COMPUTER VISION

LECTURE 4 13.9.2019

Laura Ruotsalainen, Associate Professor
Department of Computer Science

TODAY'S LECTURE

1. Feature detectors
    1. Harris feature detector
2. Feature descriptors
    1. SIFT
3. Feature matching
4. Edge detection
5. Line detection

Features are used for:

Image alignment (e.g., mosaics)

3D reconstruction

Motion tracking (e.g. for AR)

Object recognition

Image retrieval

Navigation

… other

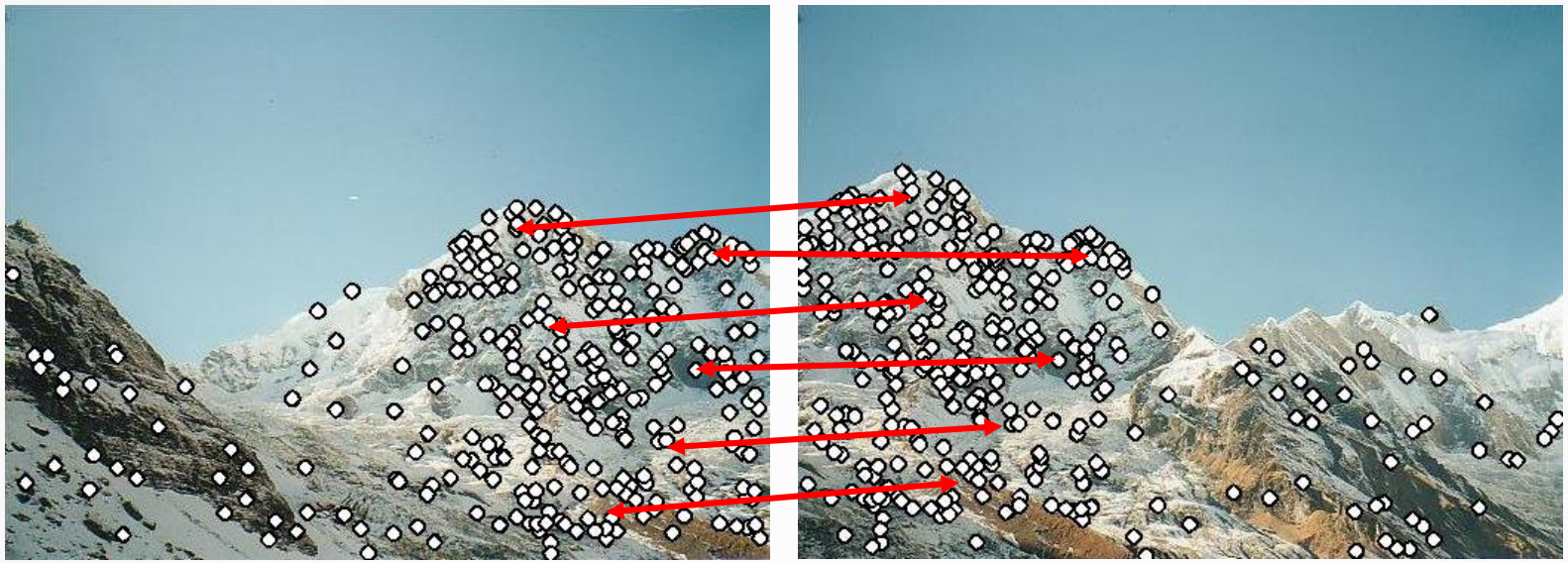
USEFUL FEATURES

APPROACH

1. **Feature detection**: find it
2. **Feature descriptor**: represent it
3. **Feature matching**: match it / **Feature tracking:** track it, when motion

EXTRACTING FEATURES

1. Feature detection 2. Feature descriptor 3. Feature matching



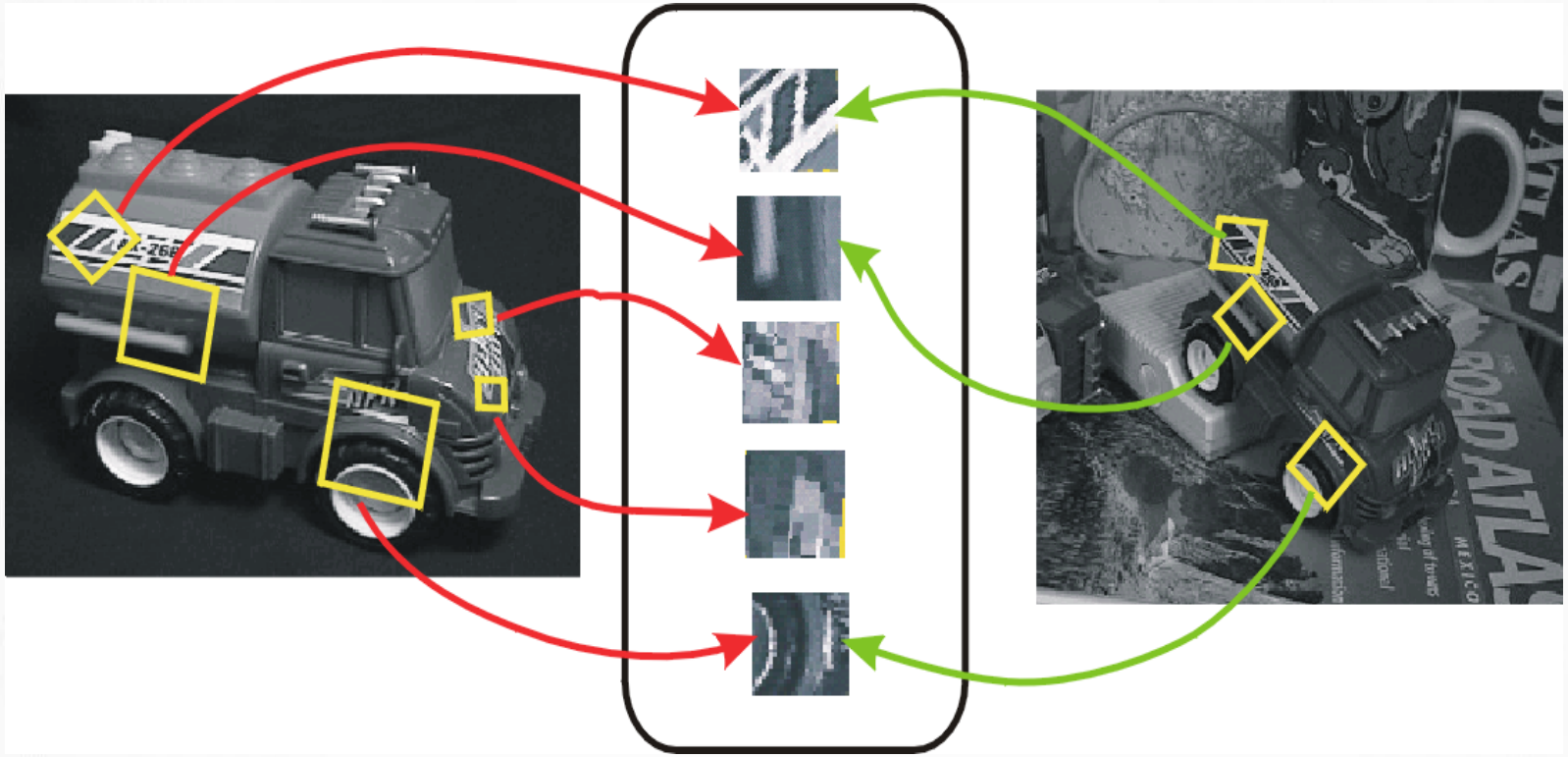
Slide credit: Kris Kitani

INVARIANT LOCAL FEATURES

Find features that are invariant to transformations

geometric invariance: translation, rotation, scale

photometric invariance: brightness, exposure, …



Feature Descriptors

Slide credit: Kris Kitani

ADVANTAGES OF LOCAL FEATURES

Locality

features are local, so robust to occlusion and clutter

Quantity

hundreds or thousands in a single image, in a favorable scene, please see my example later

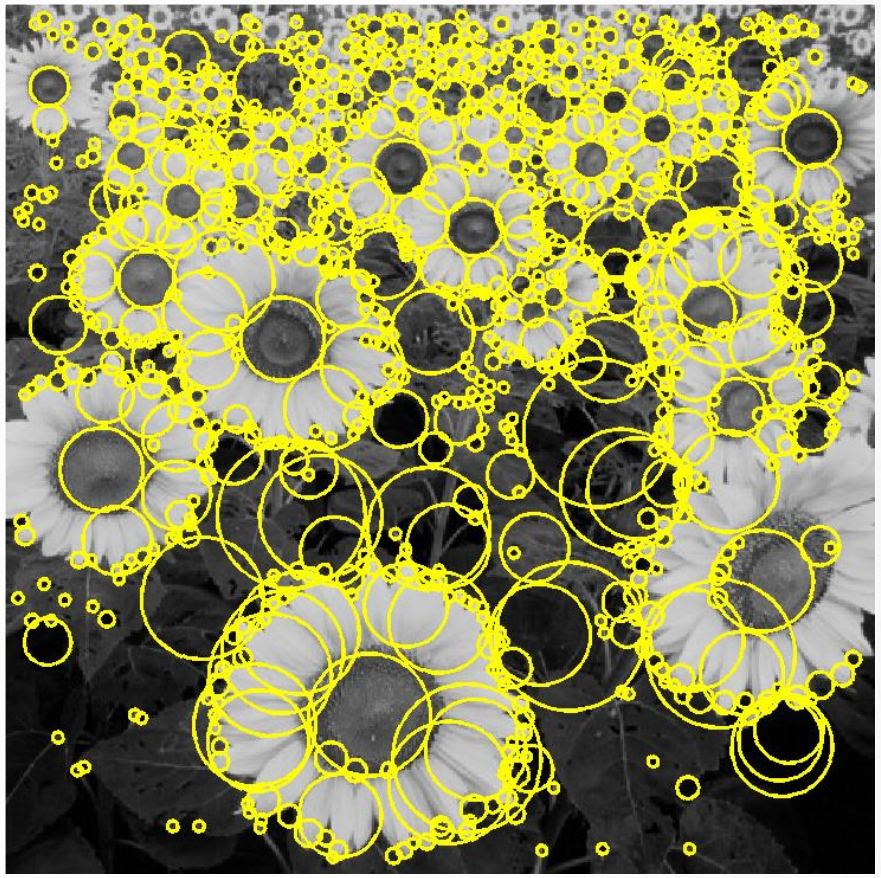
Distinctiveness:

can differentiate a large database of objects

Efficiency

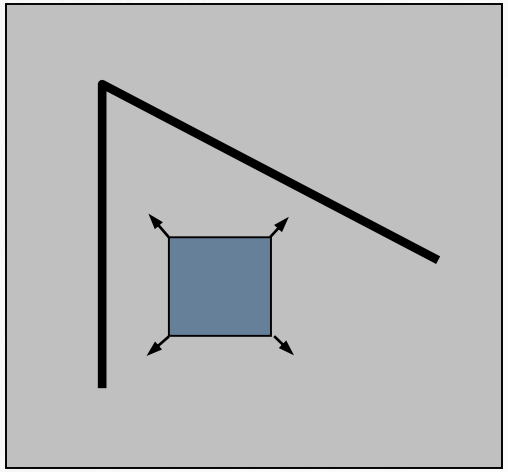
real-time performance achievable

FEATURE EXTRACTION: CORNERS AND BLOBS

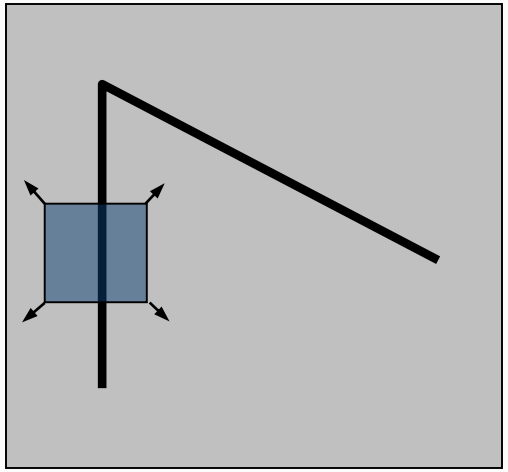


LOCAL MEASURES OF UNIQUENESS

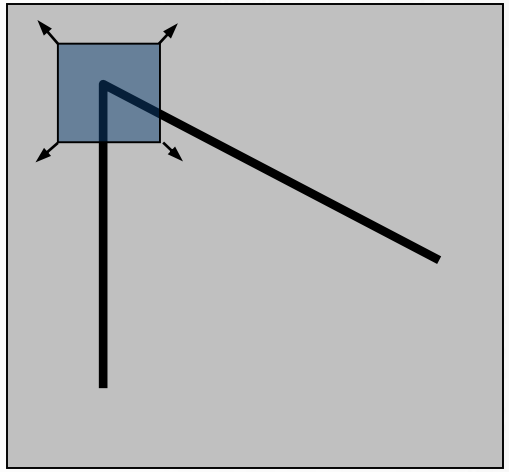
- How does the window change when you shift it?
- Shifting the window in any direction causes a big change



"flat" region:
no change in all directions



"edge":
no change along the edge direction



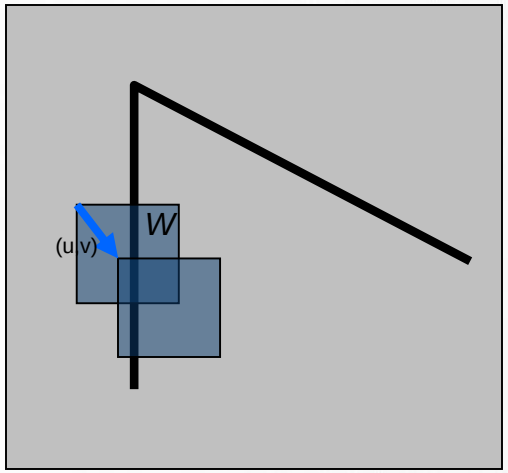
"corner":
significant change in all directions

Credit: S. Seitz, D. Frolova, D. Simakov

HARRIS CORNER DETECTION: THE MATH

- Consider shifting the window *W* by (*u,v*)
  - how do the pixels in W change?
  - compare each pixel before and after by summing up the squared differences (SSD)
  - this defines an SSD “error” *E*(*u,v*):

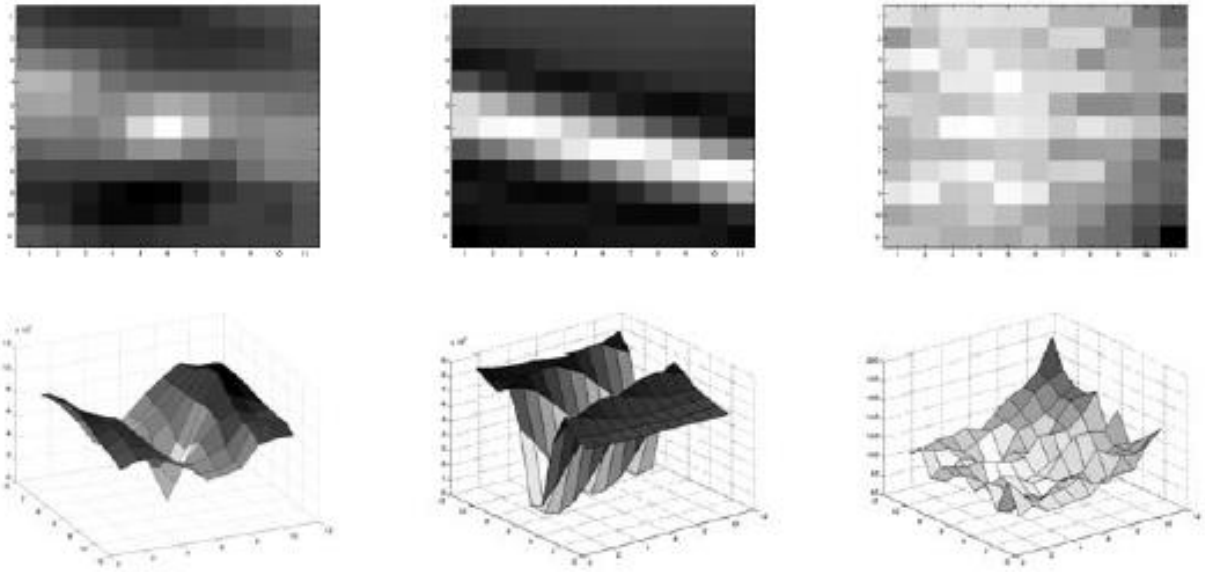
$$E(u,v) = \sum_{(x,y)\in W} [I(x+u, y+v) - I(x,y)]^2$$



  - We are happy if this error is high
  - Slow to compute exactly for each pixel and each offset (u,v)

Slide credit: Kris Kitani

(a)



SMALL MOTION ASSUMPTION

Taylor Series expansion of *I*:

$$I(x+u, y+v) = I(x,y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \text{higher order terms}$$

If the motion (u,v) is small, then first order approximation is good

$$\begin{aligned} I(x+u, y+v) &\approx I(x,y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v \\ &\approx I(x,y) + \begin{bmatrix} I_x & I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} \end{aligned}$$

$$\text{shorthand: } I_x = \frac{\partial I}{\partial x}$$

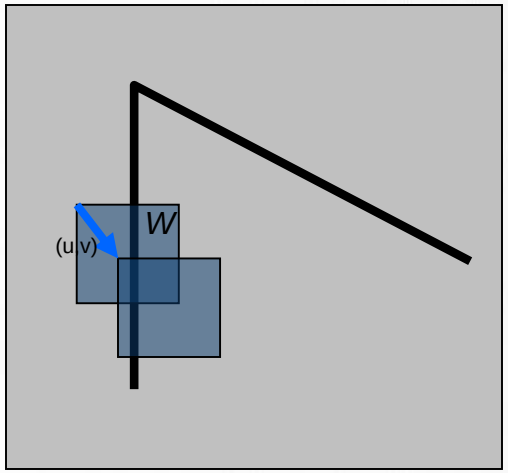
Plugging this into the formula on the previous slide…

Slide credit: Kris Kitani

CORNER DETECTION: THE MATH

Consider shifting the window *W* by (*u*,*v*)

- define an SSD "error" *E(u,v)*:



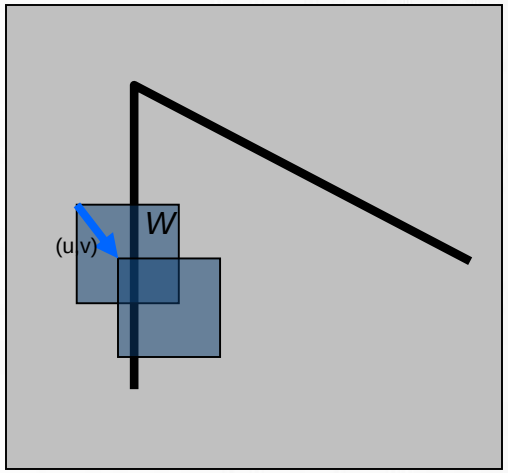
$$E(u,v) = \sum_{(x,y)\in W} [I(x+u,y+v) - I(x,y)]^2$$

$$\approx \sum_{(x,y)\in W} [I(x,y) + I_x u + I_y v - I(x,y)]^2$$

$$\approx \sum_{(x,y)\in W} [I_x u + I_y v]^2$$

Slide credit: Kris Kitani

CORNER DETECTION: THE MATH



Consider shifting the window *W* by (*u*,*v*)

- define an SSD "error" *E*(*u*,*v*):

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

$$\approx Au^2 + 2Buv + Cv^2$$

$$A = \sum_{(x,y)\in W} I_x^2 \qquad B = \sum_{(x,y)\in W} I_x I_y \qquad C = \sum_{(x,y)\in W} I_y^2$$

- Thus, *E*(*u*,*v*) is locally approximated as a quadratic error function

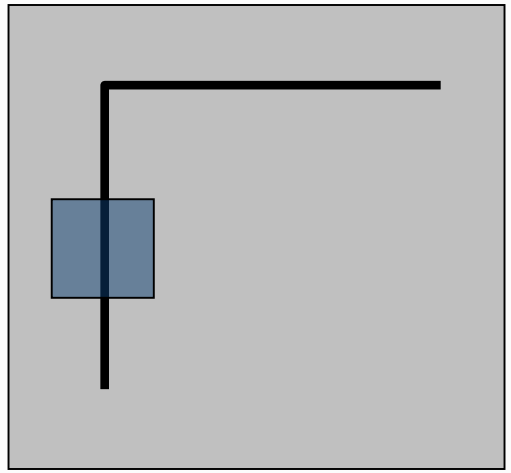
Slide credit: Kris Kitani

$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} \underbrace{\begin{bmatrix} A & B \\ B & C \end{bmatrix}}_{H} \begin{bmatrix} u \\ v \end{bmatrix}$$

$$A = \sum_{(x,y)\in W} I_x^2$$

$$B = \sum_{(x,y)\in W} I_x I_y$$

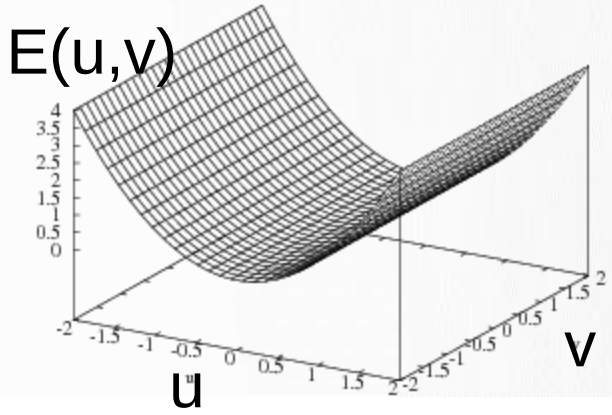
$$C = \sum_{(x,y)\in W} I_y^2$$



Vertical edge:

$$I_y = 0$$

$$H = \begin{bmatrix} A & 0 \\ 0 & 0 \end{bmatrix}$$



Slide credit: Kris Kitani

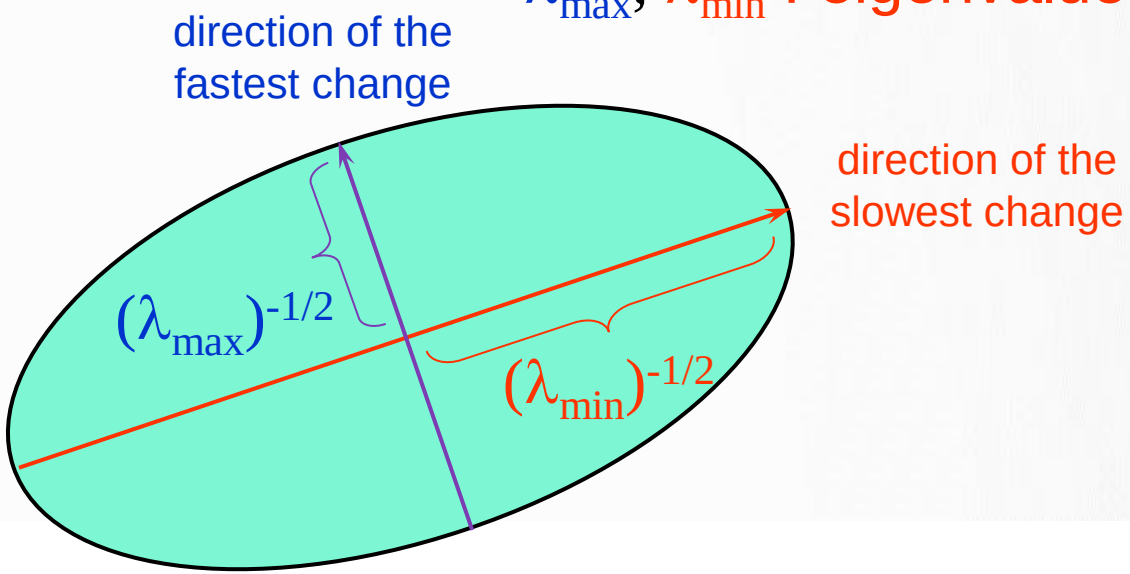
GENERAL CASE

We can visualize *H* as an ellipse with axis lengths determined by the *eigenvalues* of *H* and orientation determined by the *eigenvectors* of *H*

Ellipse equation:

$$[u \quad v] \quad H \quad \begin{bmatrix} u \\ v \end{bmatrix} = \text{const}$$

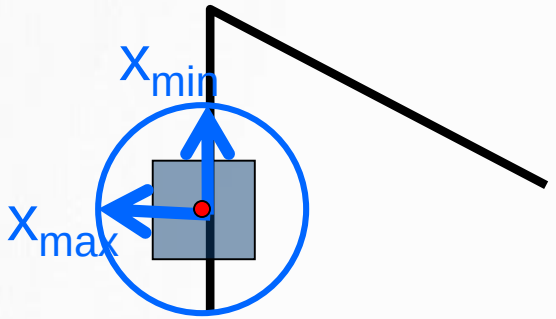
λ_{max}, λ_{min} : eigenvalues of *H*



Slide credit: Kris Kitani

CORNER DETECTION: THE MATH

$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} \underbrace{\begin{bmatrix} A & B \\ B & C \end{bmatrix}}_{H} \begin{bmatrix} u \\ v \end{bmatrix}$$



$$Hx_{\max} = \lambda_{\max} x_{\max}$$
$$Hx_{\min} = \lambda_{\min} x_{\min}$$

Eigenvalues and eigenvectors of H

- Define shift directions with the smallest and largest change in error
- x_{max} = direction of largest increase in *E*
- λ_{max} = amount of increase in direction x_{max}
- x_{min} = direction of smallest increase in *E*
- λ_{min} = amount of increase in direction x_{min}

Slide credit: Kris Kitani

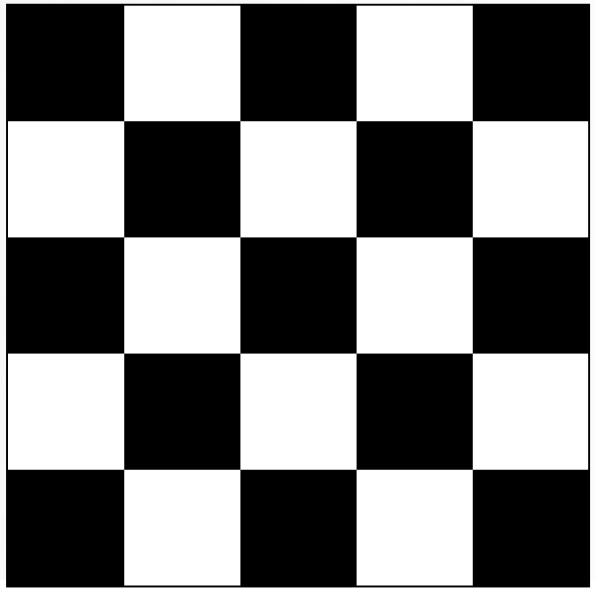
CORNER DETECTION: THE MATH

How are λ_{max}, x_{max}, λ_{min}, and x_{min} relevant for feature detection?

