Non-Linear Filtering and Hough Transform

Overview of today's lecture

New in lecture 4:

- Non-Linear Filtering

- Finding boundaries.

- Line fitting.

- Line parameterizations.

- Hough transform.

- Hough circles.

- Some applications.

Slide credits

Most of these slides were adapted from:

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

Some slides were inspired or taken from:

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

Non-Linear Filtering

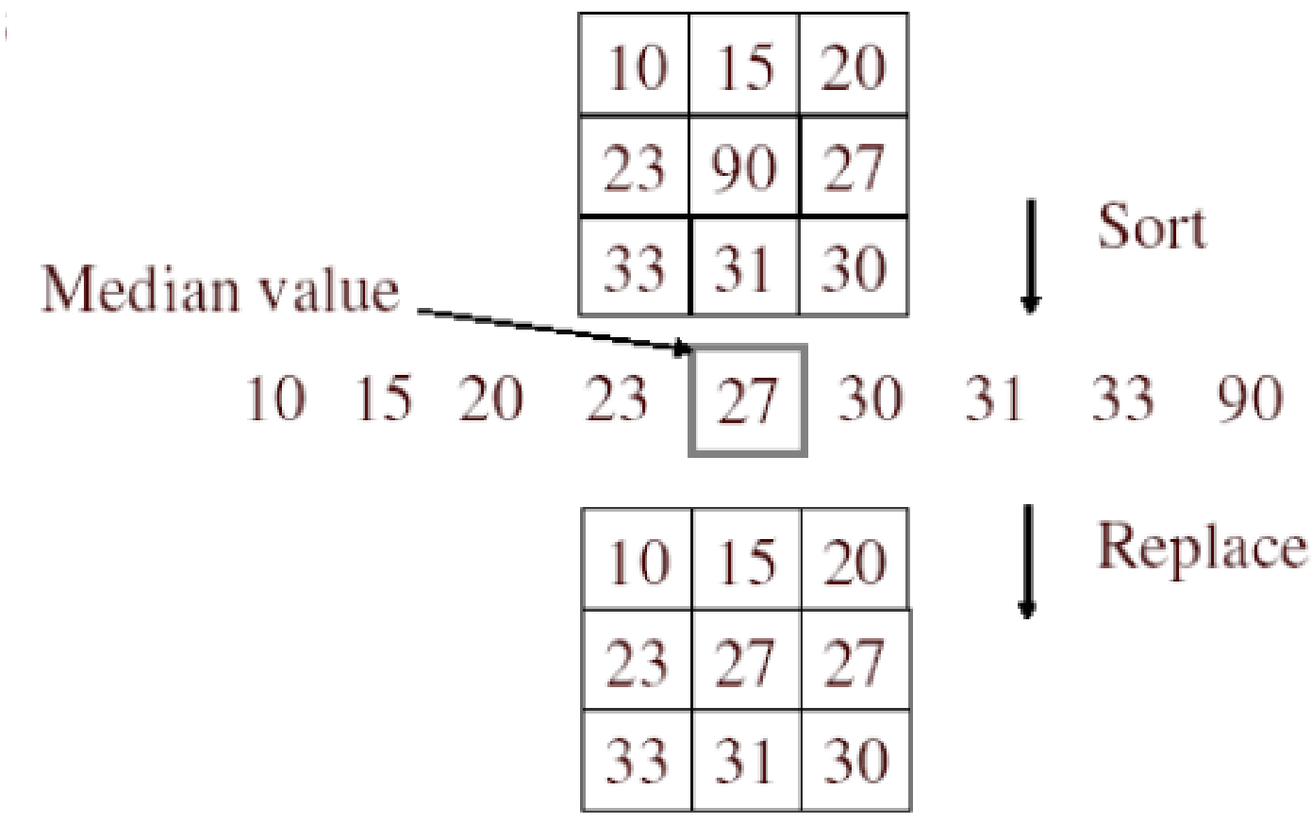
Effect of smoothing filters

5x5

Additive Gaussian noise

Salt and pepper noise

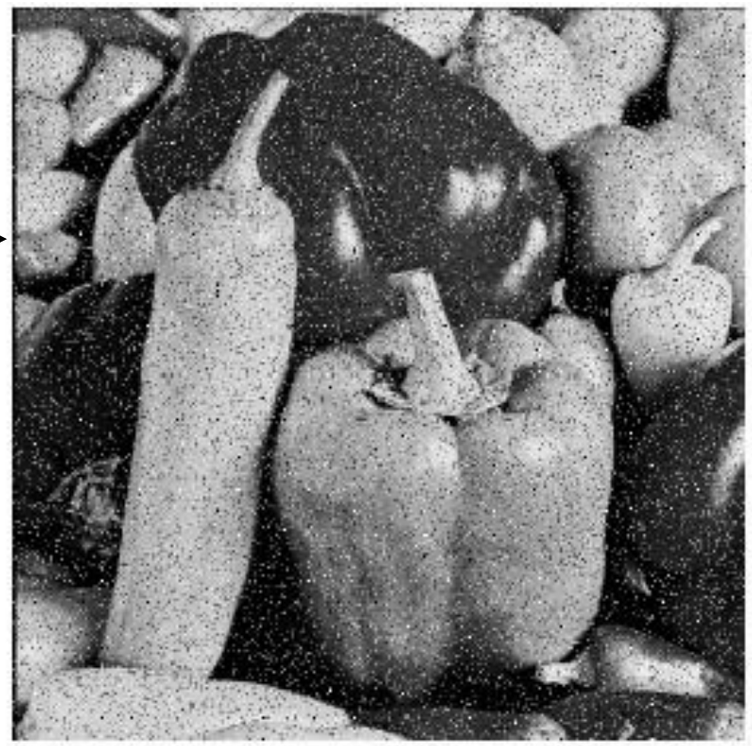
Median filter



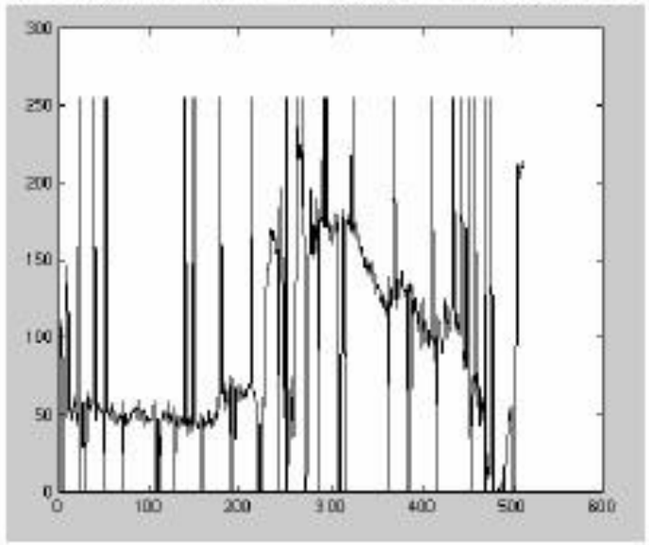
- No new pixel values introduced
- Removes spikes: good for impulse, salt & pepper noise
- Non-linear filter

