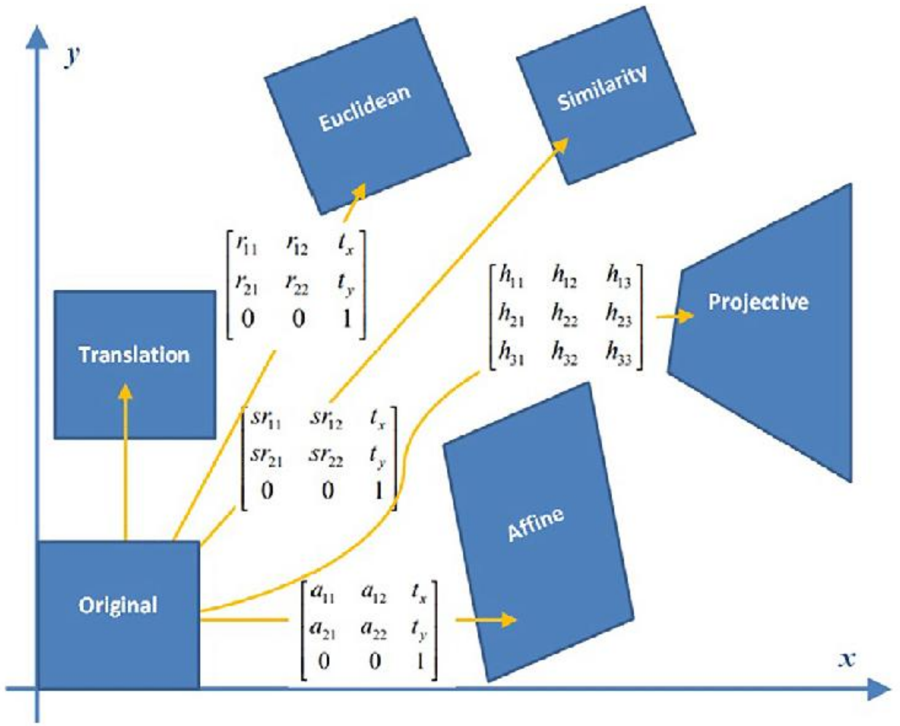


- Often modeled as a linear transformation:
    - Scaling + Offset

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 (≈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.

What are good patches?

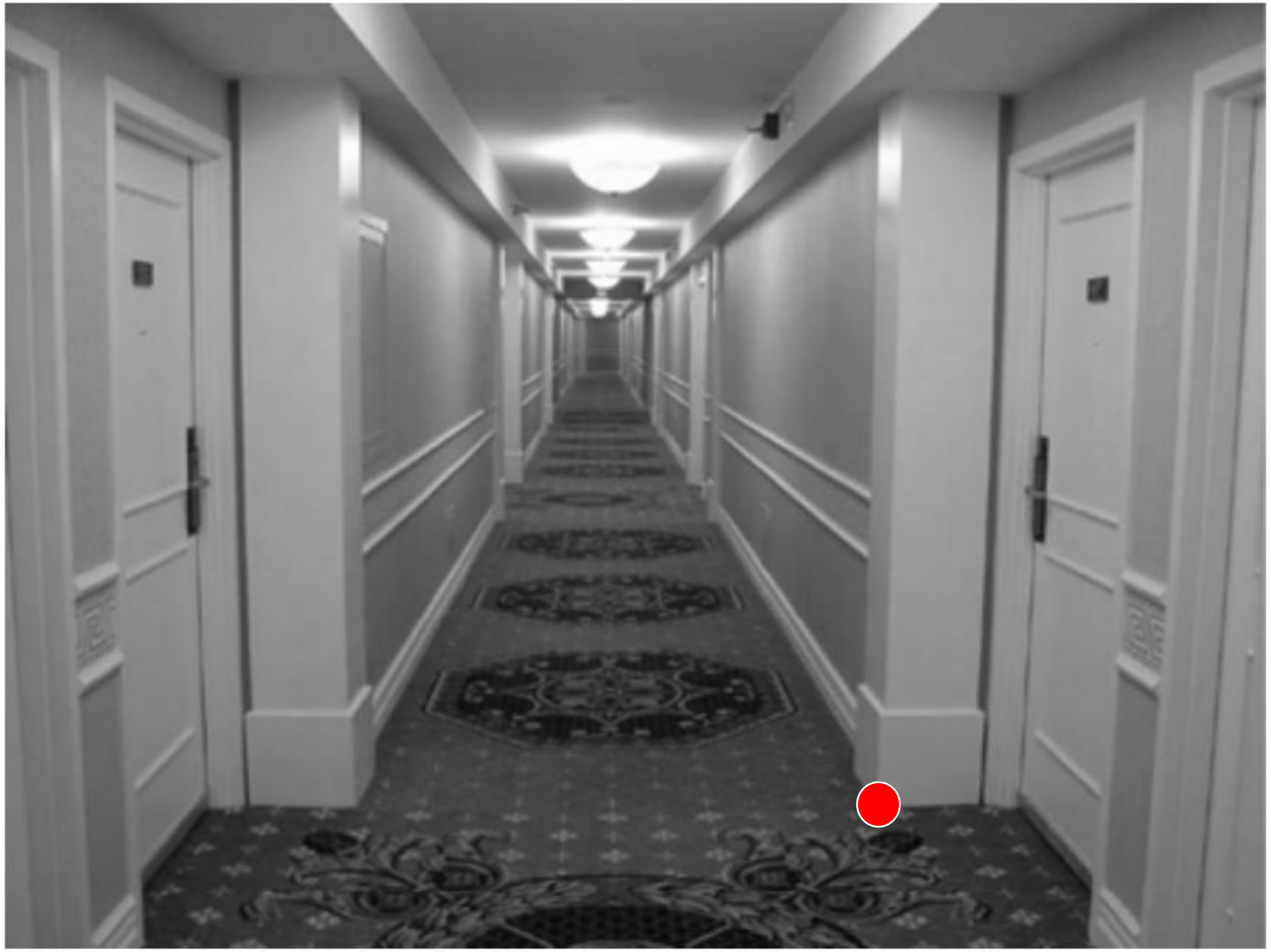
Q. Is this a good patch for image matching?

What are good patches?

Q. What about this one?

What are good patches?

Q. Let’s try another one?



Many existing feature detectors available

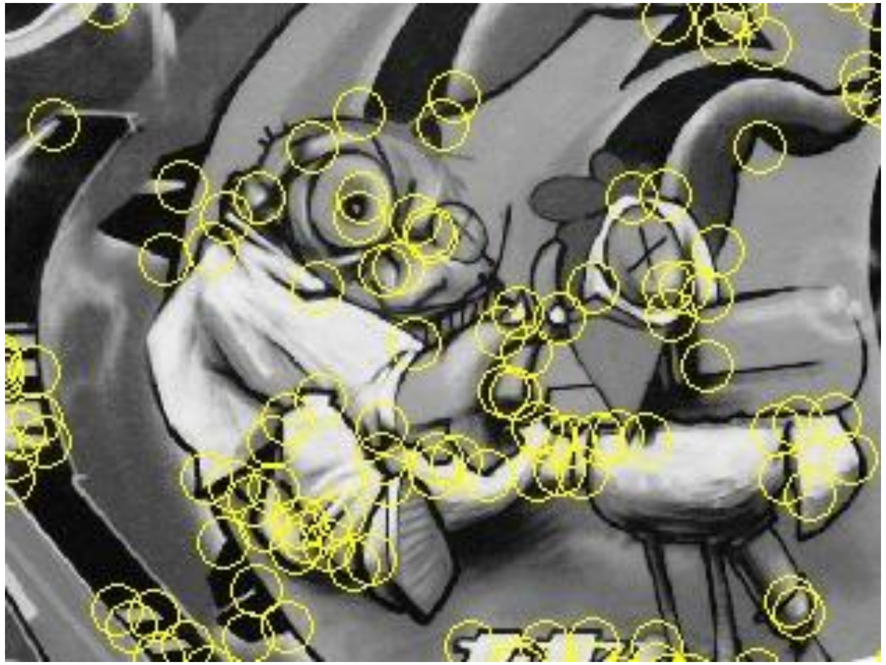
- Hessian & Harris [Beaudet '78], [Harris '88]
- Laplacian, DoG [Lindeberg '98], [Lowe '99]
- Harris-/Hessian-Laplace [Mikolajczyk & Schmid '01]
- Harris-/Hessian-Affine [Mikolajczyk & Schmid '04]
- EBR and IBR [Tuytelaars & Van Gool '04]
- MSER [Matas '02]
- Salient Regions [Kadir & Brady '01]
- Neural networks [Krichevsky '12]

- *Those detectors have become a basic building block for many applications in Computer Vision.*

