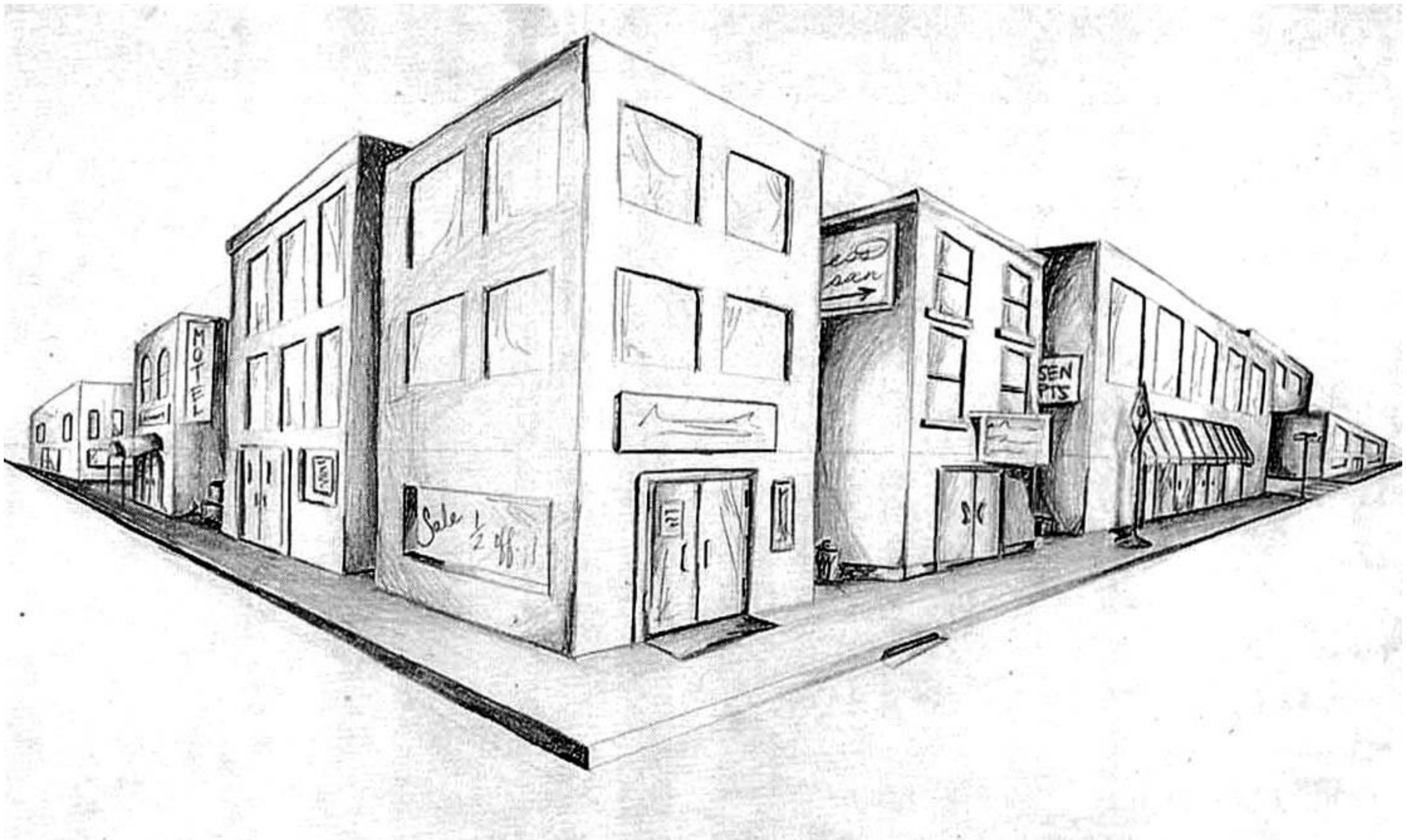
Hough Transform (cont'd)
Detecting corners



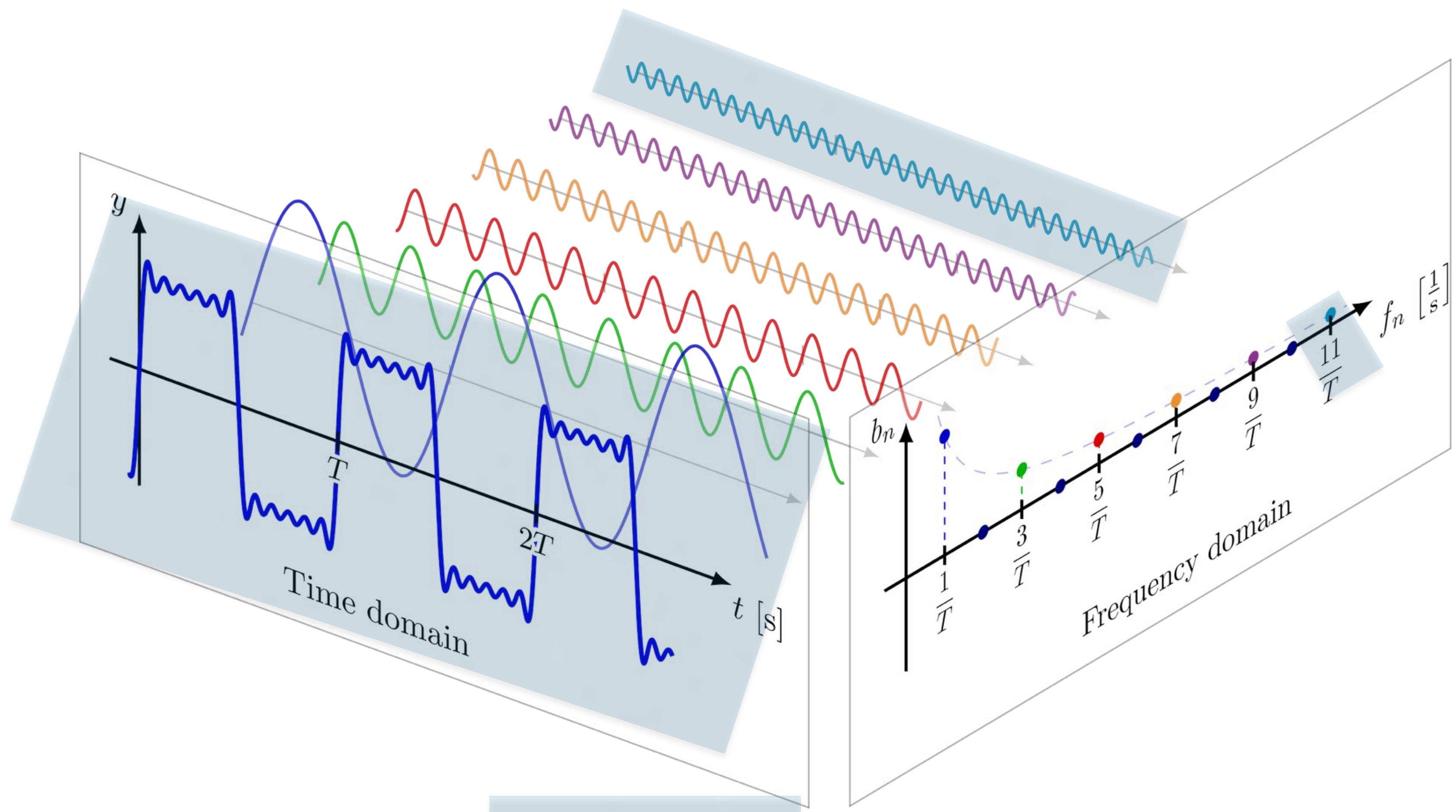
ICS 483 Computer Vision
Fall 2025, Lecture 7

Announcement

- Quiz 1 grade released. If you have not received the grade, the TA has concern about your submission (using LLM without claiming, incorrect steps lead to correct answers, etc.). Please come to talk to me.

- ICCV will be hosted here Oct. 19 - 23. Will have no classes during that week. Free student day pass to attend the conference. Details to be announced.

Recap: FT



$$\begin{bmatrix} F(0) \\ F(1) \\ F(2) \\ F(3) \\ \vdots \\ F(N-1) \end{bmatrix} = \begin{bmatrix} W^0 & W^0 & W^0 & W^0 & \cdots & W^0 \\ W^0 & W^1 & W^2 & W^3 & \cdots & W^{N-1} \\ W^0 & W^2 & W^4 & W^6 & \cdots & W^{N-2} \\ W^0 & W^3 & W^6 & W^9 & \cdots & W^{N-3} \\ \vdots & & & & \ddots & \vdots \\ W^0 & W^{N-1} & W^{N-2} & W^{N-3} & \cdots & W^1 \end{bmatrix} \begin{bmatrix} f(0) \\ f(1) \\ f(2) \\ f(3) \\ \vdots \\ f(N-1) \end{bmatrix}$$

Recap

- Finding boundaries.
- Line fitting.
- **Line parameterizations.**
- Hough transform.
- Hough circles.
- Some applications.

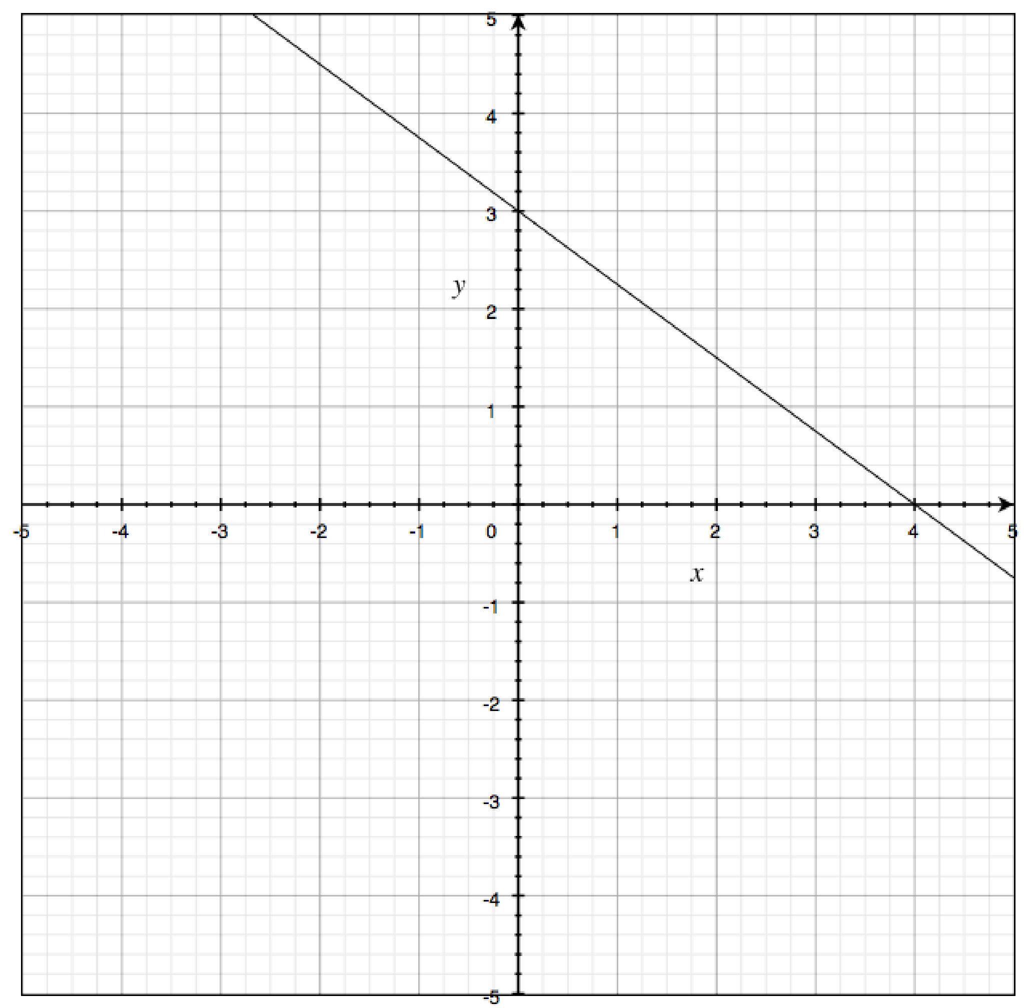
Slope intercept form

$$y = mx + b$$

slope y-intercept

normal form

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



Recap

- Finding boundaries.
- Line fitting.
- Line parameterizations.
- **Hough transform.**
- Hough circles.
- Some applications.

variables

$$y = mx + b$$

parameters

parameters

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

variables

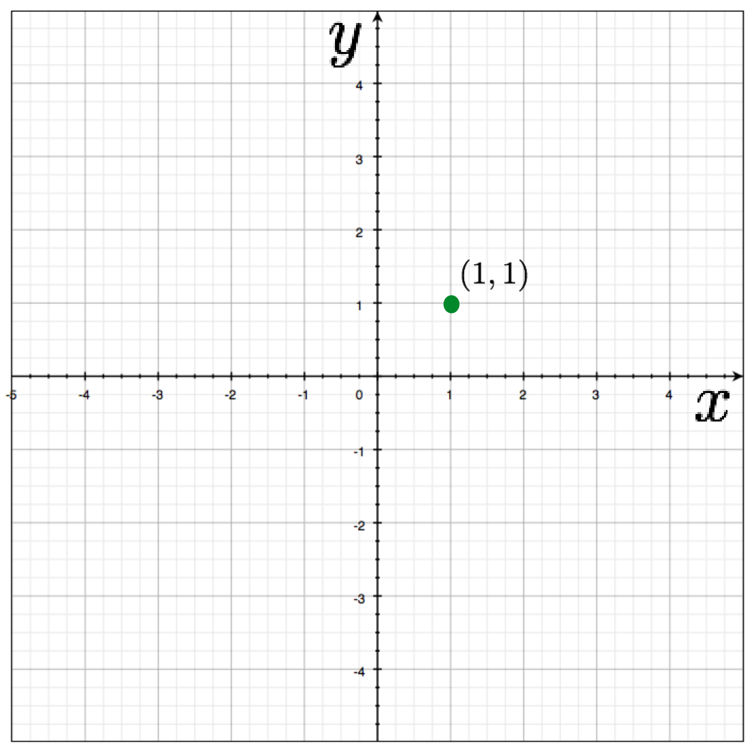


Image space

a point becomes a wave

Parameter space

Recap

- Finding boundaries.
- Line fitting.
- Line parameterizations.
- **Hough transform.**
- Hough circles.
- Some applications.

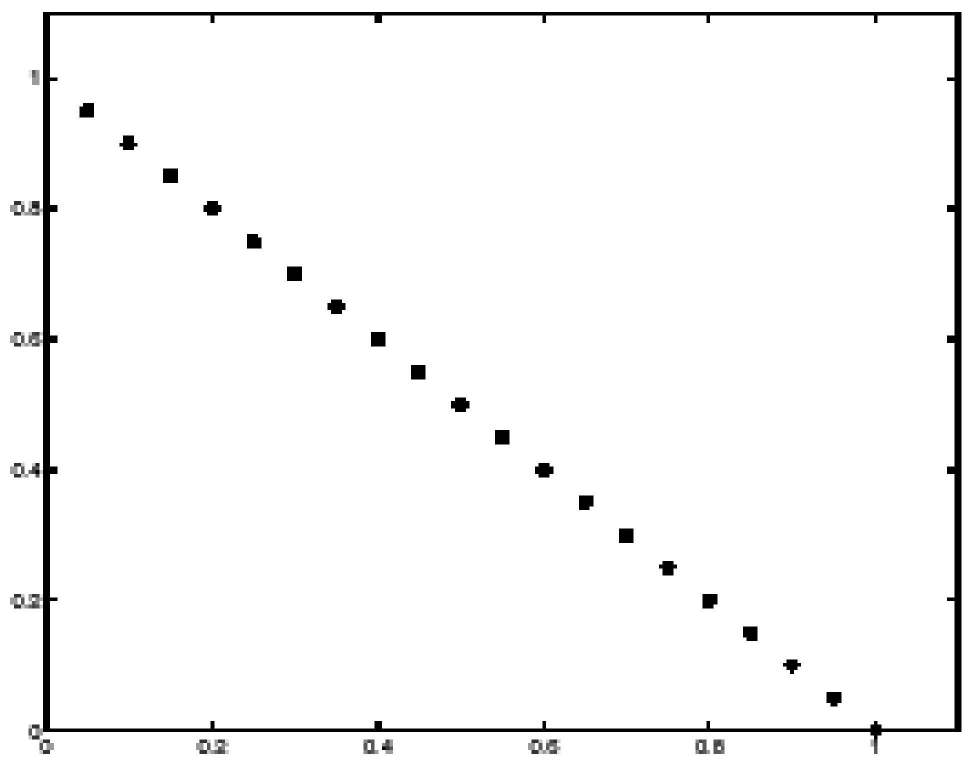
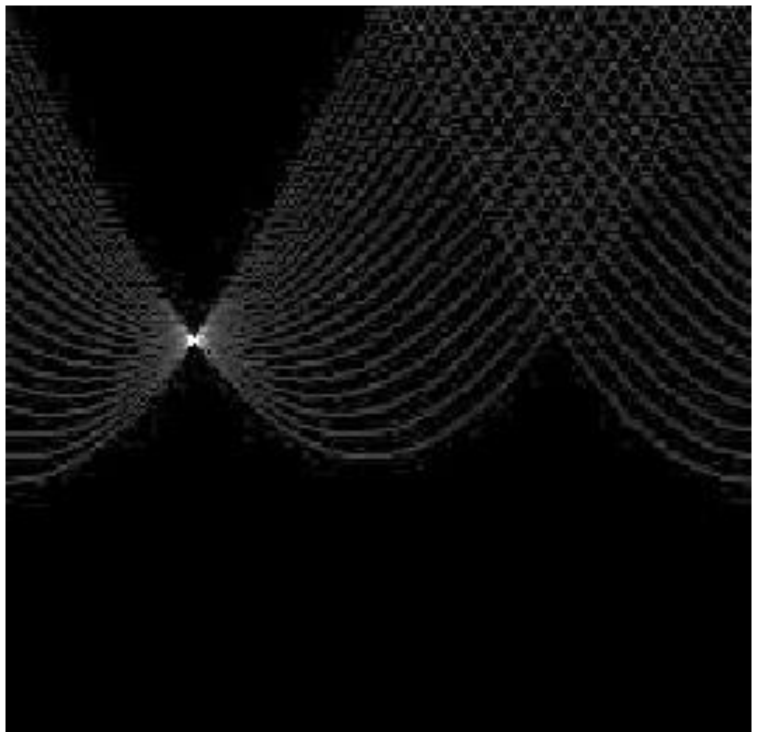


Image space



Votes

Recap

In practice, measurements are noisy…

- Finding boundaries.
- Line fitting.
- Line parameterizations.
- **Hough transform.**
- Hough circles.
- Some applications.