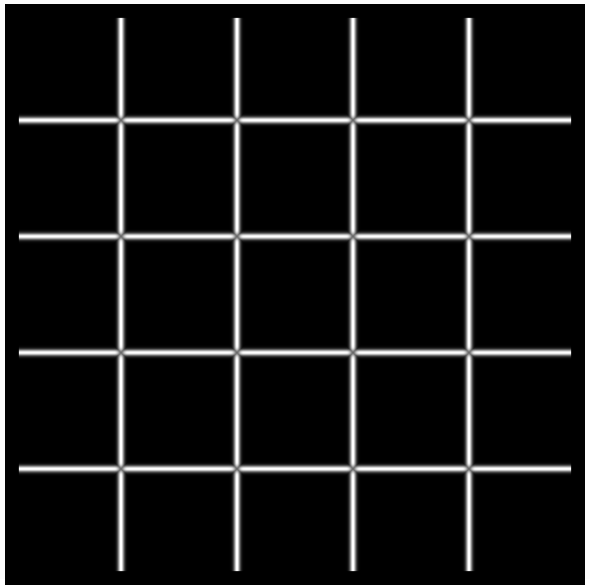
- What's our feature scoring function?

Want $E(u,v)$ to be large for small shifts in all directions

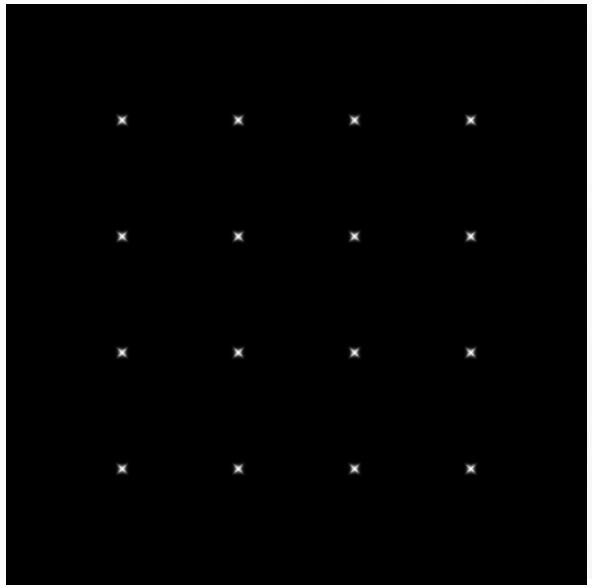
- the minimum of $E(u,v)$ should be large, over all unit vectors $[u\ v]$
- this minimum is given by the smaller eigenvalue (λ_{min}) of H



I



λ_{max}



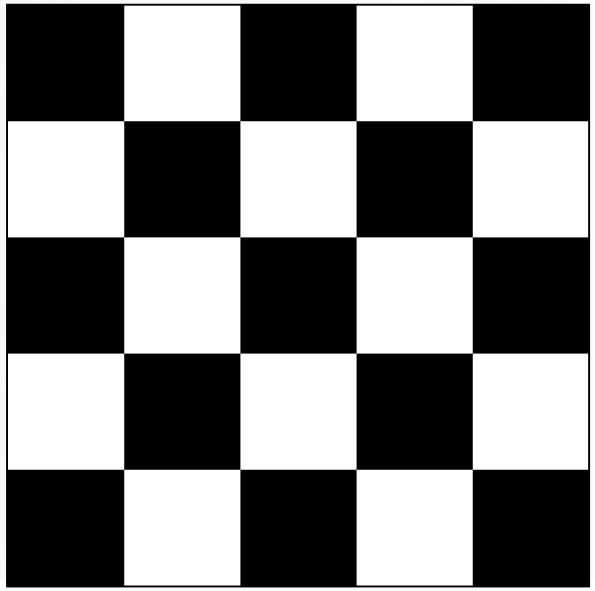
λ_{min}

Slide credit: Kris Kitani

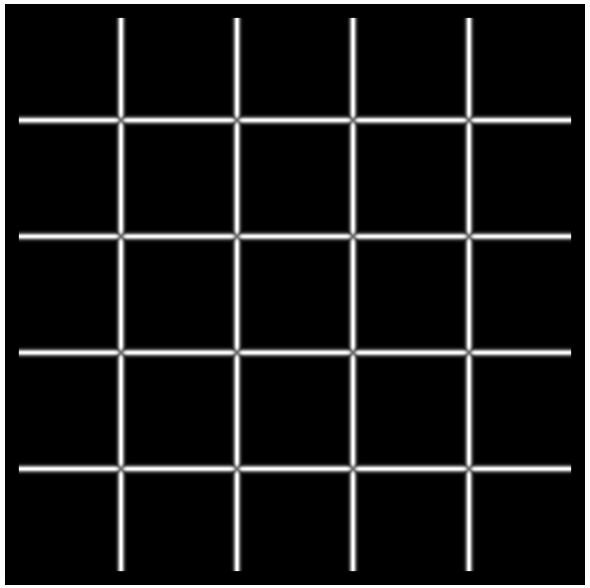
CORNER DETECTION SUMMARY

Here's what you do

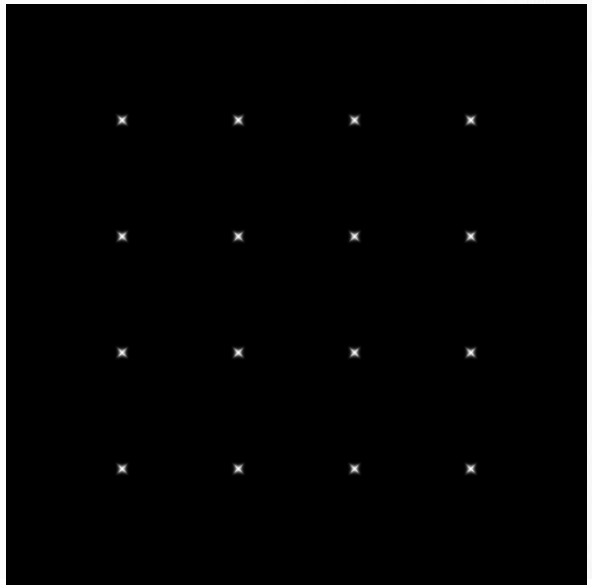
- Compute the gradient at each point in the image
- Create the *H* matrix from the entries in the gradient
- Compute the eigenvalues.
- Find points with large response (λ_{min} > threshold)
- Choose those points where λ_{min} is a local maximum as features



I



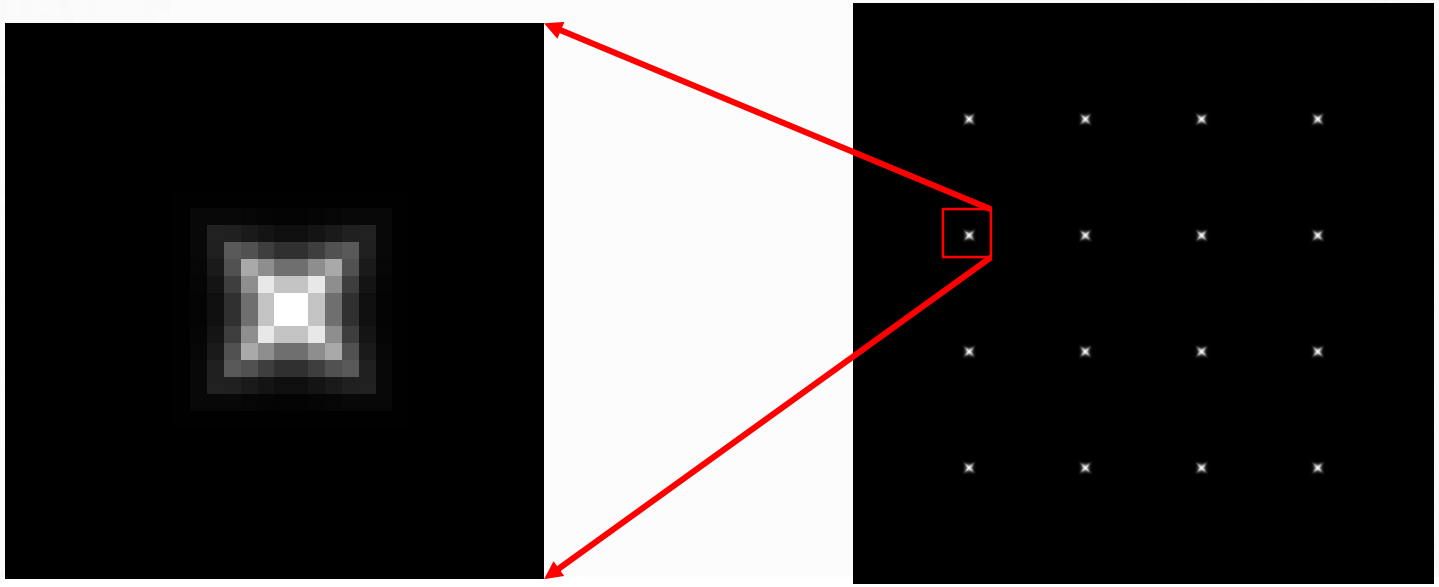
λ_{max}



λ_{min}

CORNER DETECTION SUMMARY

- Compute the gradient at each point in the image
- Create the *H* matrix from the entries in the gradient
- Compute the eigenvalues.
- Find points with large response (λ_{min} > threshold)
- Choose those points where λ_{min} is a local maximum as features



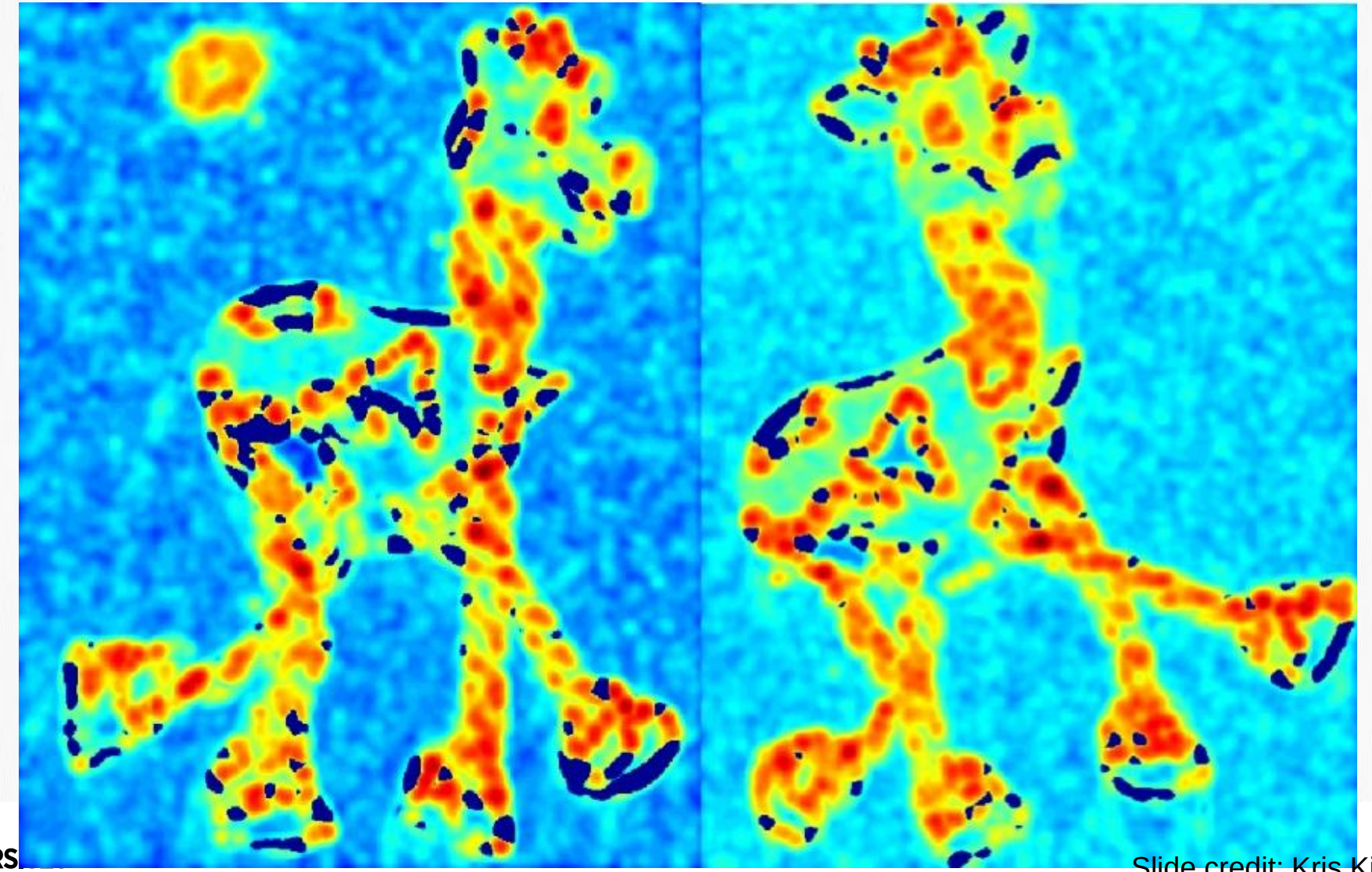
λ_{min}

Slide credit: Kris Kitani

HARRIS DETECTOR EXAMPLE

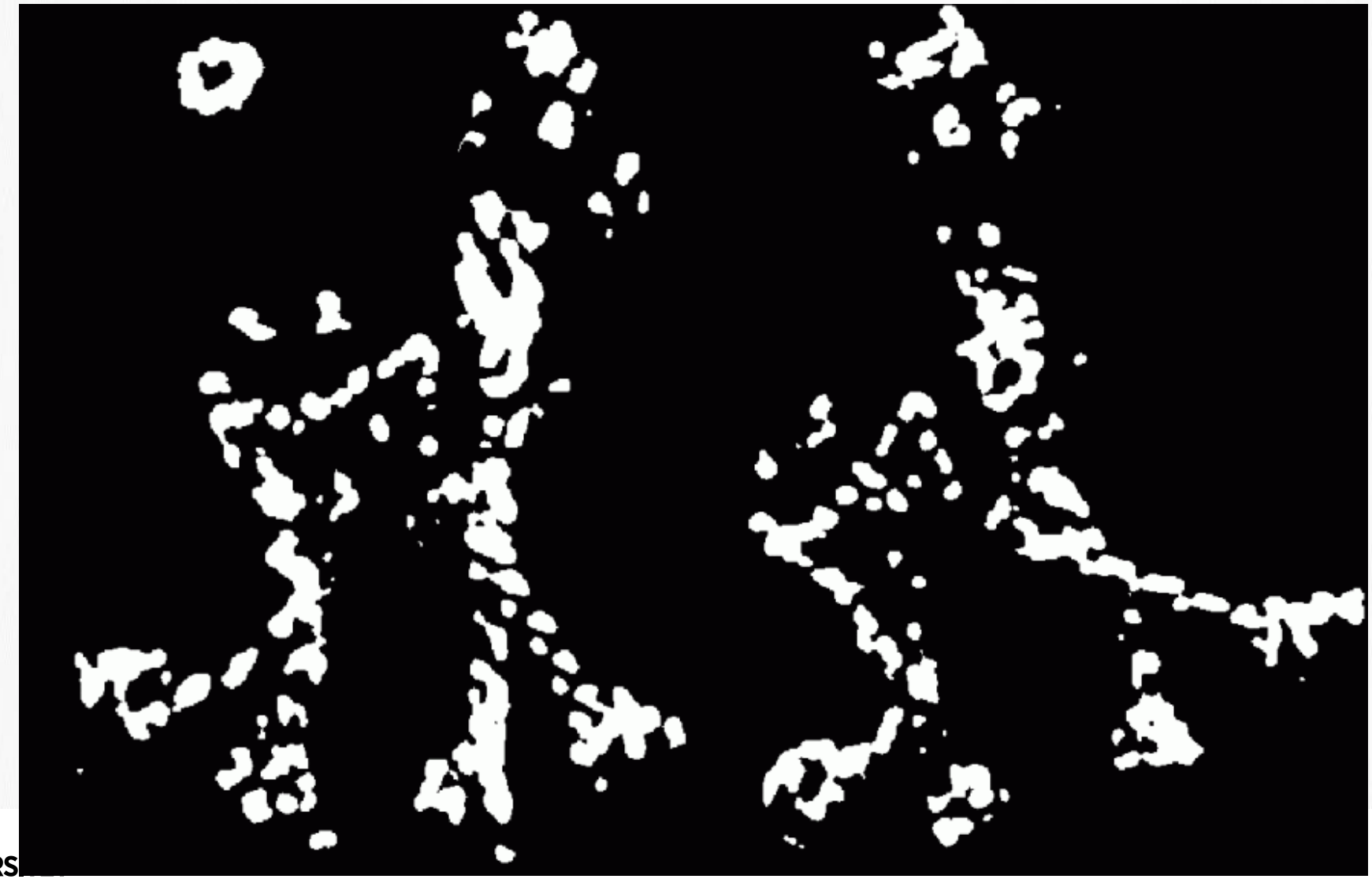
Slide credit: Kris Kitani

λ_{MIN} VALUE (RED HIGH, BLUE LOW)



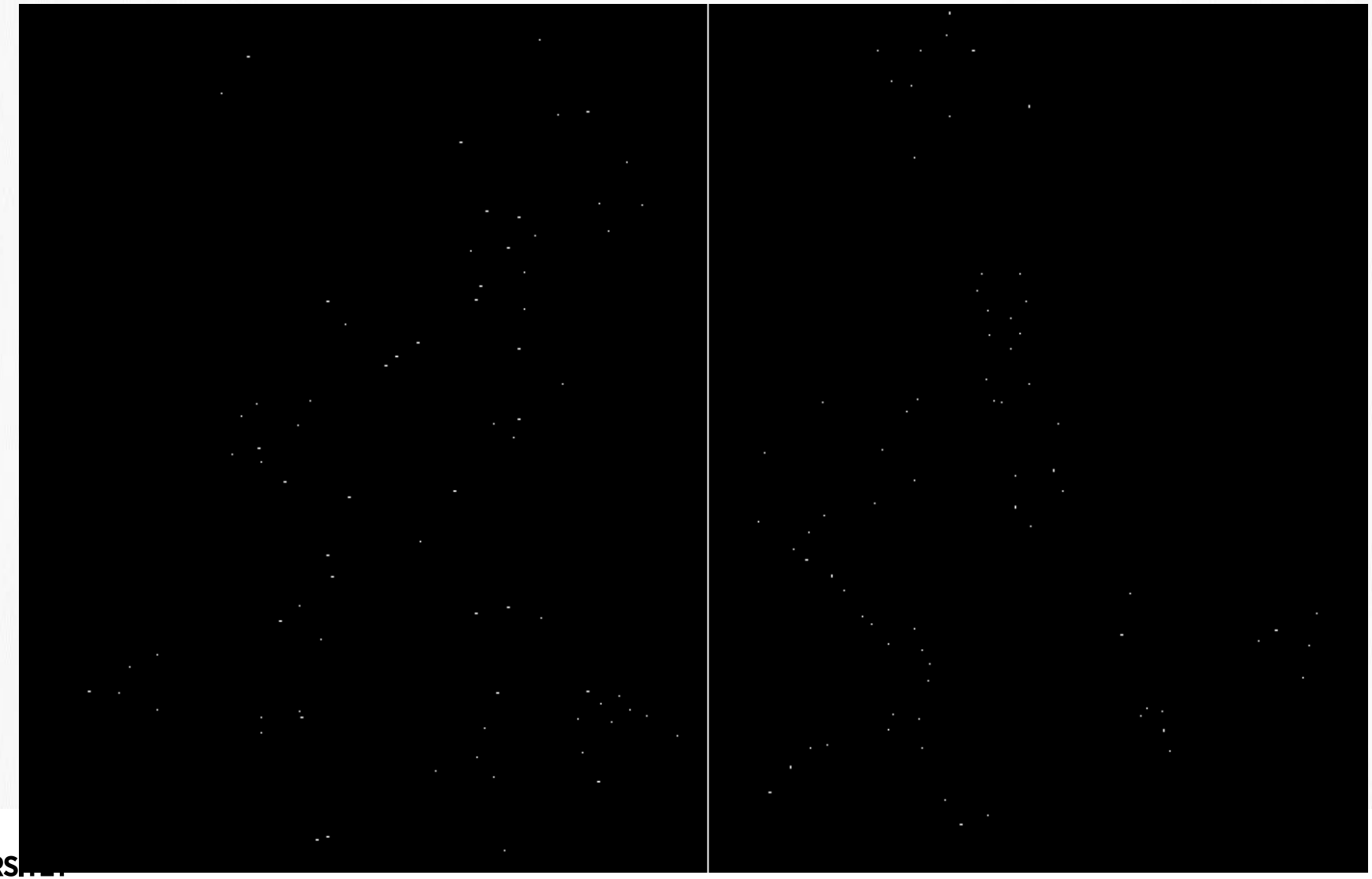
Slide credit: Kris Kitani

THRESHOLD (λ_{MIN} > VALUE)



Slide credit: Kris Kitani

FIND LOCAL MAXIMA OF λ_{MIN}



Slide credit: Kris Kitani

HARRIS FEATURES (IN RED)

Slide credit: Kris Kitani

INVARIANCE AND EQUIVARIANCE

- We want corner locations to be *invariant* to photometric transformations and *equivariant* to geometric transformations
  - **Invariance:** image is transformed and corner locations do not change
  - **Equivariance:** if we have two transformed versions of the same image, features should be detected in corresponding locations

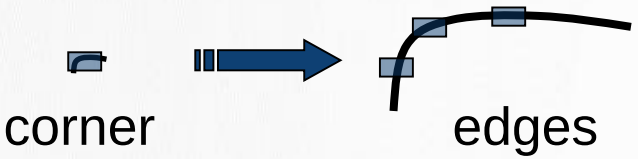
(Sometimes “invariant” and “equivariant” are both referred to as “invariant”)

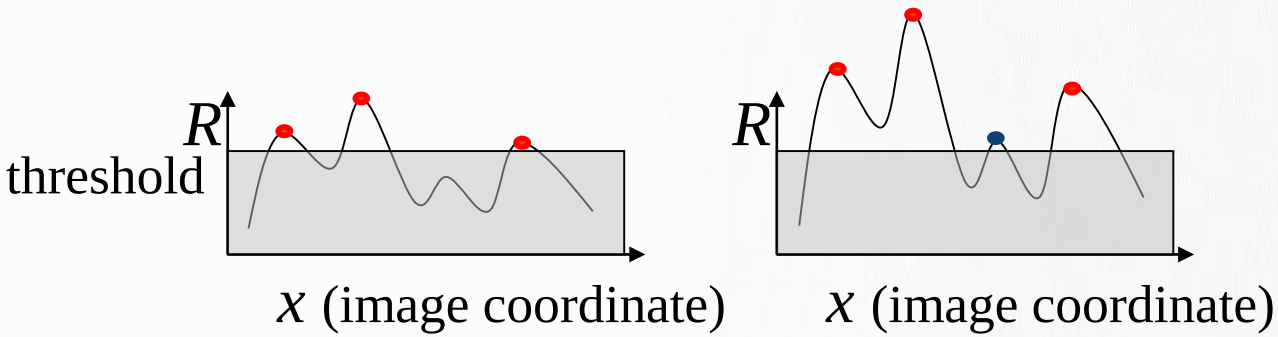
(Sometimes “equivariant” is called “covariant”)

Slide credit: Kris Kitani

HARRIS DETECTOR: INVARIANCE

- Detector is invariant to
  - translation: derivatives and window function are equivariant
  - rotation: second moment ellipse rotates but its shape (i.e. eigenvalues) remains the same
- Partially invariant to:
  - affine intensity change
- Is not invariant to scaling!