Today’s agenda

- Local Invariant Features
- Harris Corner Detector

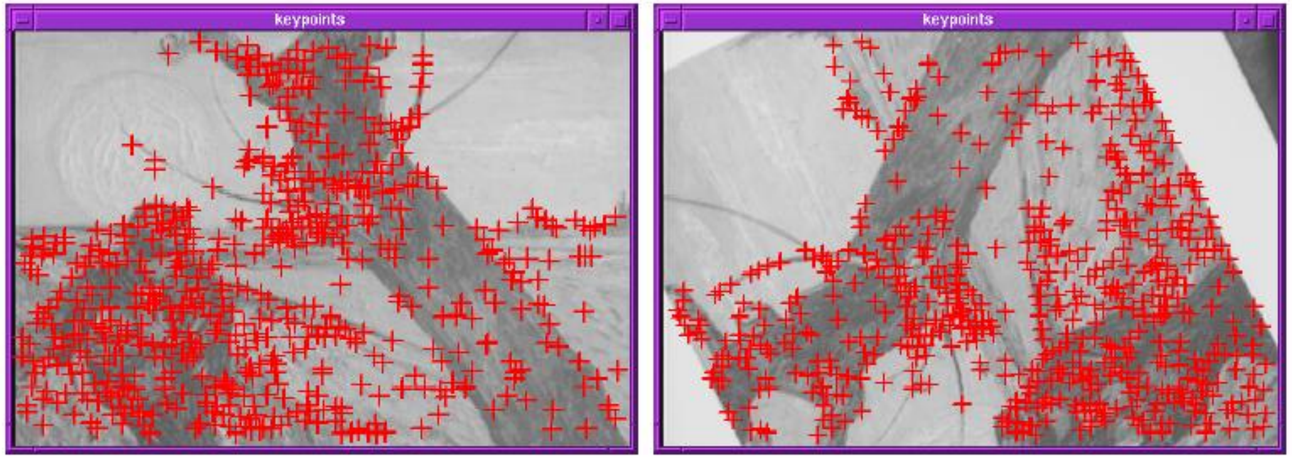
Keypoint Localization



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

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

Finding Corners



How do we find corners using LSI systems?

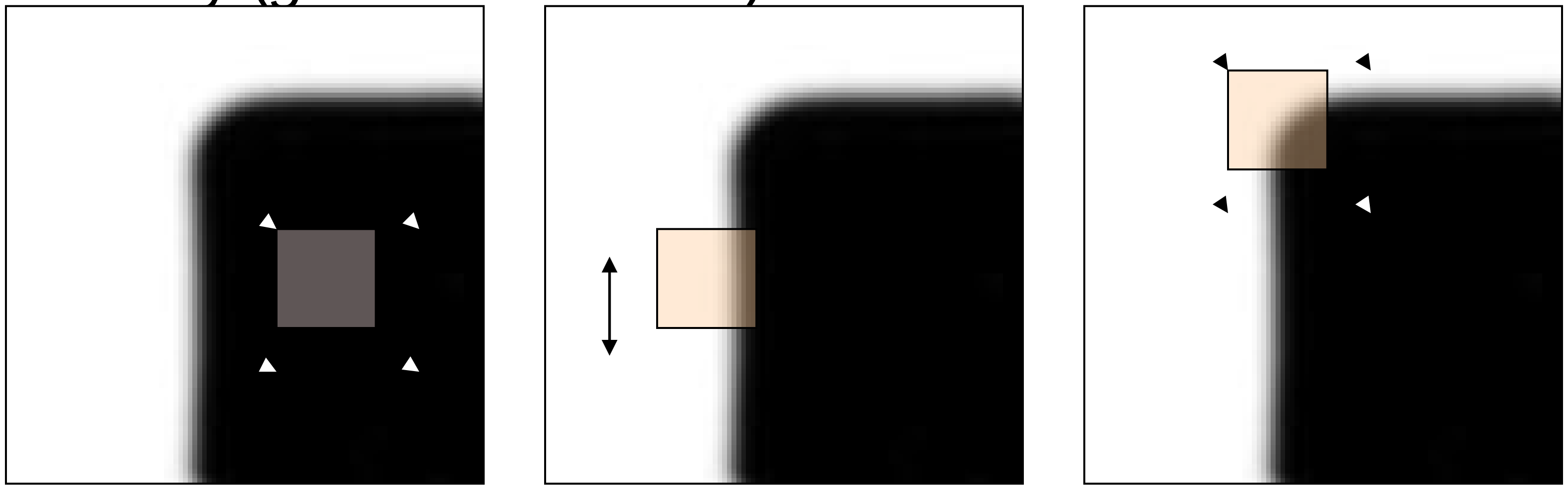
- The image gradient around a corner has two or more dominant directions

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

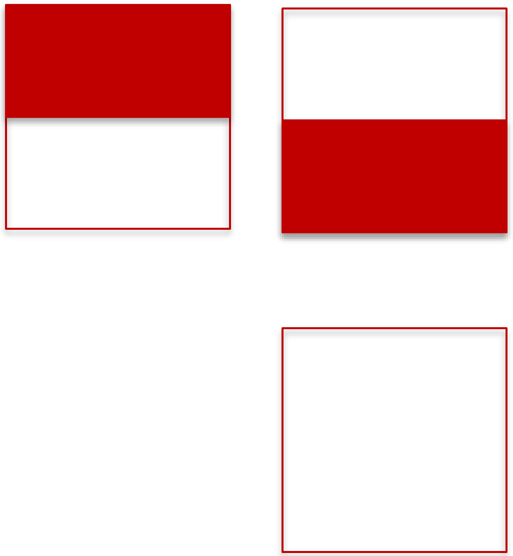
Flat patches have small image gradients

$\sum I_x^2 \longrightarrow$ Small

$\sum I_y^2 \longrightarrow$ Small

Flat

Edges have high gradient in one direction



$\sum I_x^2$ → Small

$\sum I_y^2$ → Large

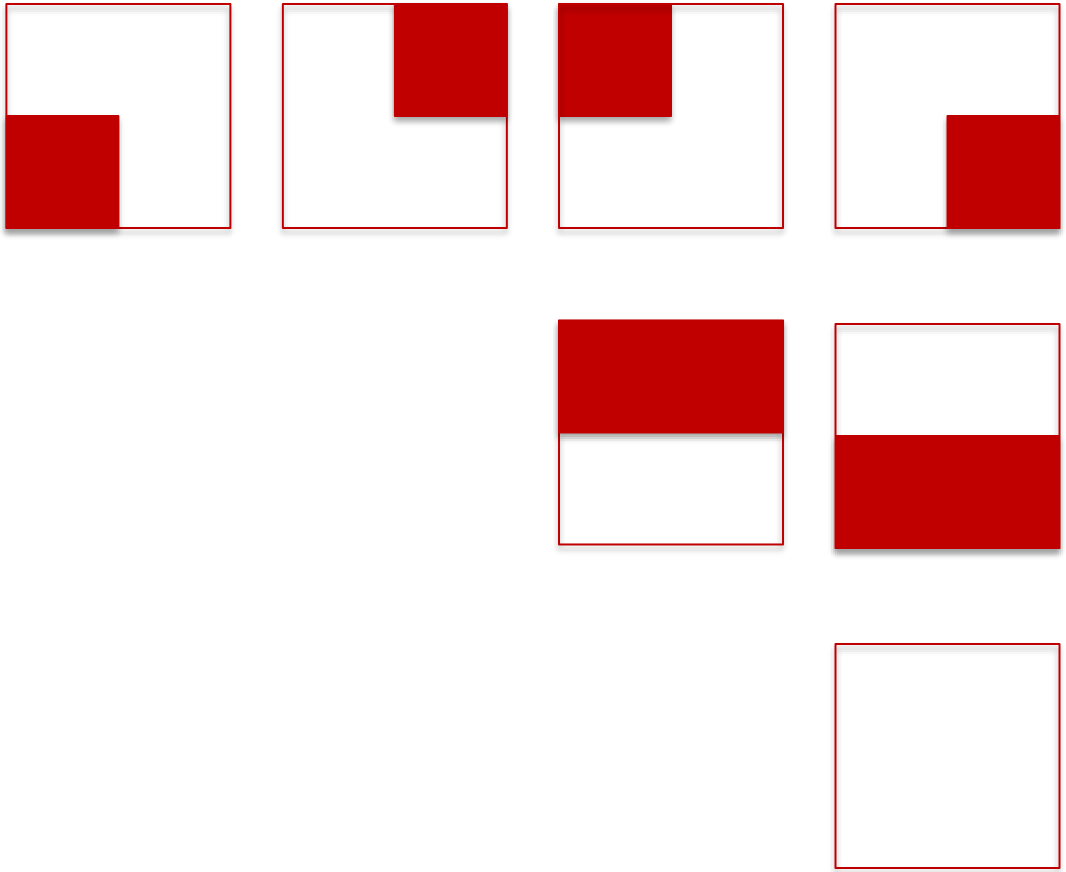
Edge

$\sum I_x^2$ → Small

$\sum I_y^2$ → Small

Flat

Corners versus edges



$\sum I_x^2$ → Large
$\sum I_y^2$ → Large

Corner

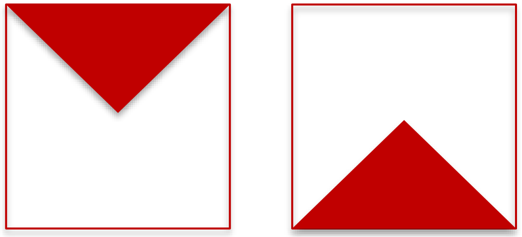
$\sum I_x^2$ → Small
$\sum I_y^2$ → Large