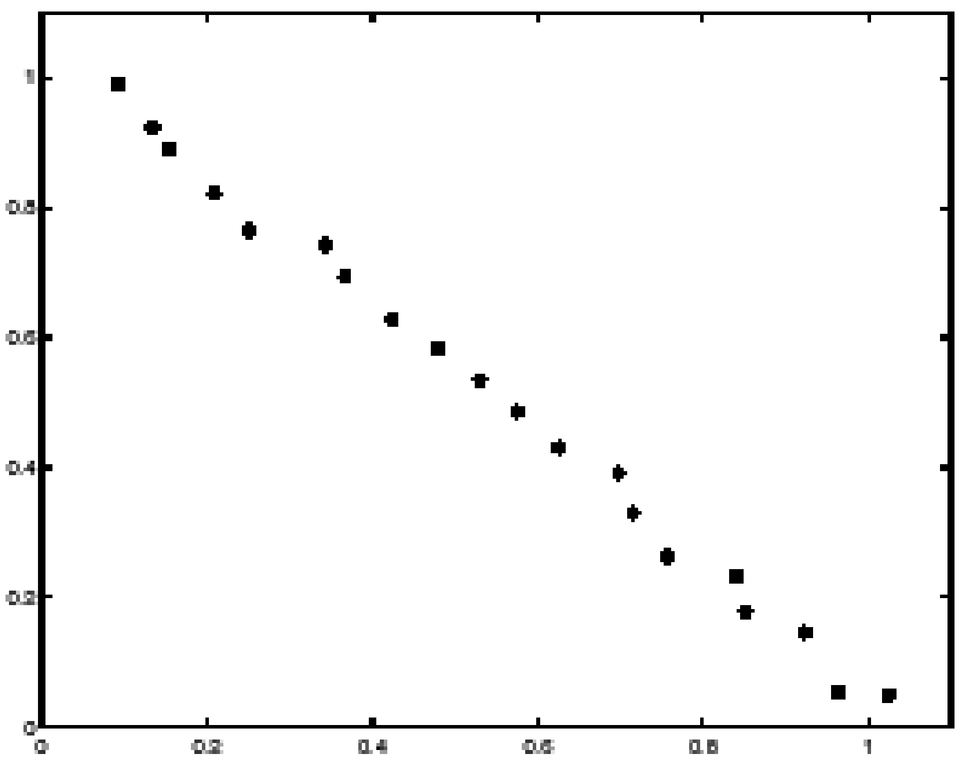
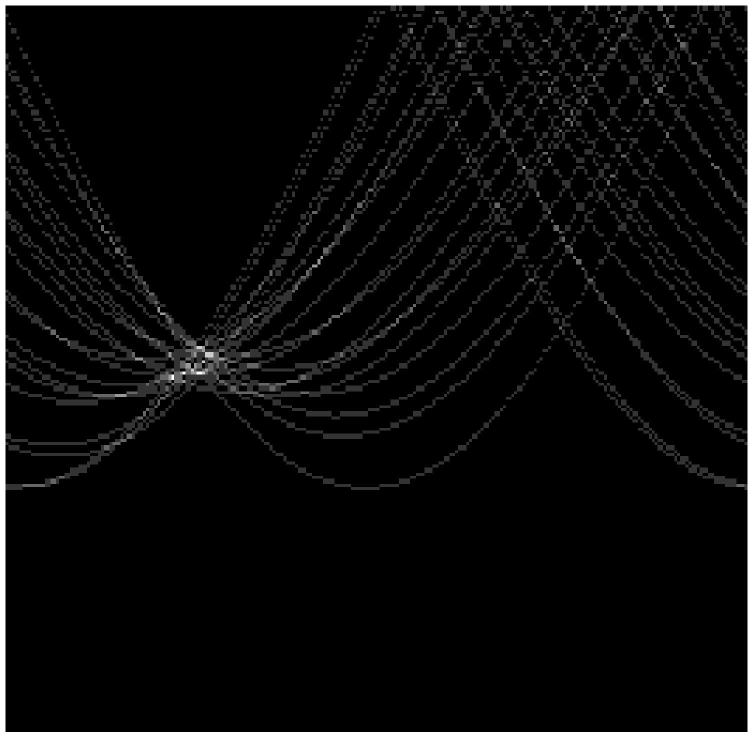


Image space



Votes

Image space

Votes

Hough Circles

Let's assume radius known

parameters

$$(x - a)^2 + (y - b)^2 = r^2$$

variables

parameters

$$(x - a)^2 + (y - b)^2 = r^2$$

variables

What is the dimension of the parameter space?

parameters

$$(x - a)^2 + (y - b)^2 = r^2$$

variables

parameters

$$(x - a)^2 + (y - b)^2 = r^2$$

variables

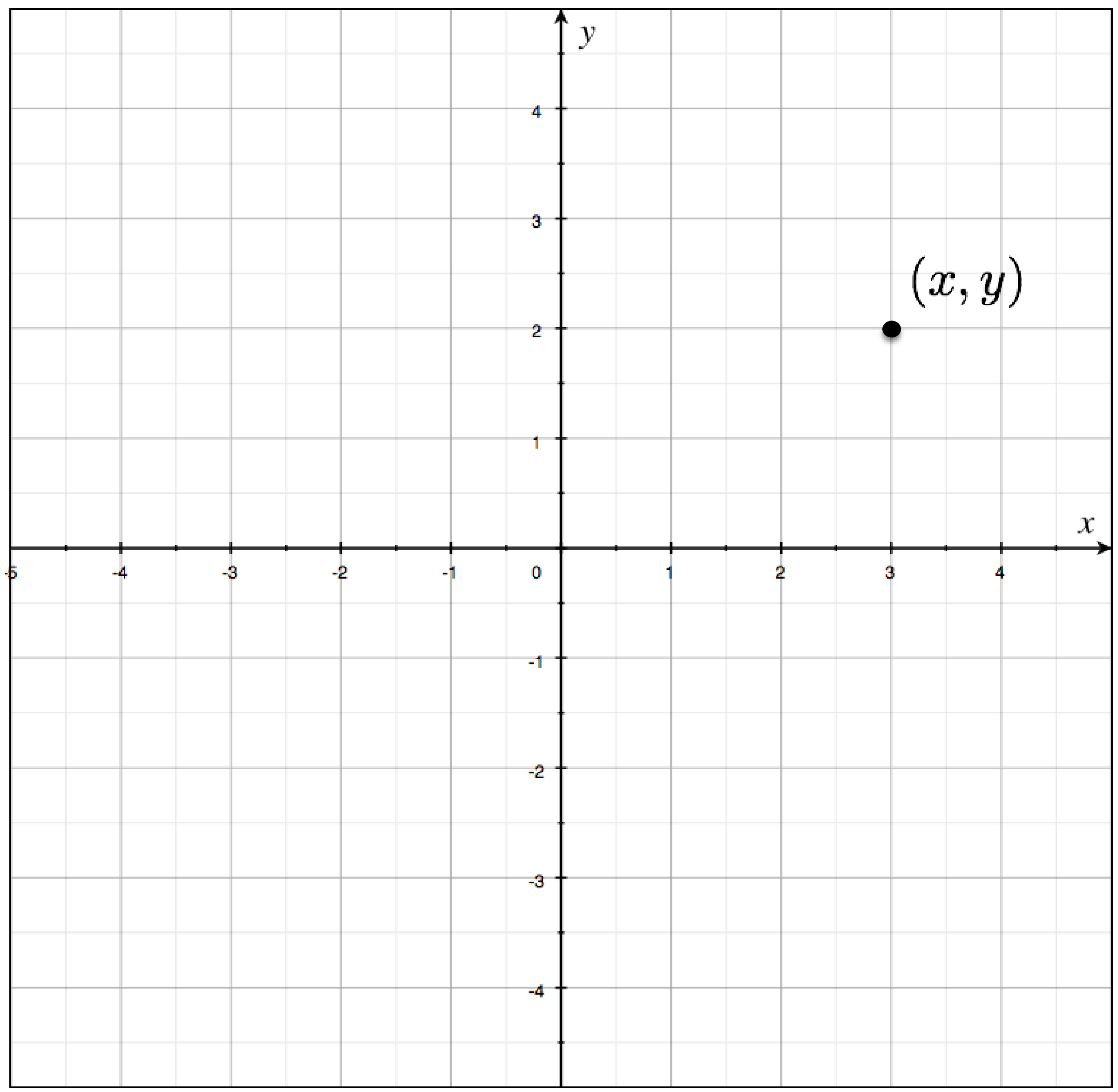
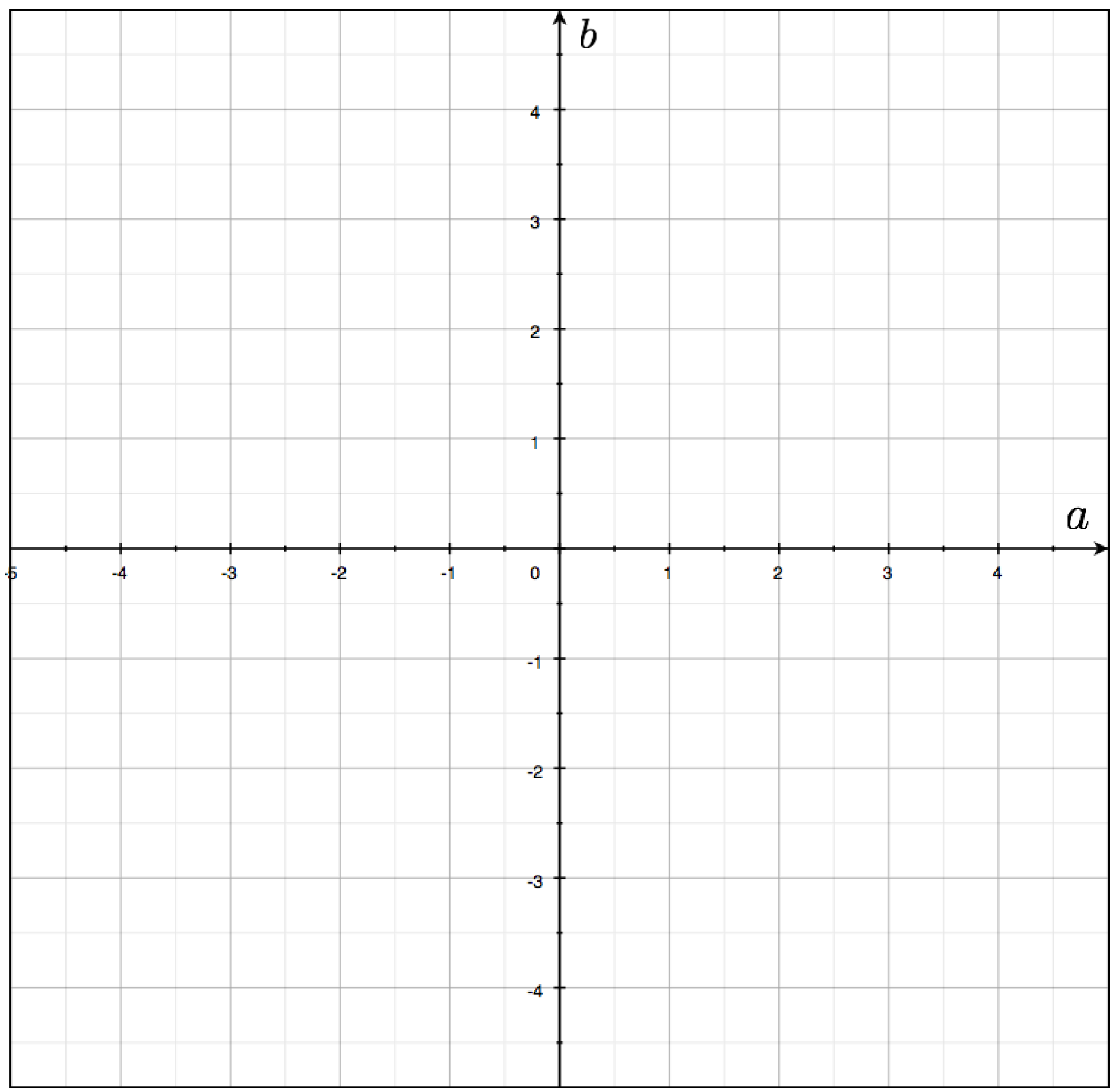


Image space



Parameter space

What does a point in image space correspond to in parameter space?