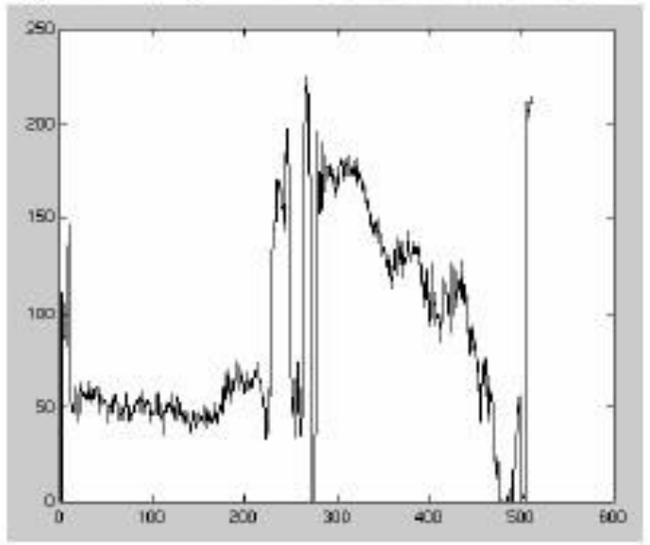
Median filter

Salt and pepper noise



Median filtered



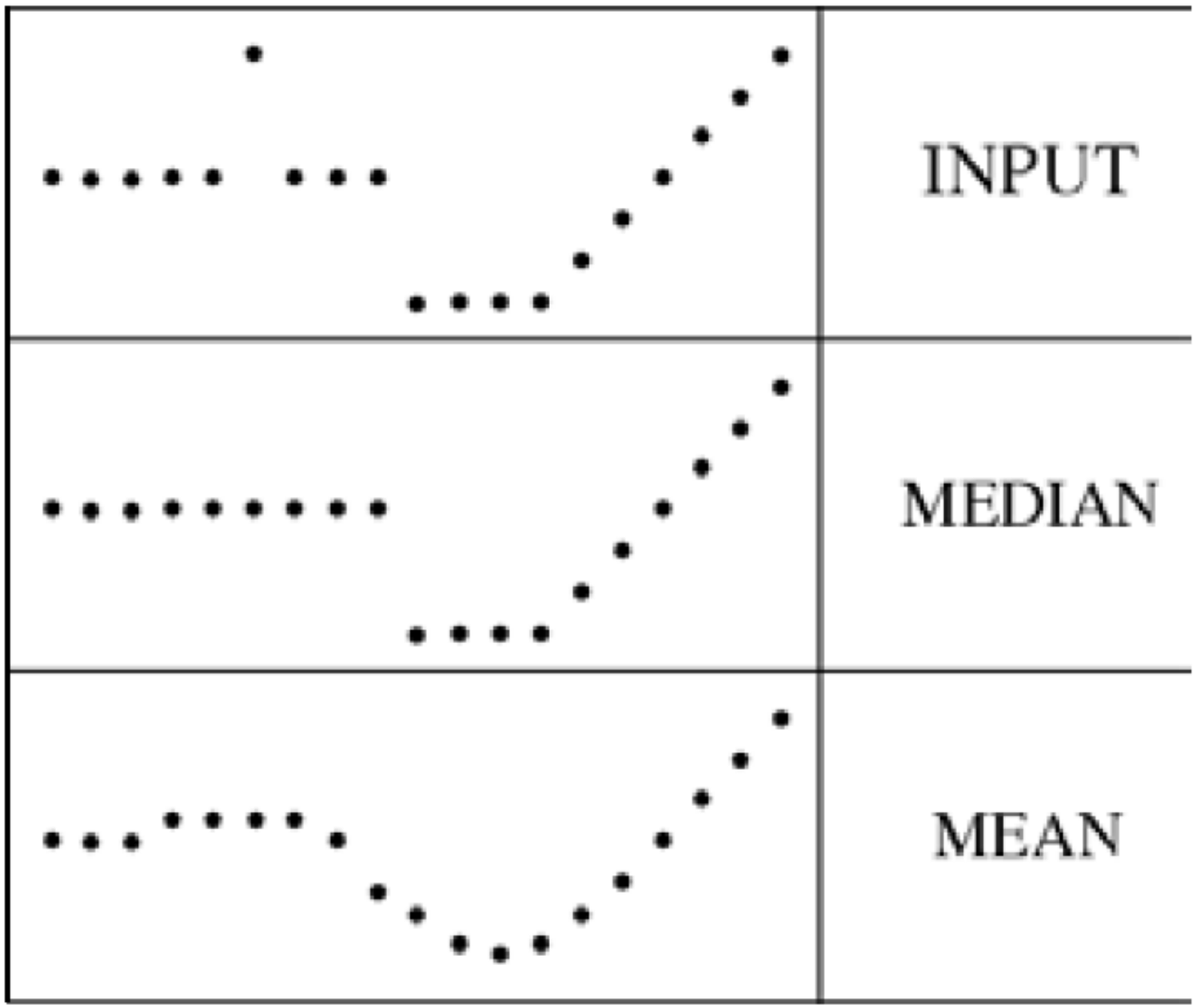


Plots of a row of the image

Source: M. Hebert

Median filter

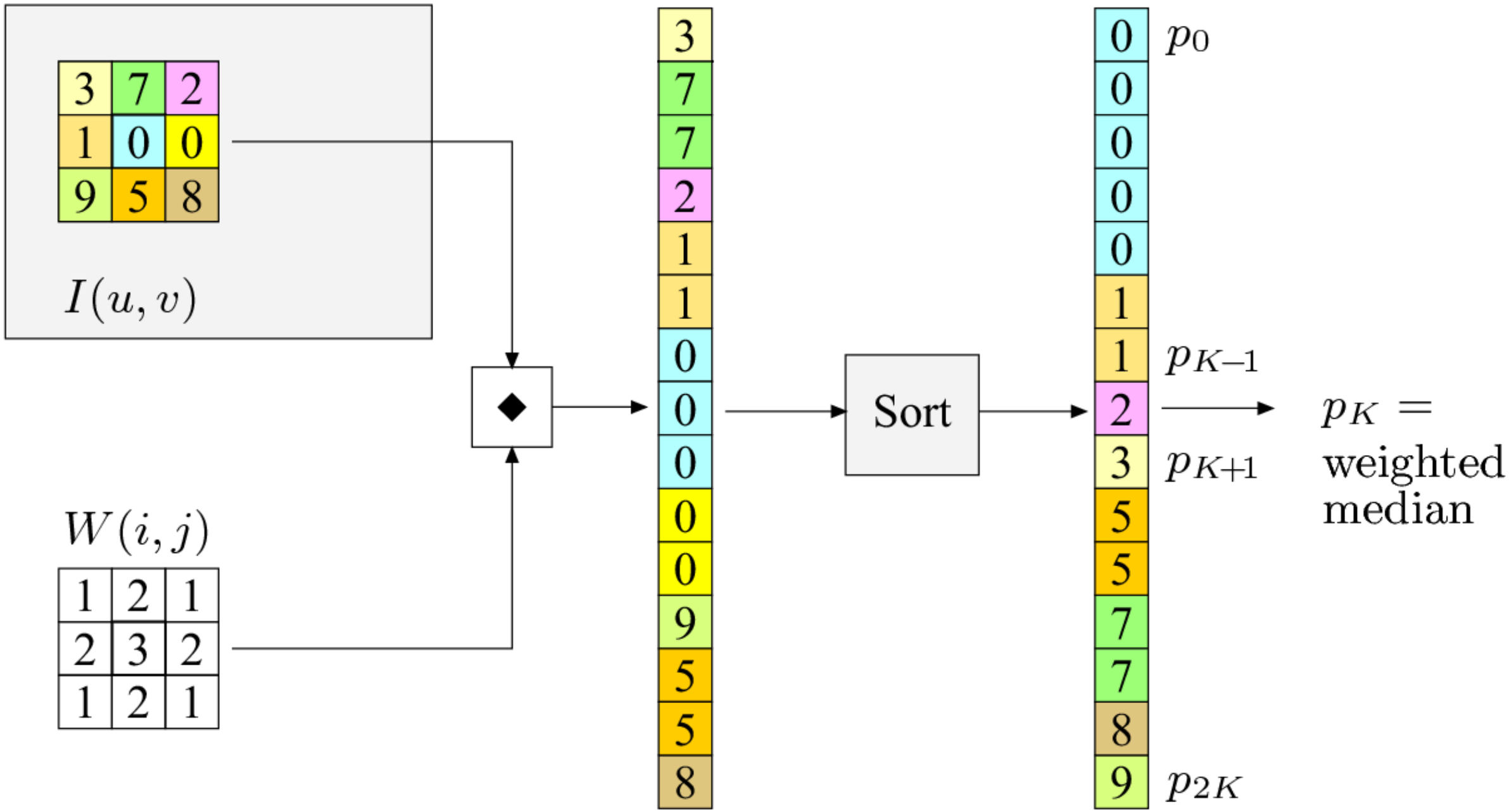
- Median filter is edge preserving



Weighted Median Filter

- What if we want to apply median filter while giving higher weights to some pixels than others?
- Solution: Replicate pixels values in proportion to their weight → Weighted Median Filter
- Note that weights have to be integers