Slide credit: Kris Kitani

FEATURE DESCRIPTORS

- A good feature descriptor should
  - be invariant: it shouldn’t change even if image is transformed
  - be highly unique for each point
- SIFT is a very popular descriptor
- Other, e.g.
  - HOG: Histogram of Gradients (HOG)
  - FREAK: Fast Retina Keypoint, used in Visual SLAM
  - LIFT: Learned Invariant Feature Transform, learned via deep learning

  https://arxiv.org/abs/1603.09114

SCALE INVARIANT FEATURE TRANSFORM

Basic idea:

- Take 16x16 square window around detected feature
- Compute edge orientation (angle of the gradient - 90°) for each pixel
- Throw out weak edges (threshold gradient magnitude)
- Create histogram of surviving edge orientations

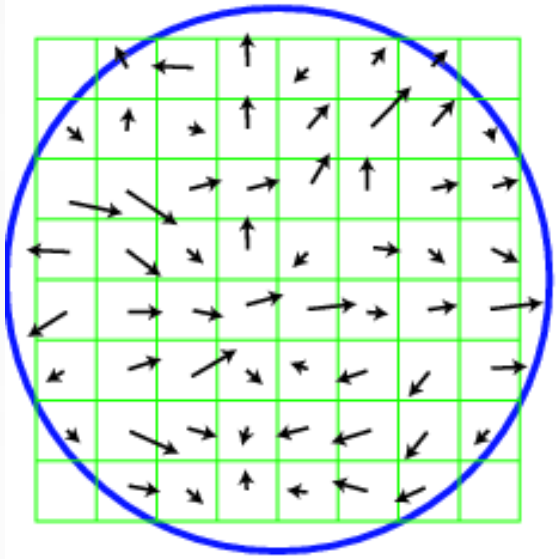
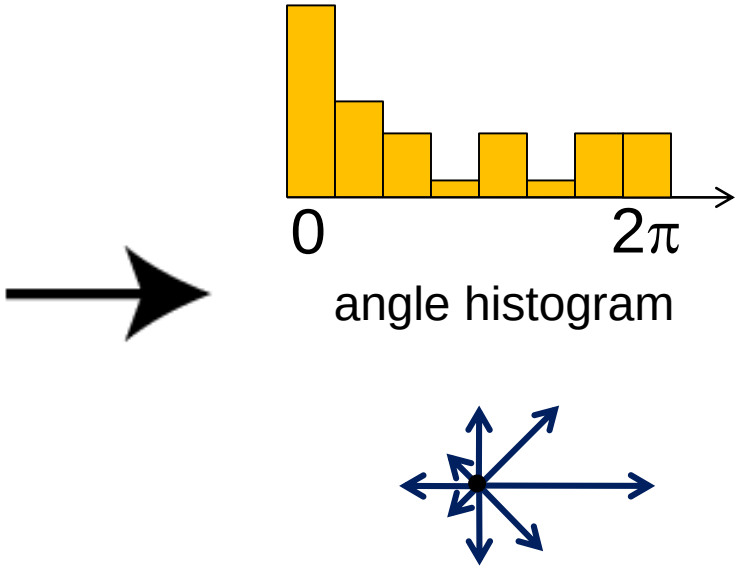


Image gradients

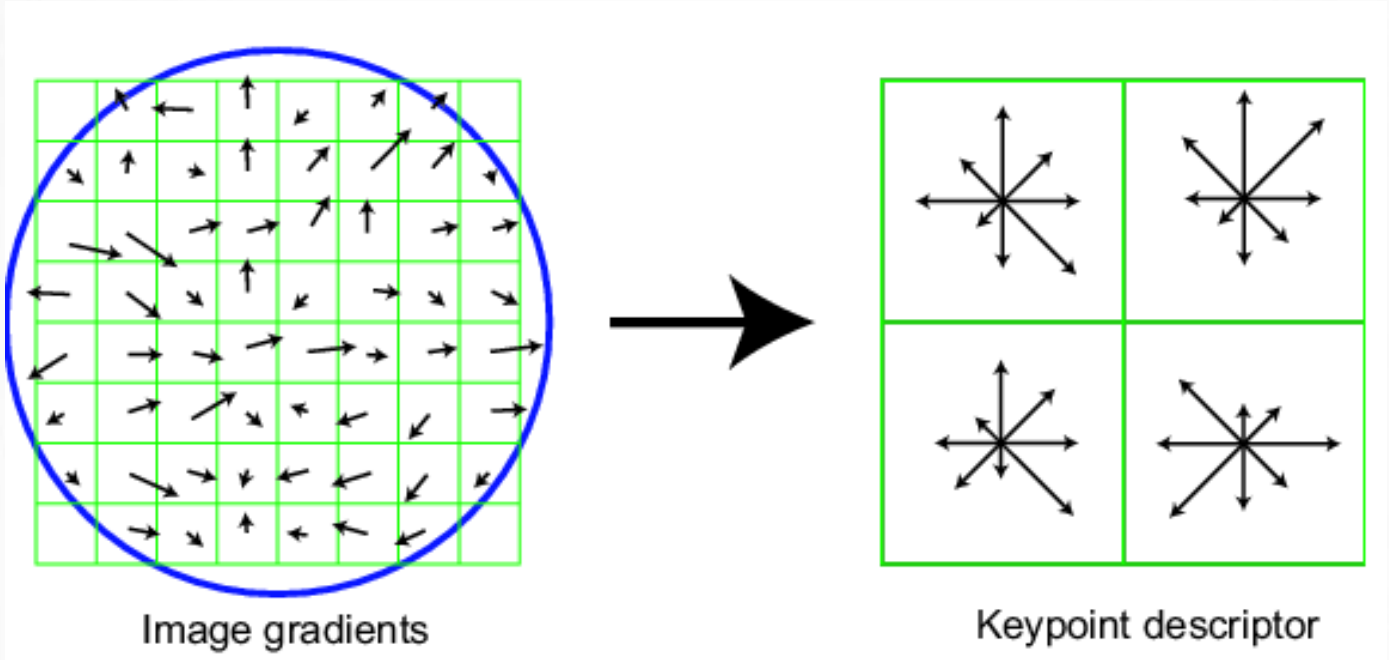


Adapted from slide by David Lowe

SIFT DESCRIPTOR

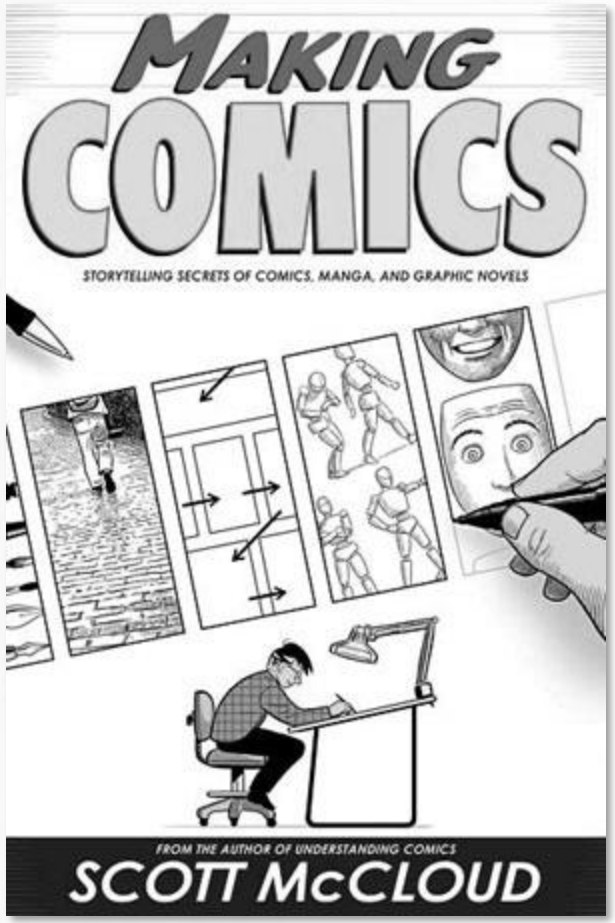
Full version

- Divide the 16x16 window into a 4x4 grid of cells (2x2 case shown below)
- Compute an orientation histogram for each cell
- 16 cells * 8 orientations = 128 dimensional descriptor

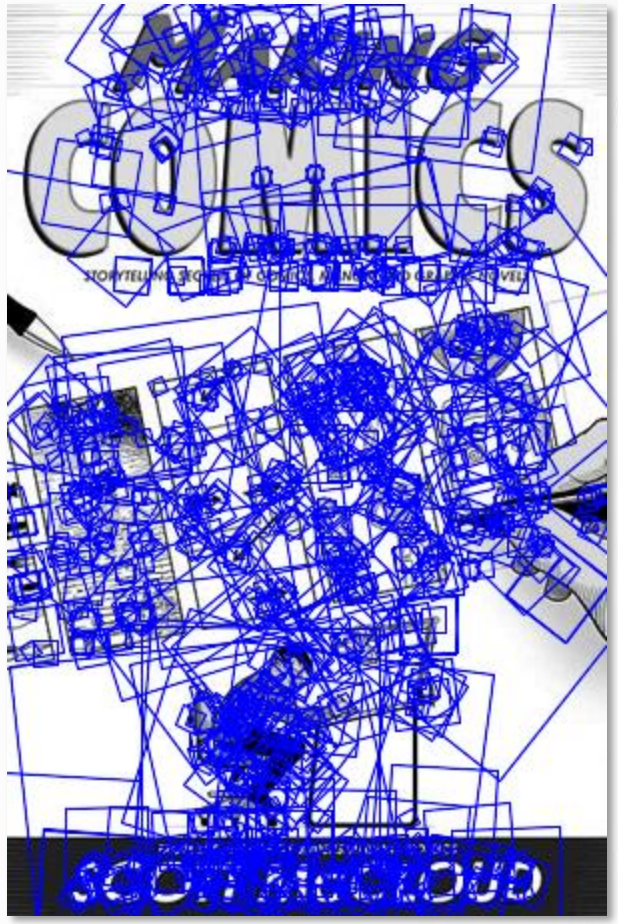


Adapted from slide by David Lowe

SIFT EXAMPLE

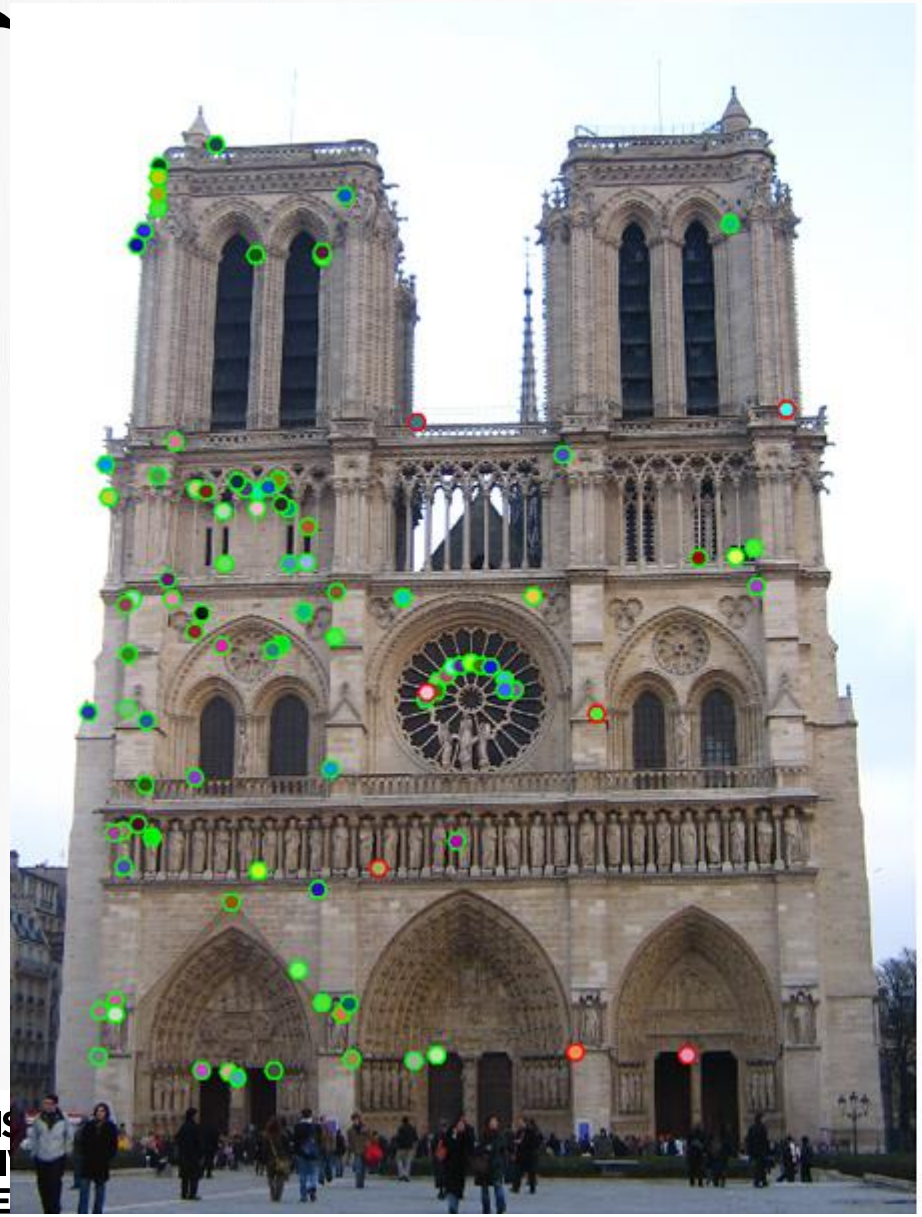


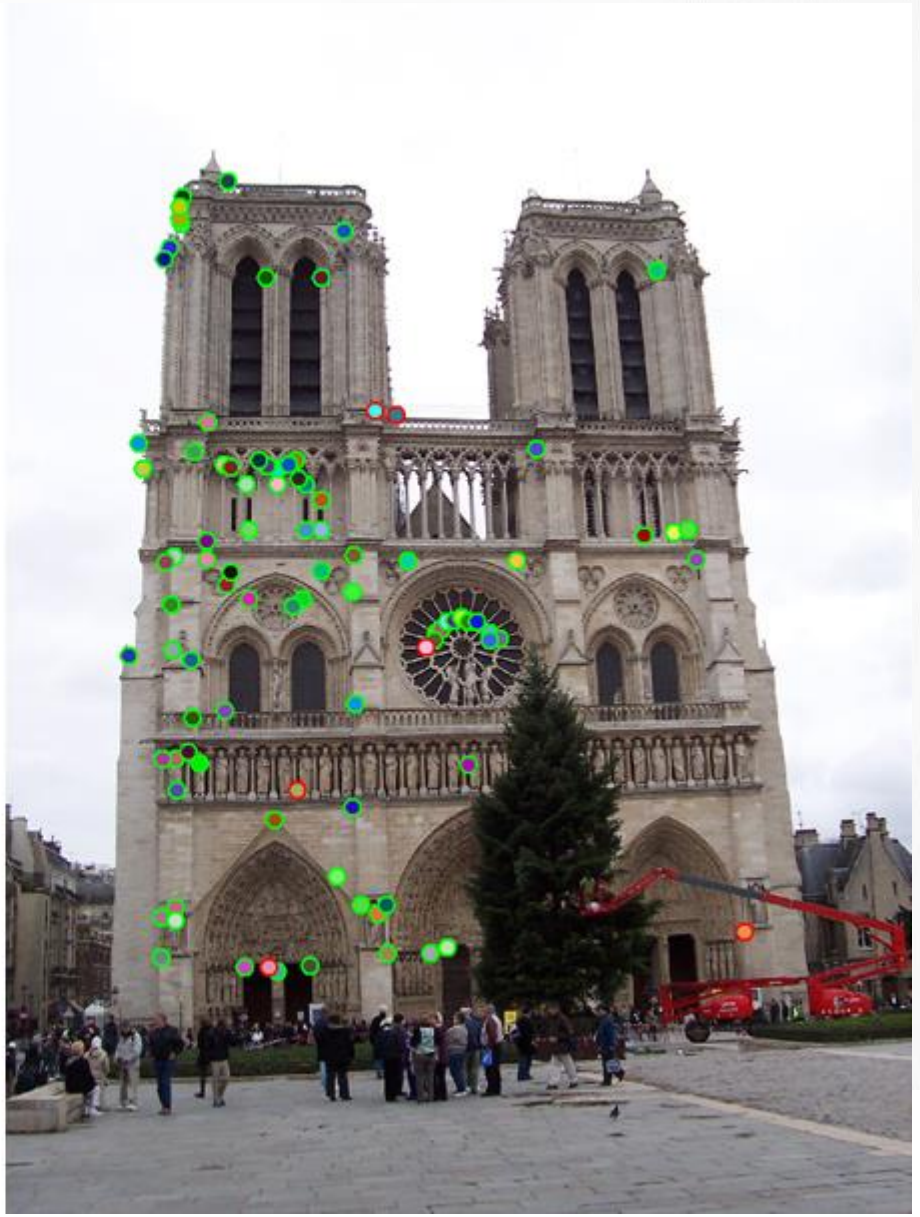
sift



868 SIFT features

WHICH FEATURES MATCH?





FEATURE MATCHING

- Given a feature in Image (I_0), how to find the best match in (I_1)
  1. Define distance function that compares two descriptors
  2. Test all the features in I_1, find the one with min distance

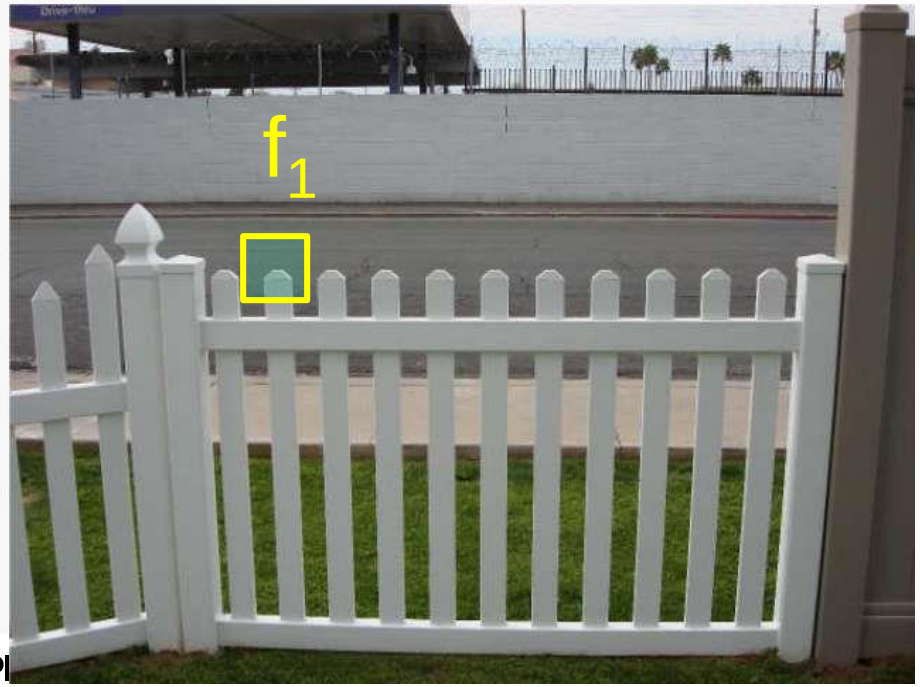
FEATURE DISTANCE

How to define the difference between two features f_0, f_1?

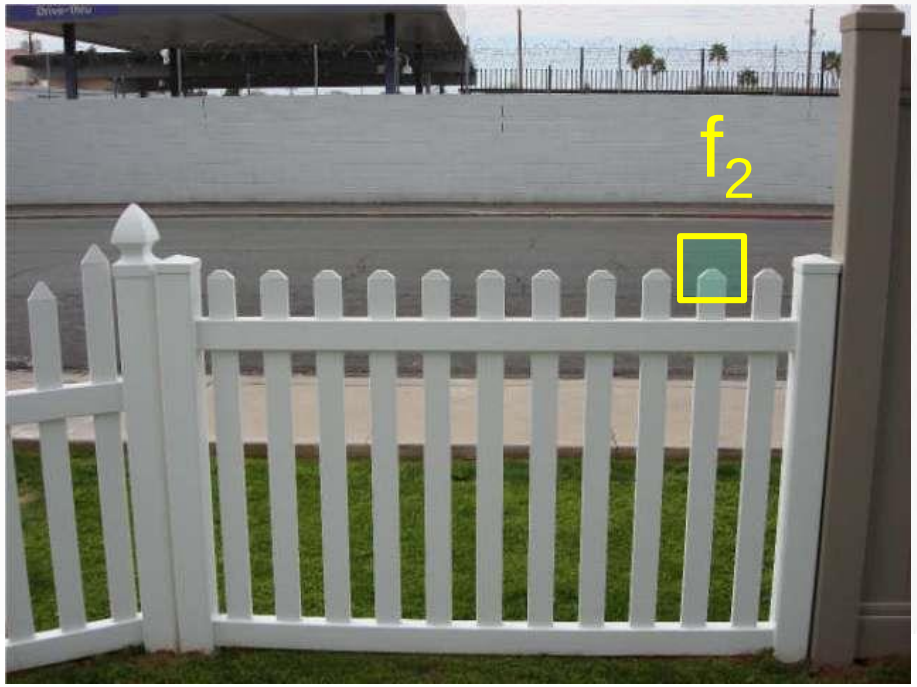
Simple approach: L_2 distance, $||f_1 - f_2||$

$$d(\mathbf{p}, \mathbf{q}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2}.$$

can give small distances for ambiguous (incorrect) matches



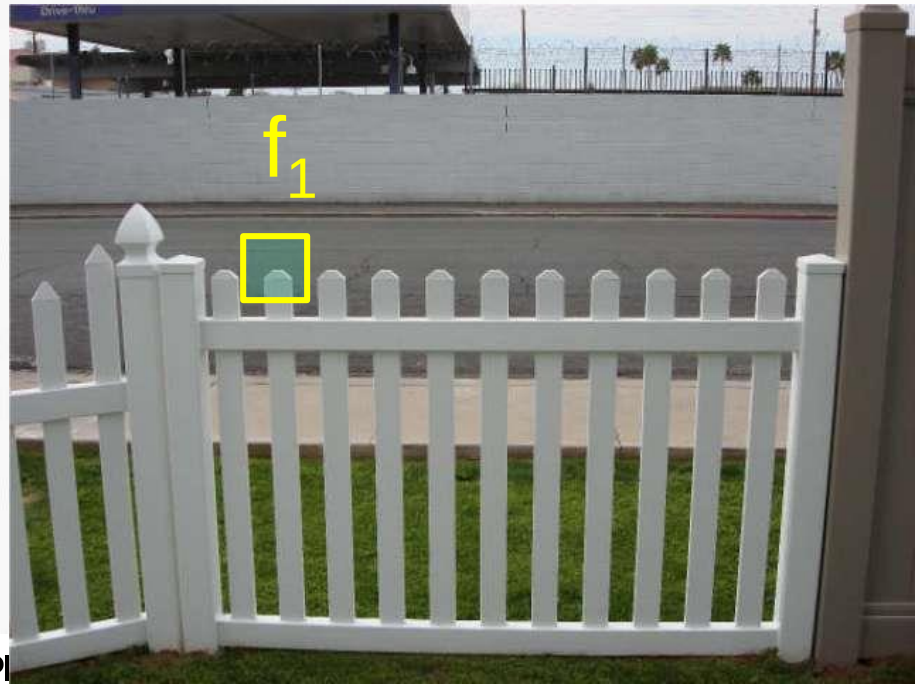
I_1



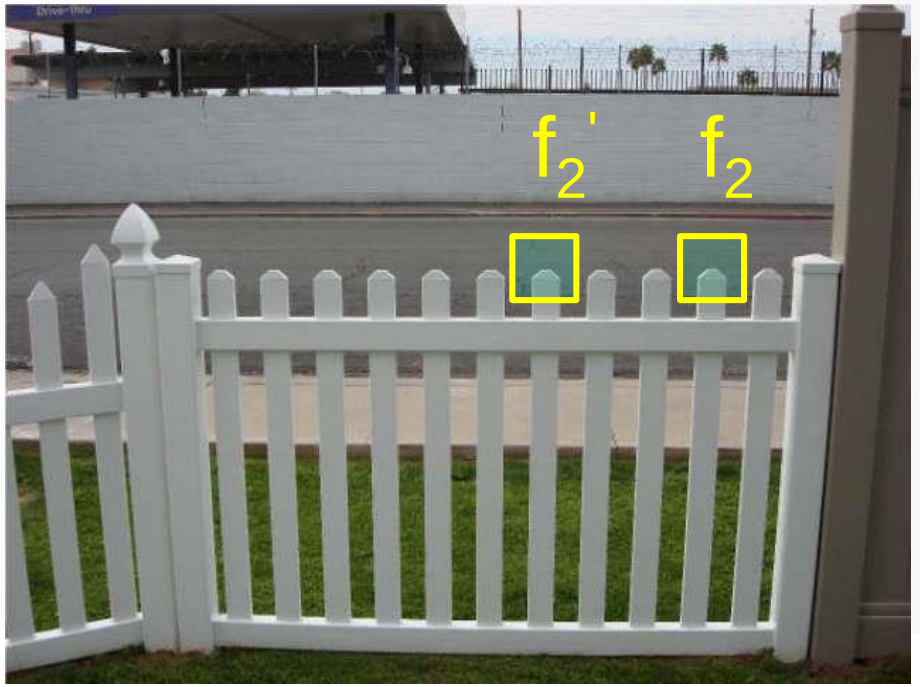
I_2

Feature distance

- Better approach: ratio distance = $||f_1 - f_2|| / ||f_1 - f_2'||$
  - f_2 is best SSD match to f_1 in I_2
  - f_2' is 2nd best SSD match to f_1 in I_2
  - gives large values for ambiguous matches