parameters

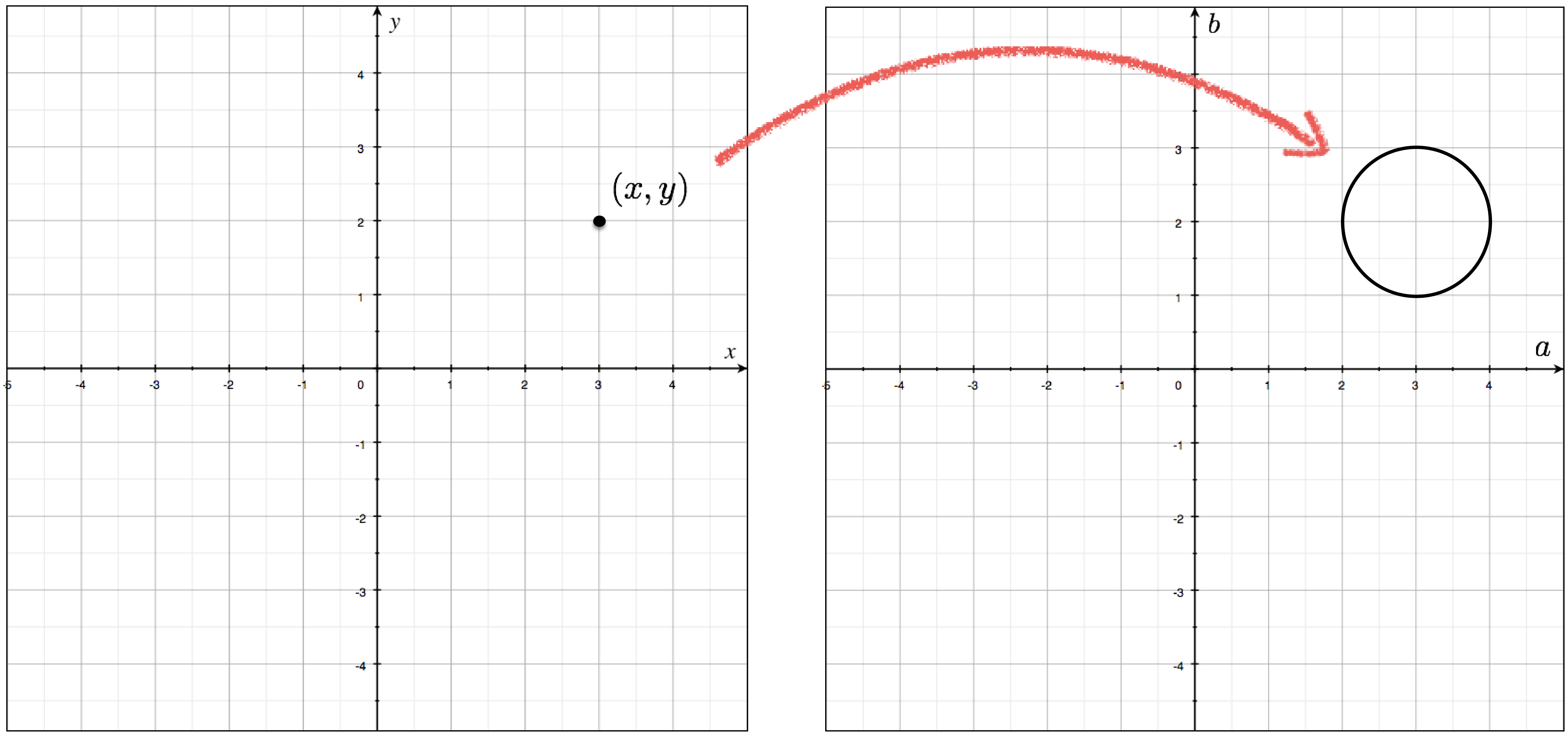
$$(x - a)^2 + (y - b)^2 = r^2$$

variables

parameters

$$(x - a)^2 + (y - b)^2 = r^2$$

variables



parameters

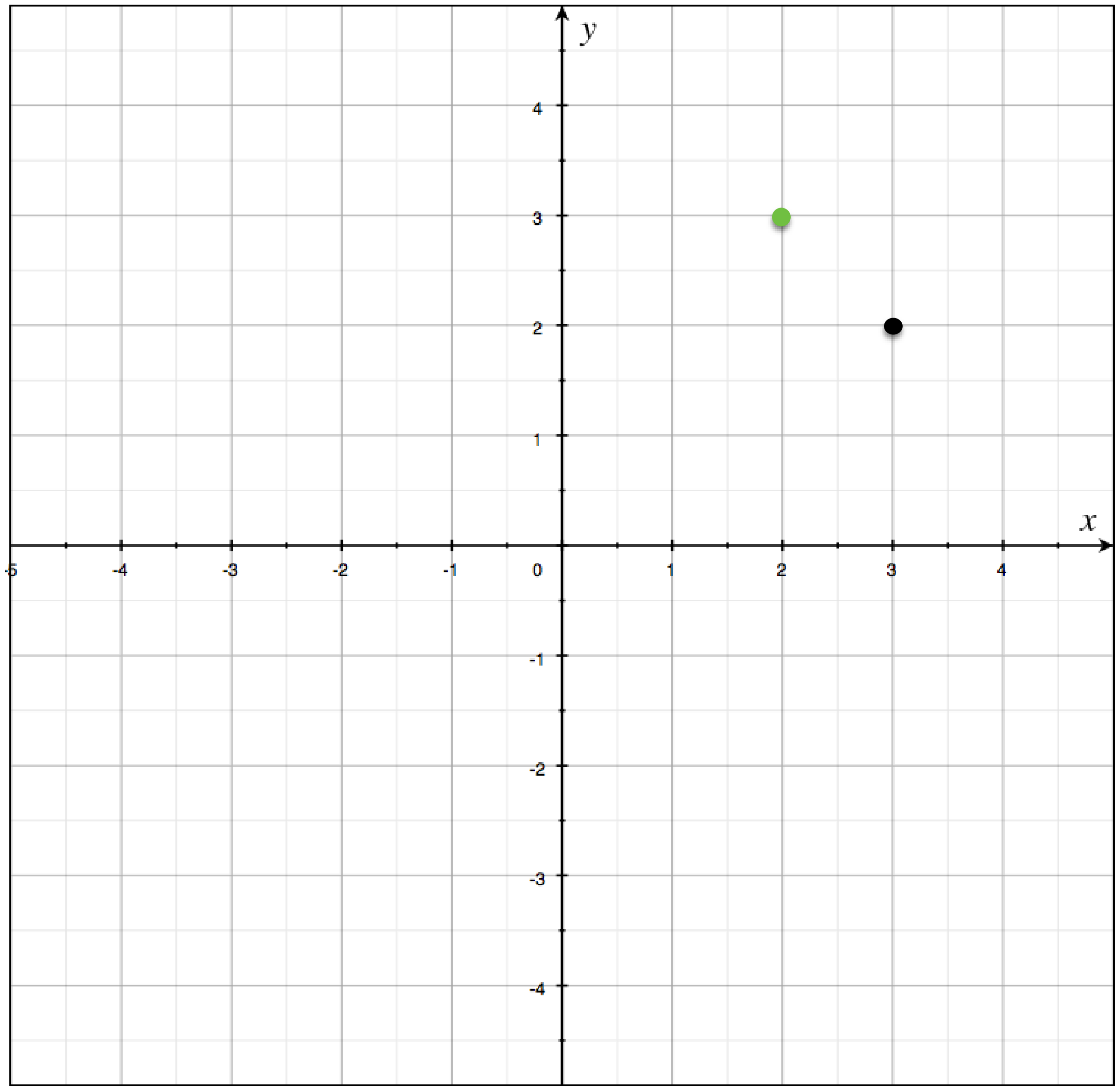
$$(x-a)^2+(y-b)^2=r^2$$

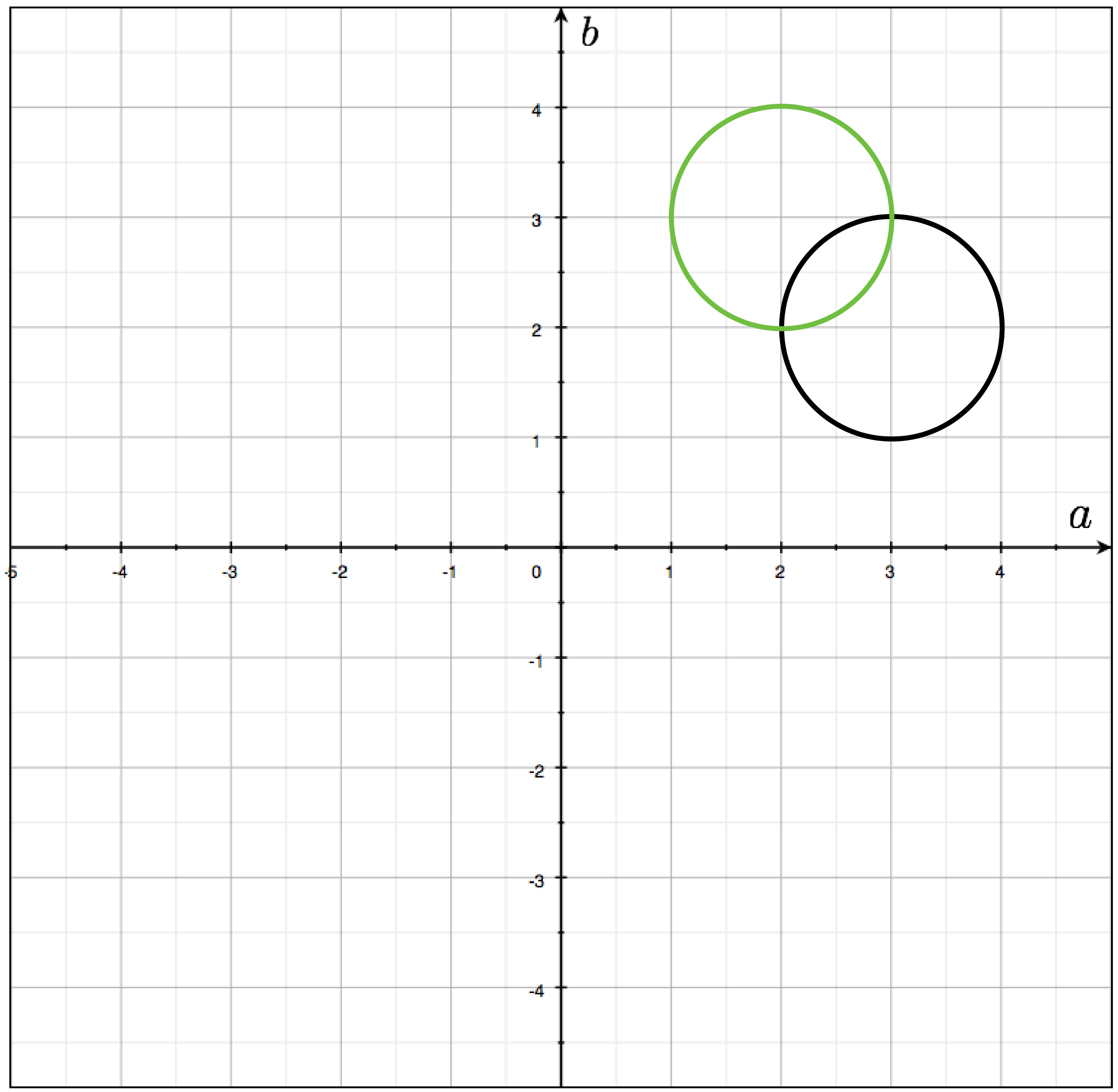
variables

parameters

$$(x-a)^2+(y-b)^2=r^2$$

variables

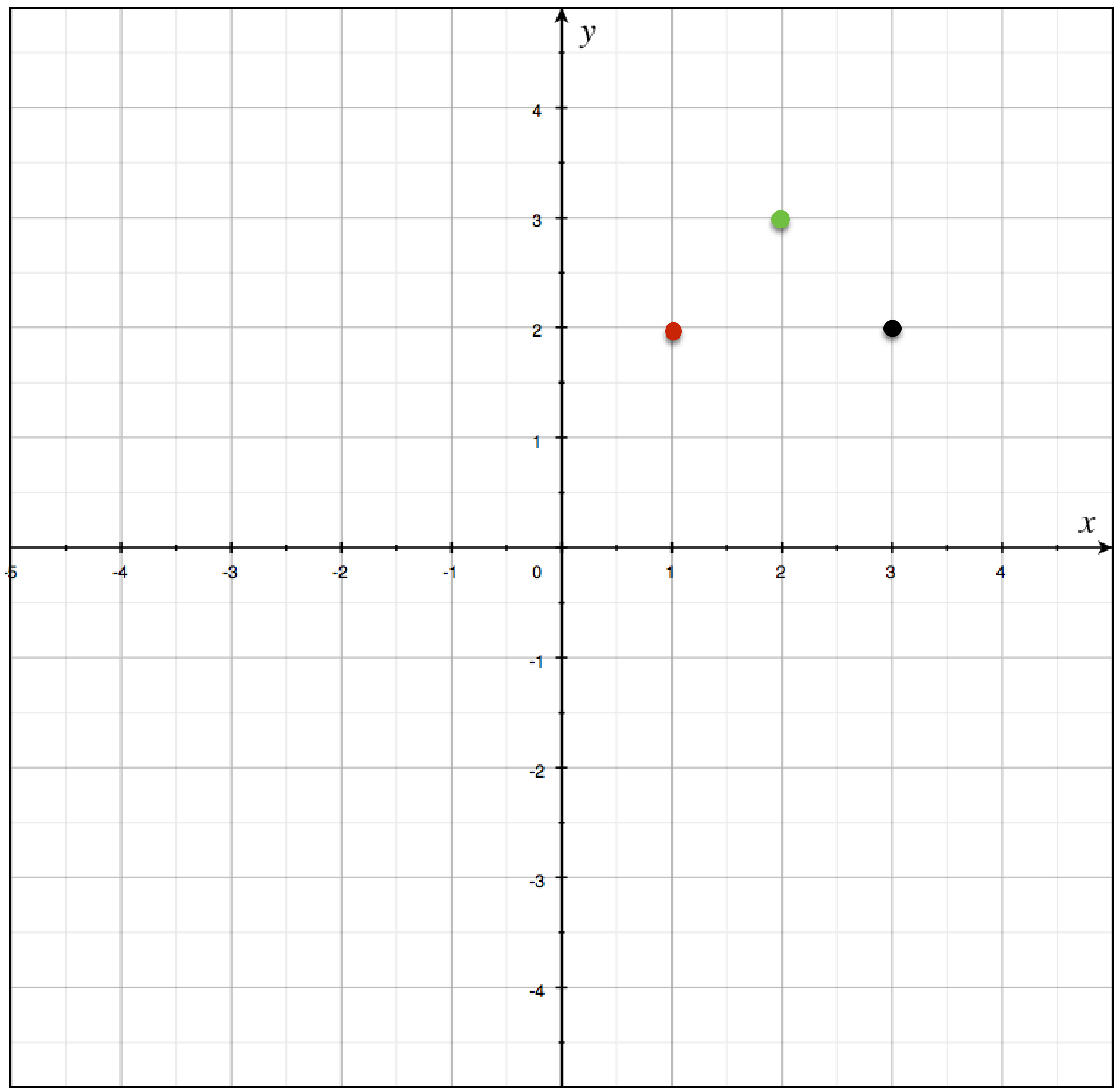




parameters

$$(x-a)^2+(y-b)^2=r^2$$

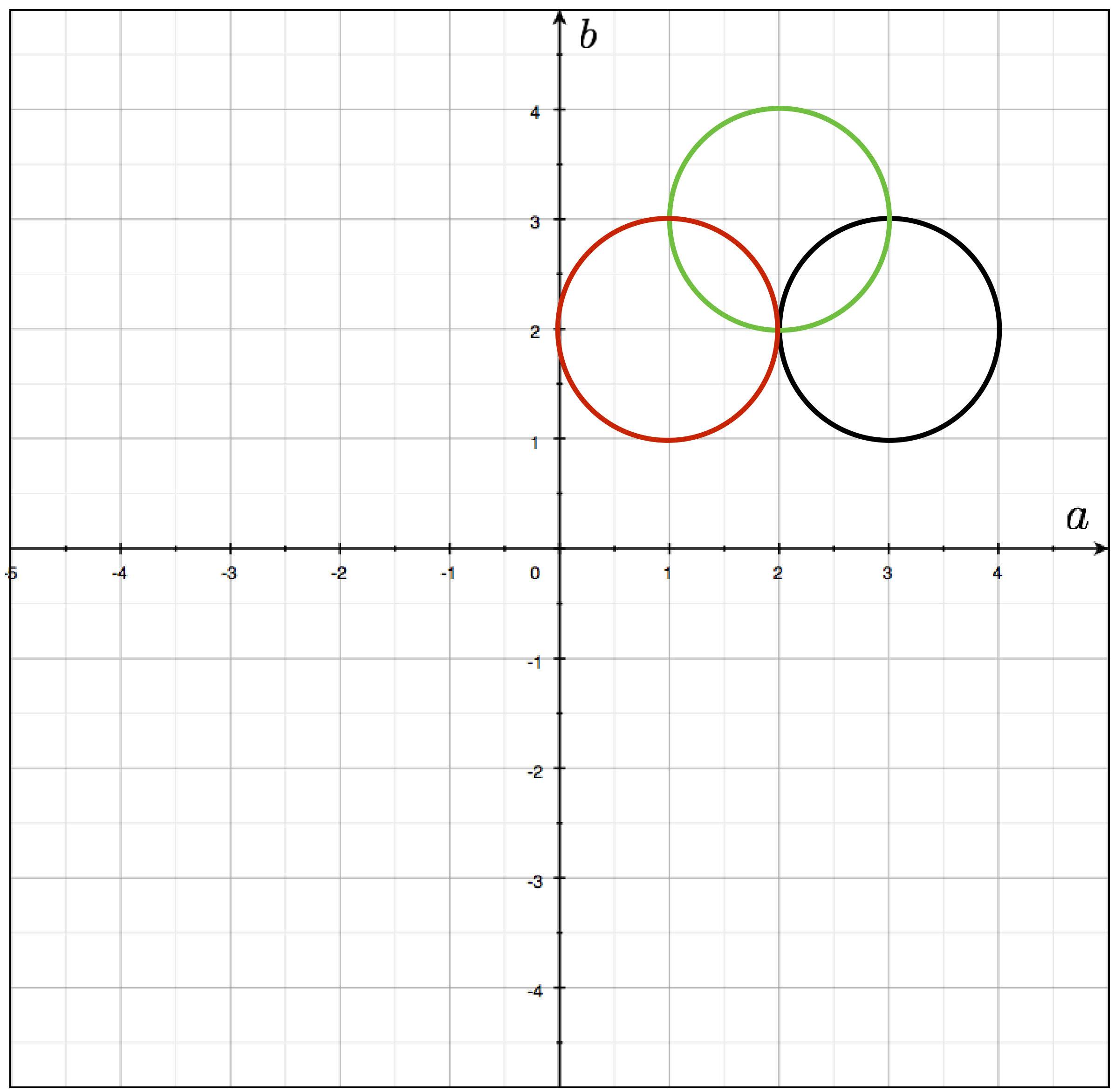
variables



parameters

$$(x-a)^2+(y-b)^2=r^2$$

variables



parameters

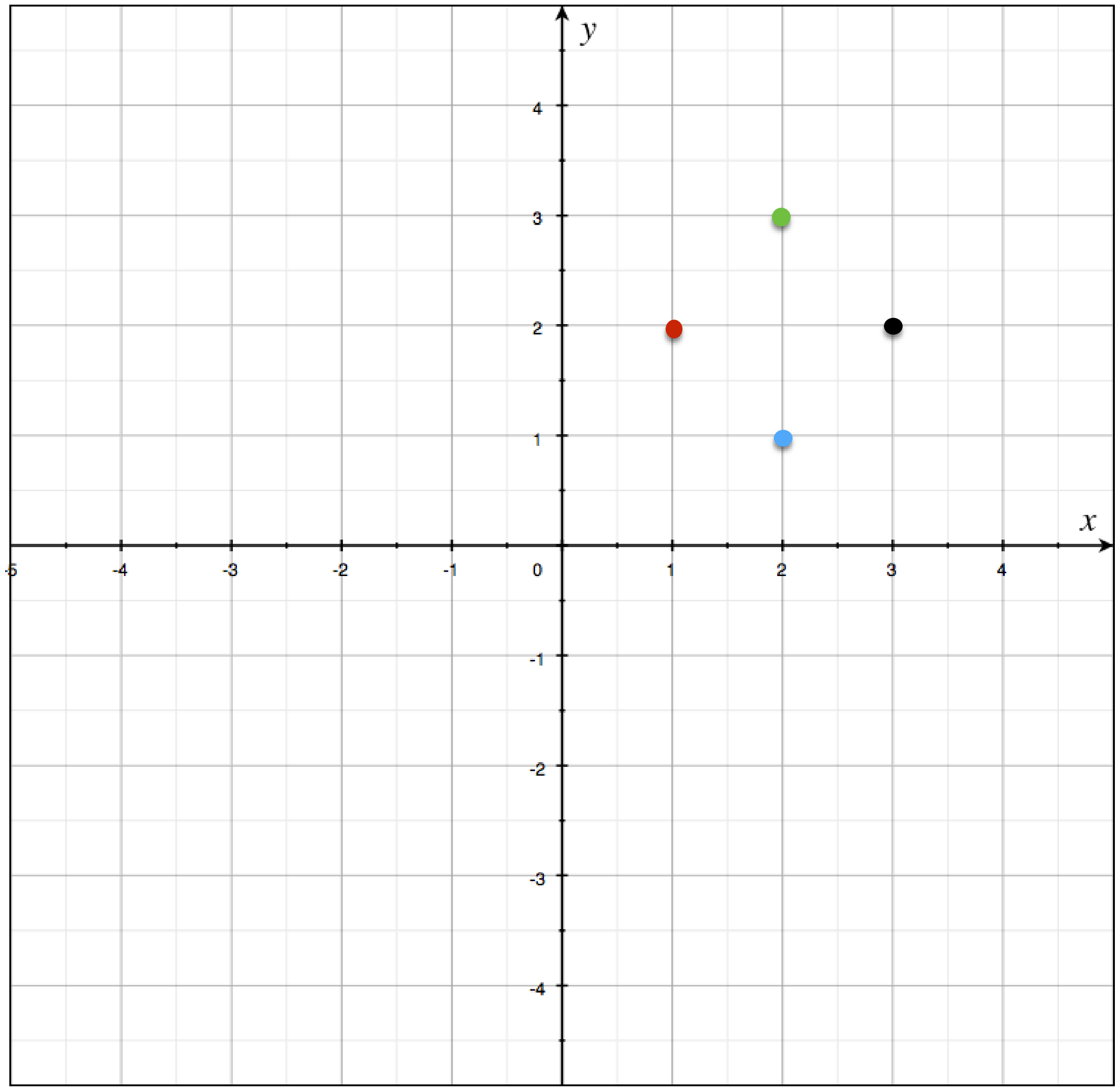
$$(x - a)^2 + (y - b)^2 = r^2$$

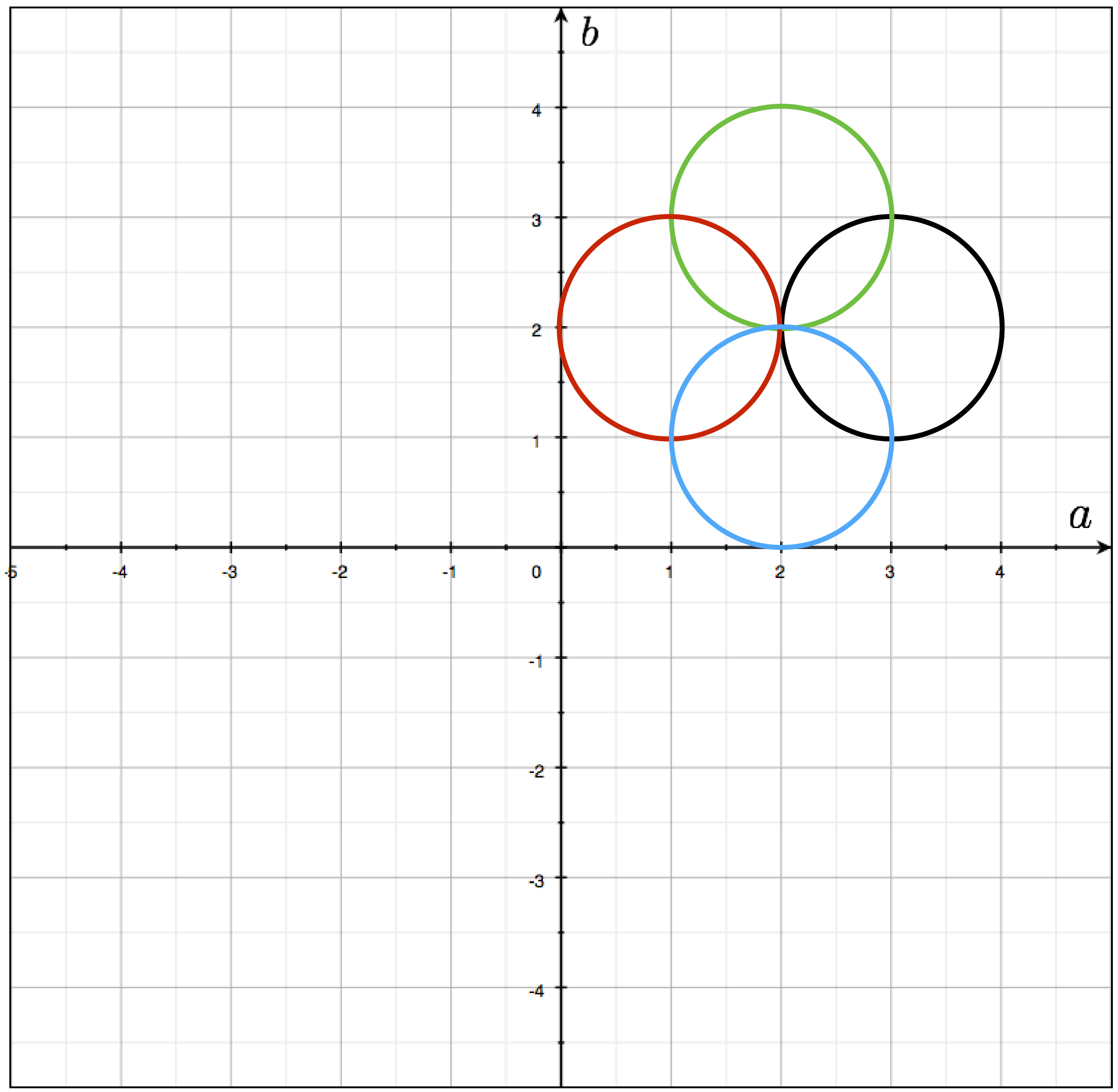
variables

parameters

$$(x - a)^2 + (y - b)^2 = r^2$$

variables





What if radius is unknown?

parameters

$$(x-a)^2+(y-b)^2=r^2$$

variables

parameters

$$(x-a)^2+(y-b)^2=r^2$$

variables

What if radius is unknown?

parameters

$$(x-a)^2+(y-b)^2=r^2$$

variables

parameters

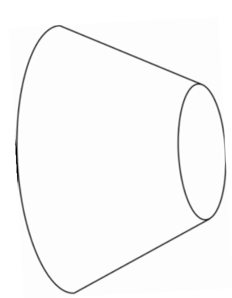
$$(x-a)^2+(y-b)^2=r^2$$

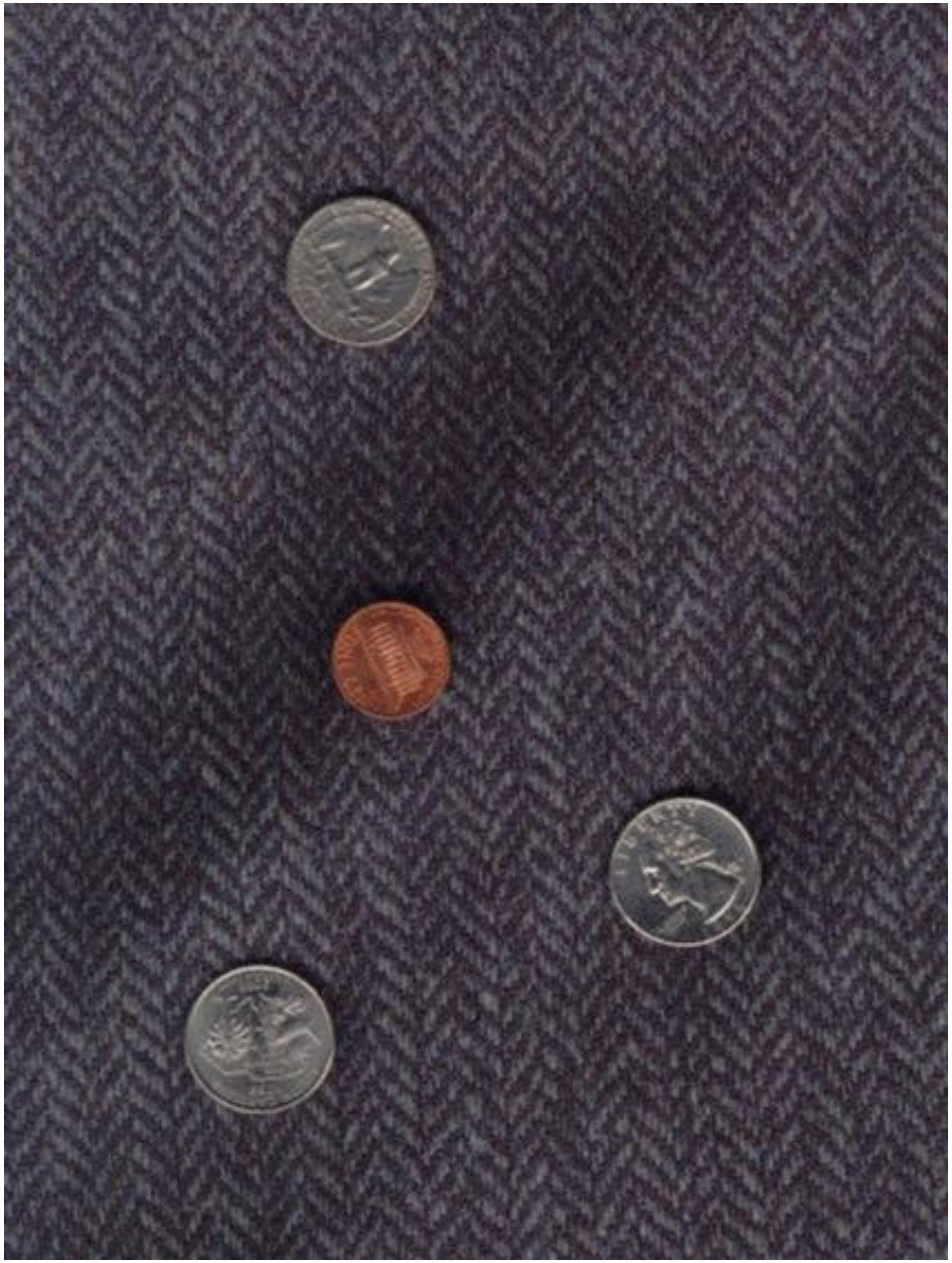
variables

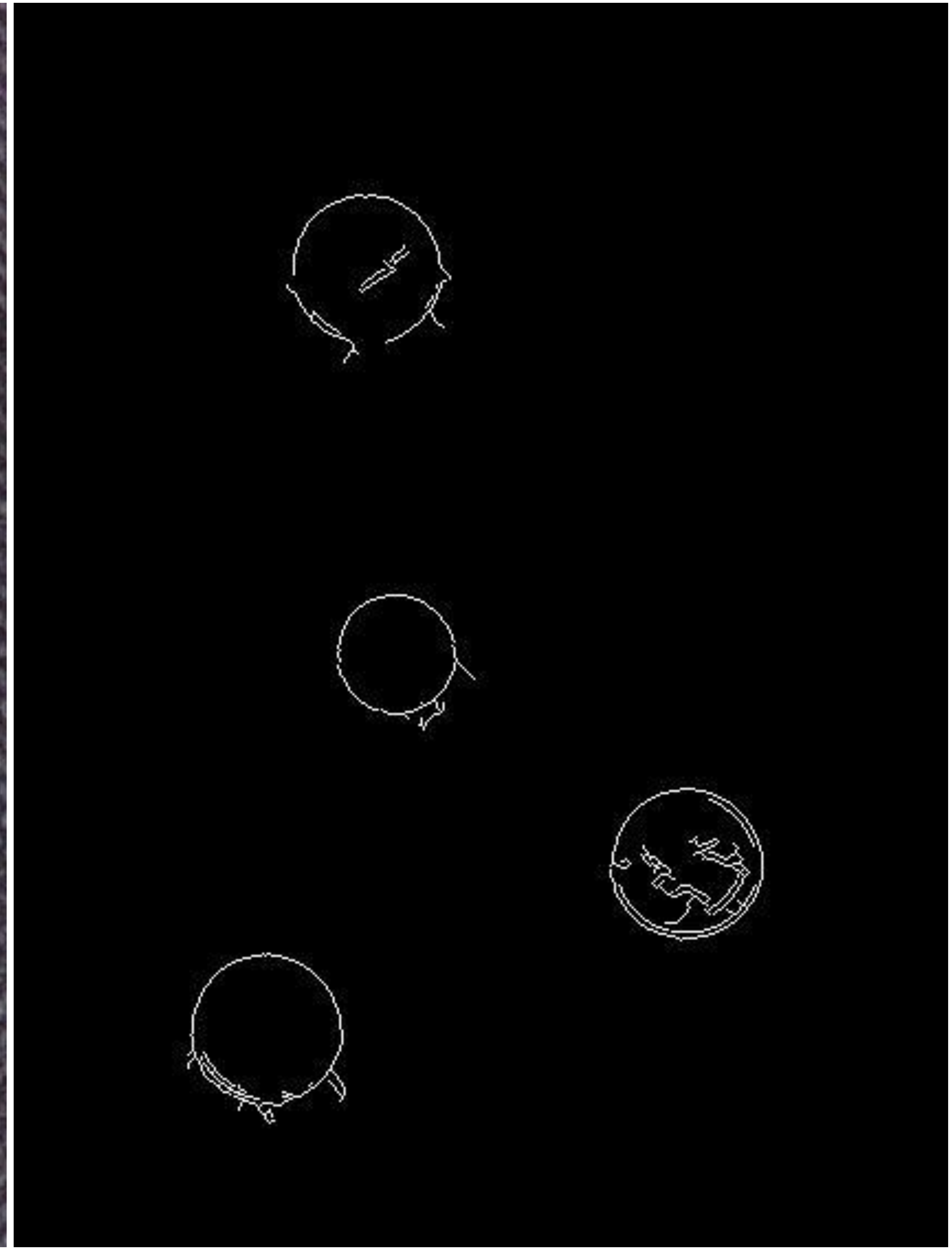
If radius is not known: 3D Hough Space!