Weighted Median Filter

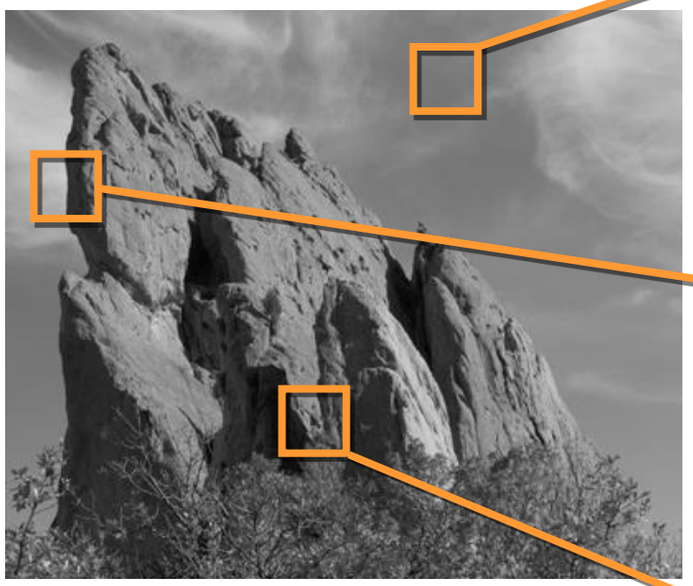


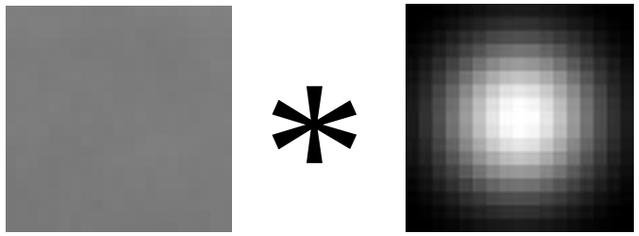
Bilateral Filtering

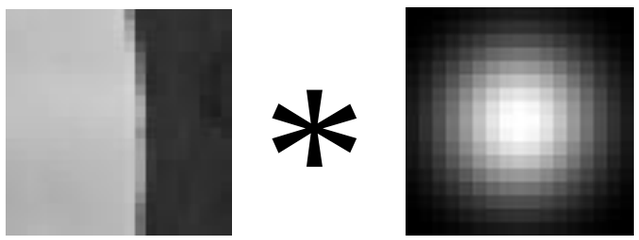
- What if want to remove Gaussian noise without blurring details of the image?
- Bilateral filtering smoothens an image with a degree inversely proportional to the local contrast

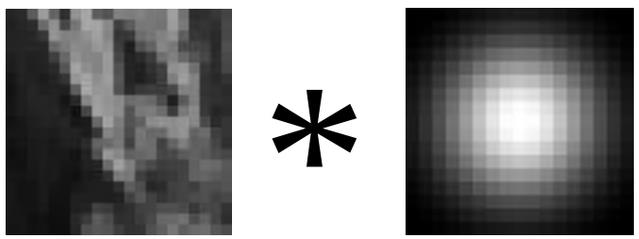
Blur Comes from Averaging across Edges

input









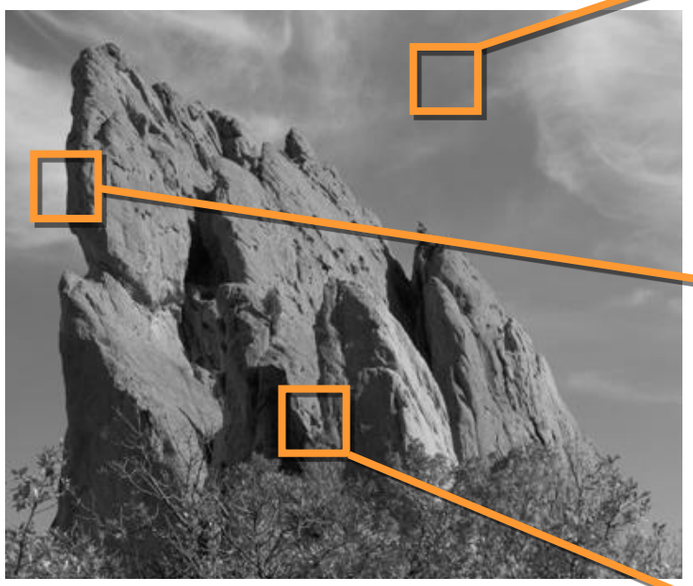
output

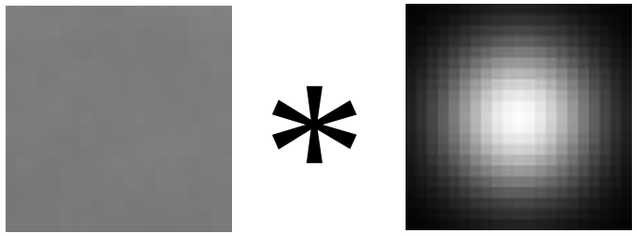
Same Gaussian kernel everywhere.