Edge

$\sum I_x^2$ → Small
$\sum I_y^2$ → Small

Flat

Generalizing to corners in any direction



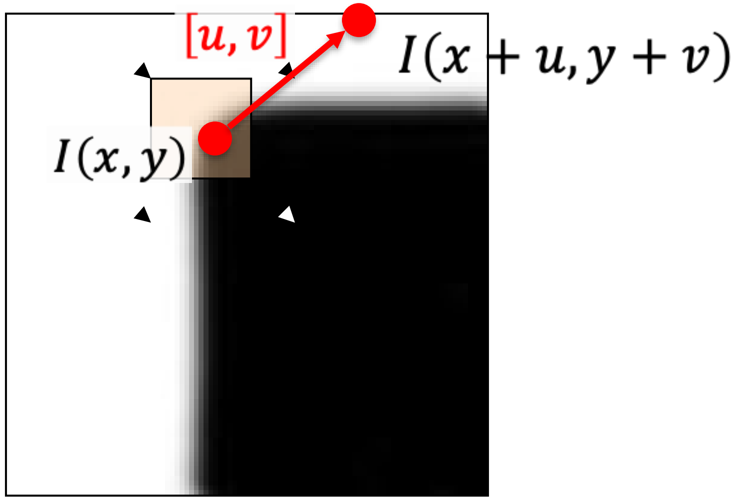
$$\sum I_x^2 \longrightarrow ??$$

$$\sum I_y^2 \longrightarrow ??$$

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:

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

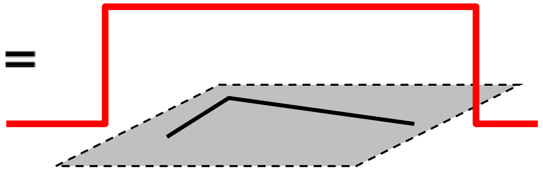
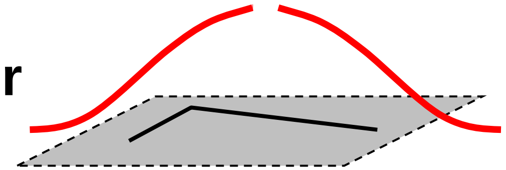
Shifted intensity

Intensity

Window function

Sum over window

Intensity change

Window function $w(x,y) =$  or 

1 in window, 0 outside

Gaussian

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 (truncated):

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

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:

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

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

Re-writing E:

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

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

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

Re-writing E:

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

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

$$= (\sum_{x,y} wI_x^2)u^2 + 2(\sum_{x,y} wI_x I_y)uv + (\sum_{x,y} wI_y^2)v^2$$

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

Re-writing E:

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

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

$$= (\sum_{x,y} wI_x^2)u^2 + 2(\sum_{x,y} wI_x I_y)uv + (\sum_{x,y} wI_y^2)v^2$$

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

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

where:

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

Simplifying M for a second:

Assuming $w(x,y)=1$

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

where:

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

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

Re-writing E:

$$E(u,v) = \boxed{\begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix}}$$

Does anyone know what this part of the equation is?

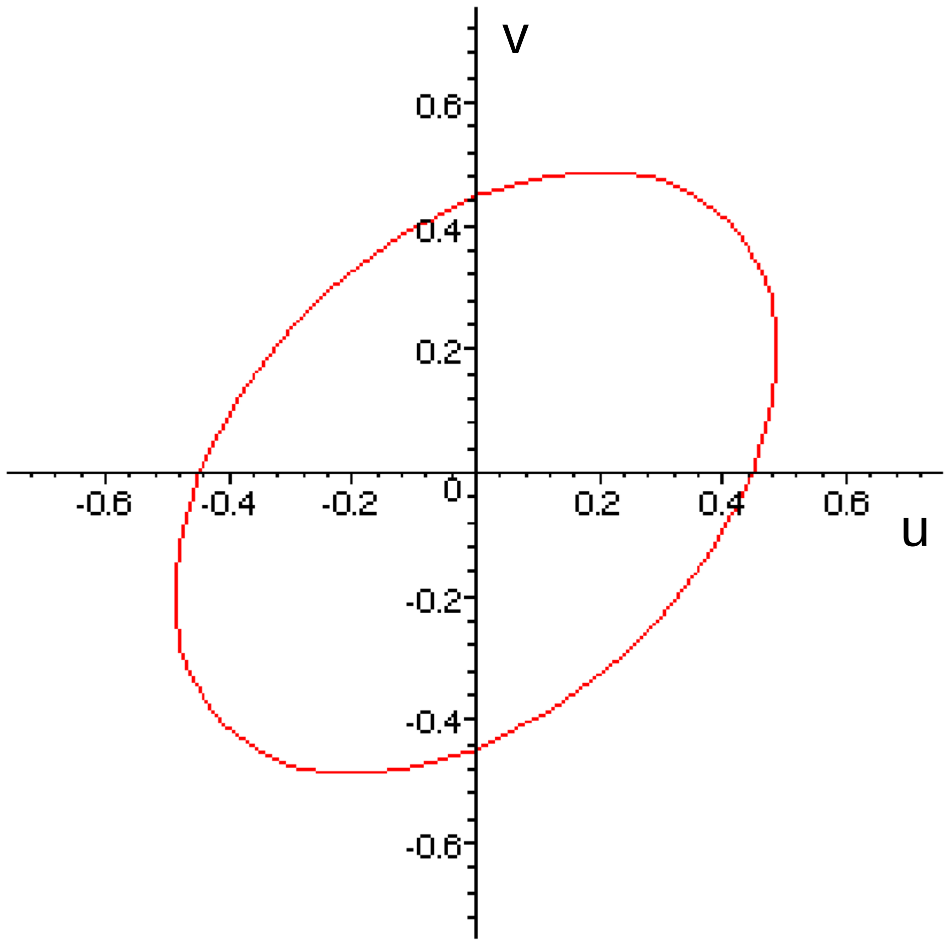
$$M = \begin{bmatrix} \sum_{x,y} I_x^2 & \sum_{x,y} I_x I_y \\ \sum_{x,y} I_x I_y & \sum_{x,y} I_y^2 \end{bmatrix}$$

It’s the equation of an ellipse

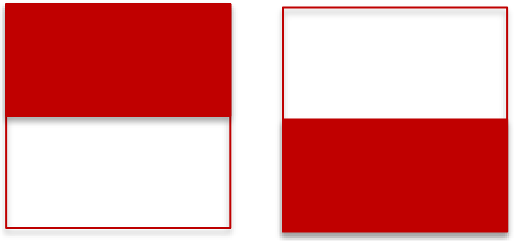
$$5u^2 - 4uv + 5v^2 = 1$$

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

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



Remember what we said about the gradients for these edges



$\sum I_x^2 \longrightarrow$ Small

$\sum I_y^2 \longrightarrow$ Large

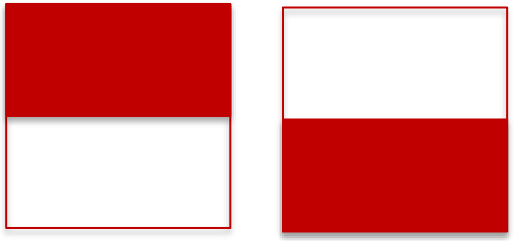
Edge

If only $\sum I_x^2 \longrightarrow$ Large (opposite picture, vertical edge) ,

Q. What is the matrix M going to look like?

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

Remember what we said about the gradients for these edges



$\sum I_x^2 \longrightarrow$ Small

$\sum I_y^2 \longrightarrow$ Large

Edge

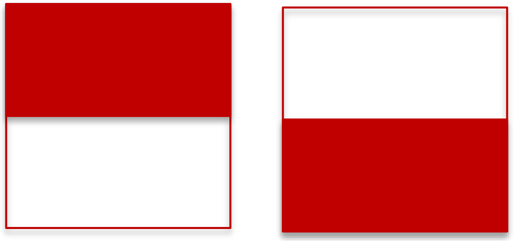
If only $\sum I_x^2 \longrightarrow$ Large ,

Q. What is the matrix M going to look like?

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

$$M = \begin{bmatrix} \text{Large} & \text{Small} \\ \text{Small} & \text{Small} \end{bmatrix}$$

Remember what we said about the gradients for these corners



$\sum I_x^2 \longrightarrow$ Small

$\sum I_y^2 \longrightarrow$ Large

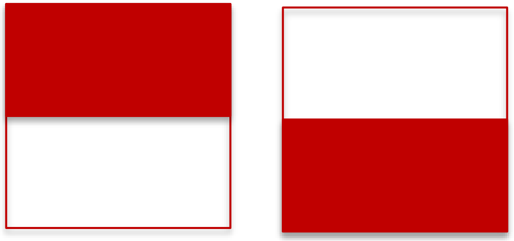
Edge

If only $\sum I_x^2 \longrightarrow$ Large , Q. what kind of ellipse would you expect to see?