Use Accumulator array $A(a,b,r)$

Surface shape in Hough space is complicated



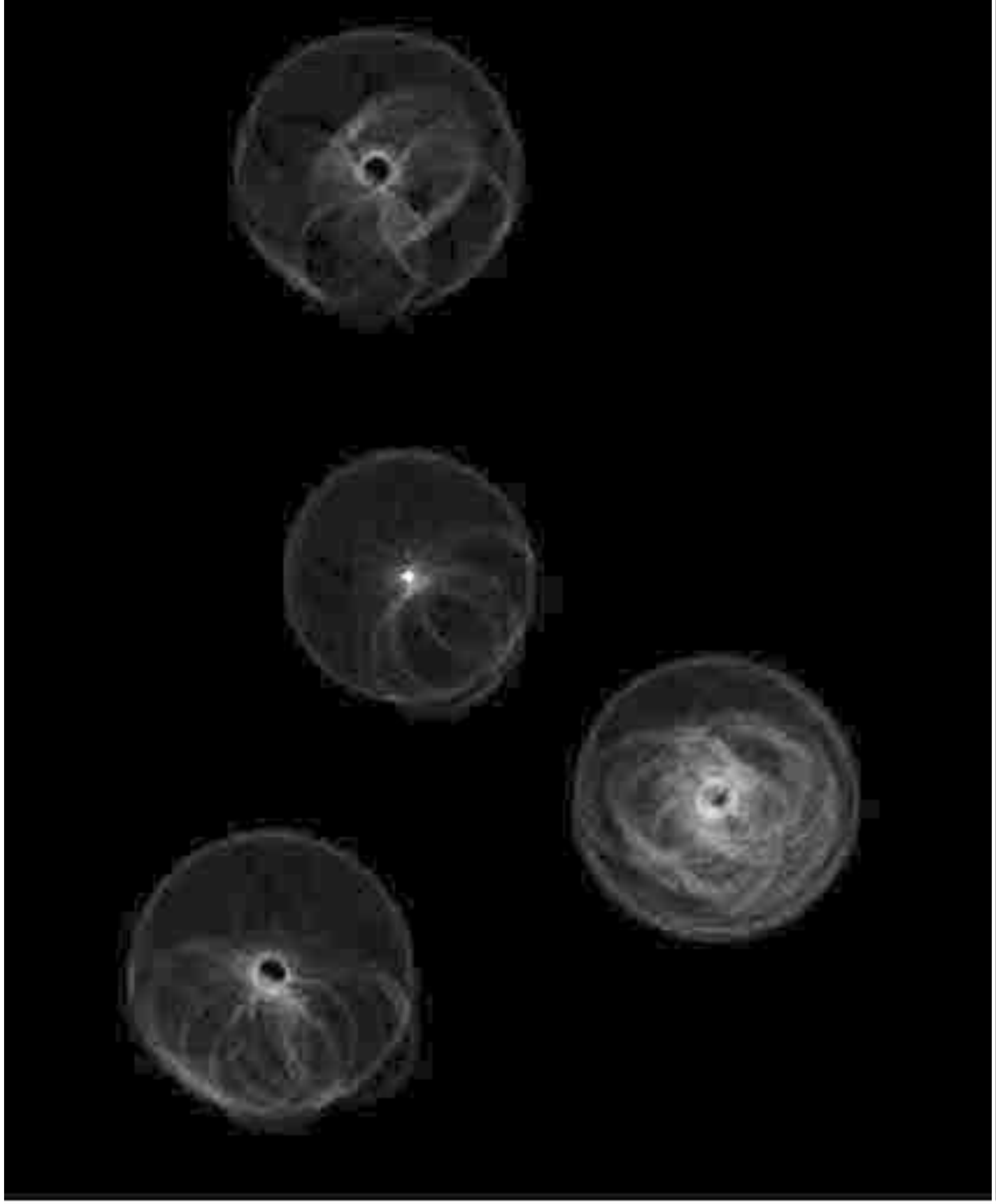




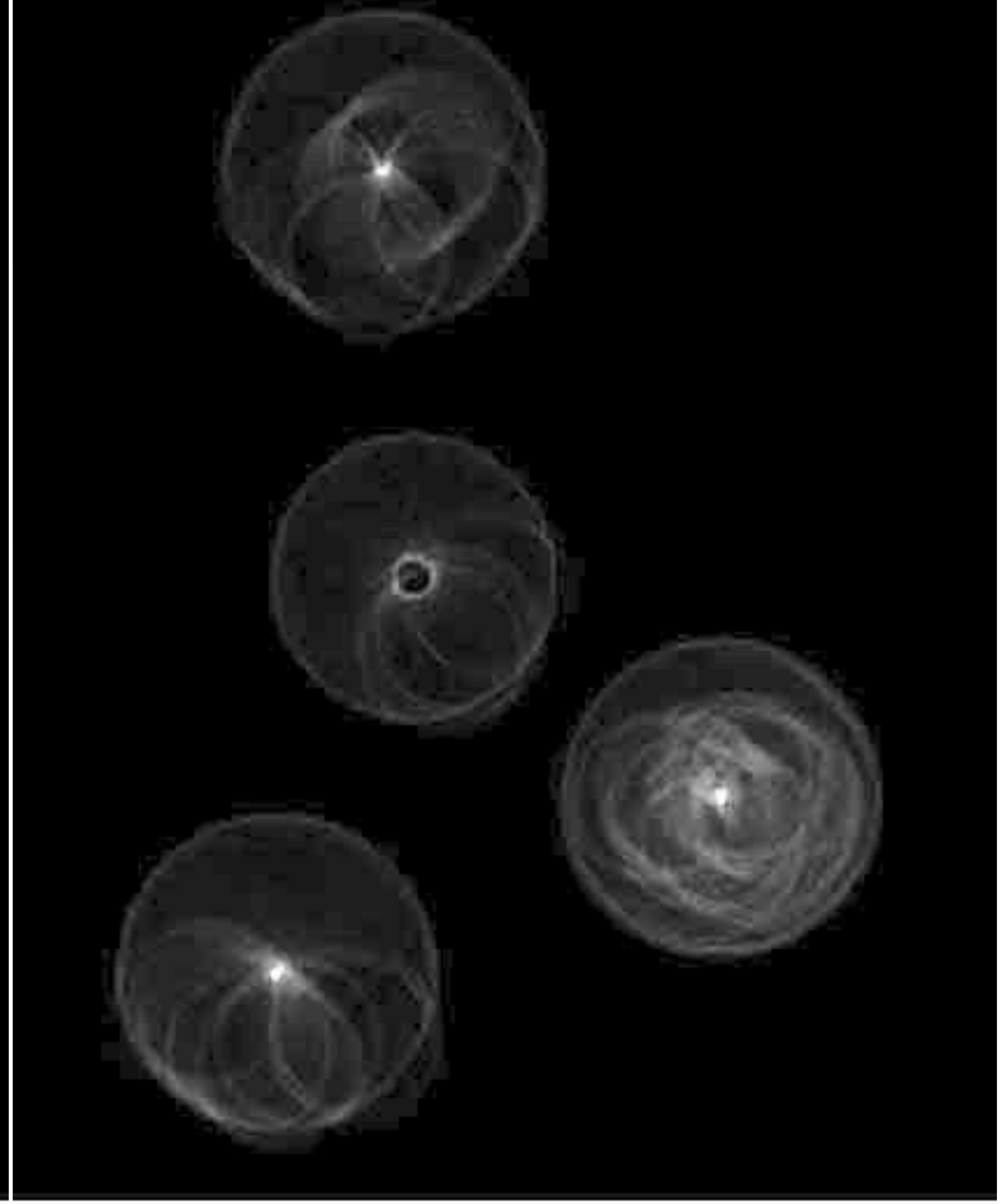

Pennie Hough detector


Quarter Hough detector

Pennie Hough detector



Quarter Hough detector

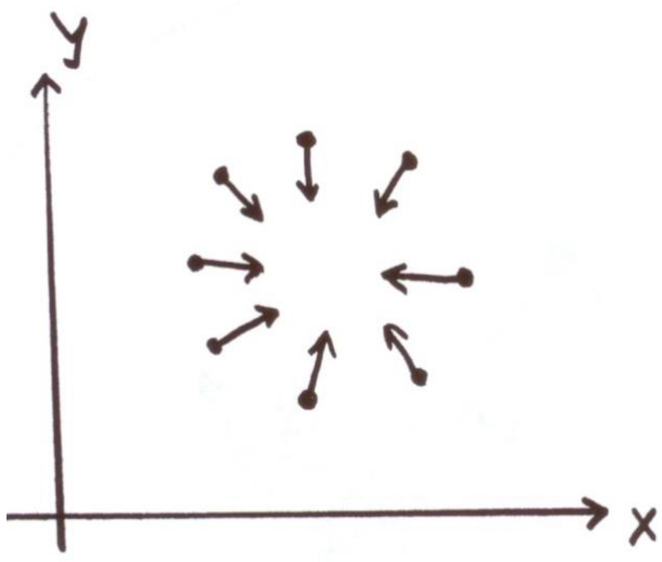


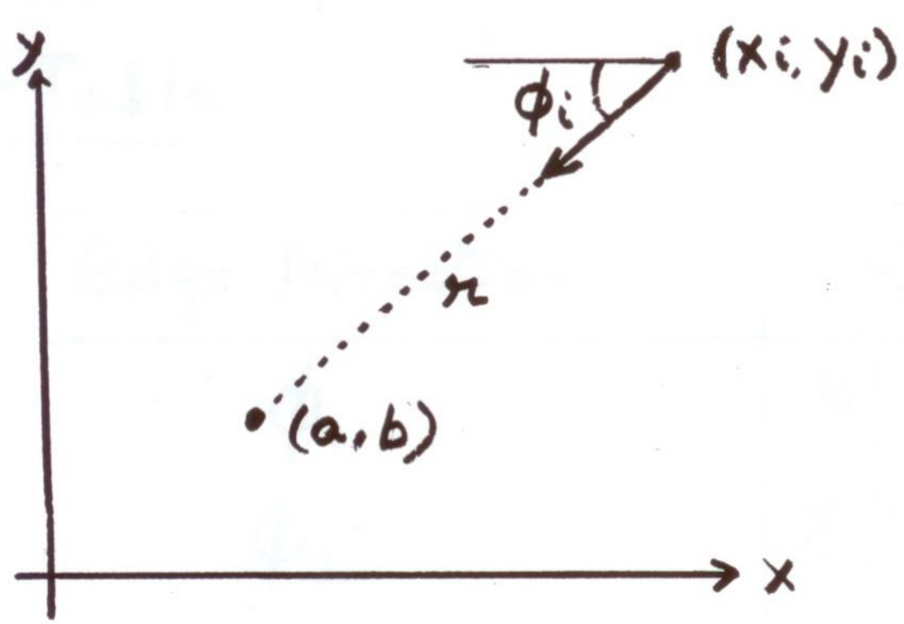
Using Gradient Information

Gradient information can save lot of computation:

Edge Location (x_i, y_i)

Edge Direction ϕ_i





$$a = x - r\cos\phi$$

$$b = y - r\sin\phi$$

If radius is known, need to increment only one point in accumulator!