I_1

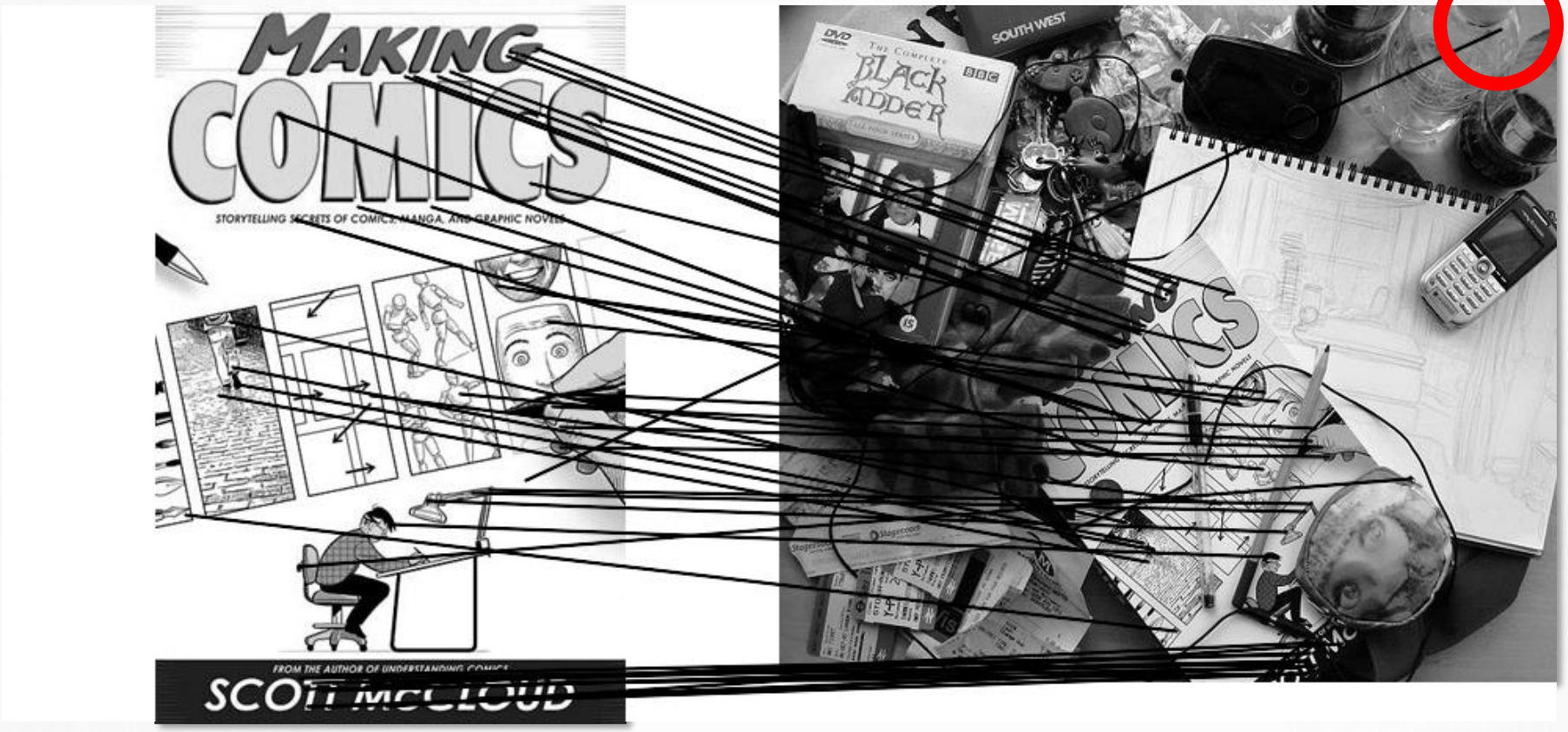


I_2

Slide credit: Kris Kitani

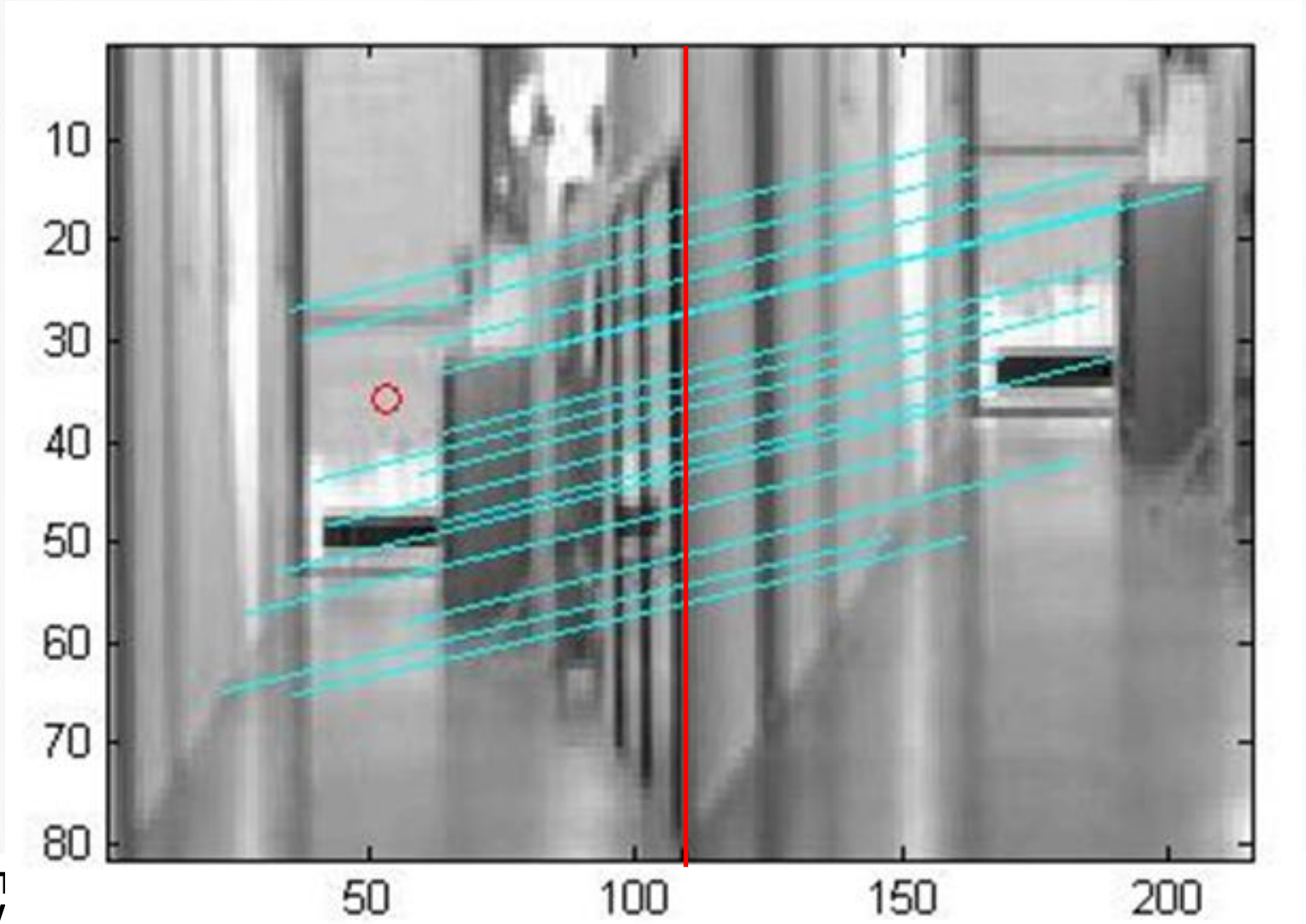
FEATURE MATCHING EXAMPLE 1

We'll deal with **outliers** later



51 matches (thresholded by ratio score)

FEATURE MATCHING EXAMPLE 2

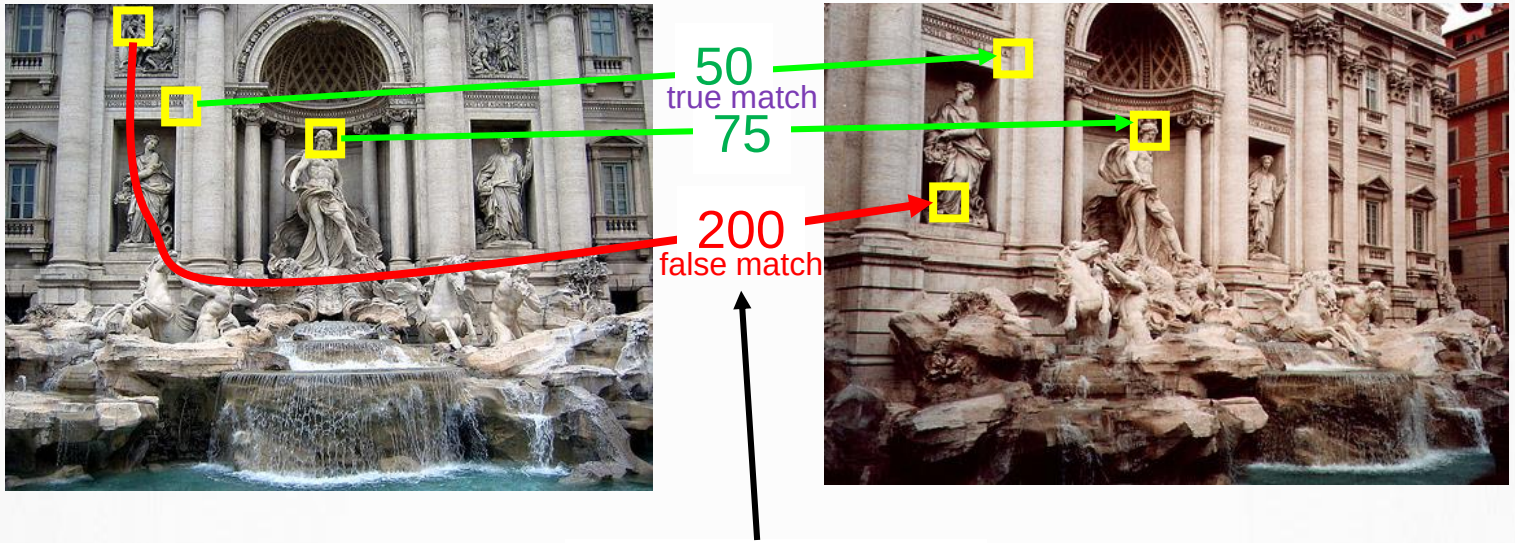


< 20 features matched,

Computer vision methods are very often tested and published by using favorable images

TRUE/FALSE POSITIVES

How can we measure the performance of a feature matcher?



The distance threshold affects performance

True positives = # of detected matches that are correct

– Suppose we want to maximize these—how to choose threshold?

False positives = # of detected matches that are incorrect

– Suppose we want to minimize these—how to choose threshold?

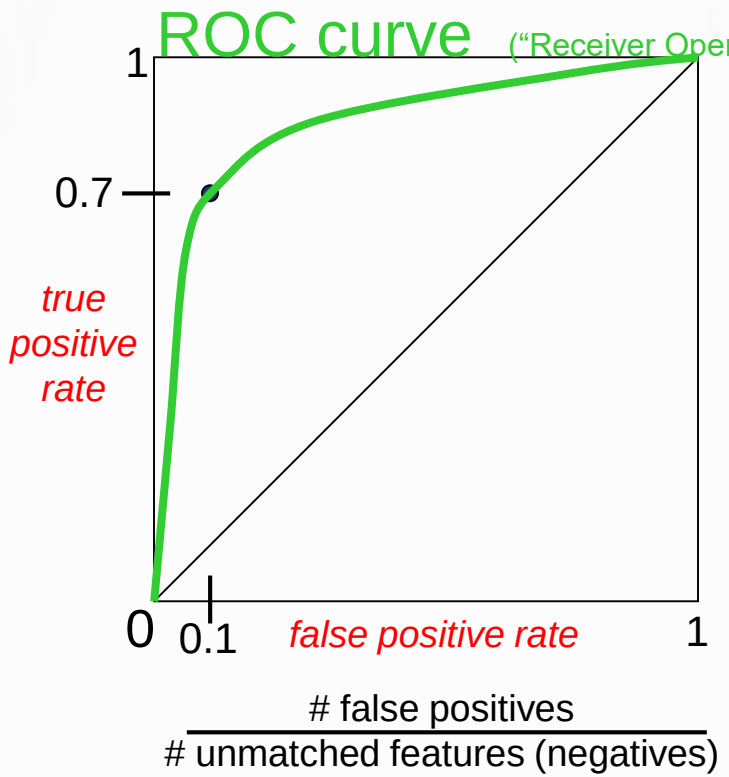
Slide credit: Kris Kitani

EVALUATING THE RESULTS

How can we measure the performance of a feature matcher?

$$\frac{\text{\# true positives}}{\text{\# matching features (positives)}}$$

recall



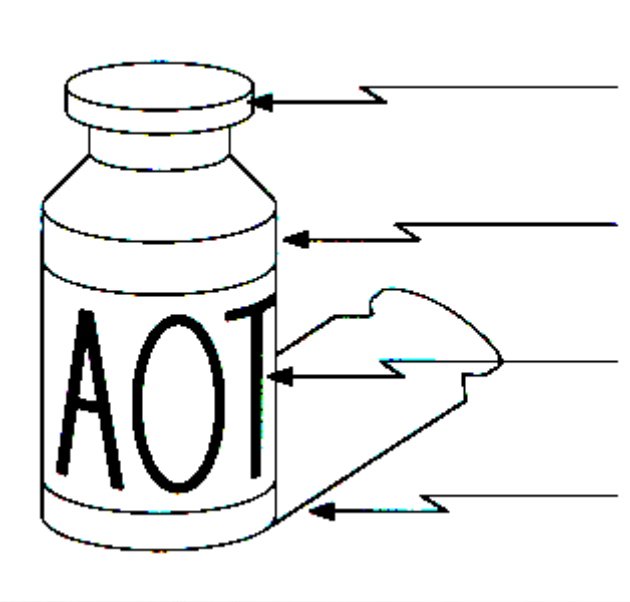
Single number: Area Under the Curve (AUC)

E.g. AUC = 0.87

1 is the best

Slide credit: Kris Kitani

EDGE DETECTION



surface normal discontinuity

depth discontinuity

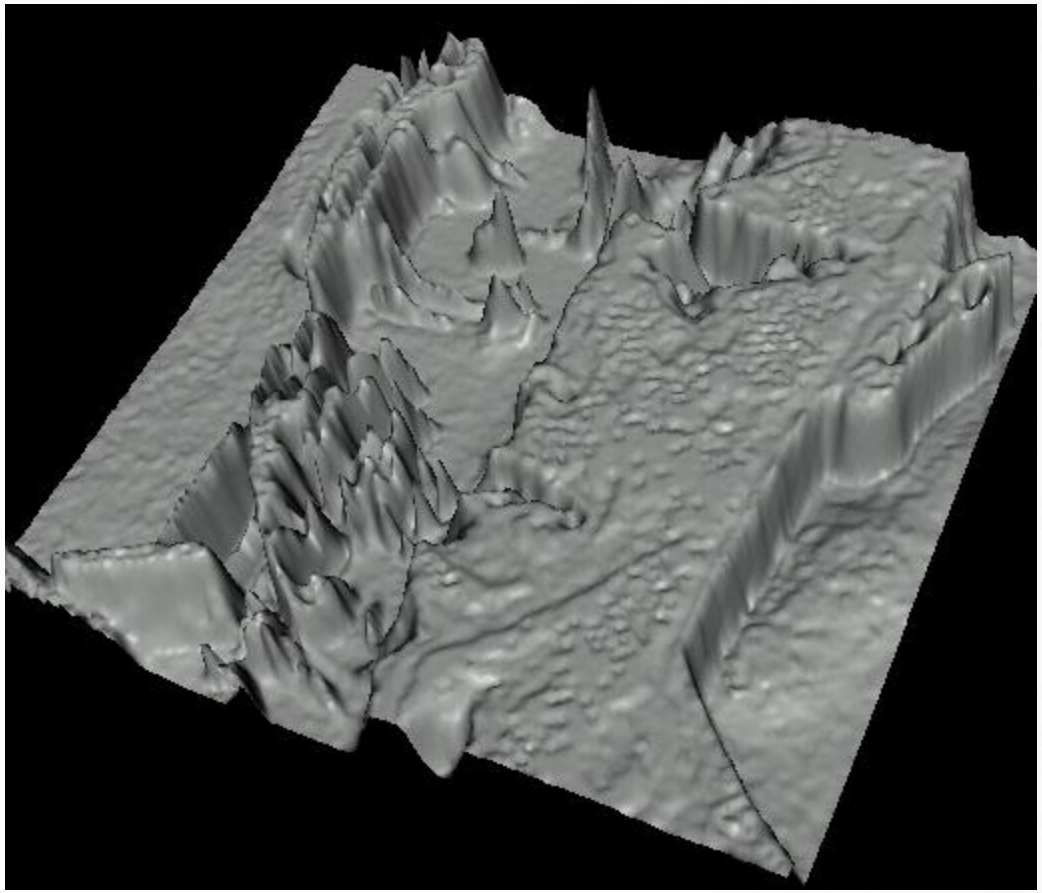
surface color discontinuity

illumination discontinuity

Edges are caused by a variety of factors

Slide credit: Kris Kitani

IMAGES AS FUNCTIONS…

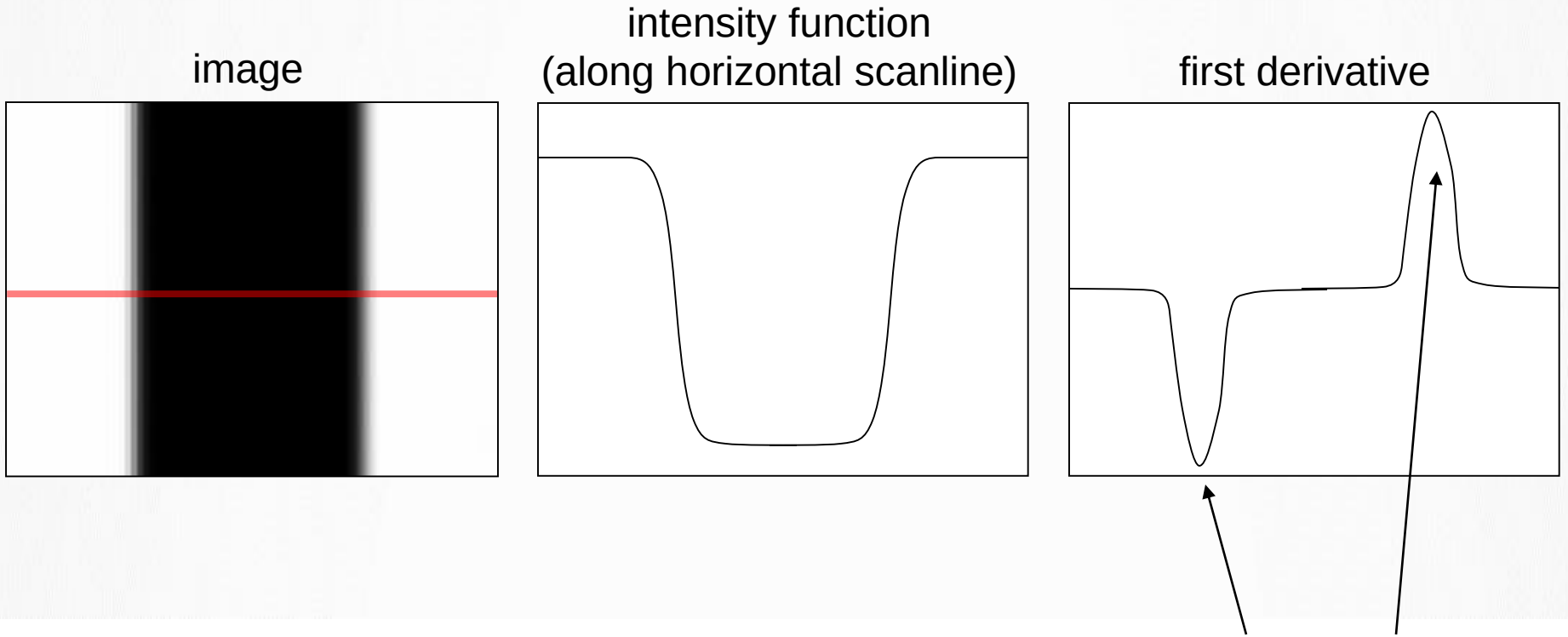


Edges look like steep cliffs

Slide credit: Kris Kitani

CHARACTERIZING EDGES

- An edge is a place of *rapid change* in the image intensity function



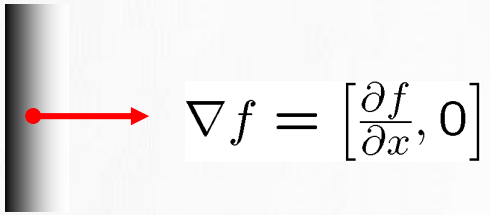
Source: L. Lazebnik

IMAGE GRADIENT

The *gradient* of an image:

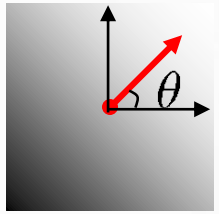
$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

The gradient points in the direction of most rapid increase in intensity



$$\nabla f = \left[\frac{\partial f}{\partial x}, 0\right]$$

$$\nabla f = \left[0, \frac{\partial f}{\partial y}\right]$$



$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

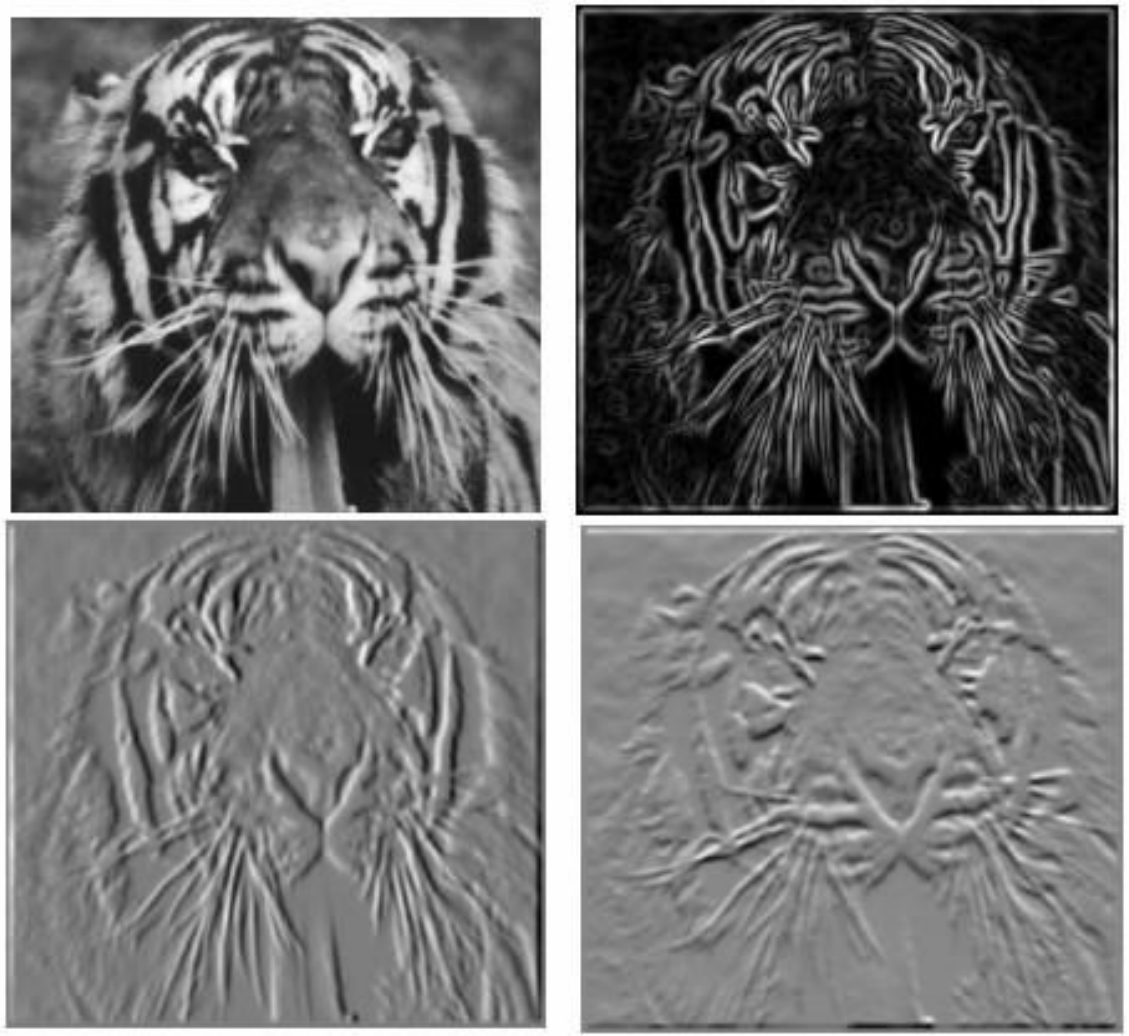
The *edge strength* is given by the gradient magnitude:

$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

The gradient direction is given by: $\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$

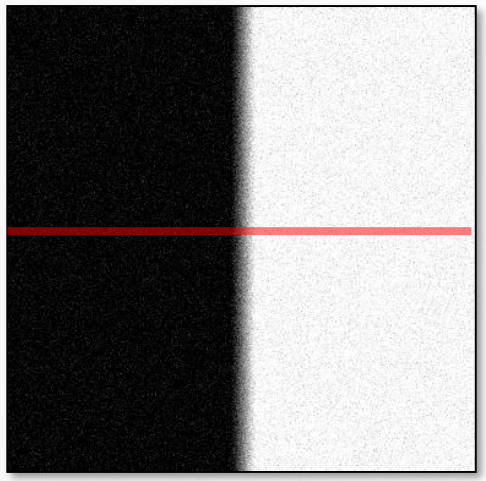
Source: Steve Seitz

IMAGE GRADIENT



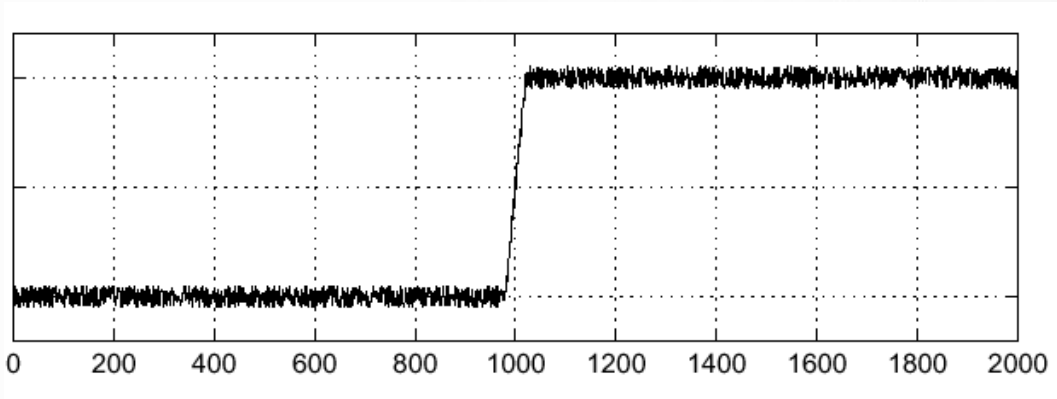
Source: L. Lazebnik

EFFECTS OF NOISE

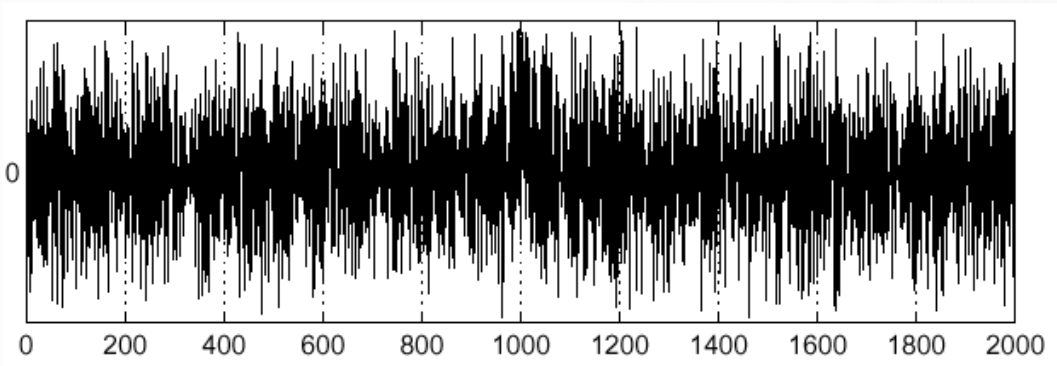


Noisy input image

$f(x)$



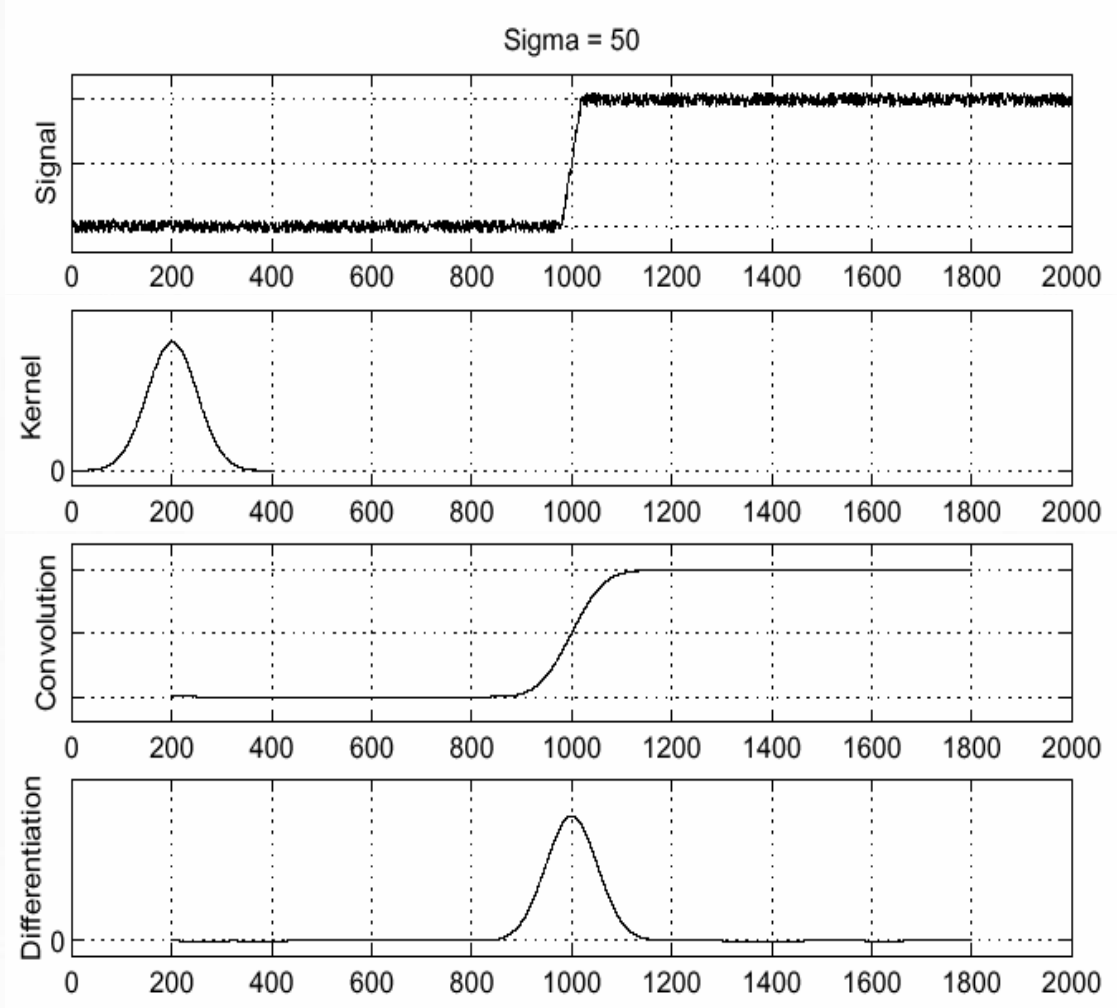
$\frac{d}{dx} f(x)$



Where is the edge?

Source: S. Seitz

SOLUTION: SMOOTH FIRST



f

h

$f * h$

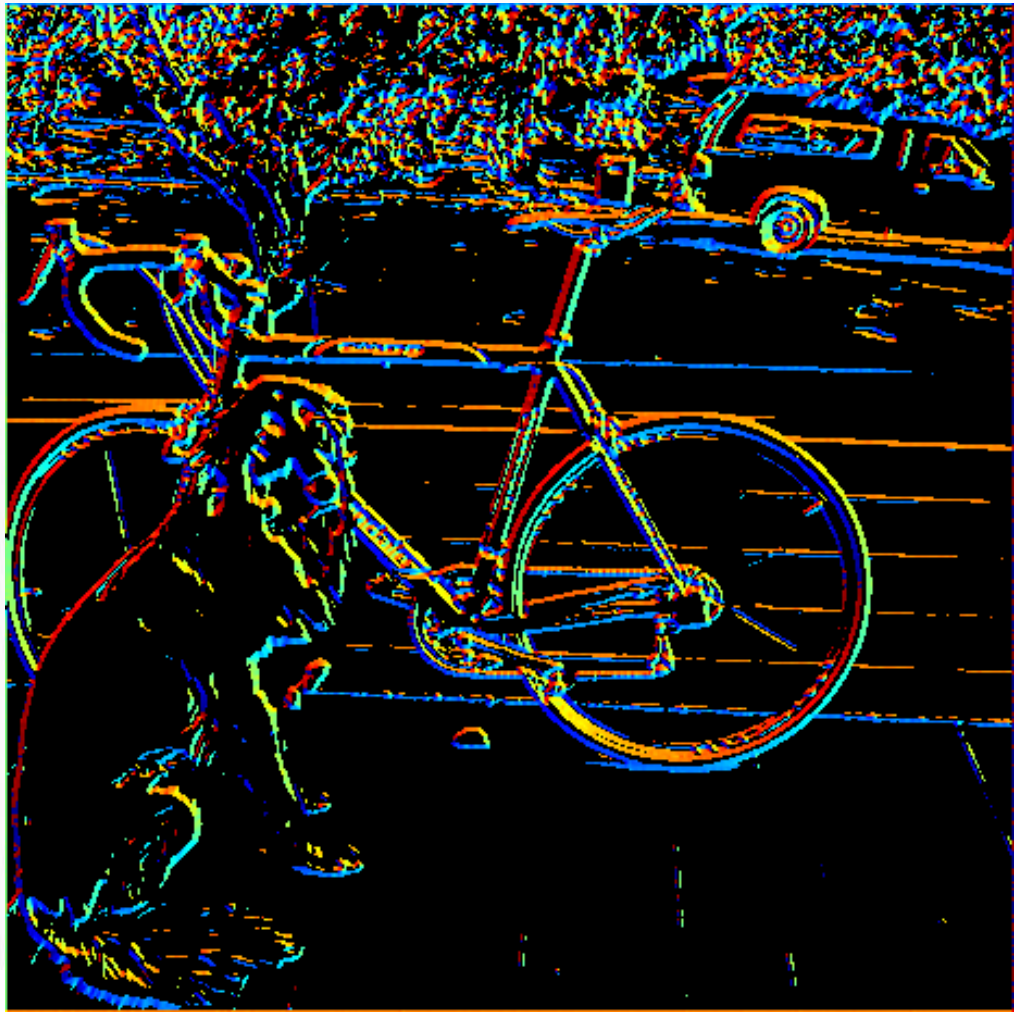
$\frac{d}{dx}(f * h)$

To find edges, look for peaks in $\frac{d}{dx}(f * h)$

Source: S. Seitz

GET ORIENTATION AT EACH PIXEL

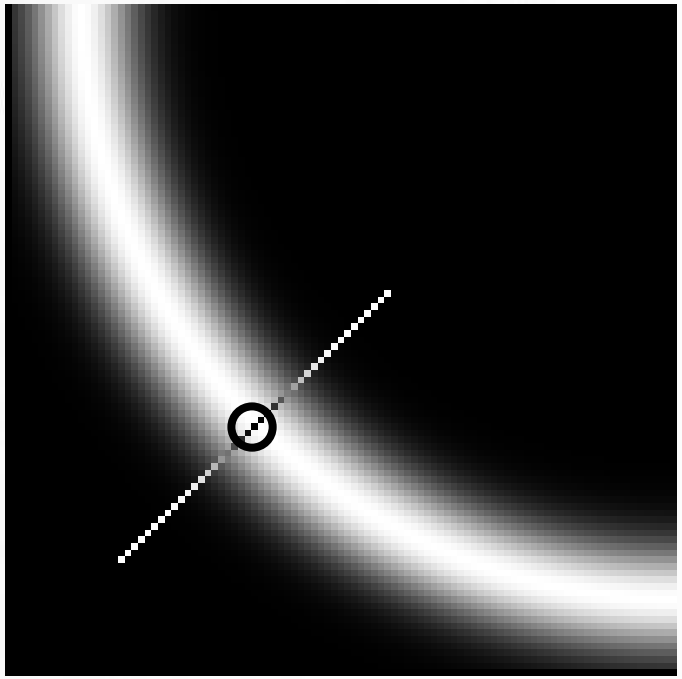
Get orientation (below, threshold at minimum gradient magnitude)

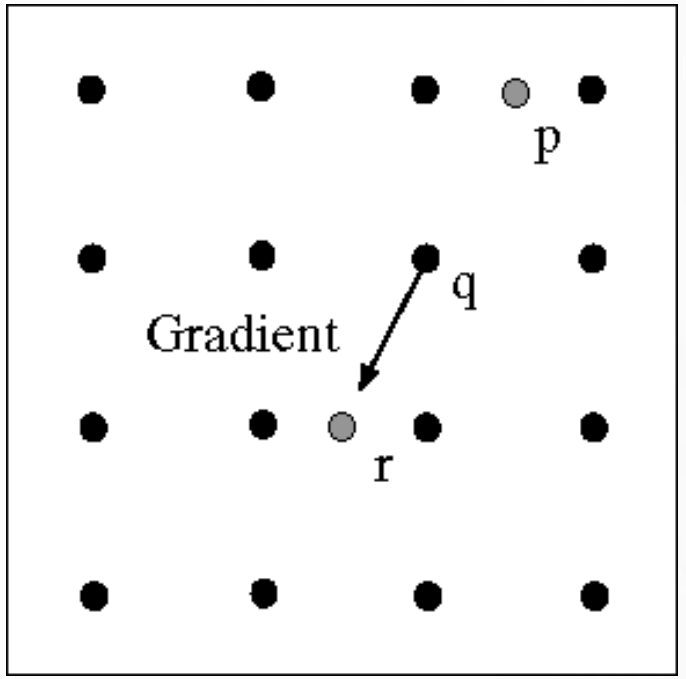


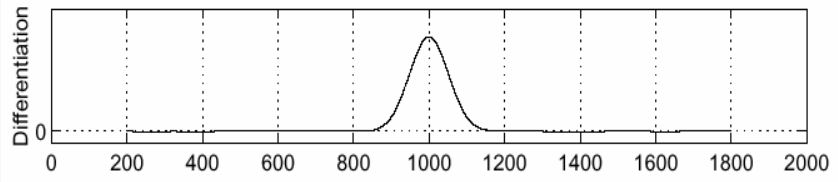
theta = atan2(gy, gx)

Gradient orientation angle

NON-MAXIMUM SUPRESSION







Check if pixel is local maximum along gradient direction

requires *interpolating* pixels p and r

BEFORE NON-MAX SUPPRESSION

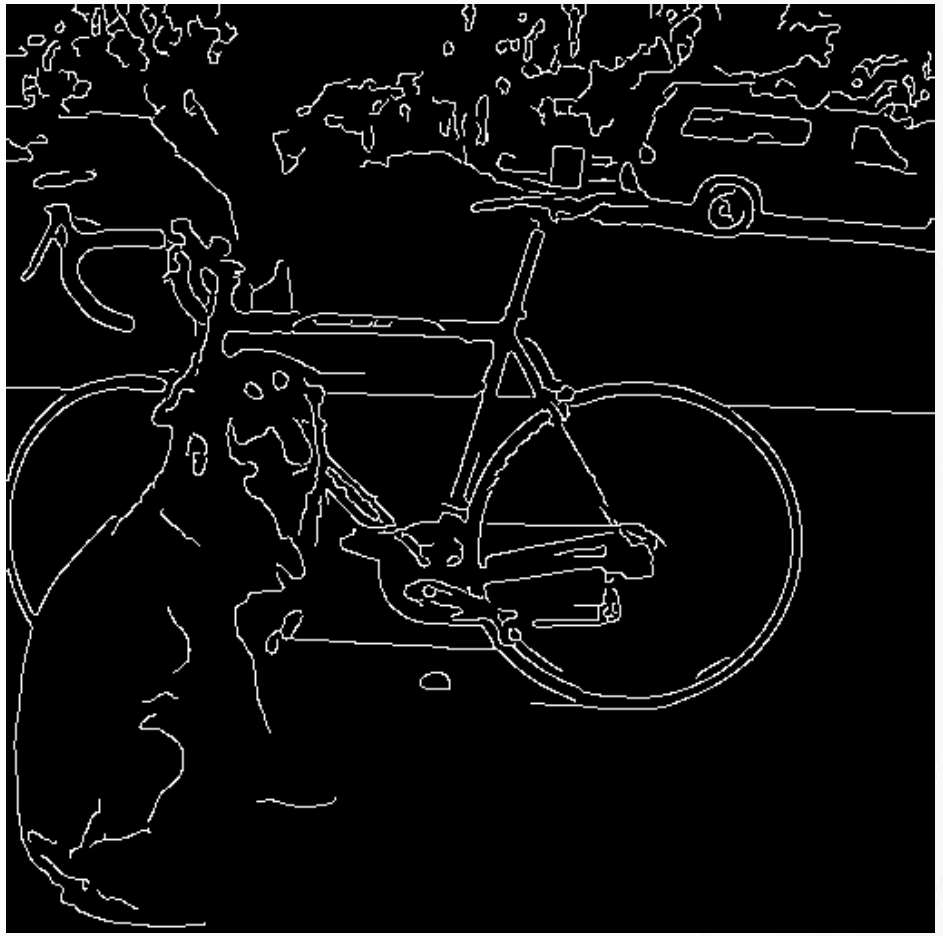
AFTER NON-MAX SUPPRESSION

THRESHOLDING EDGES

- Still some noise
- Only want strong edges
- 2 thresholds, 3 cases
  - R > T: strong edge
  - R < T but R > t: weak edge
  - R < t: no edge
- Why two thresholds?

CONNECTING EDGES

- Strong edges are edges!
- Weak edges are edges
  iff they connect to strong
- Look in some neighborhood
  (usually 8 closest)



CANNY EDGE DETECTOR

MATLAB: `edge(image,'canny')`

1. Filter image with derivative of Gaussian

2. Find magnitude and orientation of gradient

3. Non-maximum suppression

4. Linking and thresholding (hysteresis):

   Define two thresholds: low and high

   Use the high threshold to start edge curves and the low threshold to continue them

Source: D. Lowe, L. Fei-Fei, J. Redmon

CANNY EDGE DETECTOR

original

Canny with $\sigma = 1$

Canny with $\sigma = 2$

- The choice of σ depends on desired behavior
  - large σ detects “large-scale” edges
  - small σ detects fine edges

Source: S. Seitz

EDGES FROM AN IMAGE

HOUGH TRANSFORM

- Detcting lines from all edges
  - Generic framework for detecting a parametric model
  - Edges don’t have to be connected
  - Lines can be occluded
  - Key idea: edges **vote** for the possible models

IMAGE AND PARAMETER SPACE

variables

$$y = mx + b$$

parameters

variables

$$y - mx = b$$

parameters

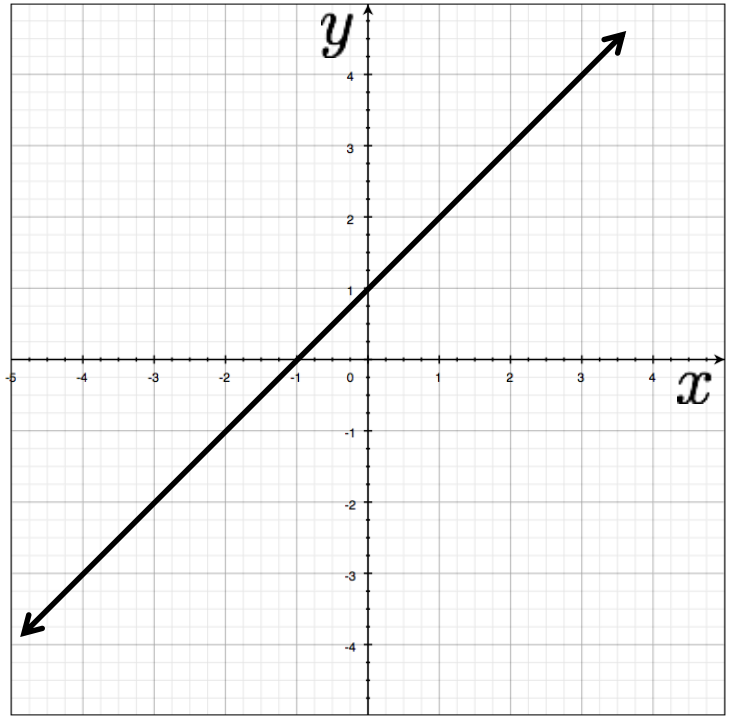
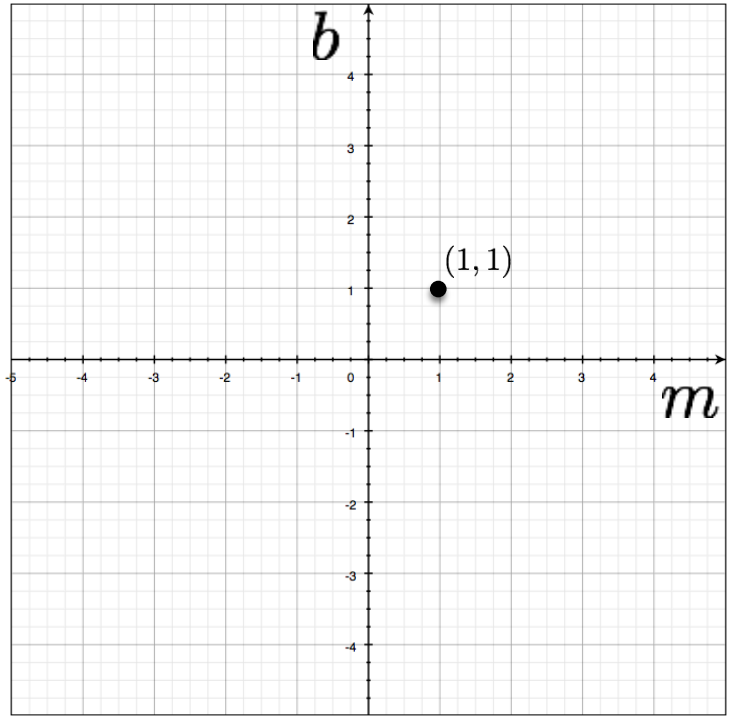