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

$$M = \begin{bmatrix} \text{Large} & \text{Small} \\ \text{Small} & \text{Small} \end{bmatrix}$$

Remember what we said about the gradients for these corners

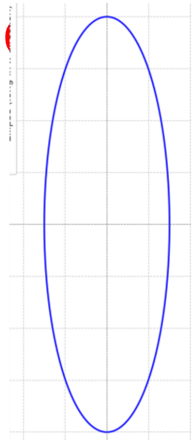


$\sum I_x^2$ → Small

$\sum I_y^2$ → Large

Edge

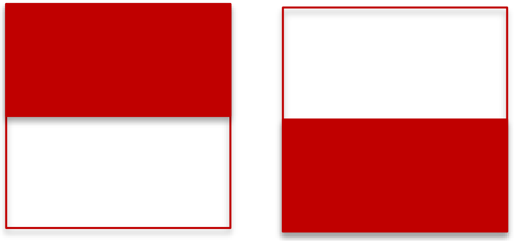
If only $\sum I_x^2$ → Large , Q. what kind of ellipse would you expect to se



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

$$M = \begin{bmatrix} \text{Large} & \text{Small} \\ \text{Small} & \text{Small} \end{bmatrix}$$

Remember what we said about the gradients for these corners



$\sum I_x^2$ → Small

$\sum I_y^2$ → Large

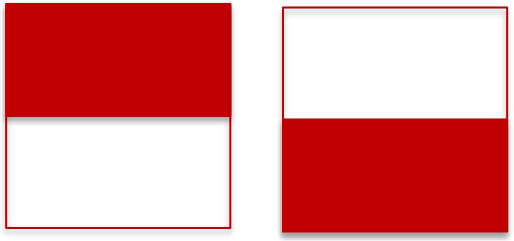
Edge

If only $\sum I_y^2$ → Large, Q. what kind of ellipse would you expect to see?

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

$$M = \begin{bmatrix} \text{Small} & \text{Small} \\ \text{Small} & \text{Large} \end{bmatrix}$$

Remember what we said about the gradients for these corners

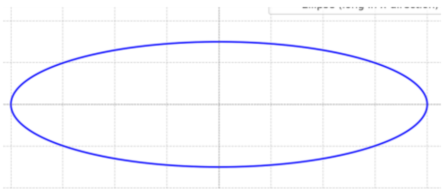


$\sum I_x^2 \longrightarrow$ Small

$\sum I_y^2 \longrightarrow$ Large

Edge

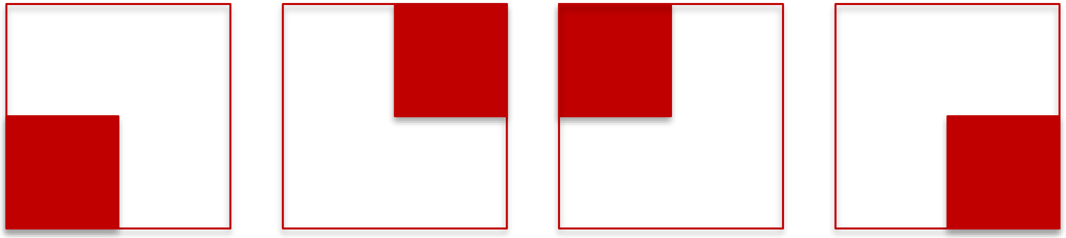
If only $\sum I_y^2 \longrightarrow$ Large , Q. what kind of ellipse would you expect to see?



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

$$M = \begin{bmatrix} \text{Small} & \text{Small} \\ \text{Small} & \text{Large} \end{bmatrix}$$

Remember what we said about the gradients for these corners



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

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

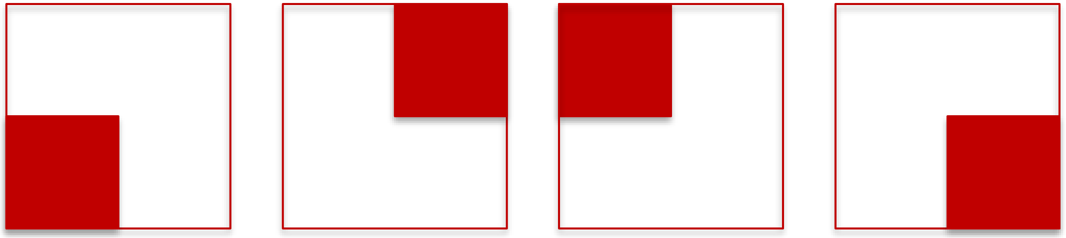
Corner

Q. What is the matrix M going to look like?

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

$$M = \begin{bmatrix} ??? & ??? \\ ??? & ??? \end{bmatrix}$$

Remember what we said about the gradients for these corners



$\sum I_x^2 \longrightarrow$ Large

$\sum I_y^2 \longrightarrow$ Large

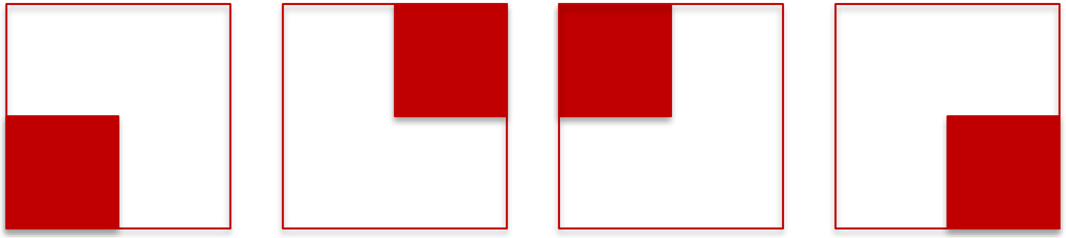
Corner

Q. What is the matrix M going to look like?

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

$$M = \begin{bmatrix} \text{Large} & \text{small} \\ \text{small} & \text{Large} \end{bmatrix}$$

Remember what we said about the gradients for these corners

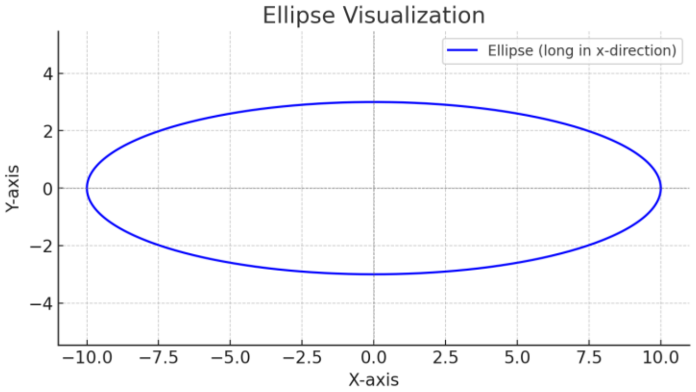


$\sum I_x^2 \longrightarrow$ Large

$\sum I_y^2 \longrightarrow$ Large

Corner

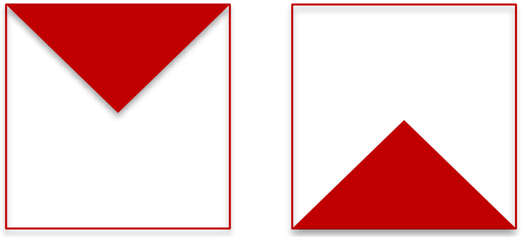
Q. What is the ellipse going to look like?