parameters

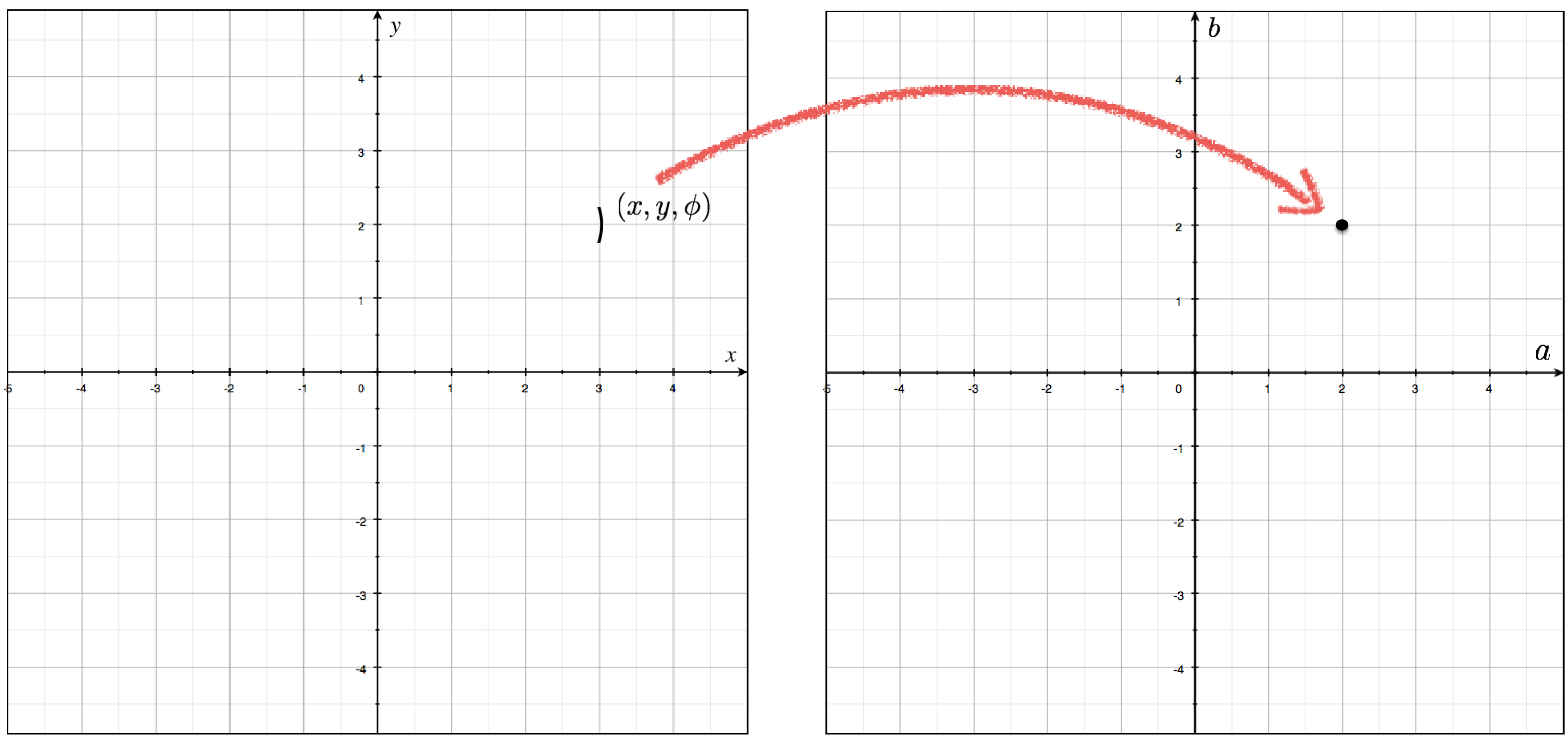
$$(x-a)^2+(y-b)^2=r^2$$

variables

parameters

$$(x-a)^2+(y-b)^2=r^2$$

variables



parameters

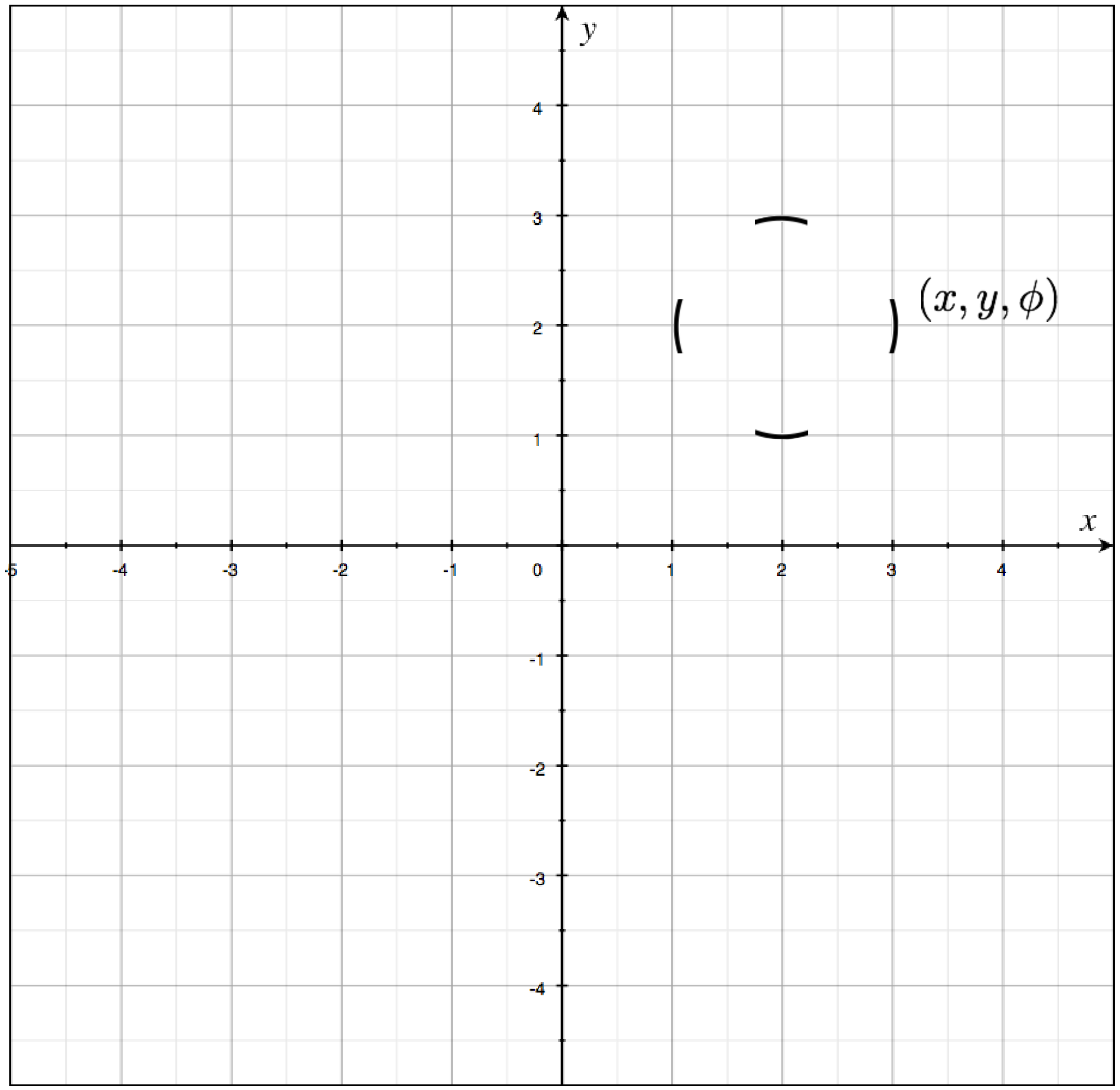
$$(x - a)^2 + (y - b)^2 = r^2$$

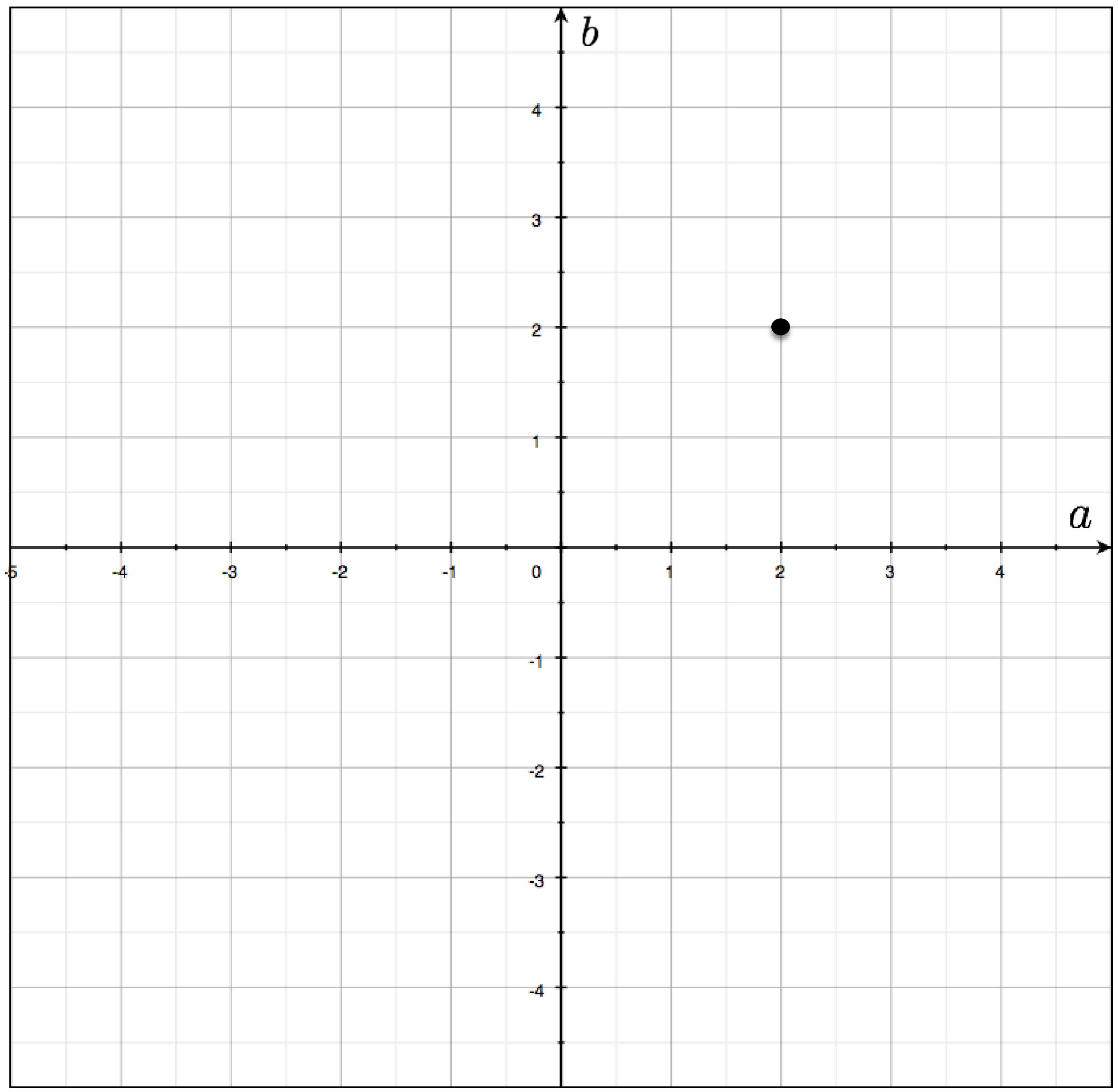
variables

parameters

$$(x - a)^2 + (y - b)^2 = r^2$$

variables





The Hough transform …

Deals with occlusion well? 

Detects multiple instances? 

Robust to noise? 

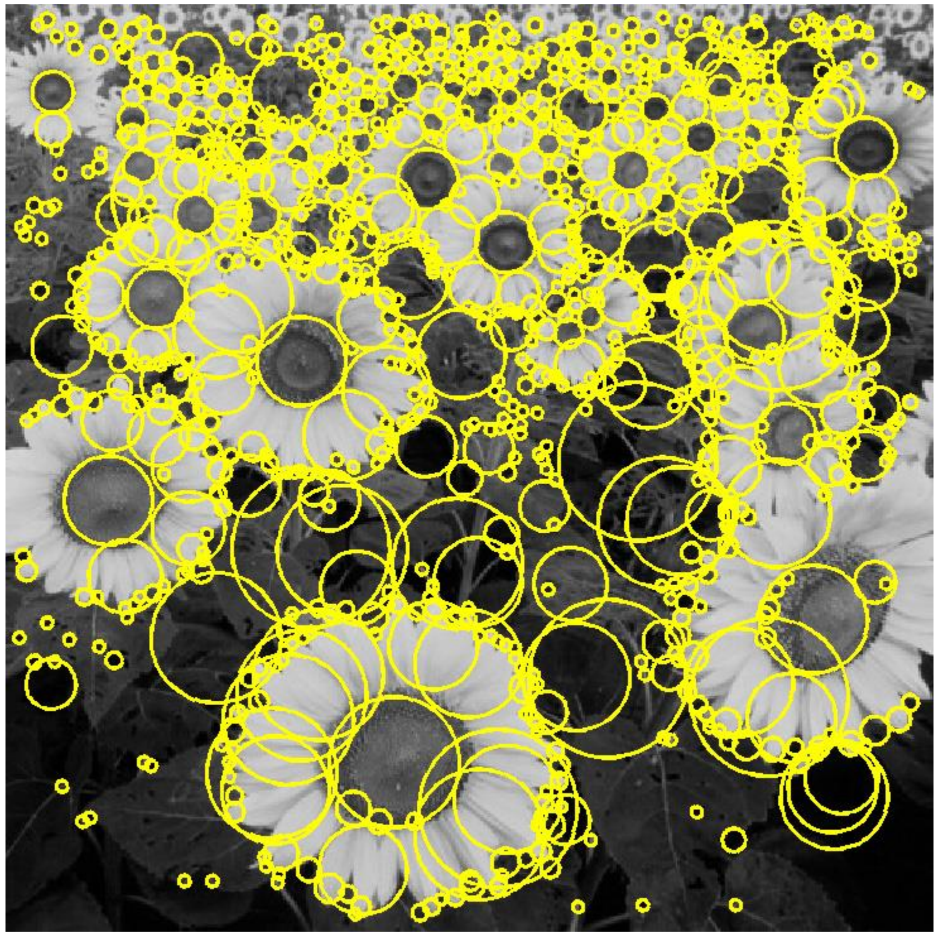
Good computational complexity? 

Easy to set parameters? 

Can you use Hough Transforms for other objects, beyond lines and circles?

Do you have to use edge detectors to vote in Hough Space?

Today: Feature extraction—Corners and blobs



Overview of today’s lecture

- Why detect corners?
- Harris corner detector.
- Multi-scale detection.
- Multi-scale blob detection.
- Visualizing quadratics. (Maybe)

Slide credits

Most of these slides were adapted from:

- Matt O’Toole (CMU, 15-463, Fall 2022).

- Ioannis Gkioulekas (CMU, 15-463, Fall 2020).

- Kris Kitani (CMU, 15-463, Fall 2016).

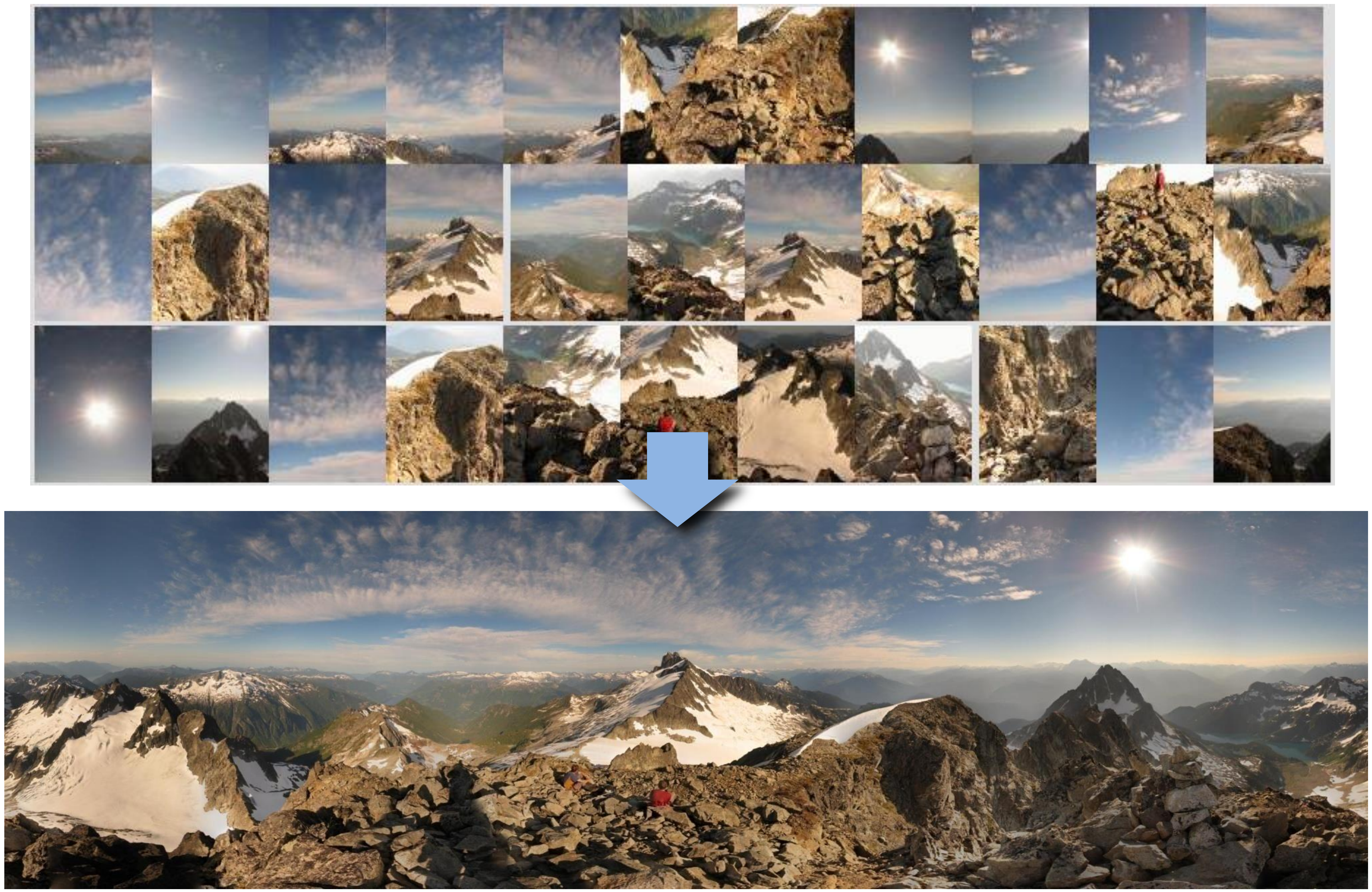
- Noah Snavely (Cornell, CS5670, Fall 2022)

Some slides were inspired or taken from:

- Fredo Durand (MIT).
- James Hays (Georgia Tech).

Why detect corners?

Motivation: Automatic panoramas



Credit: Matt Brown

Motivation: Automatic panoramas

GigaPan:
http://gigapan.com/

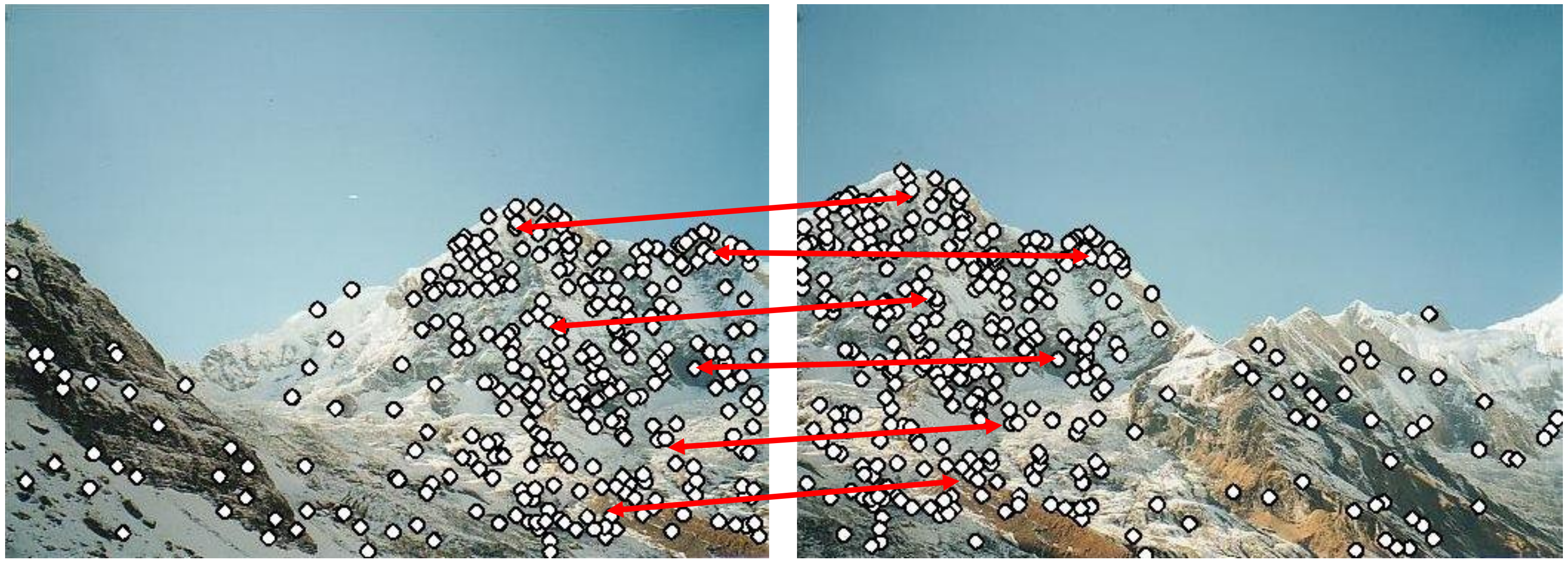
Also see Google Zoom Views:
https://www.google.com/culturalinstitute/beta/project/gigapixels

Why extract features?

- Motivation: panorama stitching
  - We have two images – how do we combine them?


Why extract features?

- Motivation: panorama stitching
  - We have two images – how do we combine them?



Step 1: extract features
Step 2: match features

Why extract features?

- Motivation: panorama stitching
  - We have two images – how do we combine them?

Step 1: extract features
Step 2: match features
Step 3: align images

Application: Visual SLAM

- (aka Simultaneous Localization and Mapping)

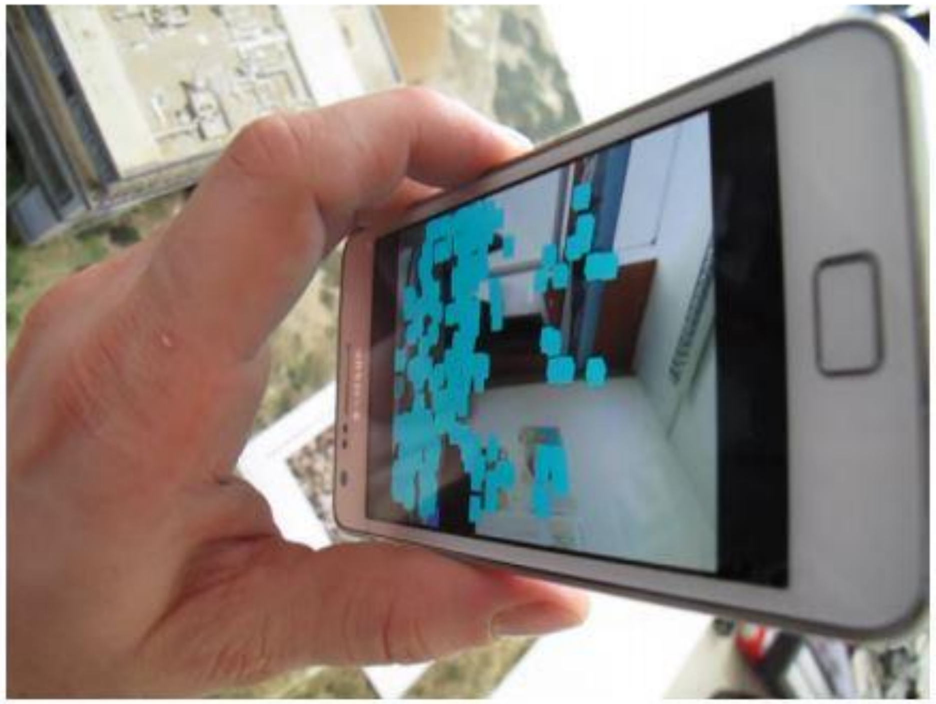
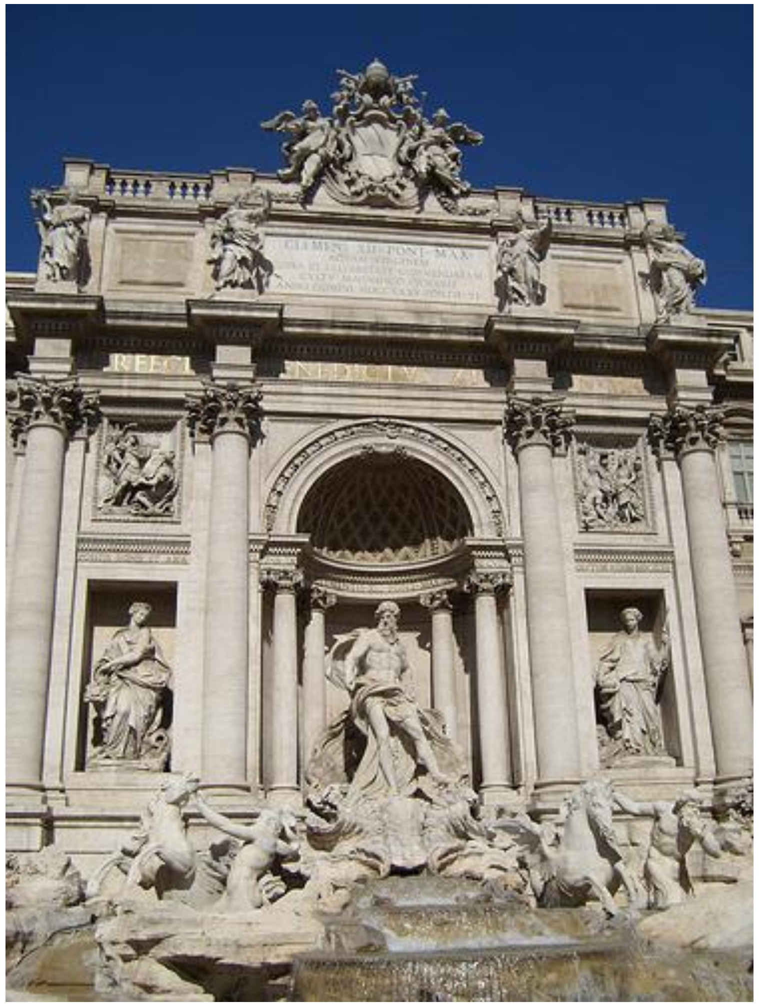