Image space

a line becomes a point



Parameter space

IMAGE AND PARAMETER SPACE

variables

$$y = mx + b$$

parameters

variables

$$y - mx = b$$

parameters

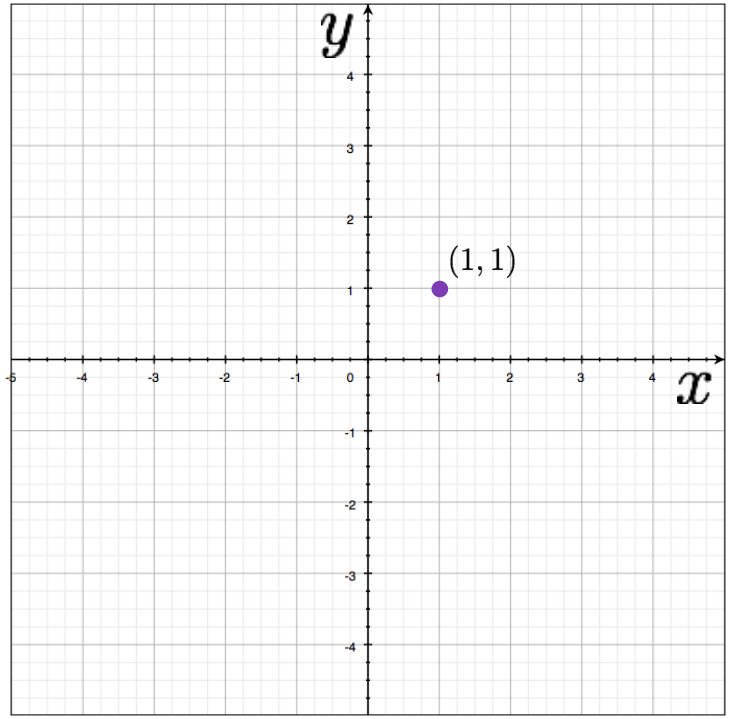
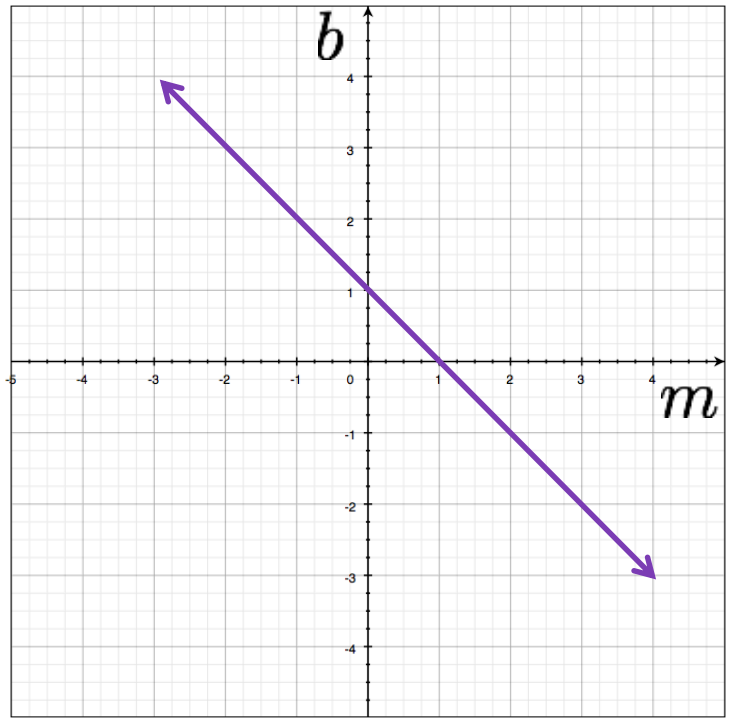


Image space

a point becomes a line



Parameter space

IMAGE AND PARAMETER SPACE

variables

$$y = mx + b$$

parameters

variables

$$y - mx = b$$

parameters

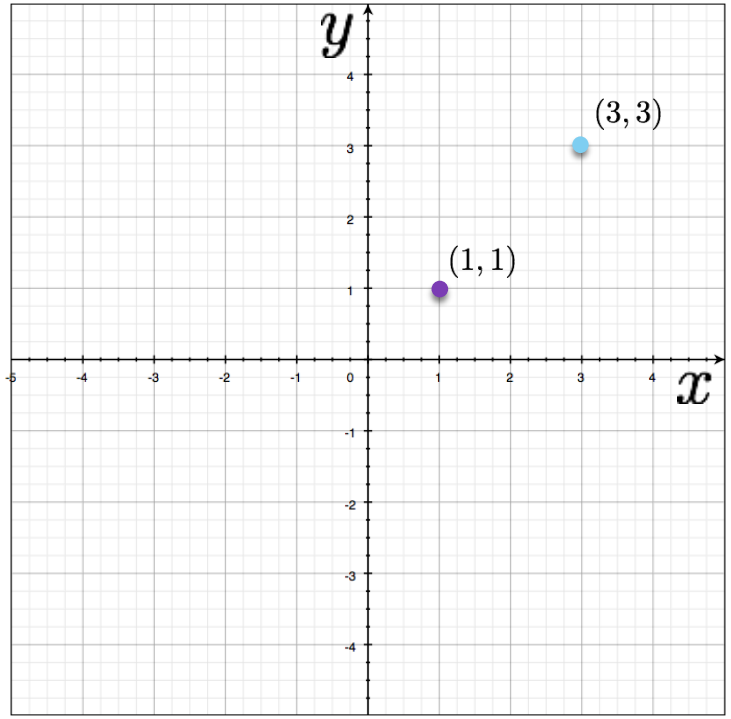
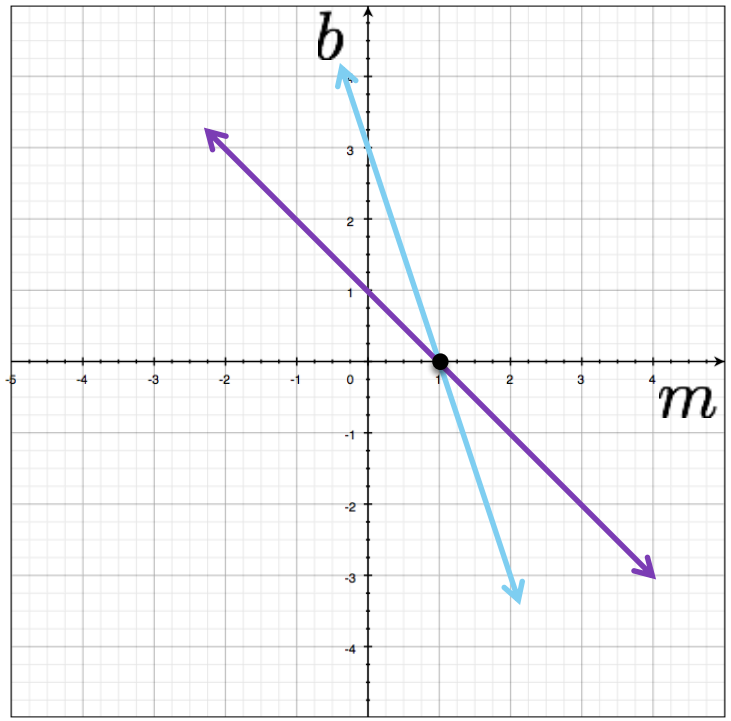


Image space

two points become ?



Parameter space

Line Detection by Hough Transform

Algorithm:

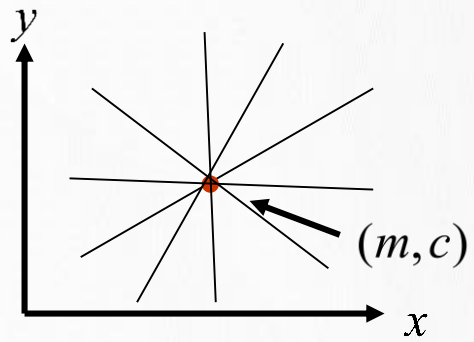
1. Quantize Parameter Space (m,c)

2. Create Accumulator Array $A(m,c)$

3. Set $A(m,c) = 0 \quad \forall m,c$

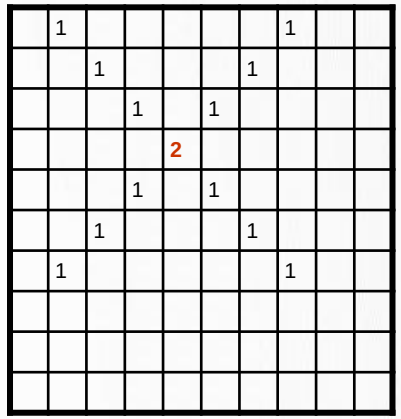
4. For each image edge (x_i, y_i)
   For each element in $A(m,c)$
   If (m,c) lies on the line: $c = -x_i m + y_i$
   Increment $A(m,c) = A(m,c) + 1$

5. Find local maxima in $A(m,c)$



Parameter Space

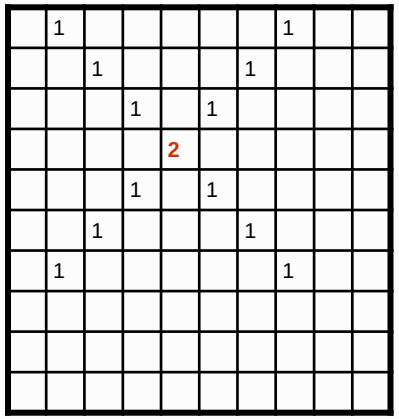
$A(m,c)$



PROBLEMS WITH PARAMETERIZATION

How big does the accumulator need to be for the parameterization (m, c) *?*

$A(m, c)$



The space of m is huge!

$$-\infty \leq m \leq \infty$$

The space of c is huge!

$$-\infty \leq c \leq \infty$$

BETTER PARAMETERIZATION

Use polar coordinates:

$$x\cos\theta + y\sin\theta = \rho$$

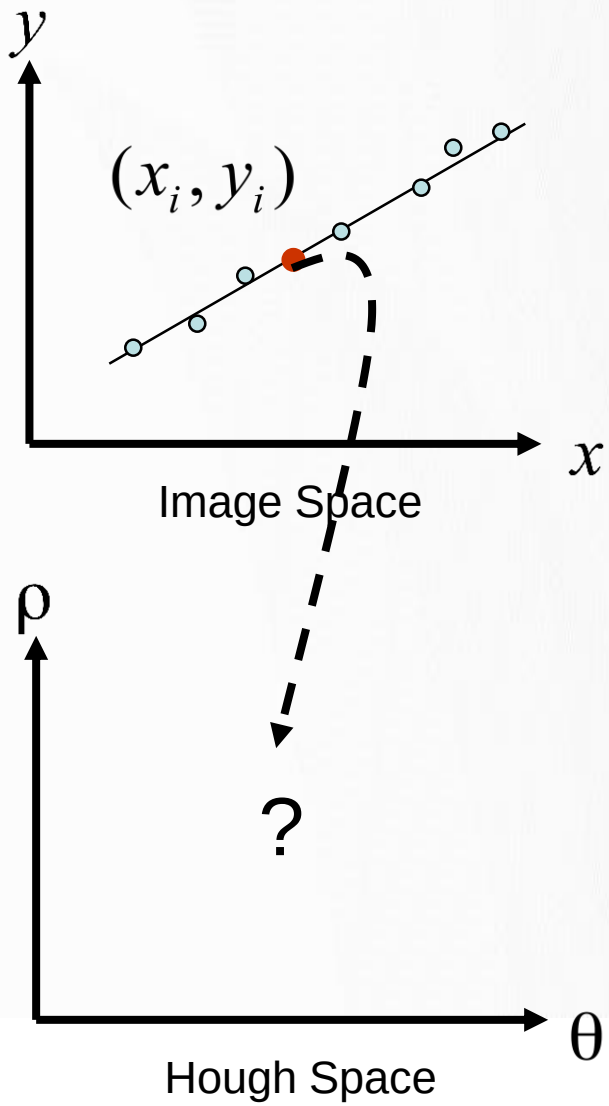
Given points (x_i, y_i) find (ρ, θ)

Hough Space Sinusoid

$$0 \le \theta \le 2\pi$$

$$0 \le \rho \le \rho_{max}$$

(Finite Accumulator Array Size)



POLAR COORDINATES

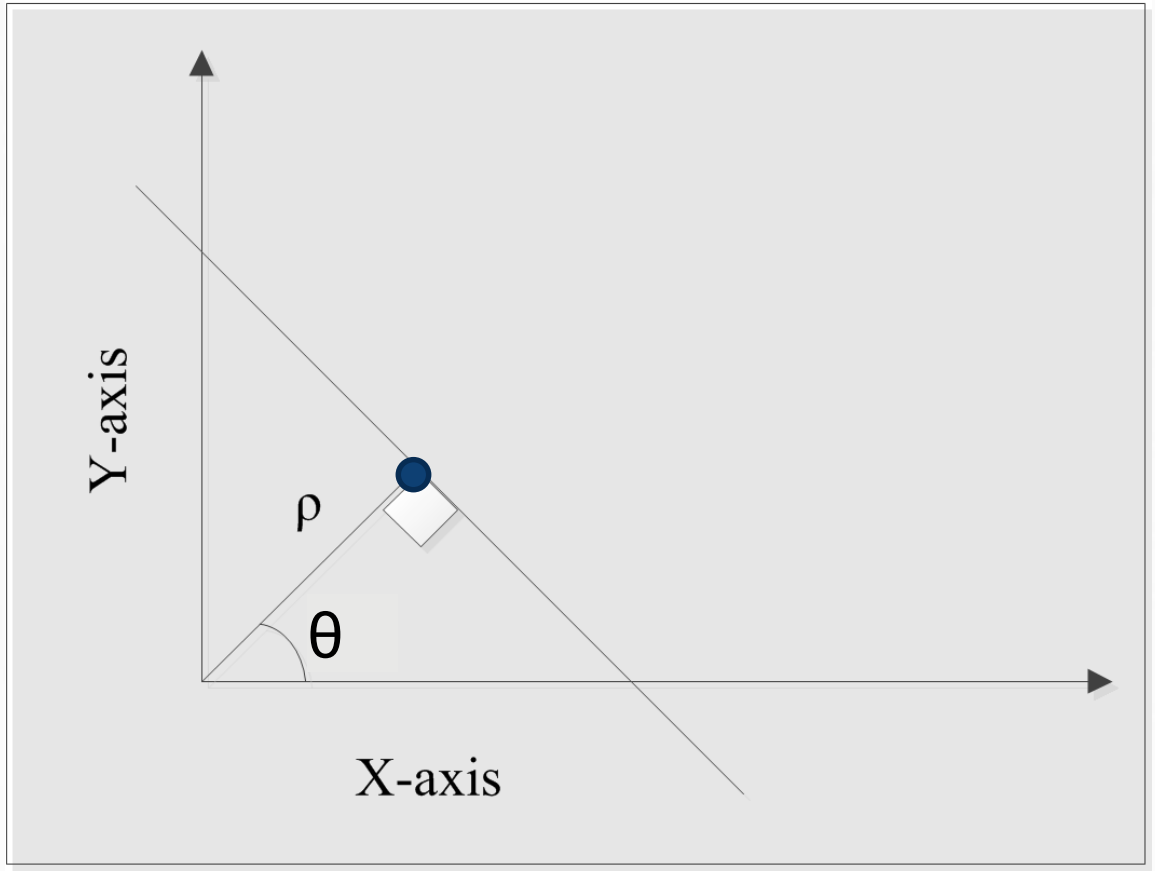


IMAGE AND PARAMETER SPACE

variables

$$y = mx + b$$

parameters

parameters

$$x\cos\theta + y\sin\theta = \rho$$

variables

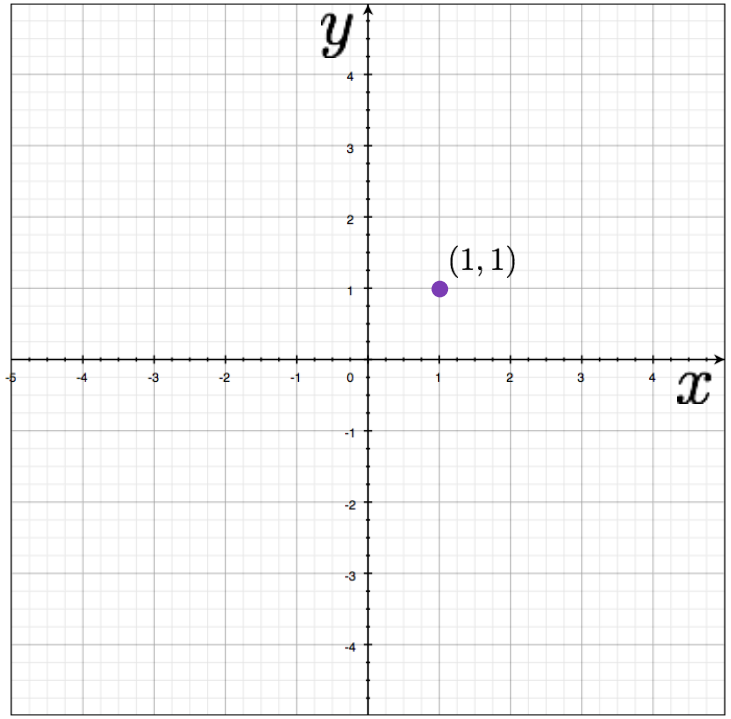
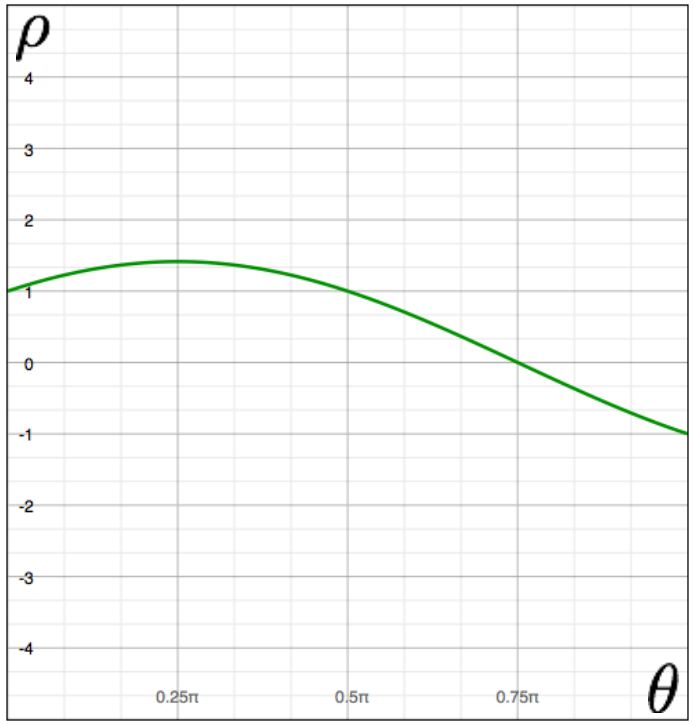


Image space

a point becomes a wave



Parameter space

IMAGE AND PARAMETER SPACE

variables

$$y = mx + b$$

parameters

$$x\cos\theta + y\sin\theta = \rho$$

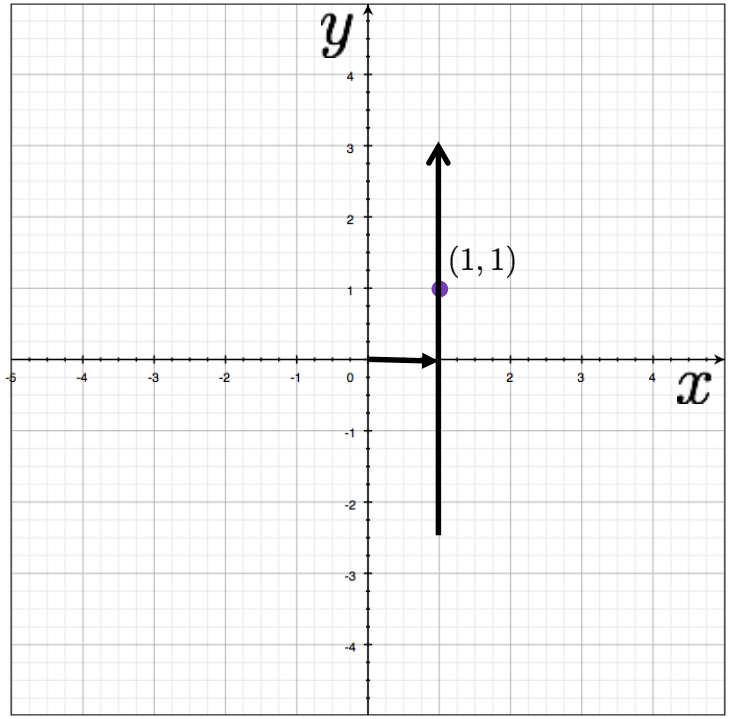
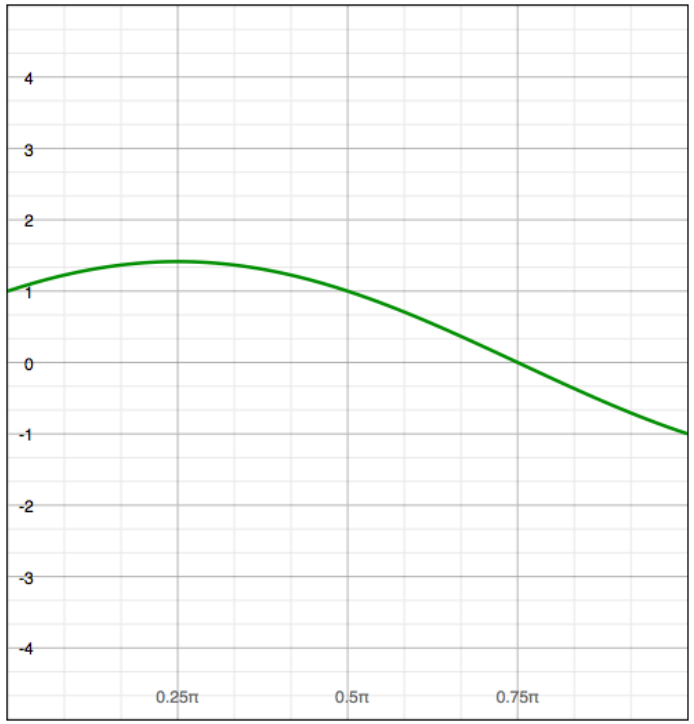


Image space

a line becomes?