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

$$M = \begin{bmatrix} \text{Large} & \text{small} \\ \text{small} & \text{Large} \end{bmatrix}$$

But what about these ones?



$$\sum I_x^2 \longrightarrow ??$$

$$\sum I_y^2 \longrightarrow ??$$

Corner

Q. What would the matrix and ellipses look like?

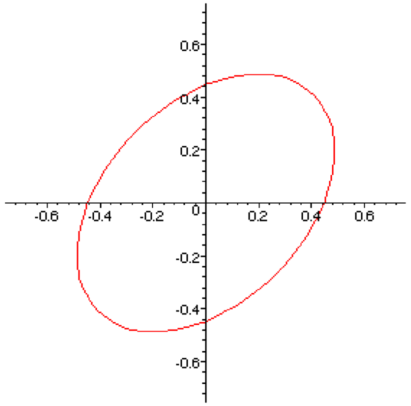
Hint:

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

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

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

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



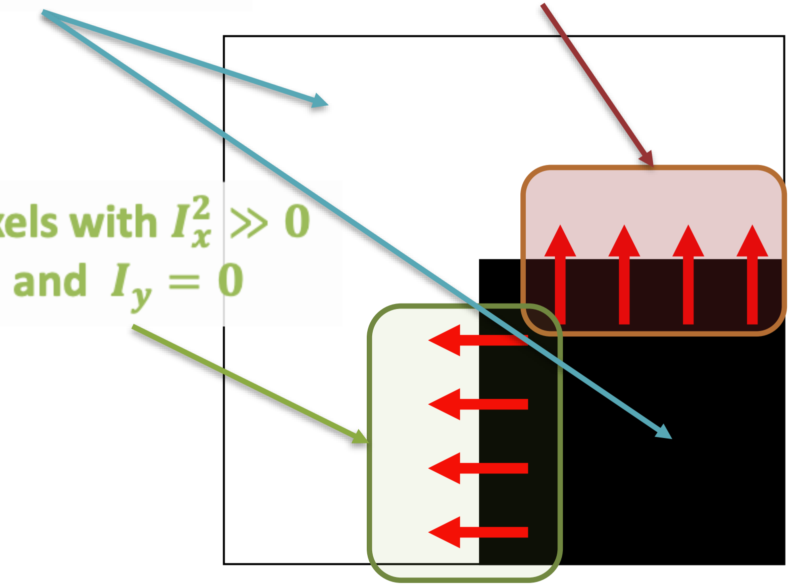
What Does This Matrix Reveal?

- 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?

Pixels with $I_x = 0$ and $I_y = 0$

Pixels with $I_y^2 \gg 0$ and $I_x = 0$

Pixels with $I_x^2 \gg 0$ and $I_y = 0$

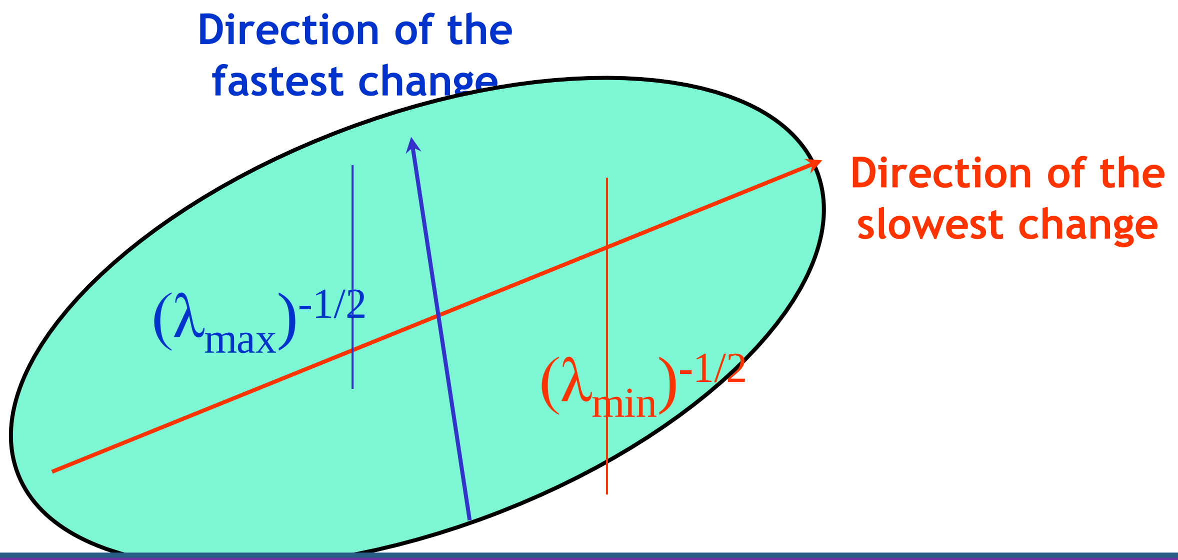


M defines an ellipse in the direction (u, v)

- 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)

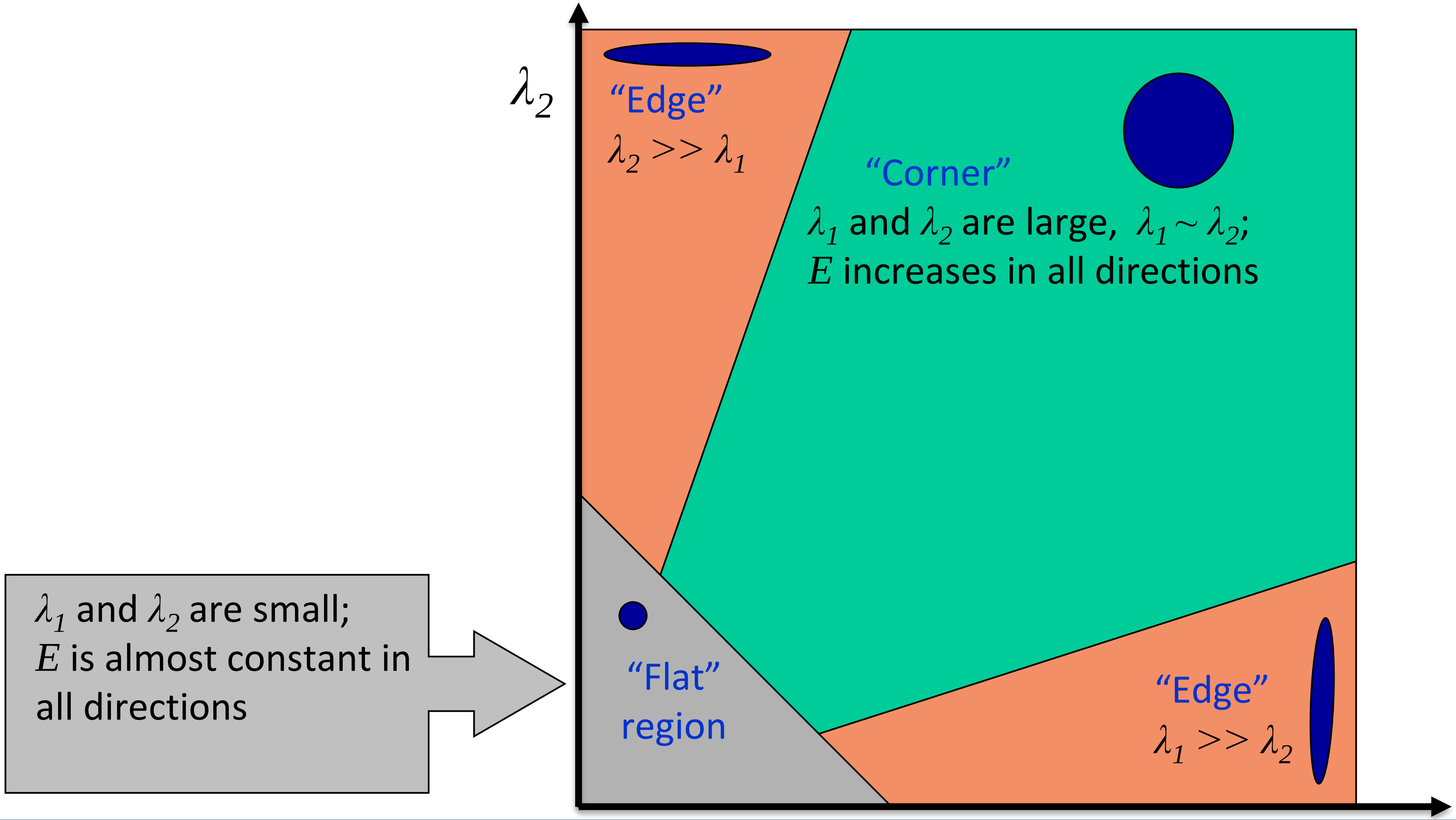
- We can think of M as an ellipse with its axis lengths determined by the eigenvalues λ_1 and λ_2; and its orientation determined by R