Image matching

by Diva Sian



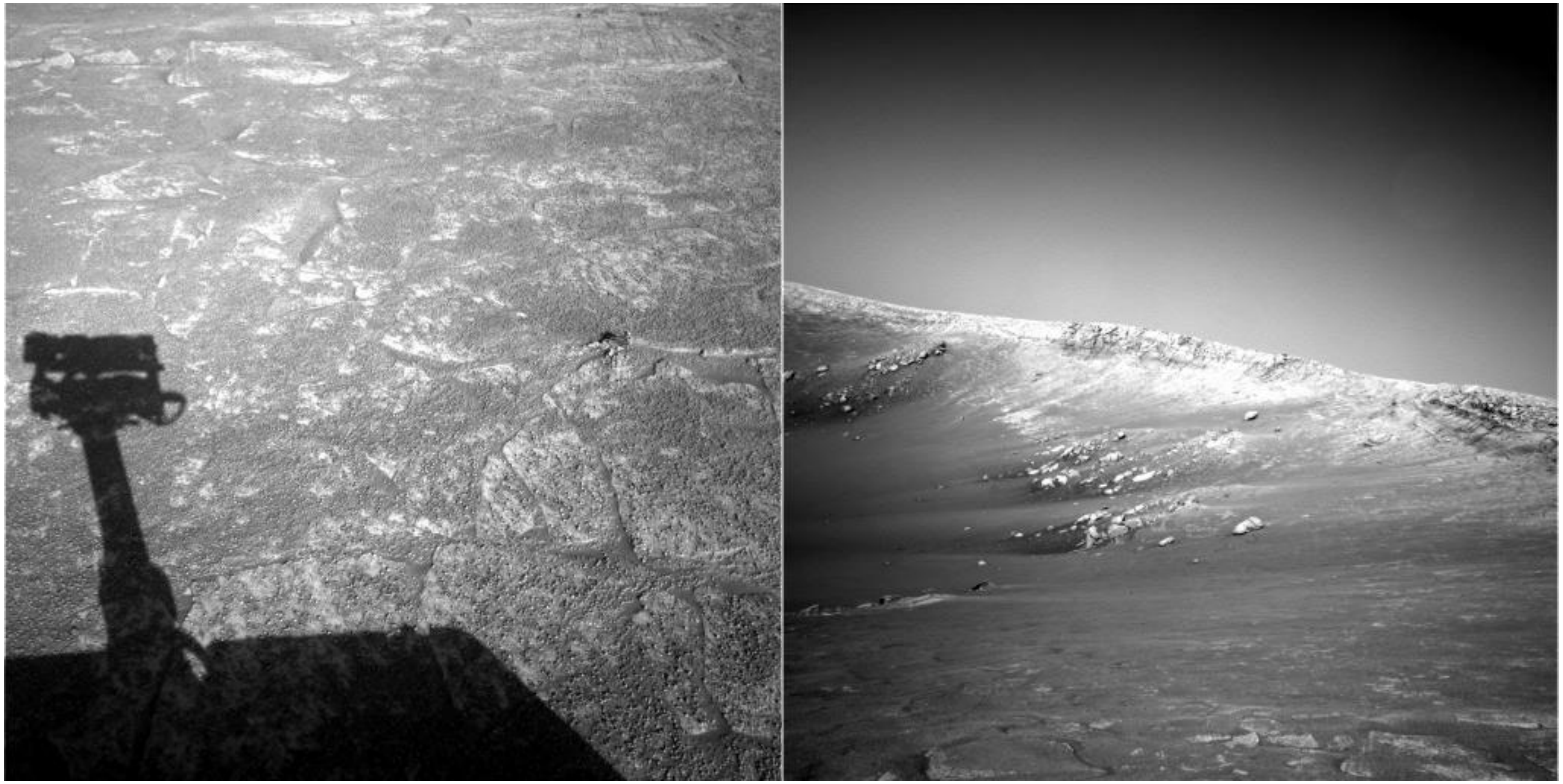
by swashford

Harder case

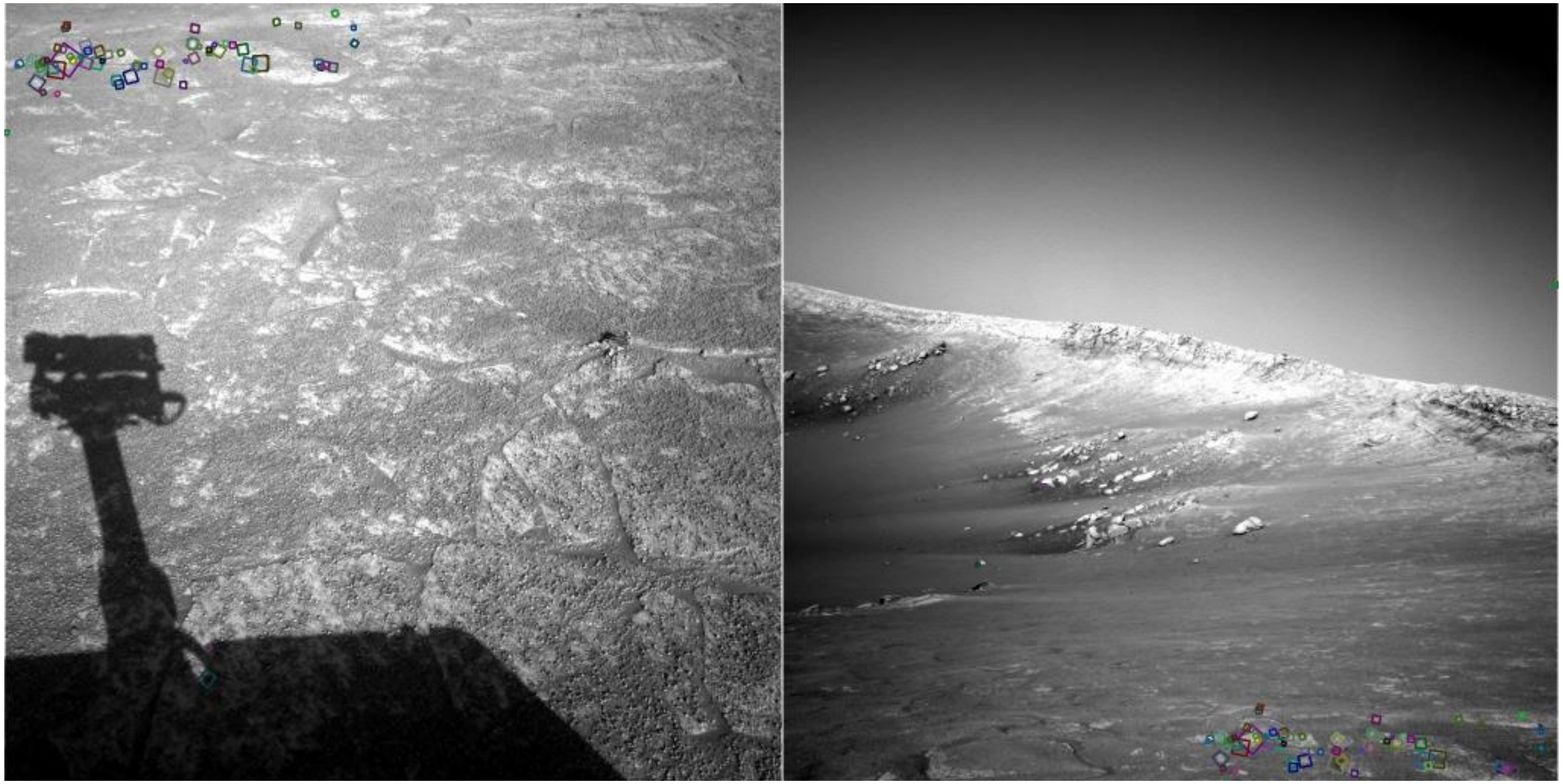
by Diva Sian

by scgbt

Harder still?

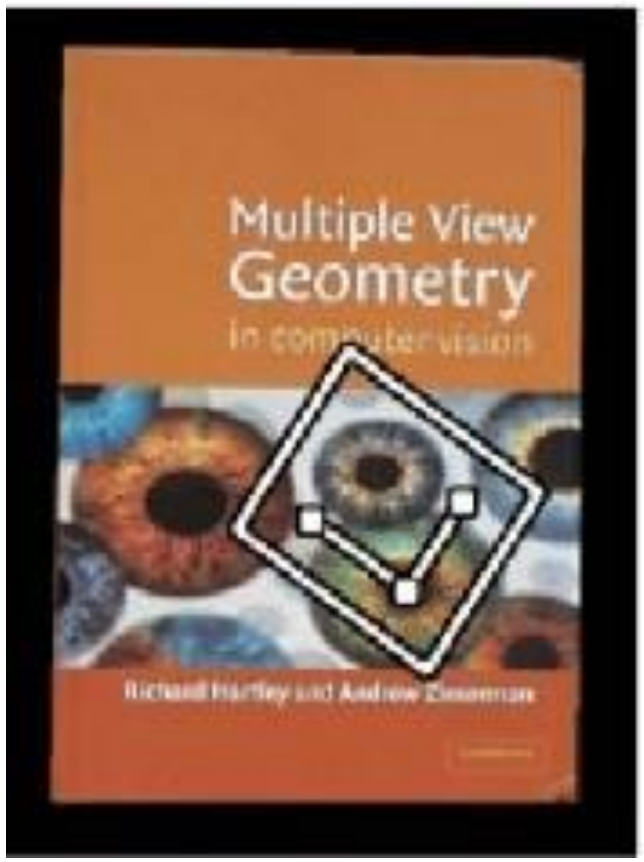


Answer below (look for tiny colored squares…)



NASA Mars Rover images
with SIFT feature matches

Feature matching for object search

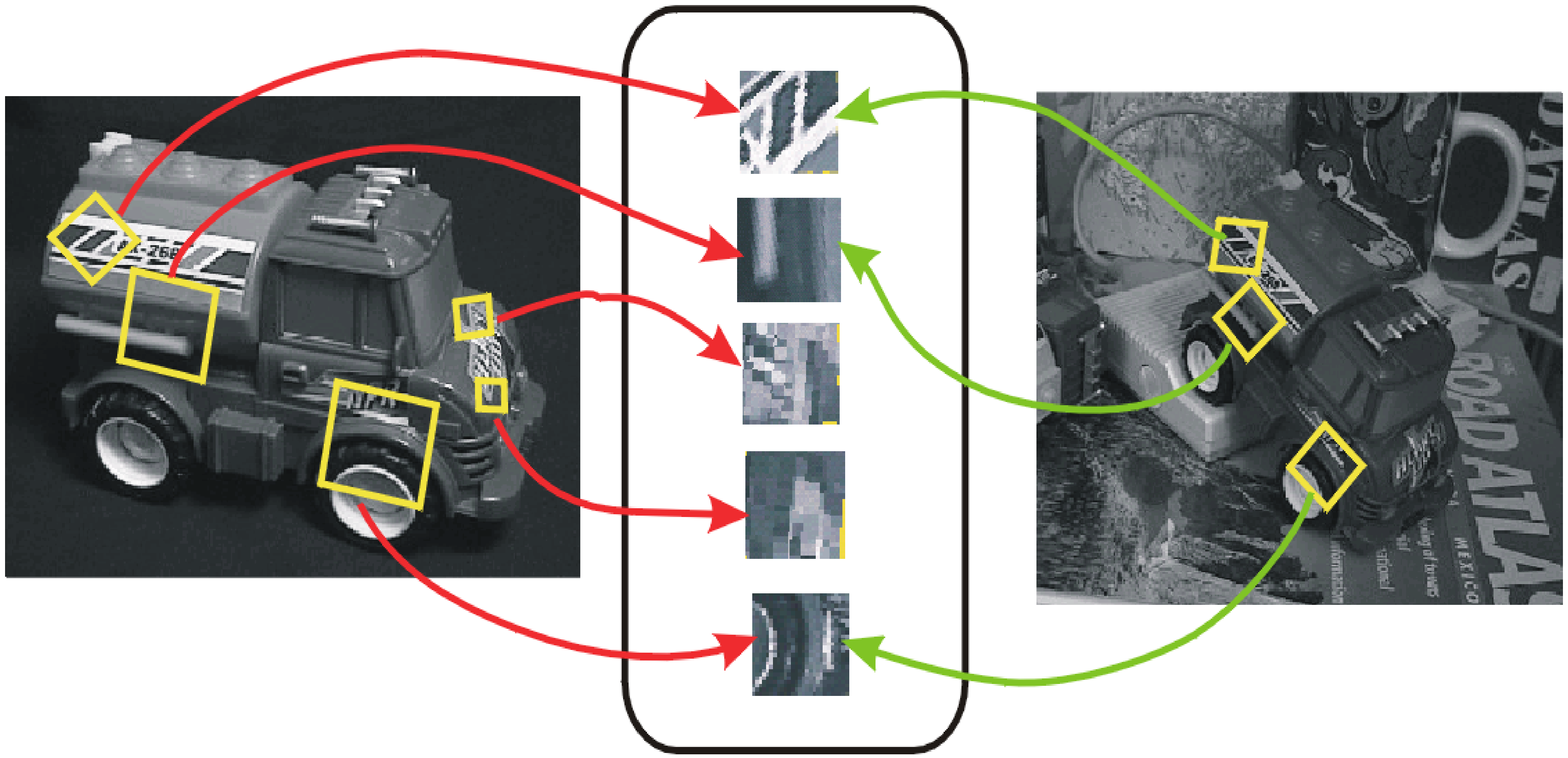


Feature matching

Invariant local features

Find features that are invariant to transformations

- geometric invariance: translation, rotation, scale
- photometric invariance: brightness, exposure, …



Feature Descriptors

Advantages of local features

Locality

– features are local, so robust to occlusion and clutter

Quantity

– hundreds or thousands in a single image

Distinctiveness:

– can differentiate a large database of objects

Efficiency

– real-time performance achievable

More motivation…

Feature points are used for:

- Image alignment (e.g., mosaics)
- 3D reconstruction
- Motion tracking (e.g. for AR)
- Object recognition
- Image retrieval
- Robot/car navigation
- … other

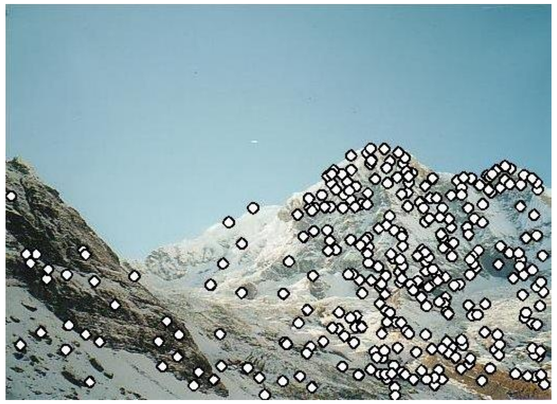
Approach

1. **Feature detection**: find it
2. **Feature descriptor**: represent it
3. **Feature matching**: match it

Feature tracking: track it, when motion

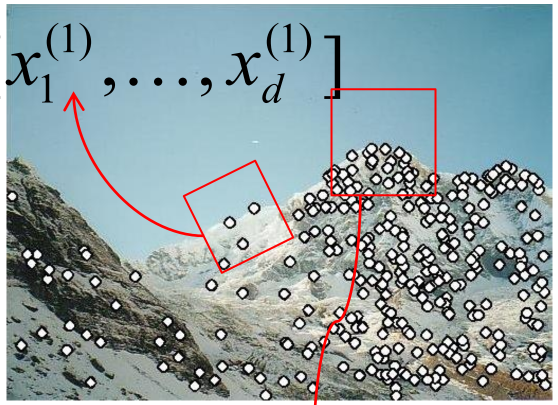
Local features: main components

1) **Detection**: Identify the interest points



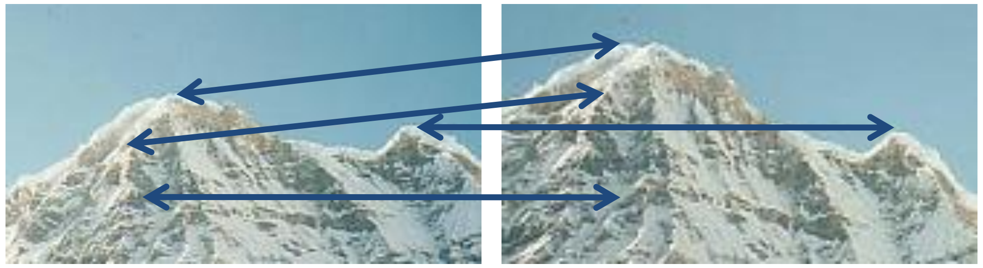
2) **Description**: Extract vector feature descriptor surrounding each interest point

$$\mathbf{x}_1 = [x_1^{(1)}, \ldots, x_d^{(1)}]$$



$$\mathbf{x}_2 = [x_1^{(2)}, \ldots, x_d^{(2)}]$$

3) **Matching**: Determine correspondence between descriptors in two views



Credit: Kristen Grauman

What makes a good feature?

Want uniqueness

Look for image regions that are unusual

- Lead to unambiguous matches in other images

How to define “unusual”?

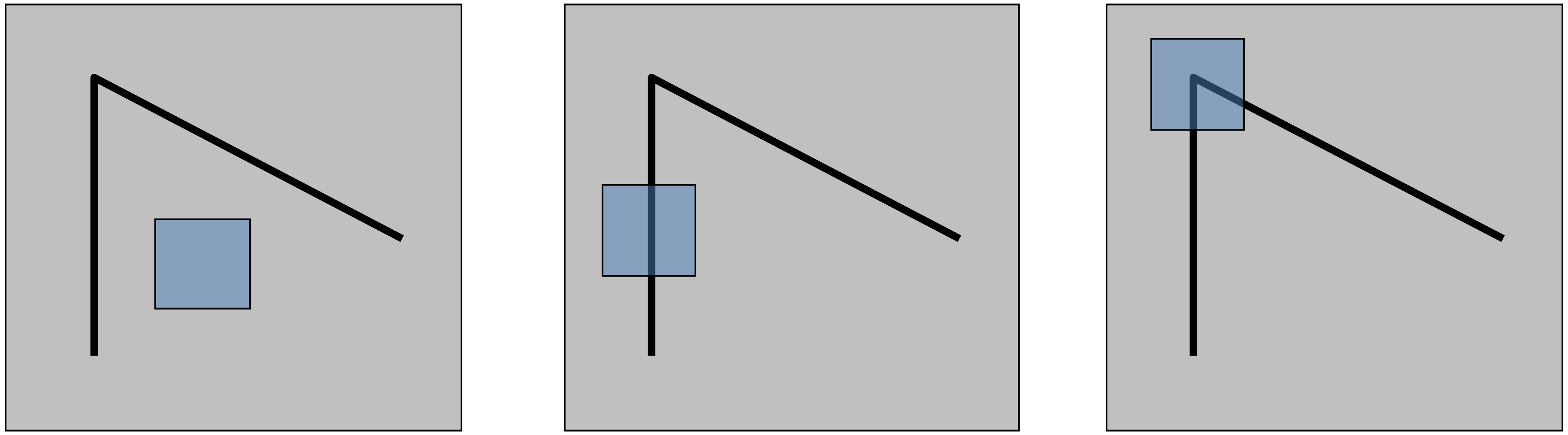
Why extract features?

- Motivation: panorama stitching
  - We have two images – how do we combine them?

Local measures of uniqueness

Suppose we only consider a small window of pixels

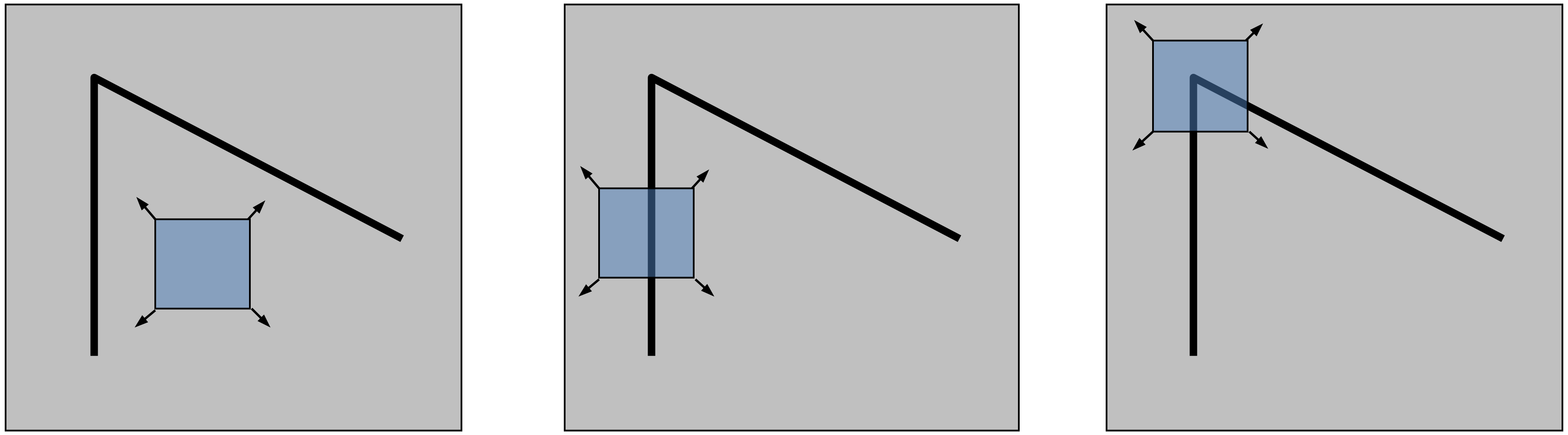
– What defines whether a feature is a good or bad candidate?