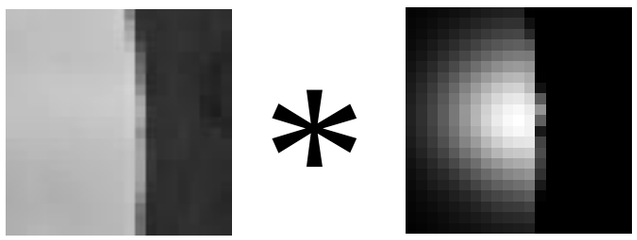
Bilateral Filter [Aurich 95, Smith 97, Tomasi 98]

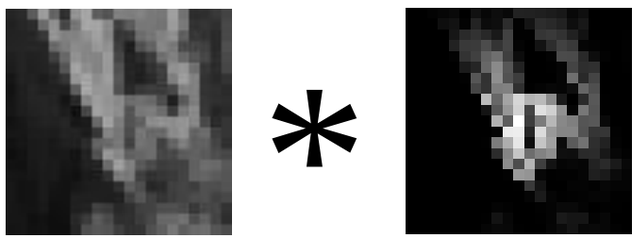
No Averaging across Edges

input









output

The kernel shape depends on the image content.

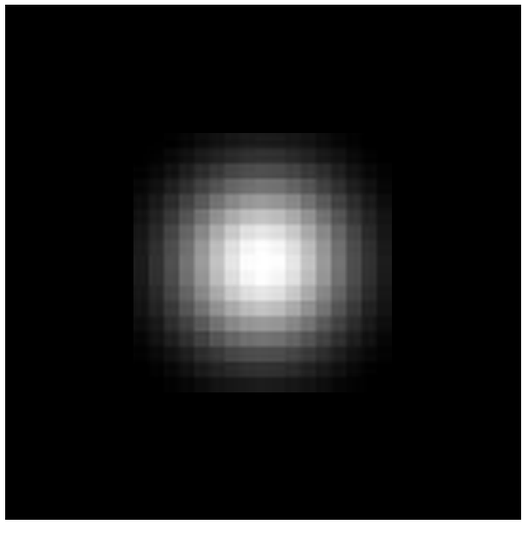
Bilateral Filter Definition: an Additional Edge Term

Same idea: **weighted average of pixels**.

$$BF[I]_{\mathbf{p}} = \underbrace{\frac{1}{W_{\mathbf{p}}}}_{\text{new}} \sum_{\mathbf{q}\in S} \underbrace{G_{\sigma_s}(\|\mathbf{p}-\mathbf{q}\|)}_{\text{not new}} \underbrace{G_{\sigma_r}(|I_{\mathbf{p}}-I_{\mathbf{q}}|)}_{\text{new}} I_{\mathbf{q}}$$