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

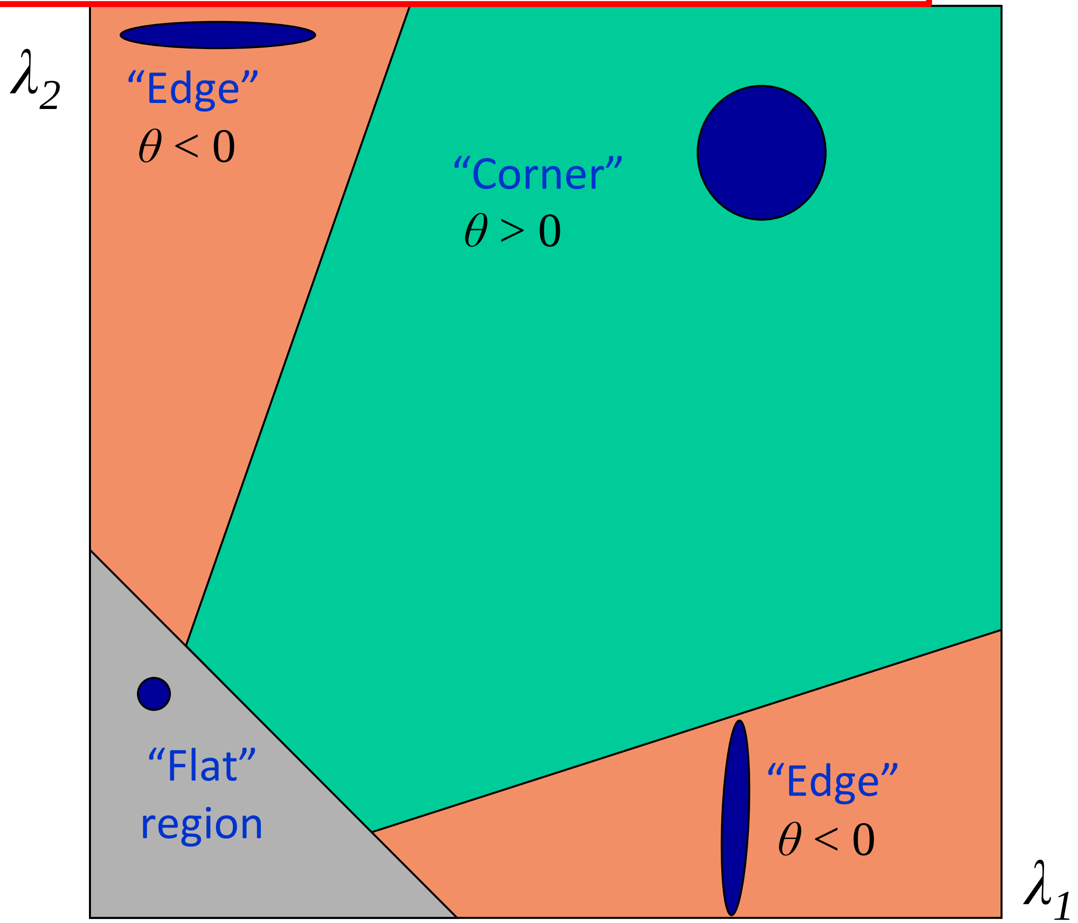
Interpreting the Eigenvalues

- Classification of image points using eigenvalues of M:



But calculating eigenvalues is expensive.
Solution: Corner Response Function

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



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

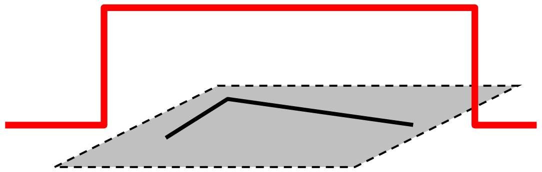
Window Function $w(x,y)$

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

- Option 1: uniform window
  - Sum over square window

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

  - Problem: not rotation invariant

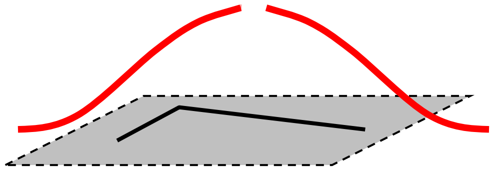


1 in window, 0 outside

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

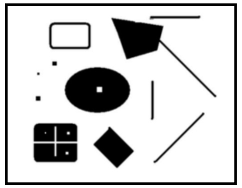
$$M = g(\sigma) * \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

  - Result is rotation invariant



Gaussi
an

Summary: Harris Detector [Harris88]



- Compute second moment matrix (autocorrelation matrix)

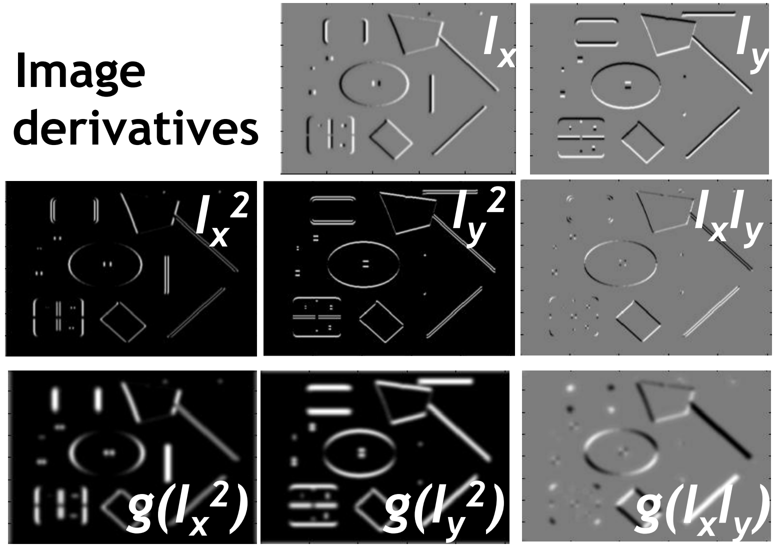
$$M(\sigma_I,\sigma_D)=g(\sigma_I)*\begin{bmatrix} I_x^2(\sigma_D) & I_xI_y(\sigma_D) \\ I_xI_y(\sigma_D) & I_y^2(\sigma_D) \end{bmatrix}$$

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

1. Image derivatives

2. Square of derivatives

3. Gaussian filter $g(\sigma_I)$



4. Cornerness function – two strong eigenvalues

$$\theta=\det[M(\sigma_I,\sigma_D)]-\alpha[\mathrm{trace}(M(\sigma_I,\sigma_D))]^2$$
$$=g(I_x^2)g(I_y^2)-[g(I_xI_y)]^2-\alpha[g(I_x^2)+g(I_y^2)]^2$$