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

Local measures of uniqueness

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

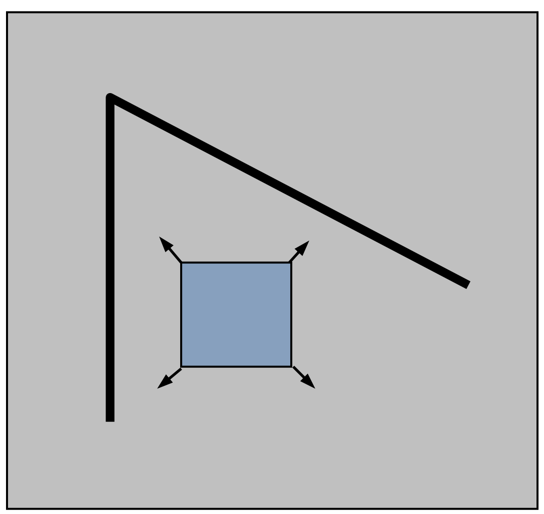


"flat" region: "edge": "corner":

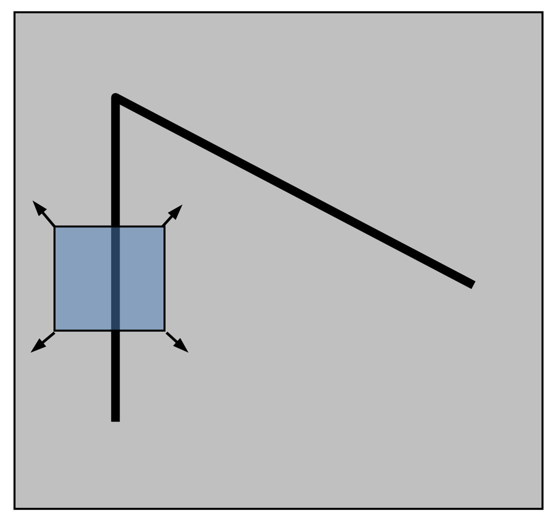
Credit: S. Seitz, D. Frolova, D. Simakov

Local measures of uniqueness

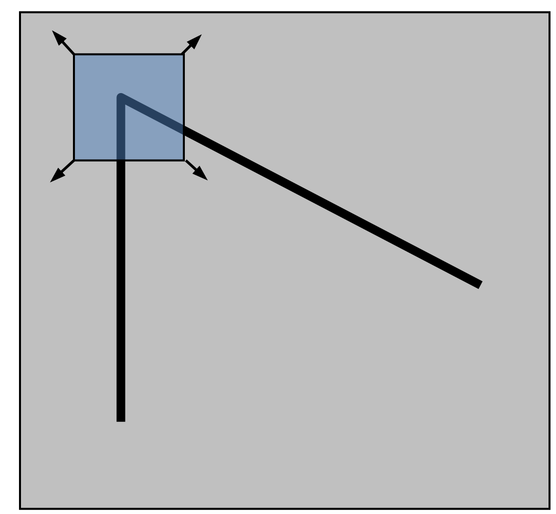
- How does the window change when you shift it?
- Shifting the window in any direction causes a big change



"flat" region:
no change in all
directions



"edge":
no change along
the edge direction

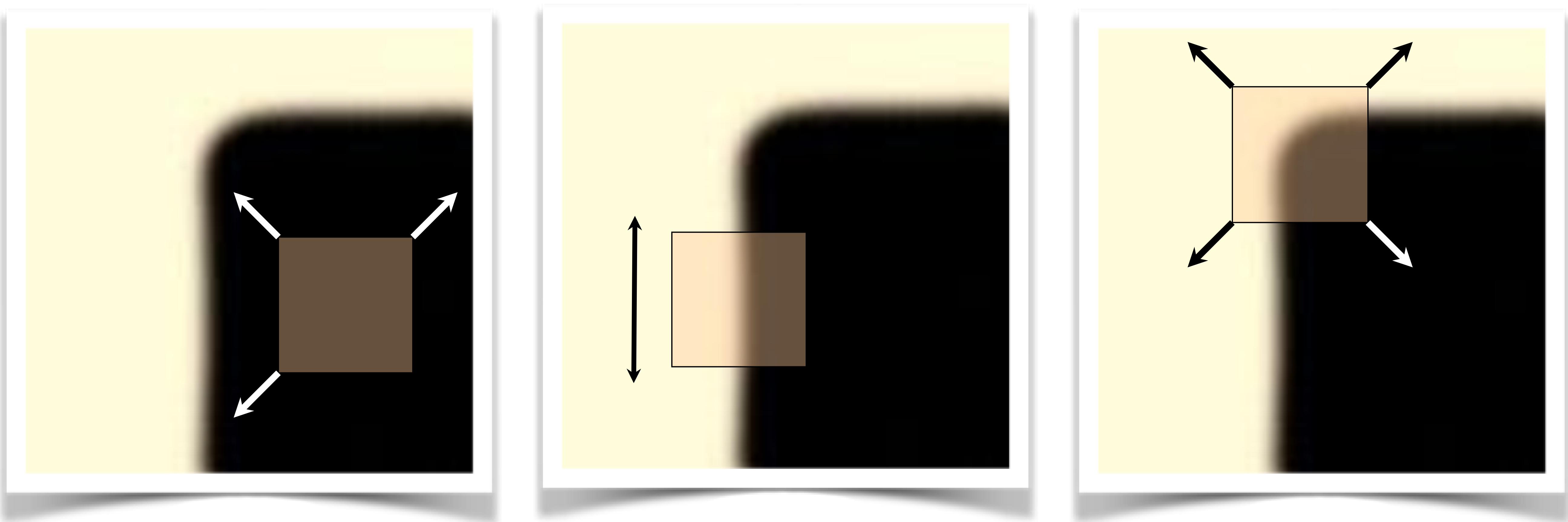


"corner":
significant change
in all directions

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

Easily recognized by looking through a small window

Shifting the window should give large change in intensity



"flat" region:
no change in all directions

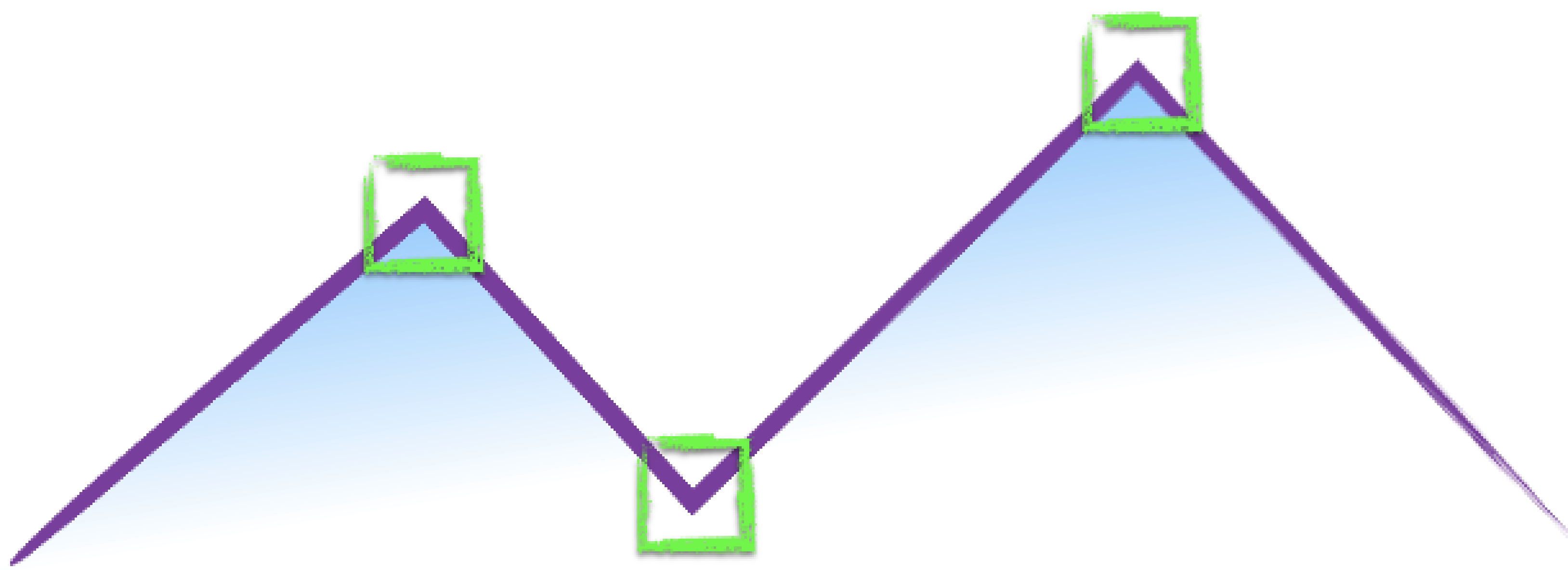
"edge":
no change along the edge direction

"corner":
significant change in all directions

[Moravec 1980]

How do you find a corner?

[Moravec 1980]

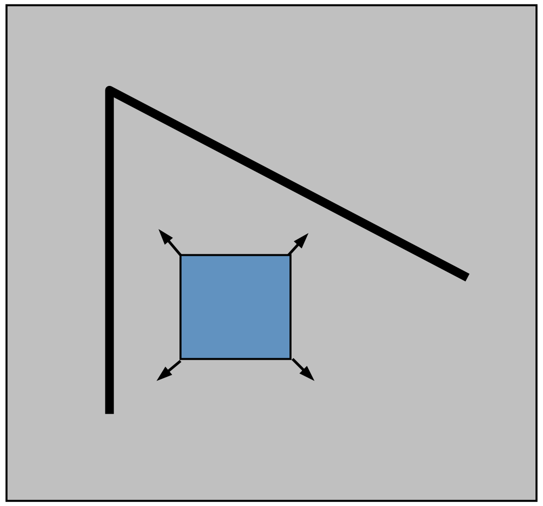


Easily recognized by looking through a small window

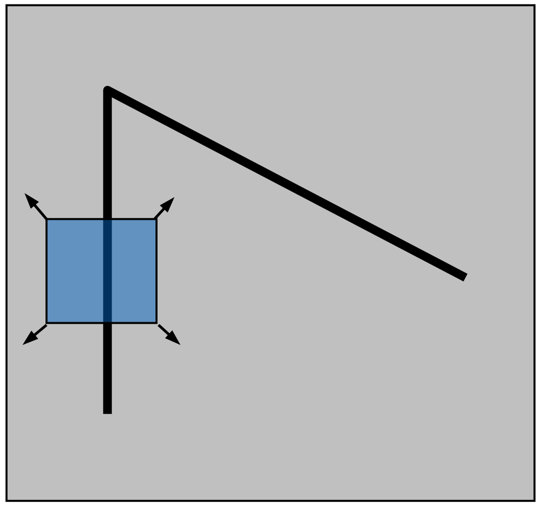
Shifting the window should give large change in intensity

Local measures of uniqueness

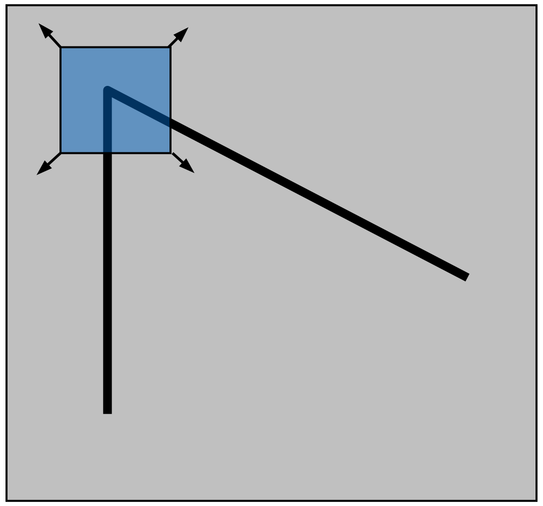
- How does the window change when you shift it?
- Shifting the window in any direction causes a big change



"flat" region:
no change in all directions



"edge":
no change along the edge direction



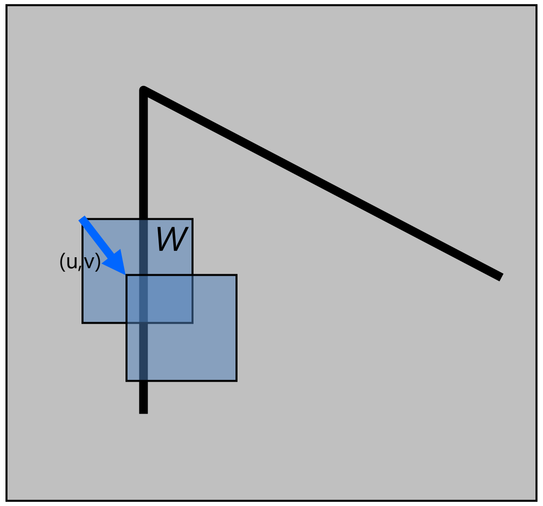
"corner":
significant change in all directions

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*):



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

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

Chris Harris and Mike Stephens (1988). "A Combined Corner and Edge Detector". *Alvey Vision Conference*.

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

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

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

Plugging this into the formula on the previous slide…

Taylor Series Expansion

- The Taylor series or Taylor expansion of a function is an infinite sum of terms that are expressed in terms of the function's derivatives at a single point.

Taylor Series

$$\begin{aligned} f(x) &= \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n \\ &= f(a) + f'(a)(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \frac{f'''(a)}{3!}(x-a)^3 + \cdots \end{aligned}$$

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

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

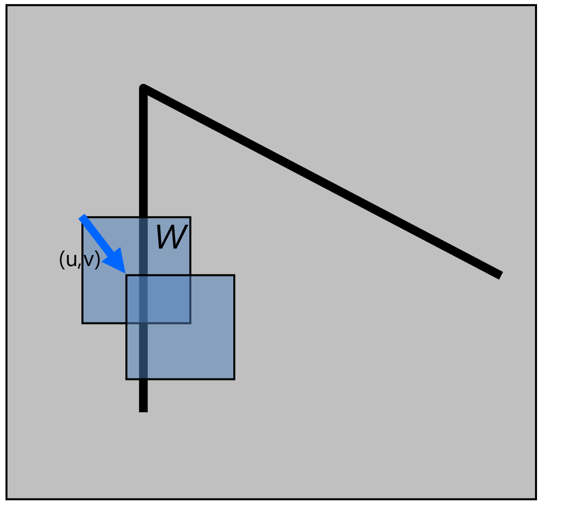
$$\text{shorthand: } I_x = \frac{\partial I}{\partial x}$$

Plugging this into the formula on the previous slide…

Corner detection: the math

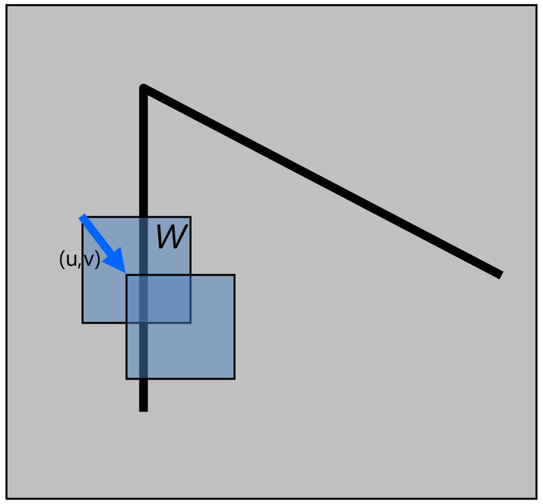
Consider shifting the window *W* by (*u*,*v*)

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



$$
\begin{aligned}
E(u,v) &= \sum_{(x,y)\in W} \left[I(x+u,y+v) - I(x,y)\right]^2 \\
&\approx \sum_{(x,y)\in W} \left[I(x,y) + I_x u + I_y v - I(x,y)\right]^2 \\
&\approx \sum_{(x,y)\in W} \left[I_x u + I_y v\right]^2
\end{aligned}
$$

Corner detection: the math



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

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

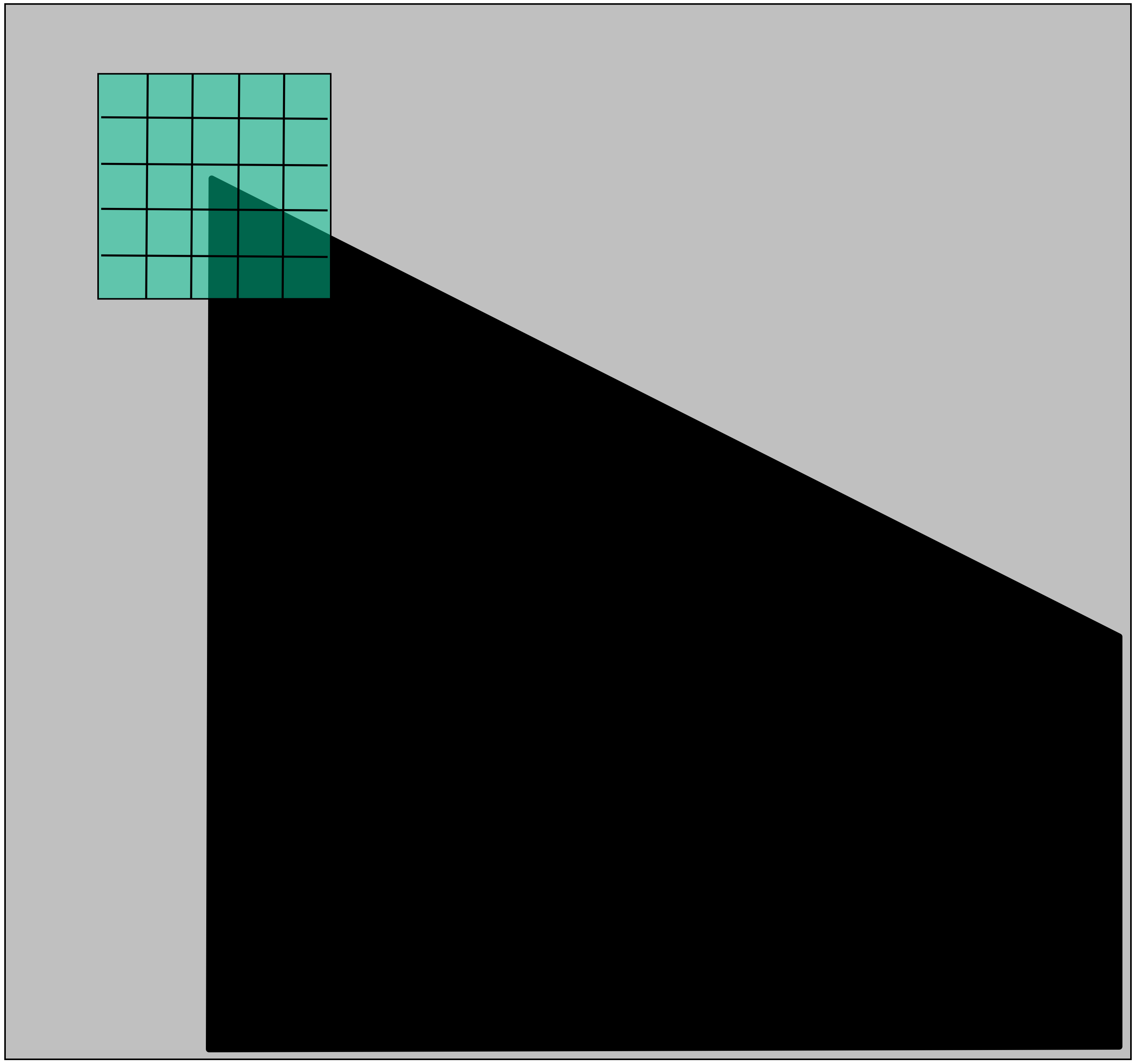
$$E(u,v) \approx \sum_{(x,y)\in W} [I_x u + I_y v]^2$$

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

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

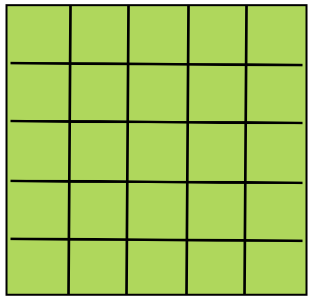
- Thus, *E*(*u*,*v*) is locally approximated as a quadratic error function

Compute image gradients over a small region
(not just a single pixel)