normalization factor

space weight



range weight

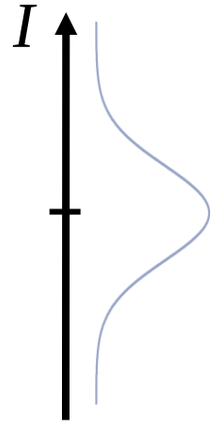
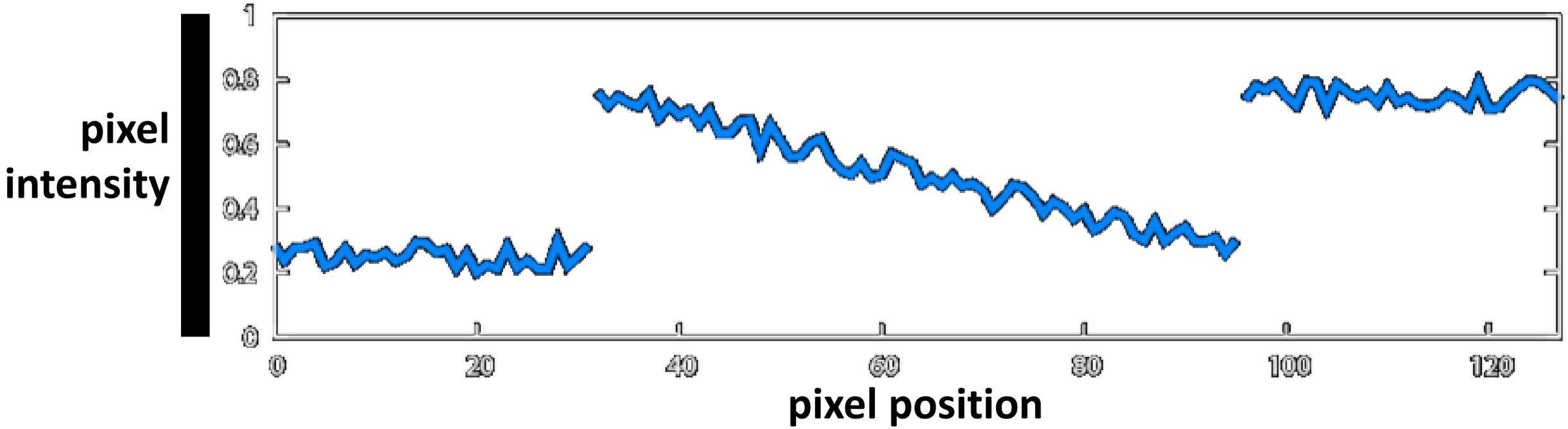


Illustration a 1D Image

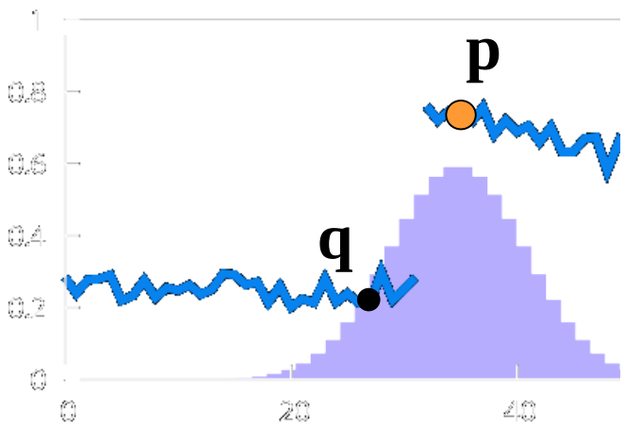
- 1D image = line of pixels

- Better visualized as a plot



Gaussian Blur and Bilateral Filter

Gaussian blur

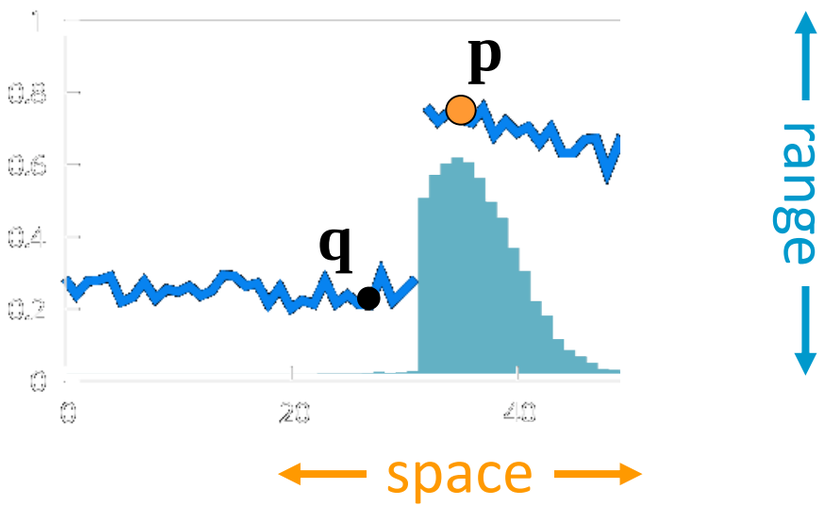


$$GB[I]_{\mathbf{p}} = \sum_{\mathbf{q}\in S} G_\sigma(\|\mathbf{p}-\mathbf{q}\|)\, I_{\mathbf{q}}$$

space

Bilateral filter

[Aurich 95, Smith 97, Tomasi 98]