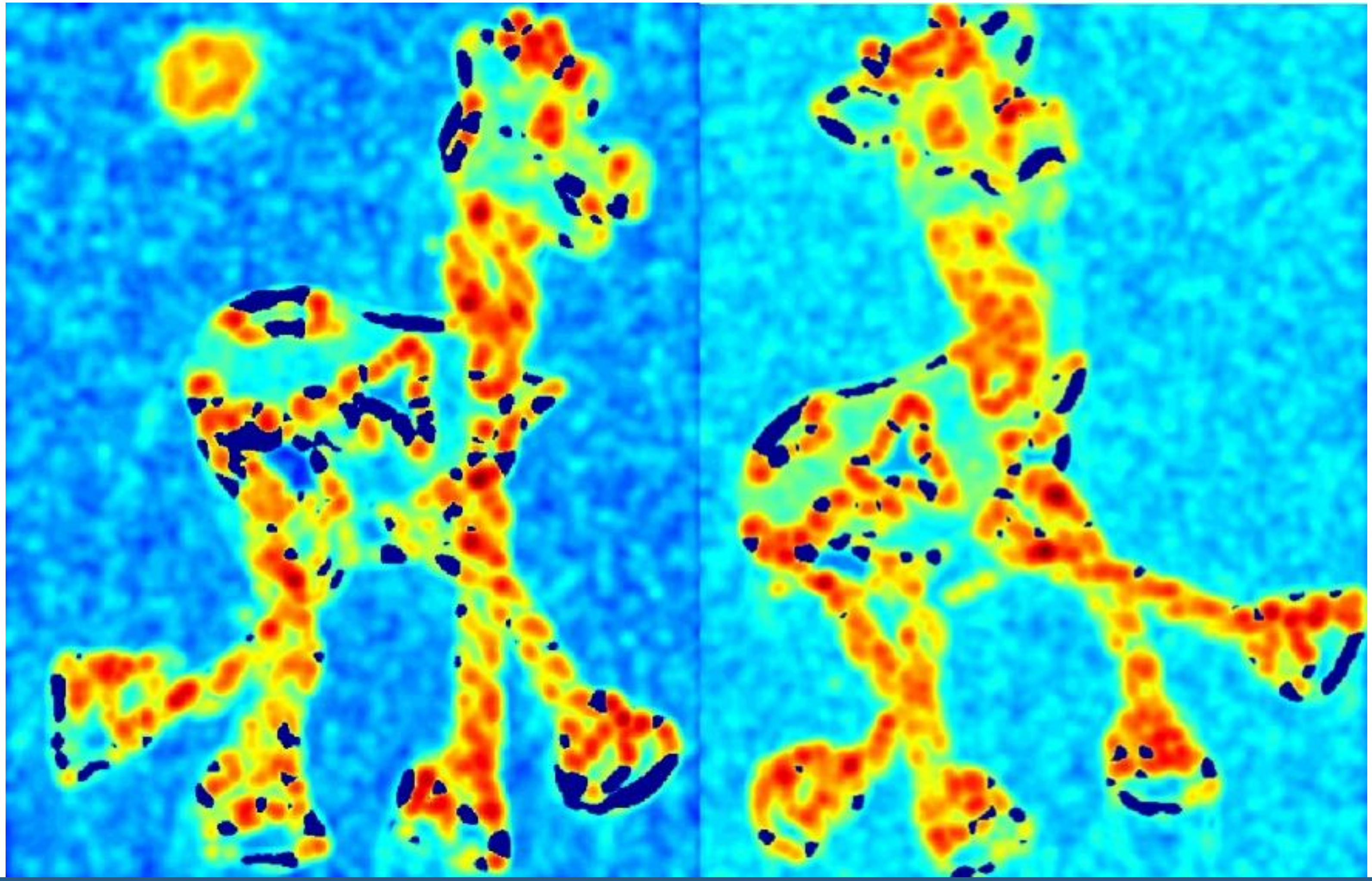
5. Perform non-maximum suppression

Harris Detector: Example

- Input Image

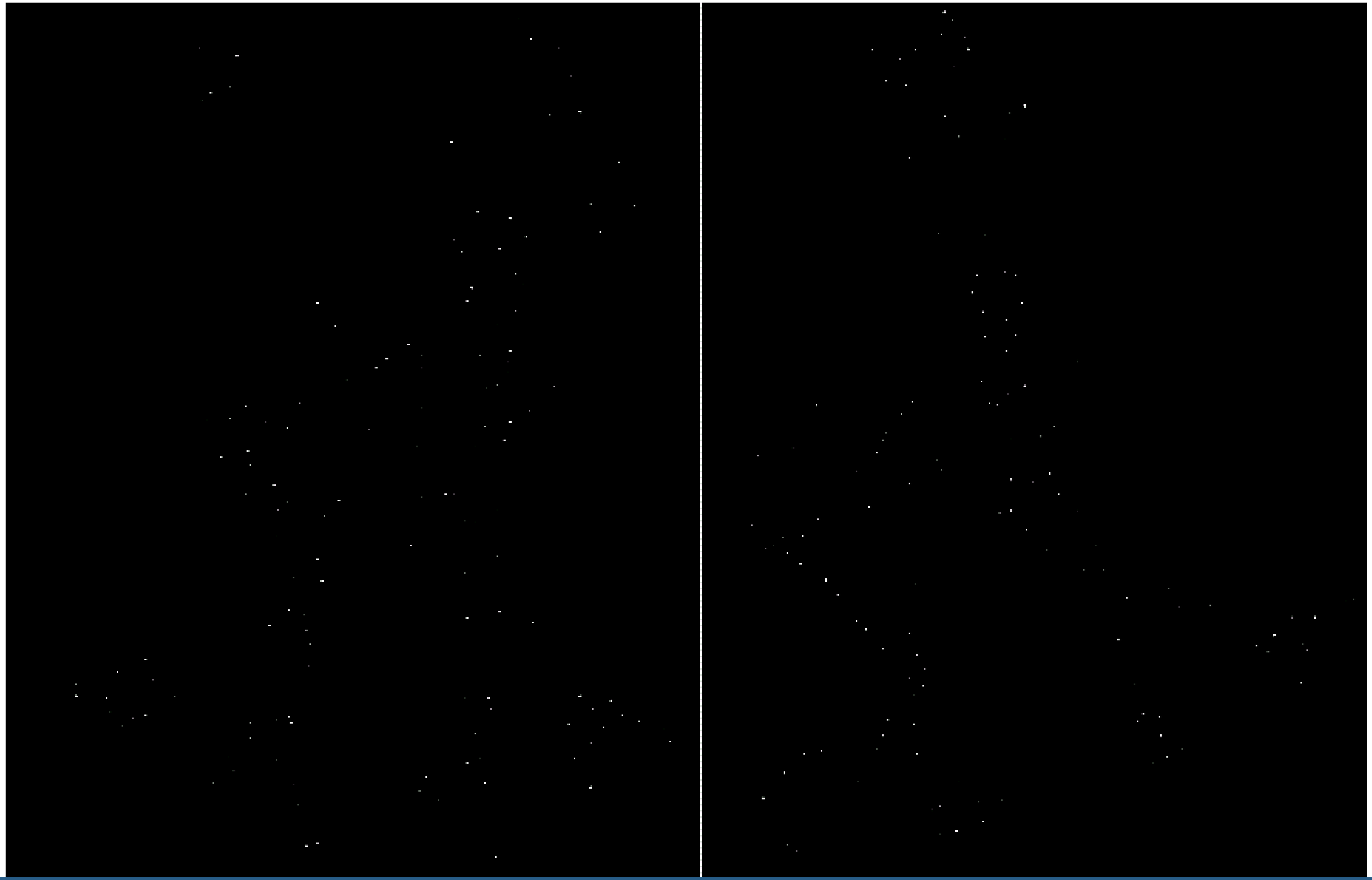
Harris Detector: Example

- Input Image
- Compute corner response function θ



Harris Detector: Example

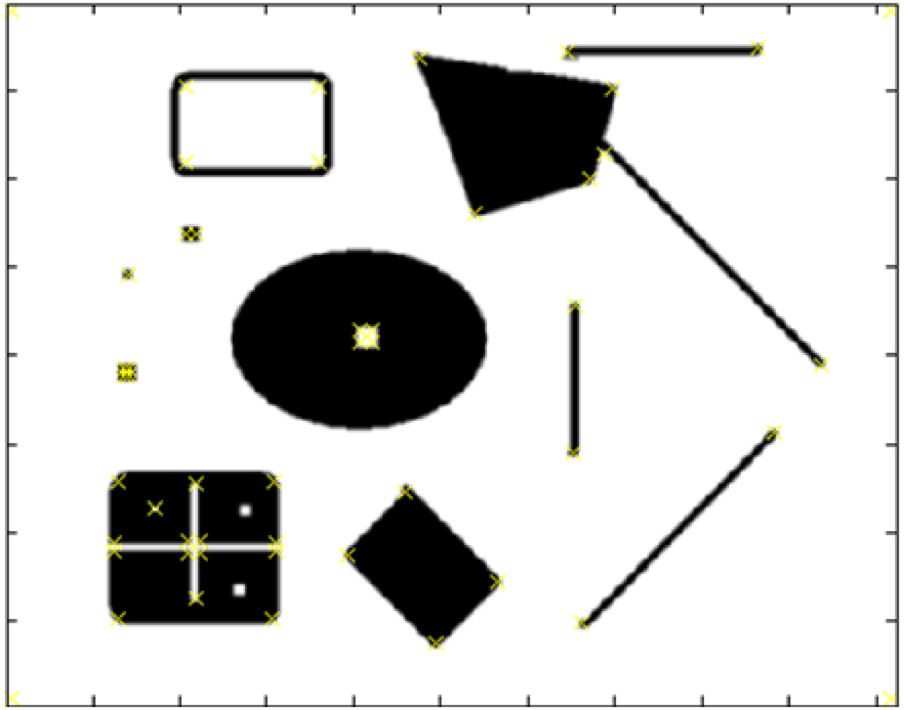
- Input Image
- Compute corner response function θ
- Take only the local maxima of θ, where θ > threshold