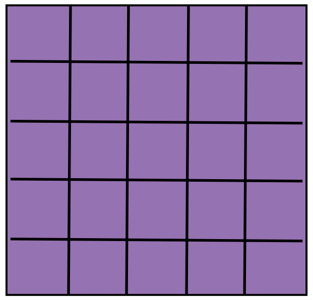
array of x gradients

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

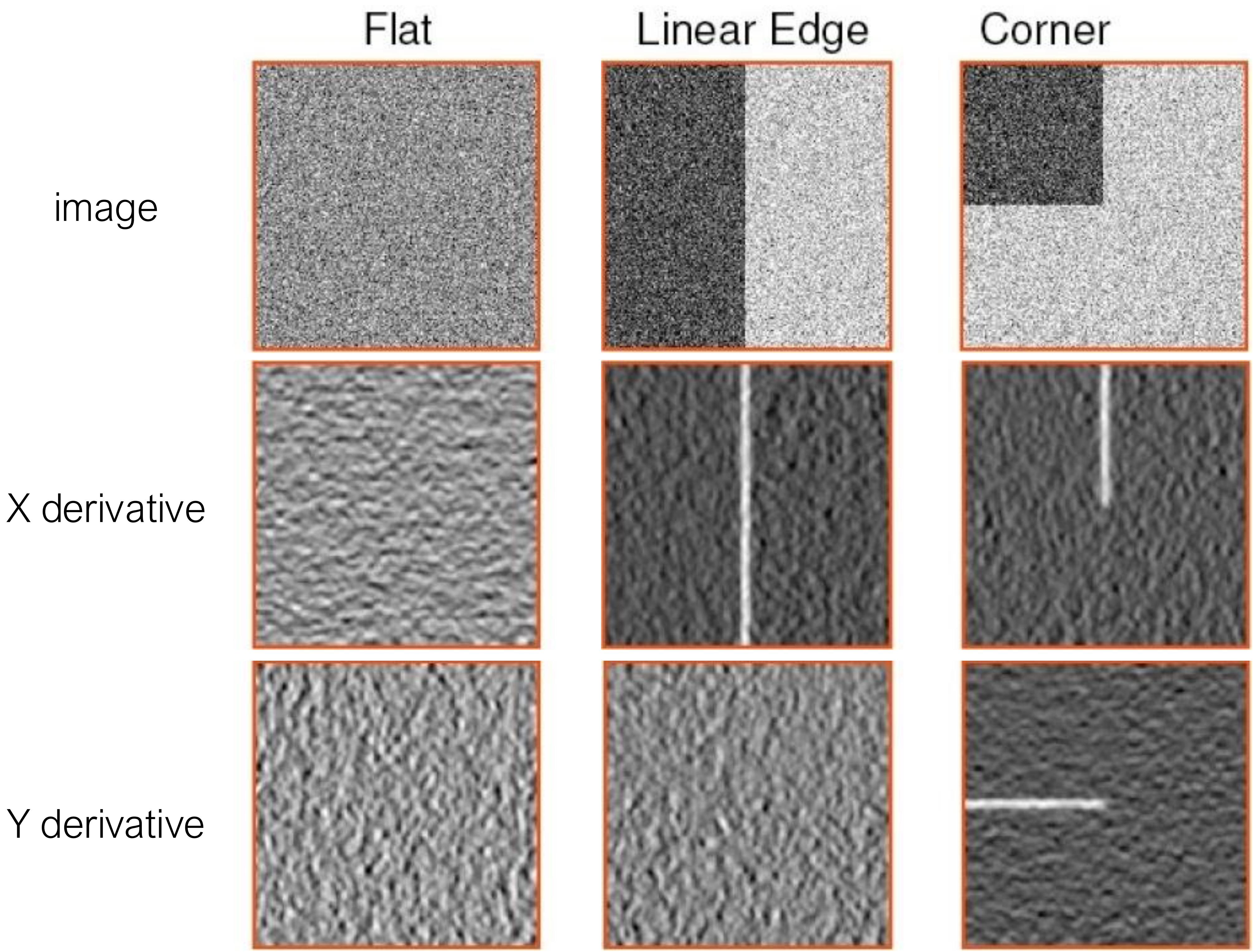


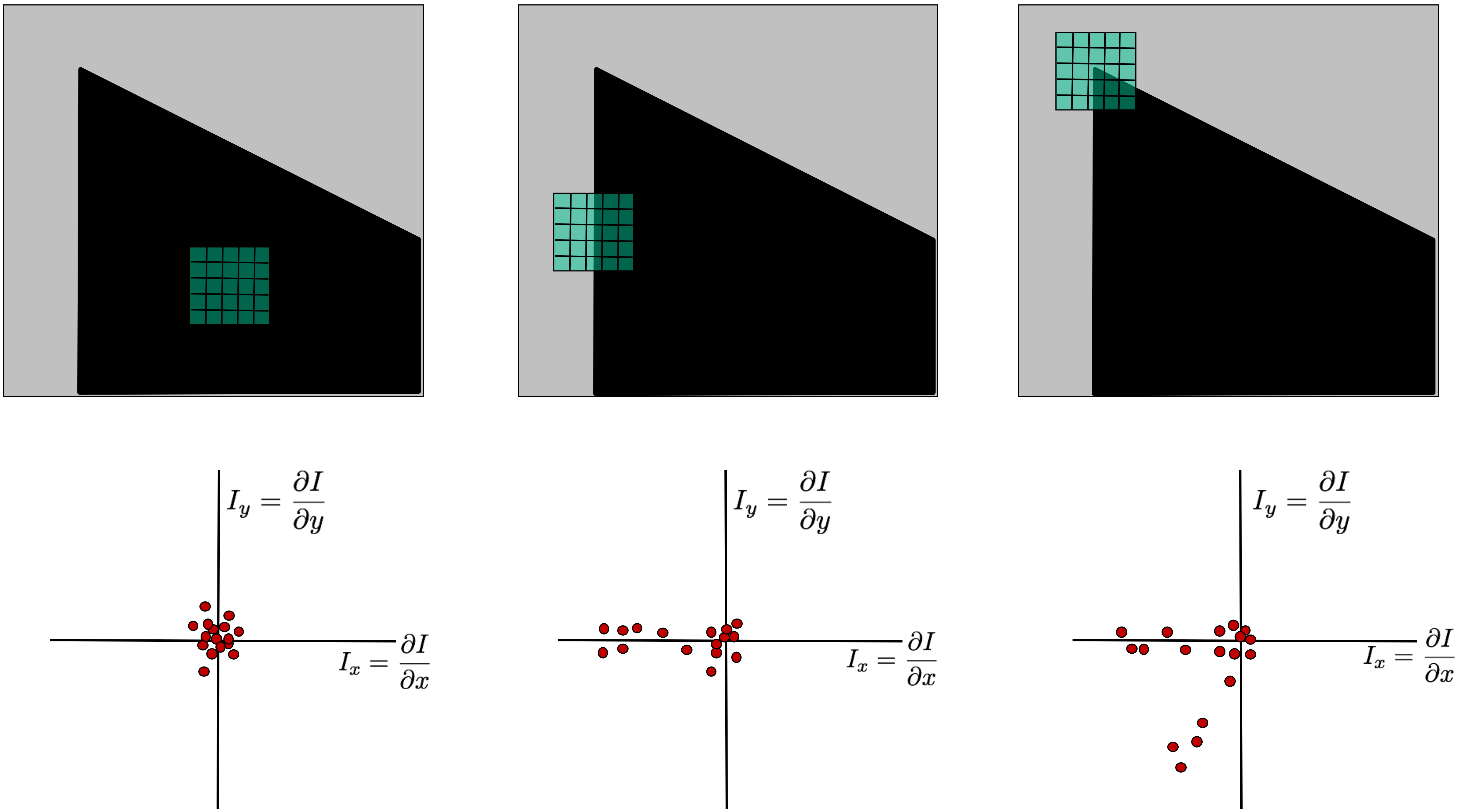
array of y gradients

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

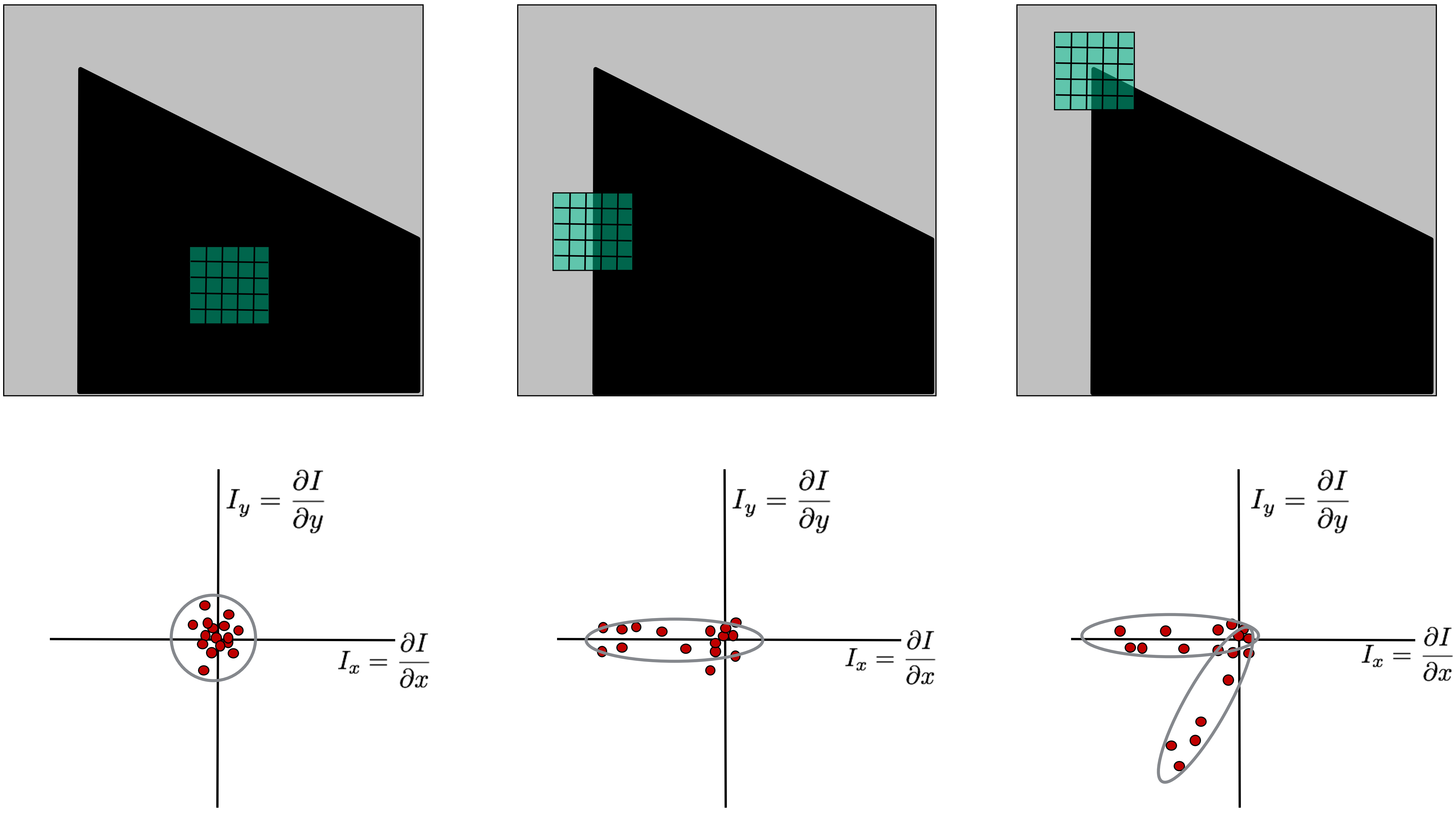


visualization of gradients

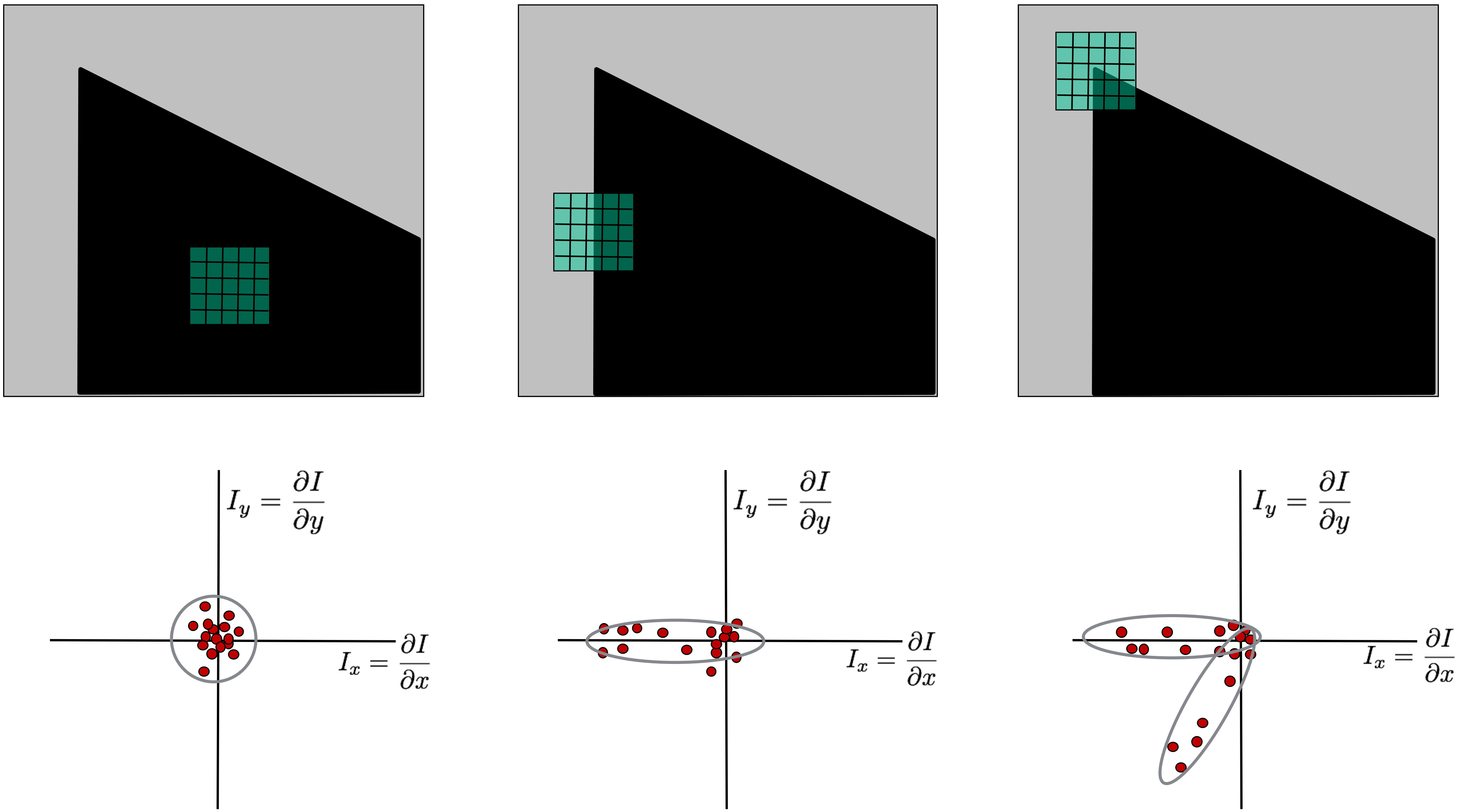




What does the distribution tell you about the region?



distribution reveals edge orientation and magnitude

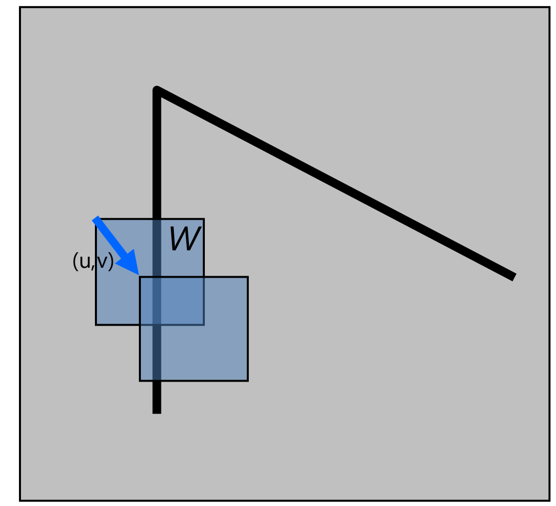


How do you quantify orientation and magnitude?

Corner detection: the math

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

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

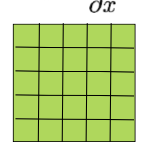
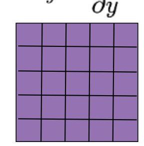


$$E(u,v) \approx \sum_{(x,y)\in W} [I_x u + I_y v]^2$$

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

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

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

=sum( * )

array of x gradients $\quad$ array of y gradients

The second moment / covariance matrix

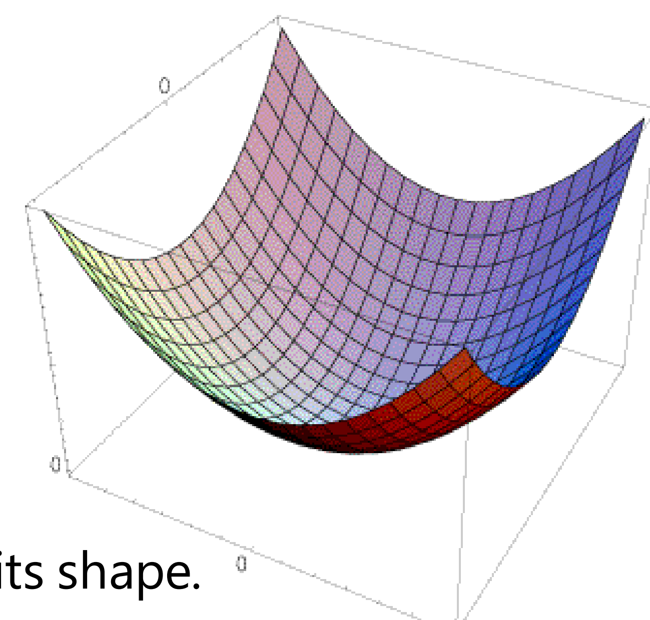
The surface $E(u,v)$ is locally approximated by a quadratic form.

$$
\begin{aligned}
E(u,v) &\approx Au^2 + 2Buv + Cv^2 \\
&\approx \begin{bmatrix} u & v \end{bmatrix} \underbrace{\begin{bmatrix} A & B \\ B & C \end{bmatrix}}_{H} \begin{bmatrix} u \\ v \end{bmatrix}
\end{aligned}
$$

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

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

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



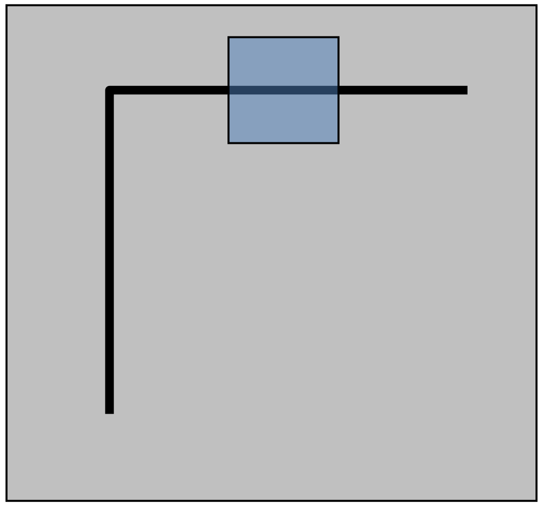
Let’s try to understand its shape.

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

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

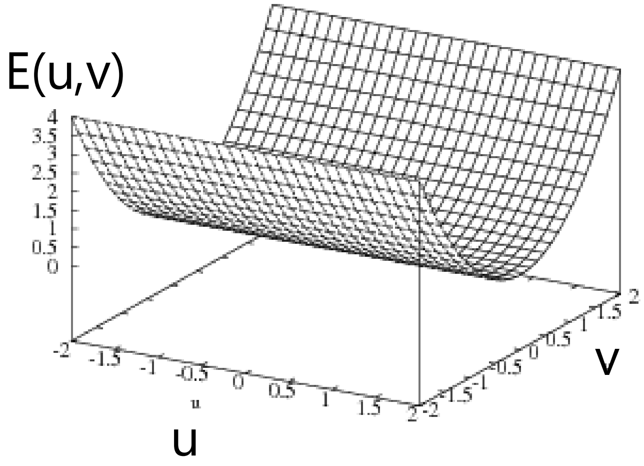
$$B = \sum_{(x,y)\in W} I_x I_y$$

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



Horizontal edge: $I_x = 0$

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

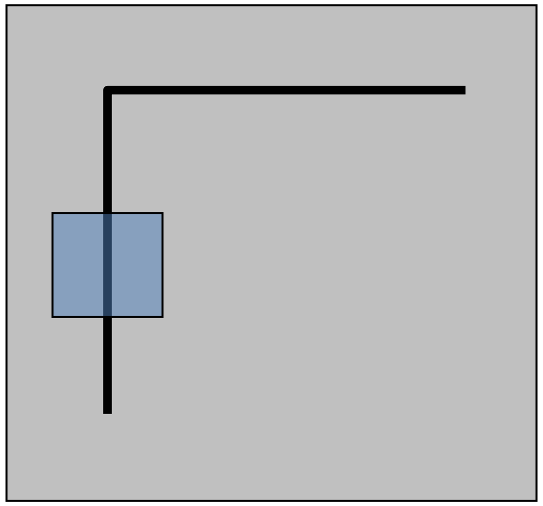


$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} \underbrace{\begin{bmatrix} A & B \\ B & C \end{bmatrix}}_{H} \begin{bmatrix} u \\ v \end{bmatrix}$$

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

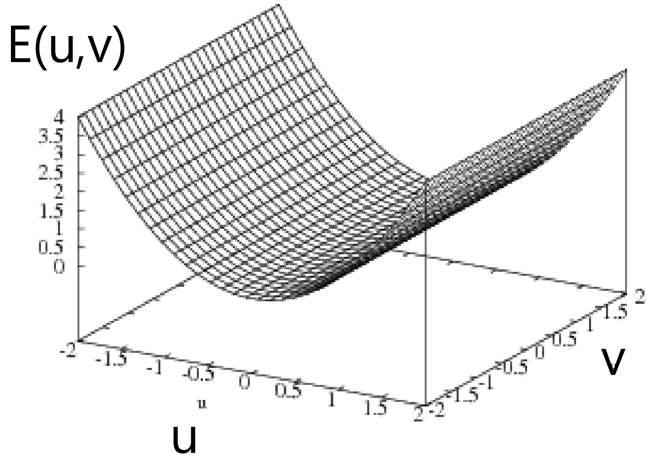
$$B = \sum_{(x,y)\in W} I_x I_y$$

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



Vertical edge: $I_y = 0$

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



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:

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