Parameter space

IMAGE AND PARAMETER SPACE

variables

$$y = mx + b$$

parameters

$$x\cos\theta + y\sin\theta = \rho$$

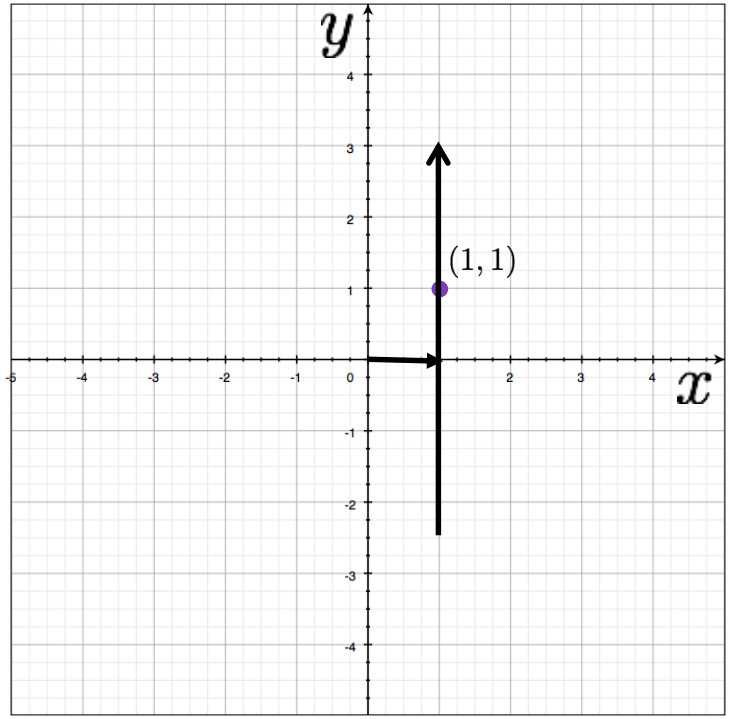
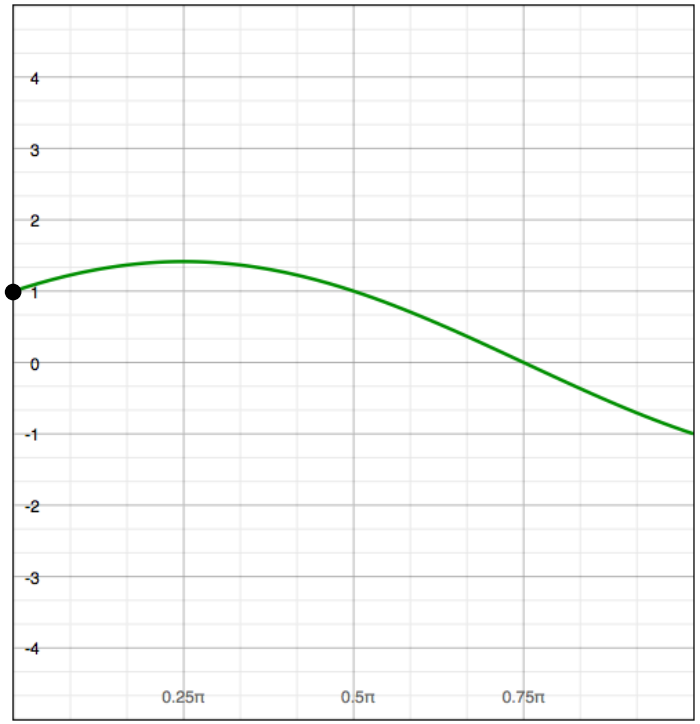


Image space

a line becomes a point



Parameter space

IMAGE AND PARAMETER SPACE

variables

$$y = mx + b$$

parameters

$$x\cos\theta + y\sin\theta = \rho$$

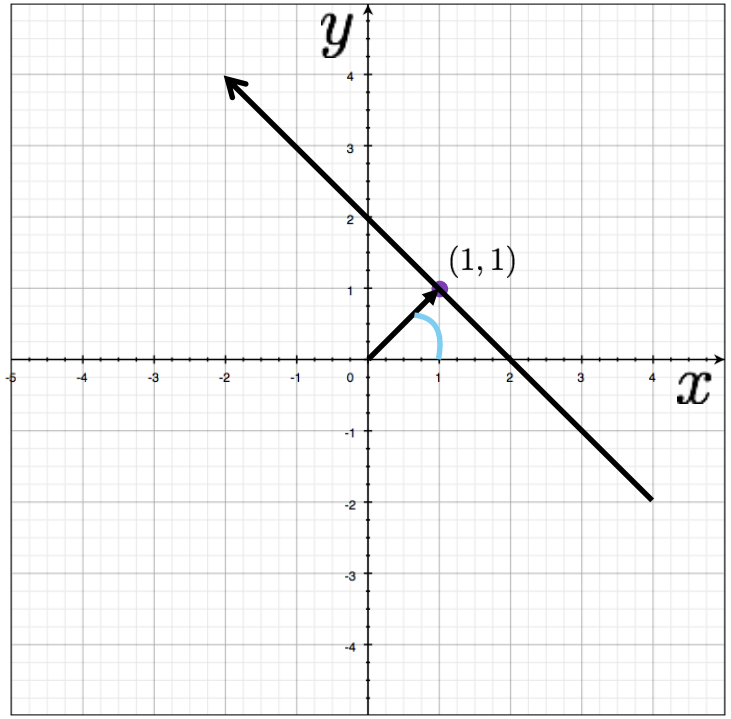
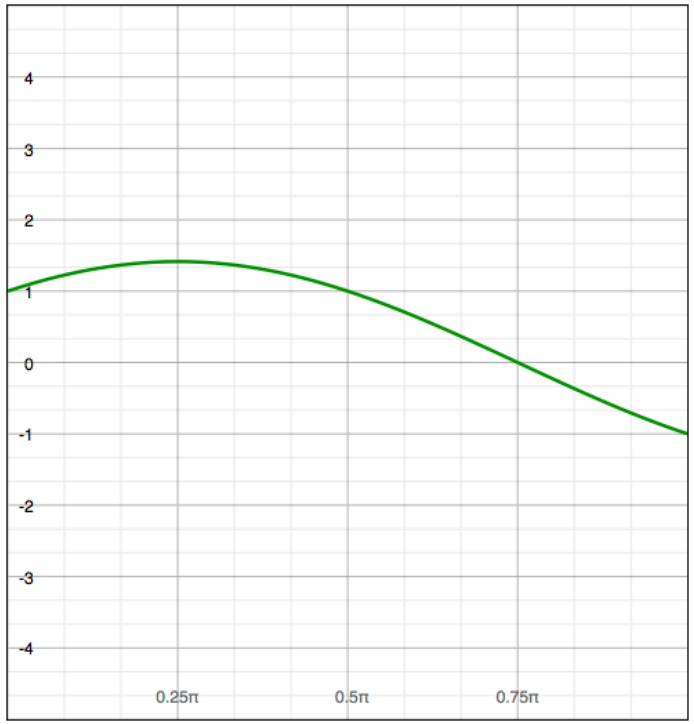


Image space

a line becomes?



Parameter space

IMAGE AND PARAMETER SPACE

variables

$$y = mx + b$$

parameters

$$x\cos\theta + y\sin\theta = \rho$$

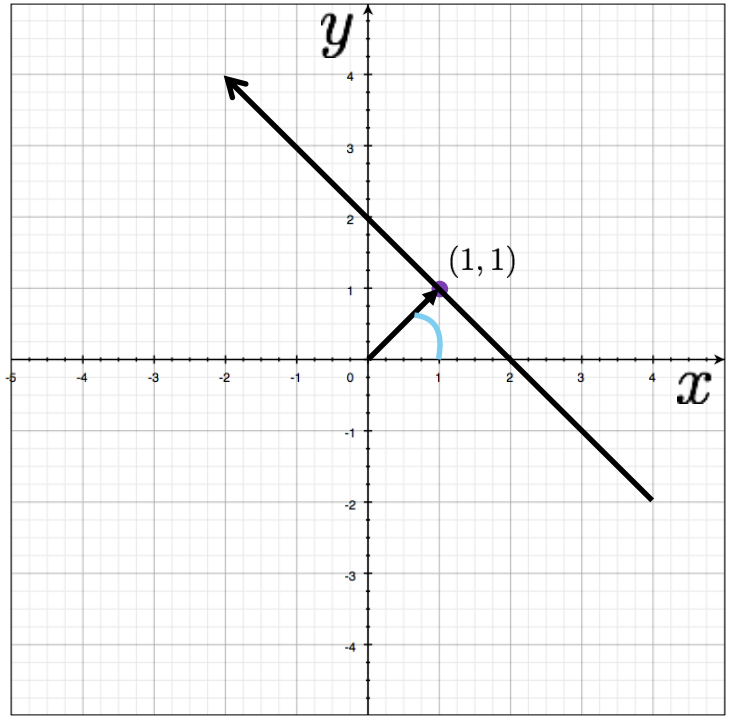
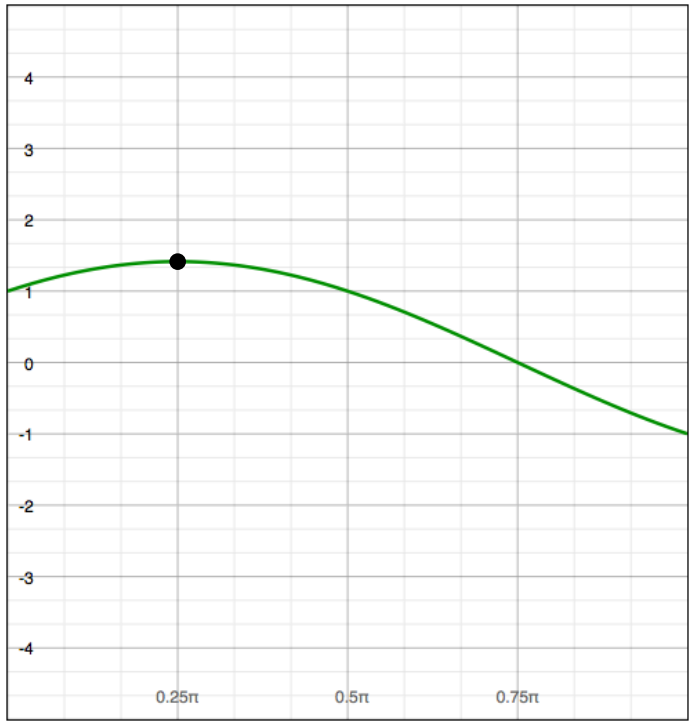


Image space

a line becomes a point



Parameter space

IMAGE AND PARAMETER SPACE

variables

$$y = mx + b$$

parameters

$$x\cos\theta + y\sin\theta = \rho$$

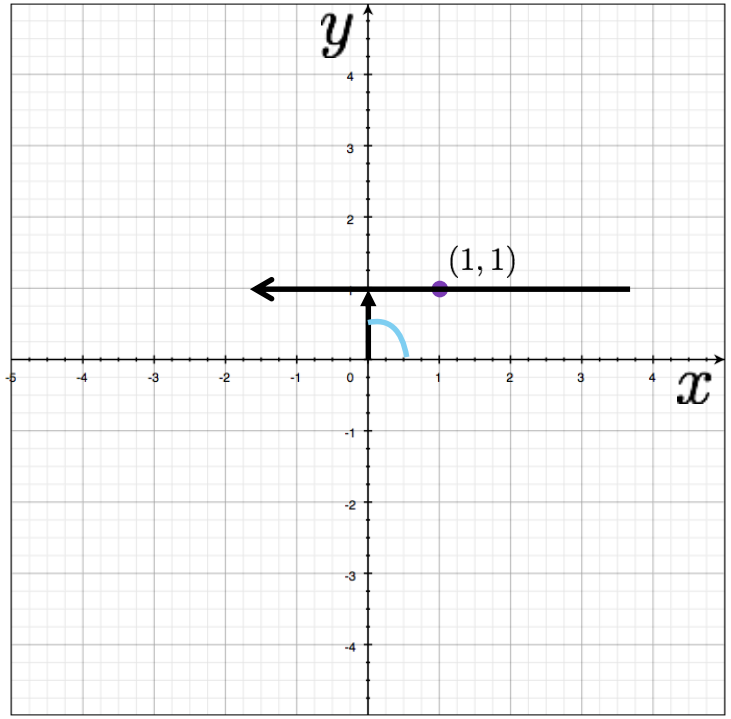
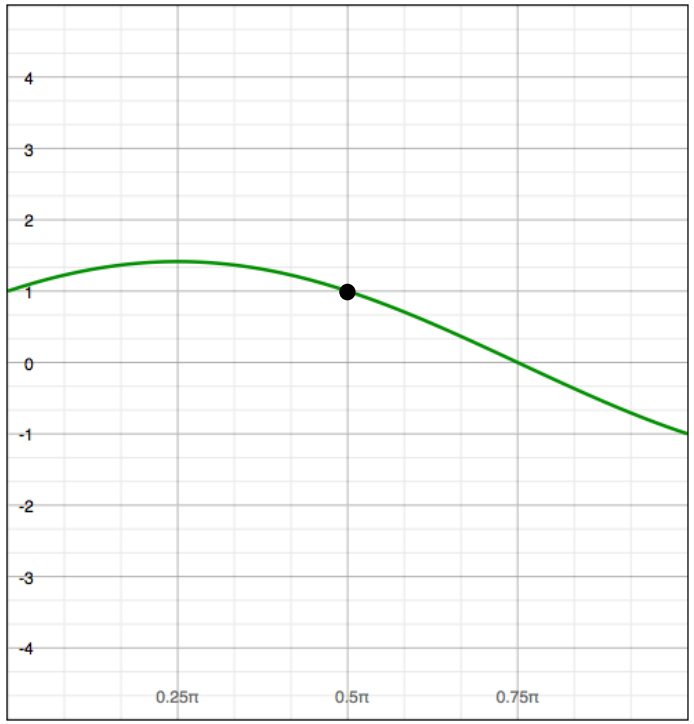


Image space

a line becomes a point



Parameter space

IMAGE AND PARAMETER SPACE

variables

$$y = mx + b$$

parameters

$$x\cos\theta + y\sin\theta = \rho$$

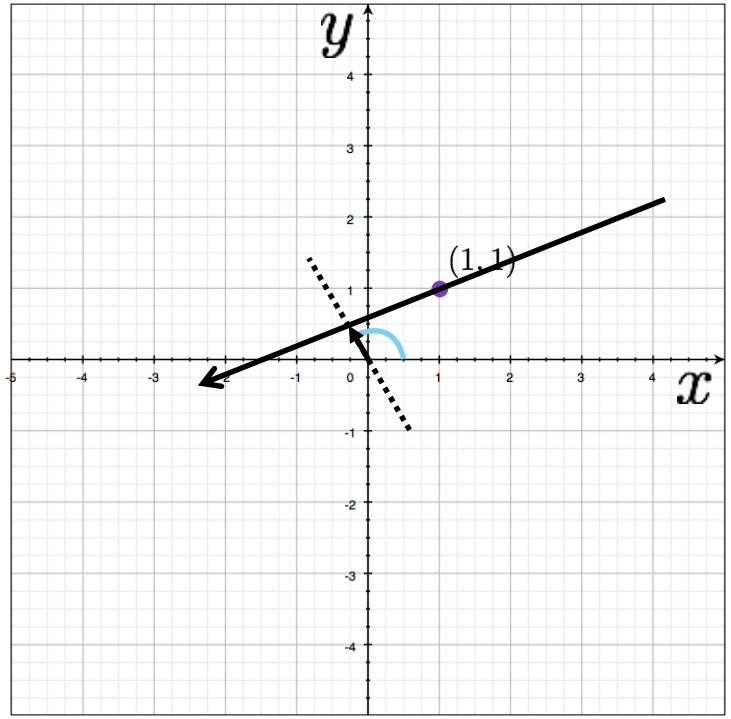
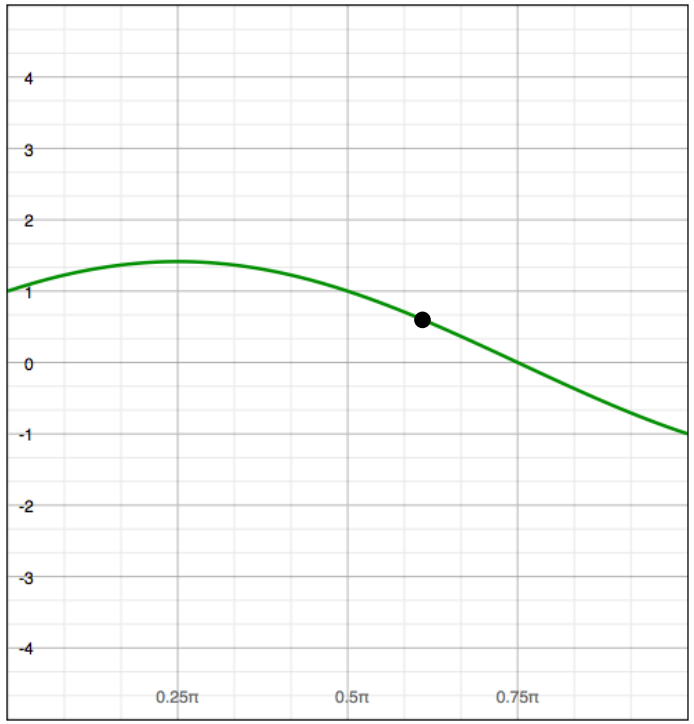


Image space

a line becomes a point



Parameter space

IMAGE AND PARAMETER SPACE

variables

$$y = mx + b$$

parameters

$$x\cos\theta + y\sin\theta = \rho$$

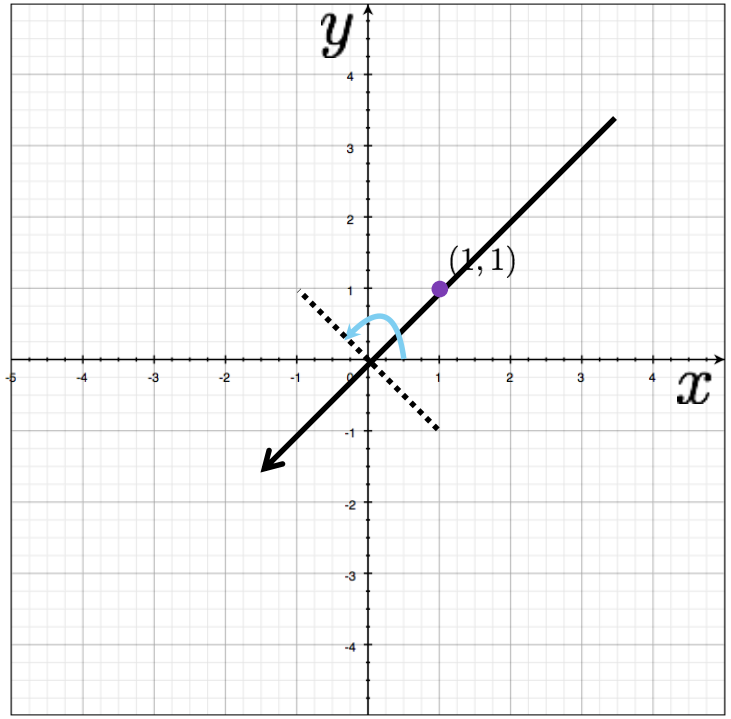
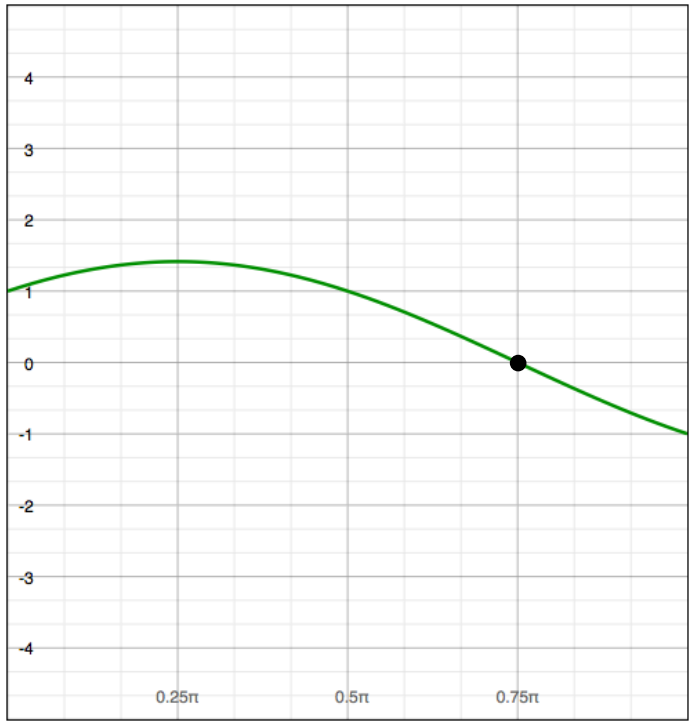


Image space

a line becomes a point



Parameter space

IMAGE AND PARAMETER SPACE

variables

$$y = mx + b$$

parameters

$$x\cos\theta + y\sin\theta = \rho$$

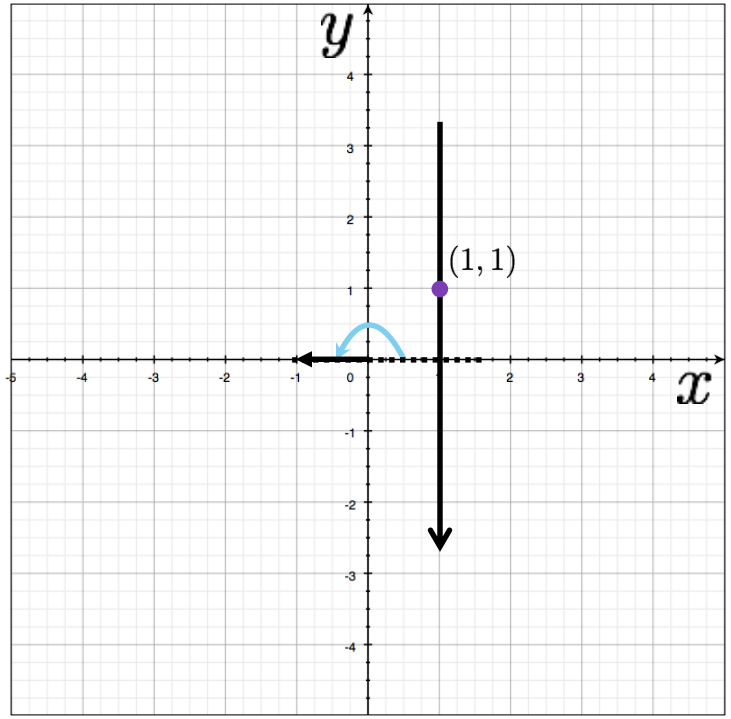
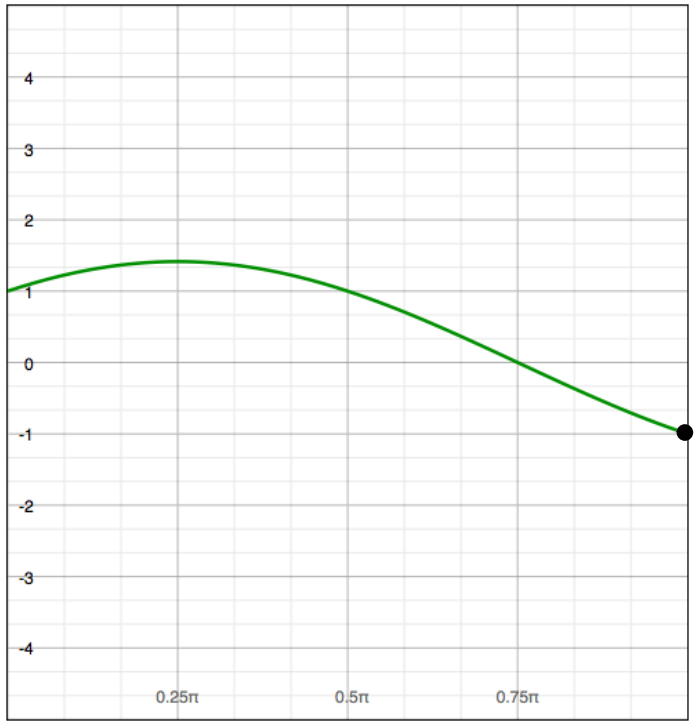


Image space

a line becomes a point



Parameter space

IMAGE AND PARAMETER SPACE

variables

$$y = mx + b$$

parameters

$$x\cos\theta + y\sin\theta = \rho$$

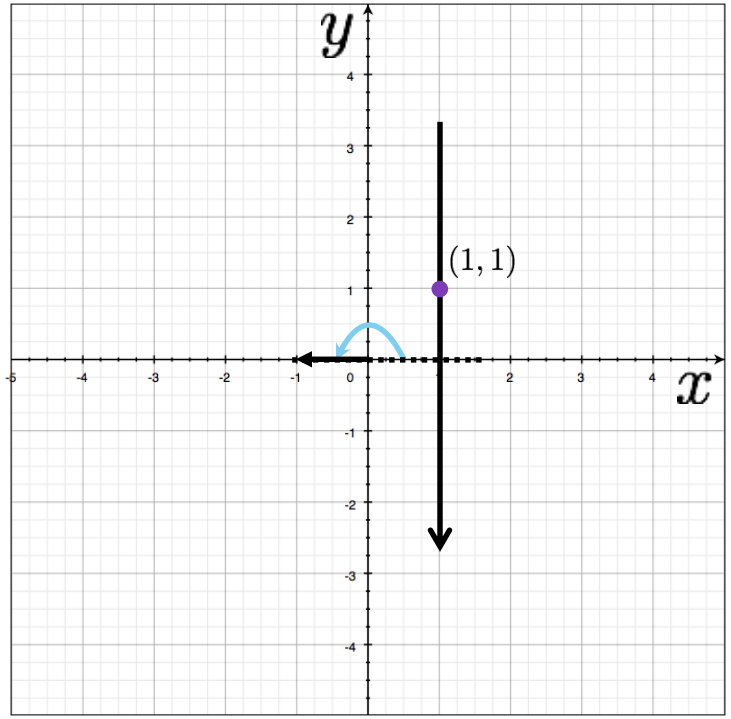
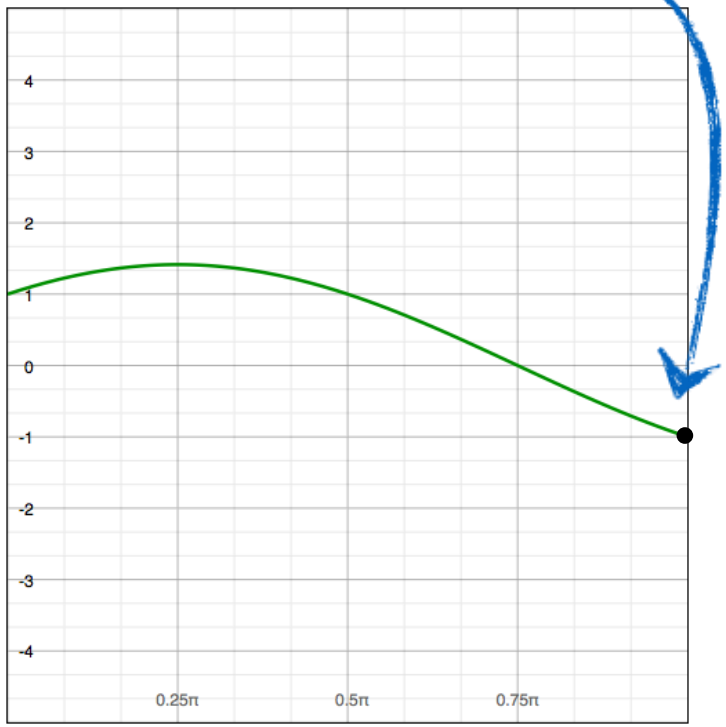


Image space

a line becomes a point

Wait …why is rho negative?