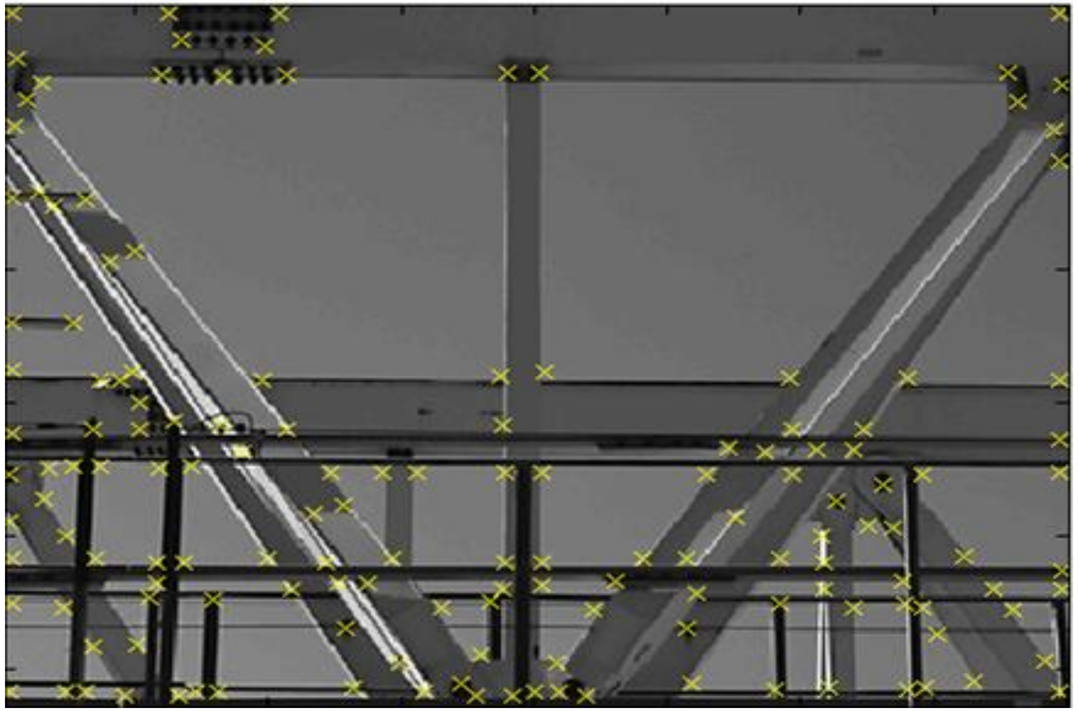
Harris Detector: Example

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

Harris Detector – Responses [Harris88]

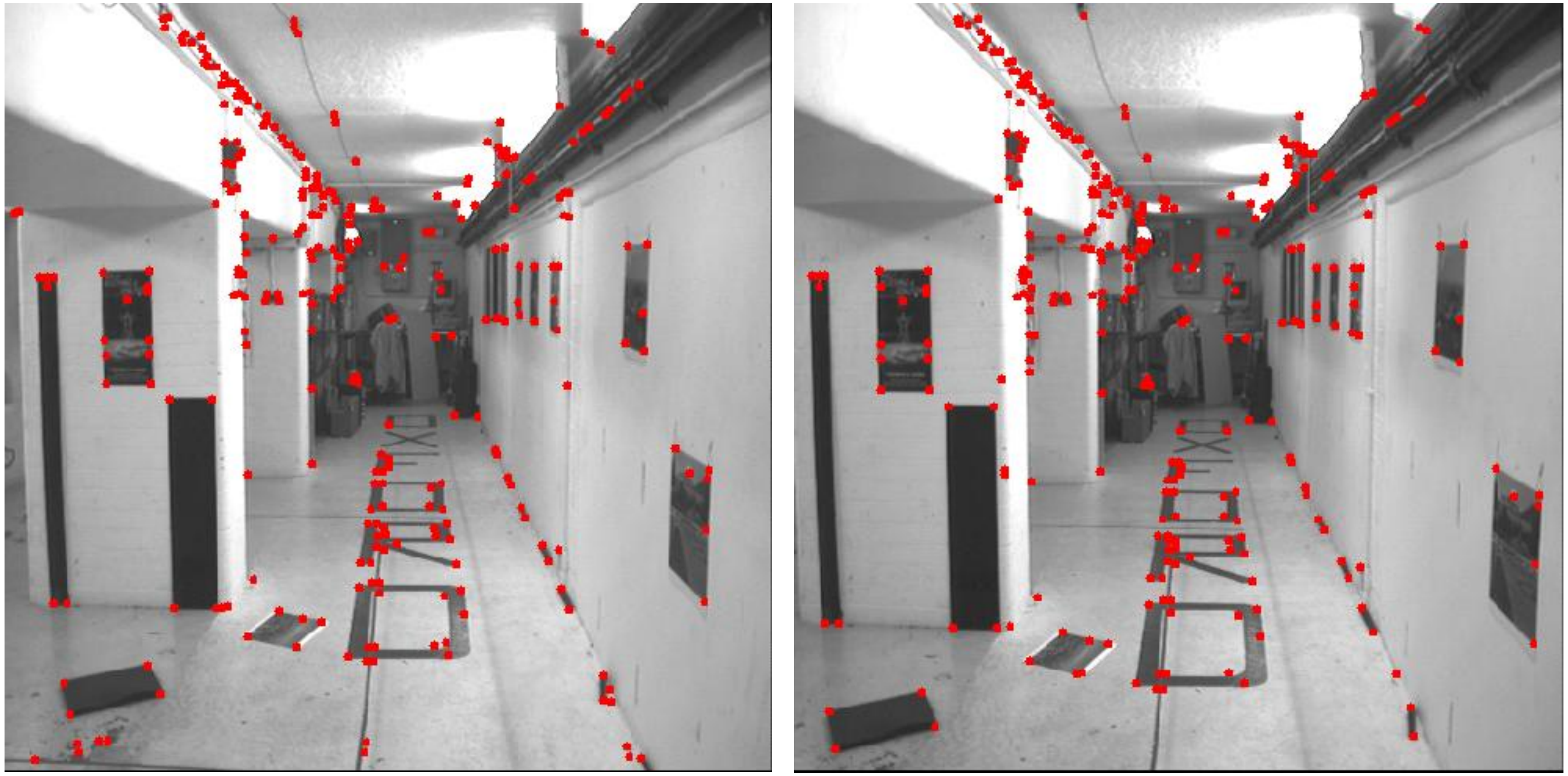


Effect: A very precise corner detector.



Harris Detector – Responses [Harris88]

Harris Detector – Responses [Harris88]



- Results are great for finding correspondences matches between images

Summary

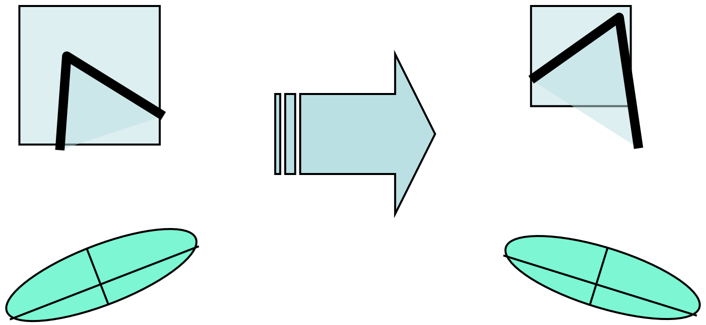
- Local Invariant Features
- Harris Corner Detector

Harris Detector: Properties

- Translation invariance?

Harris Detector: Properties

- Translation invariance
- Rotation invariance?

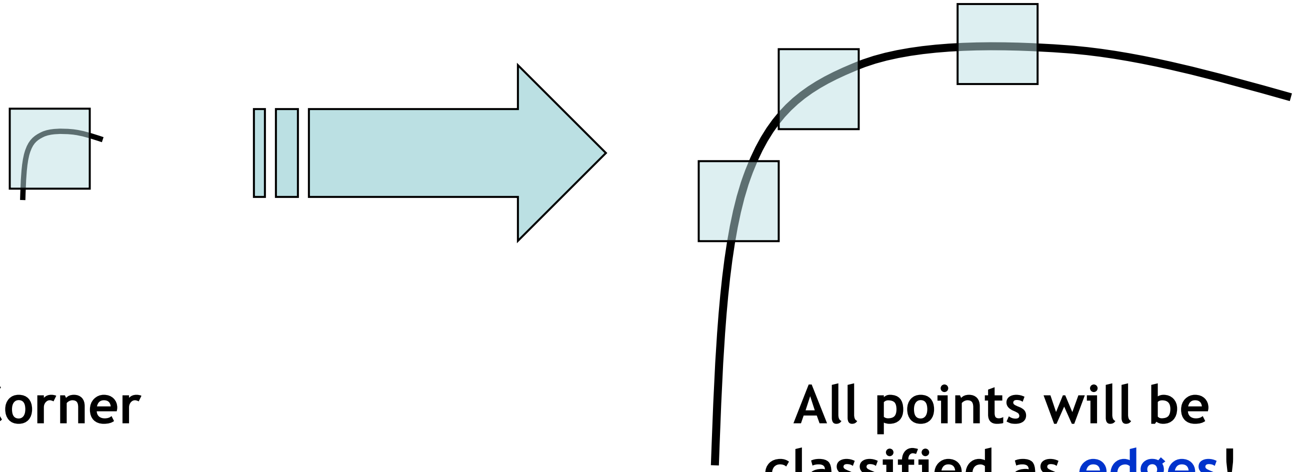


Ellipse rotates but its shape (i.e. eigenvalues) remains the same

***It* is invariant to image rotation**

Harris Detector: Properties

- Translation invariance
- Rotation invariance
- Scale invariance?



Corner

All points will be classified as edges!

***Not* invariant to image scale!**

Next time

Detectors and Descriptors