Parameter space

IMAGE AND PARAMETER SPACE

variables

$$y = mx + b$$

parameters

$$x\cos\theta + y\sin\theta = \rho$$

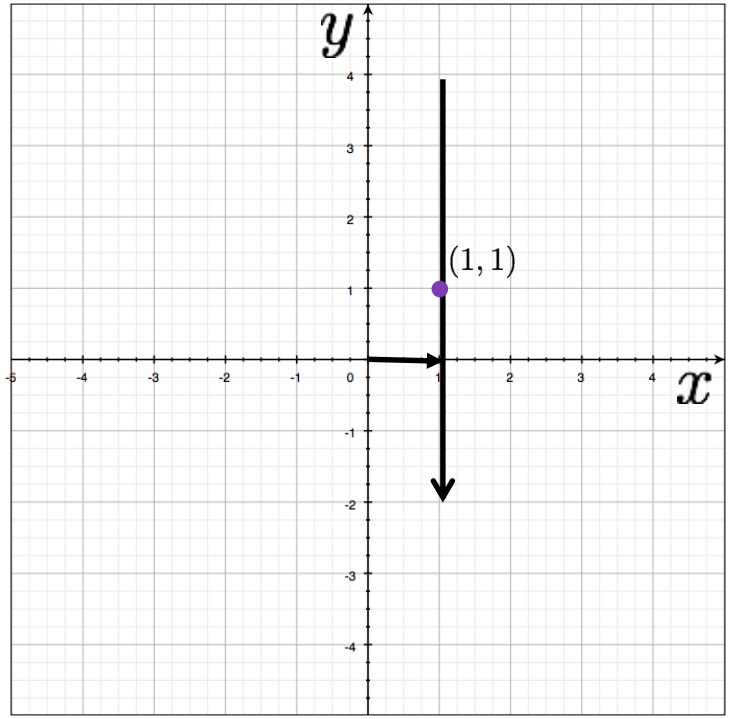
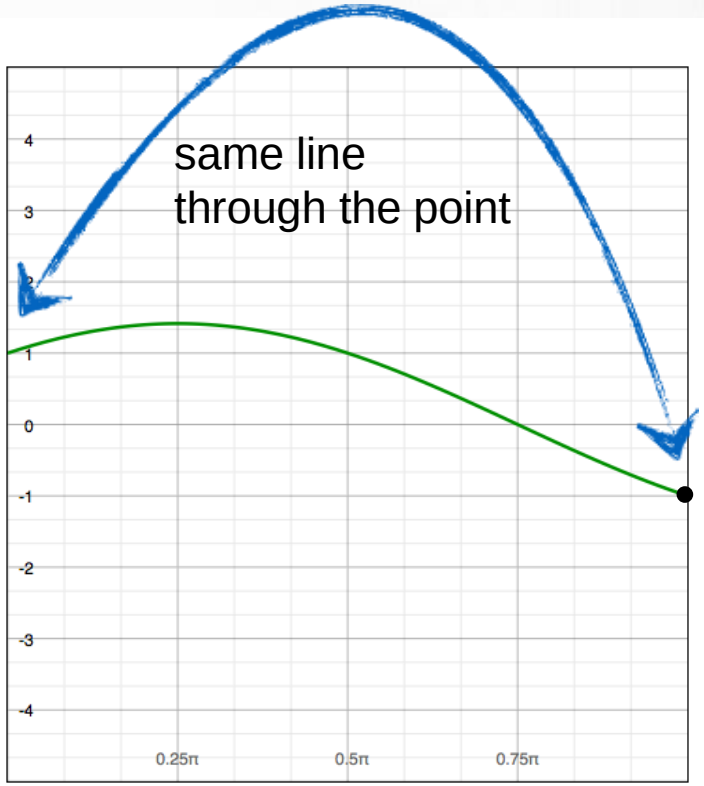


Image space

a line becomes a point

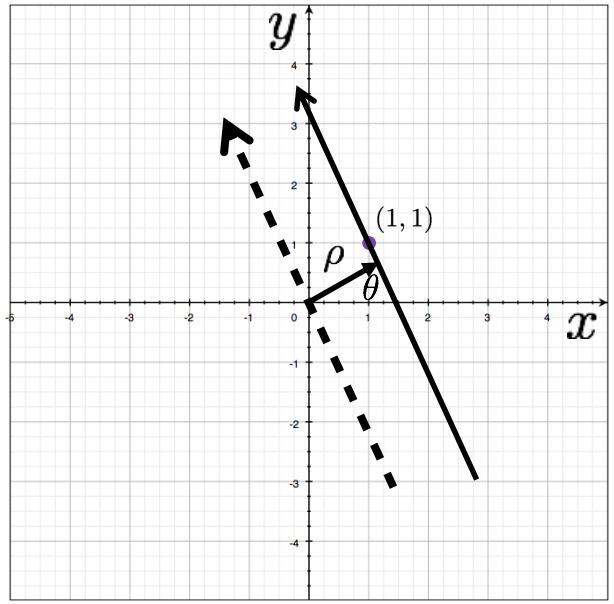


Parameter space

There are two ways to write the same line:

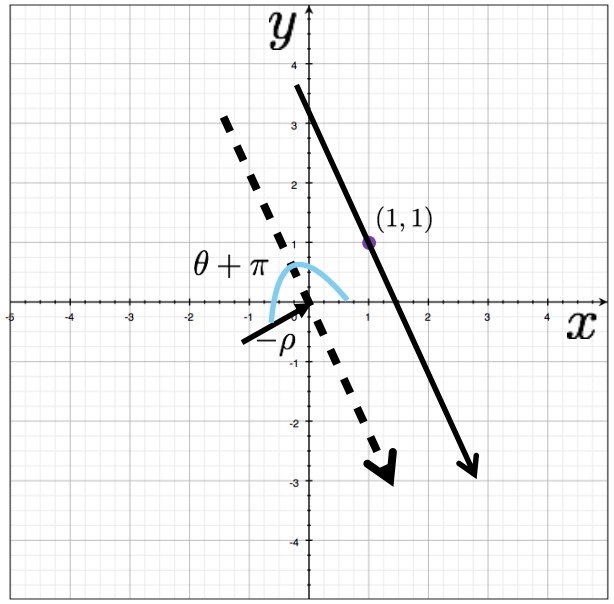
Positive rho version:

$$x\cos\theta + y\sin\theta = \rho$$



Negative rho version:

$$x\cos(\theta+\pi) + y\sin(\theta+\pi) = -\rho$$



Recall:

$$\sin(\theta) = -\sin(\theta+\pi)$$
$$\cos(\theta) = -\cos(\theta+\pi)$$

IMAGE AND PARAMETER SPACE

variables

$$y = mx + b$$

parameters

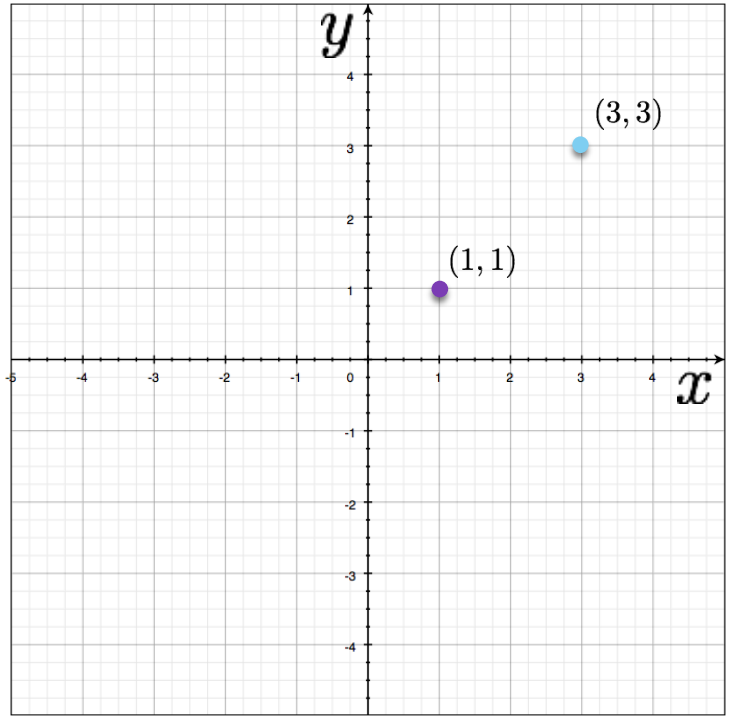
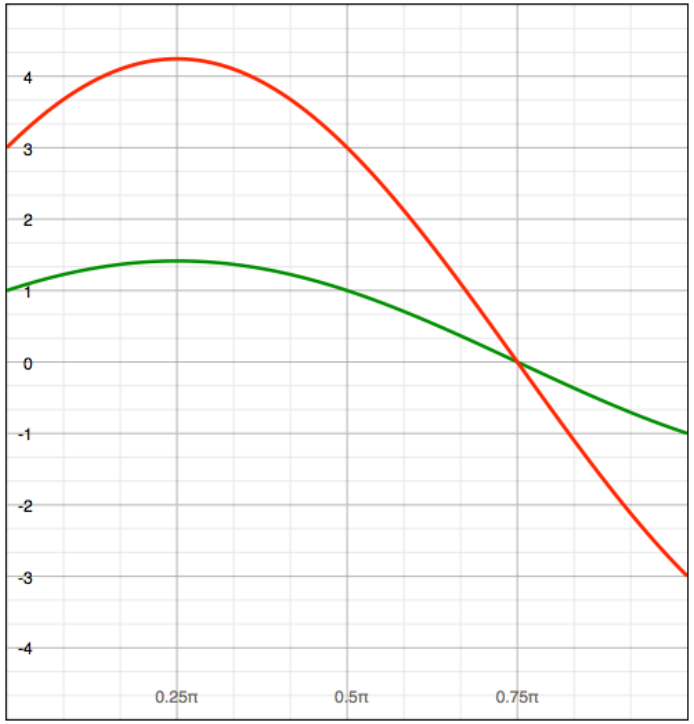


Image space

two points become ?



Parameter space

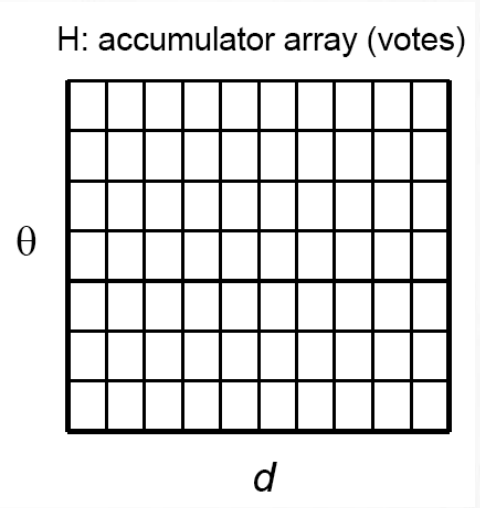
IMPLEMENTATION

1. Initialize accumulator H to all zeros

2. For each edge point (x,y) in the image
   For θ = 0 to 180
   ρ = x cos θ + y sin θ
   H(θ, ρ) = H(θ, ρ) + 1
   end
   end

3. Find the value(s) of (θ, ρ) where H(θ, ρ) is a local maximum

4. The detected line in the image is given by
   ρ = x cos θ + y sin θ



NOTE: Watch your coordinates. Image origin is top left!

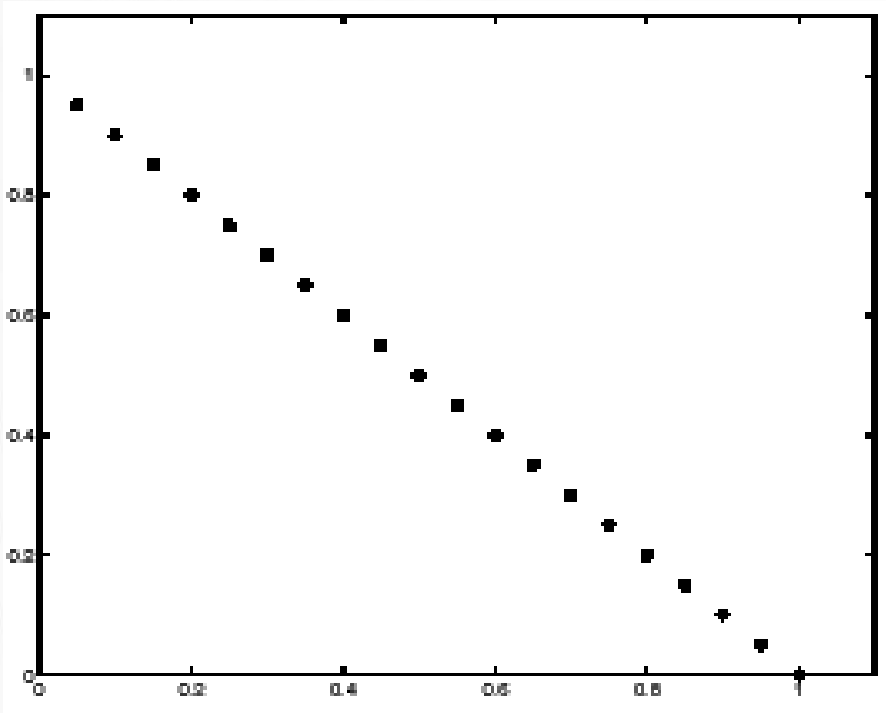
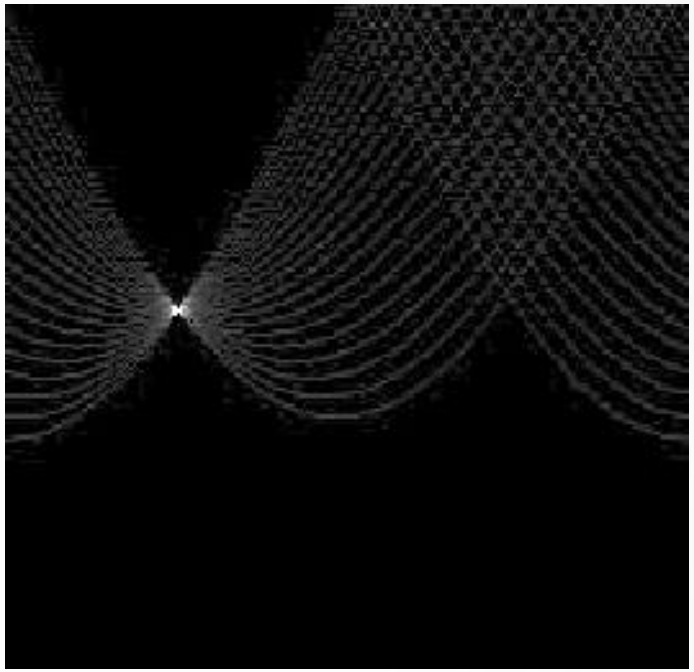


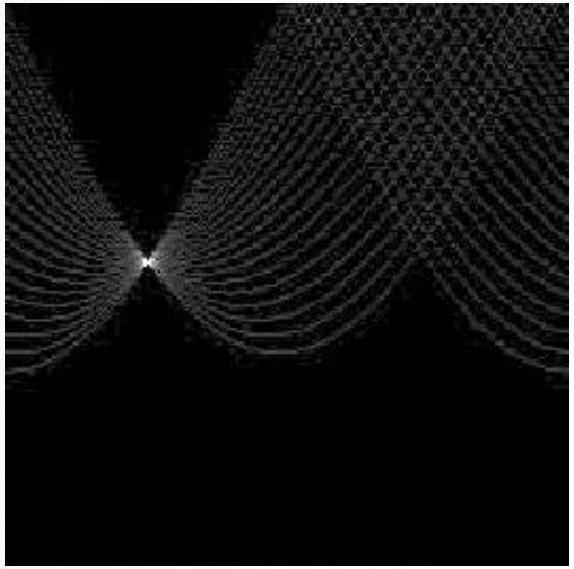
Image space



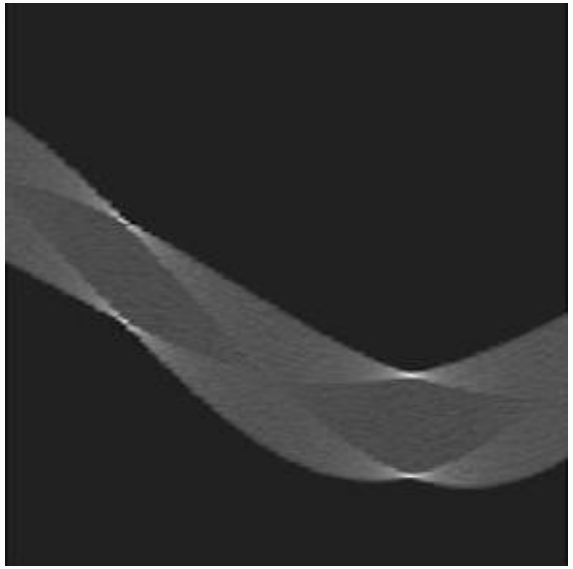
Votes

Basic shapes

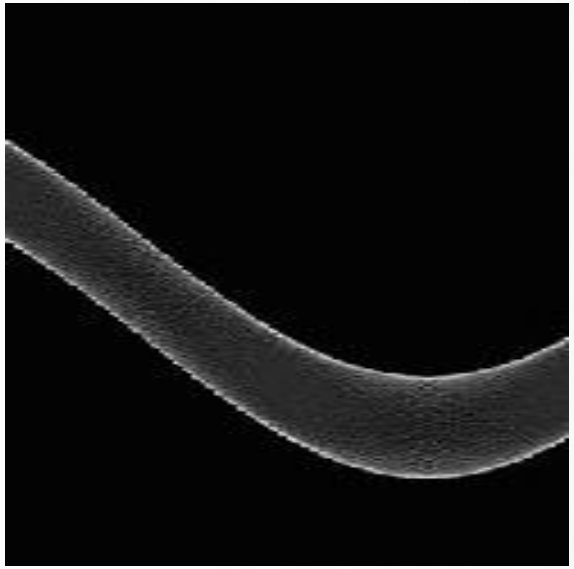
(in parameter space)



line



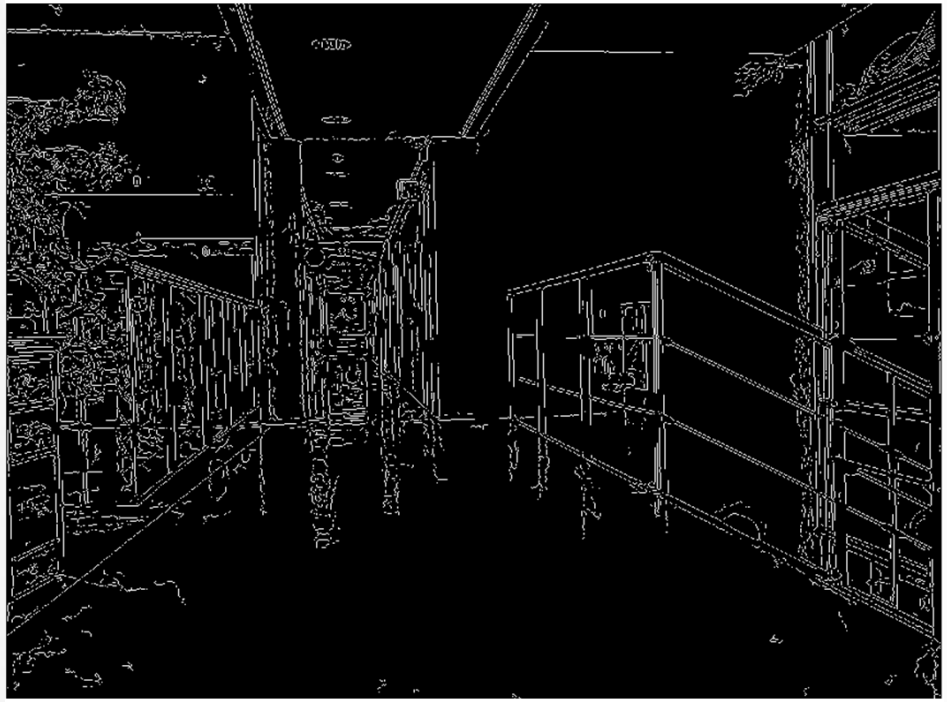
rectangle

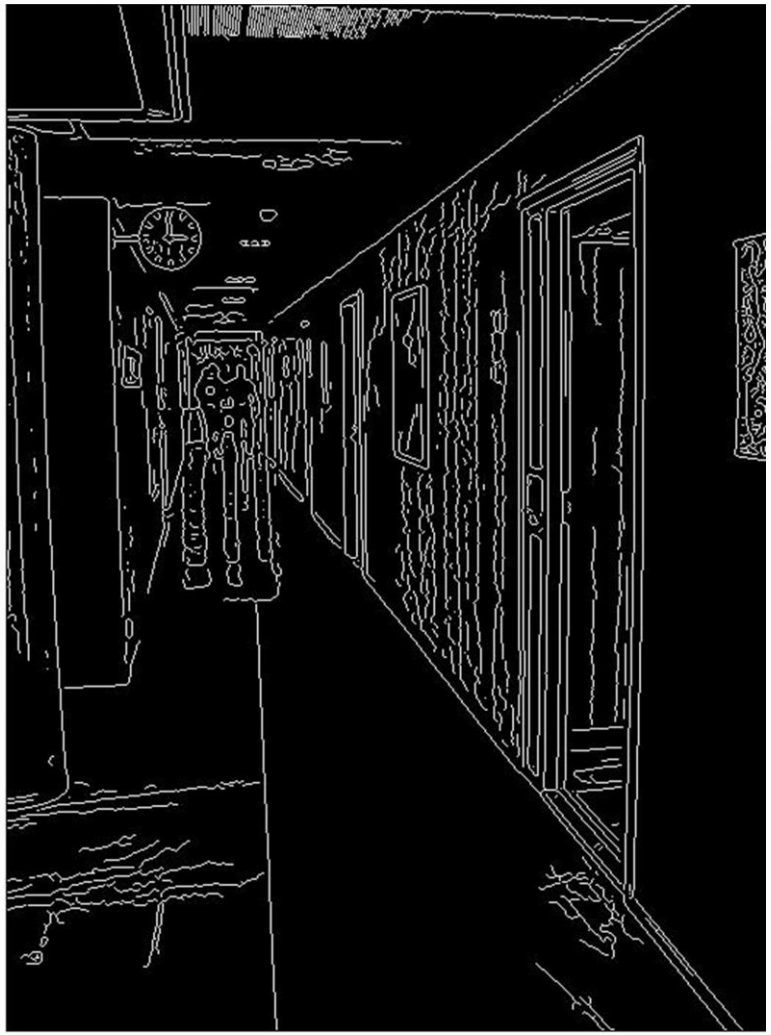


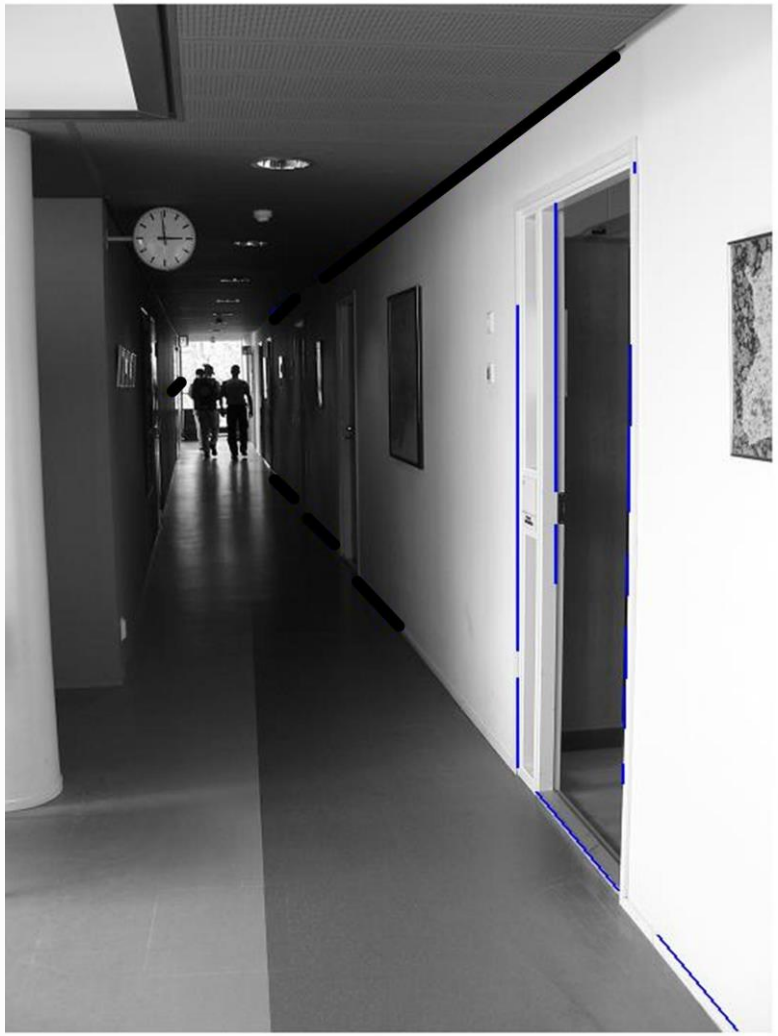
circle

HOUGH TRANSFORM

- Why don’t we find all lines existing in the image?








60˚ 10 1.2 N, 24˚ 57 18 E