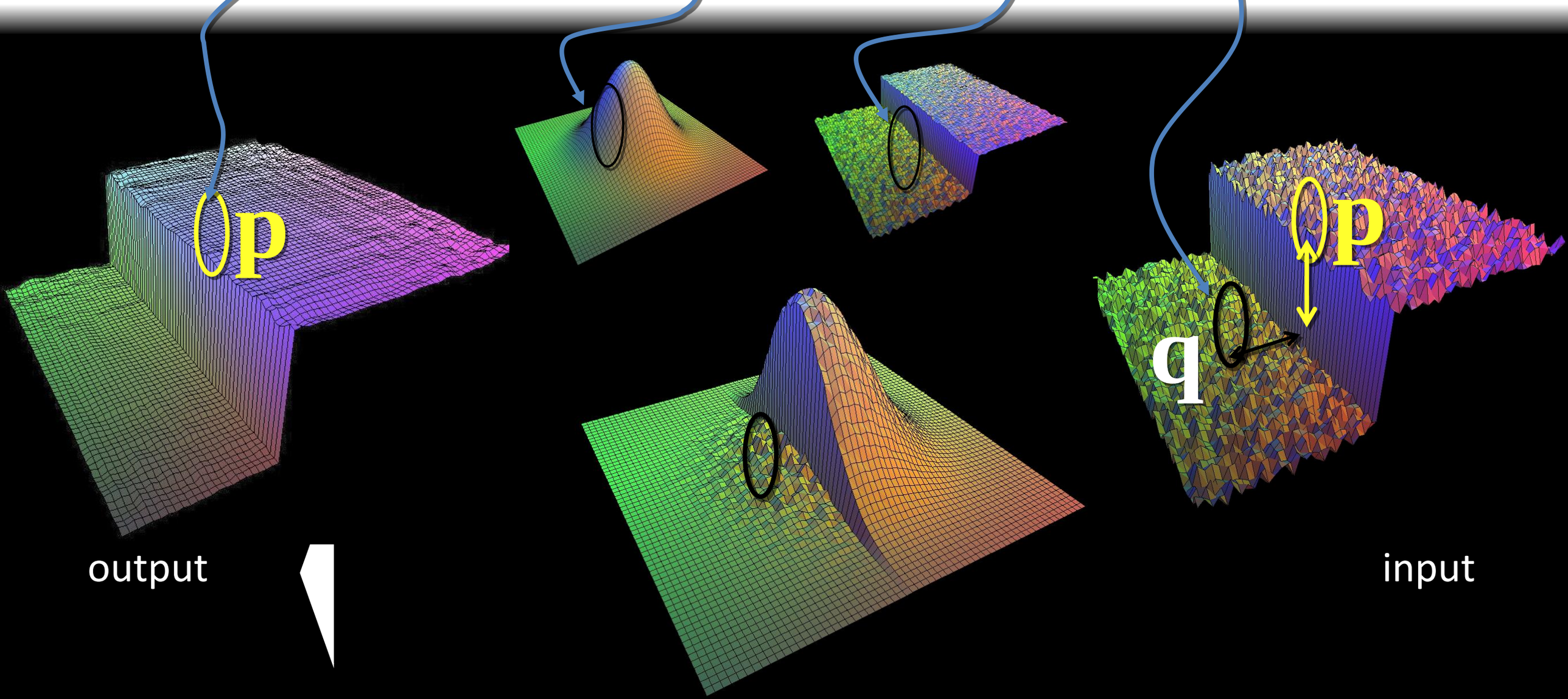
$$BF[I]_{\mathbf{p}} = \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q}\in S} G_{\sigma_s}(\|\mathbf{p}-\mathbf{q}\|)\, G_{\sigma_r}(|I_{\mathbf{p}}-I_{\mathbf{q}}|)\, I_{\mathbf{q}}$$

space range

normalization

Bilateral Filter on a Height Field

$$\underbrace{BF[I]_{\mathbf{p}}}_{\text{output}} = \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q} \in S} \underbrace{G_{\sigma_s}(\|\mathbf{p} - \mathbf{q}\|)}_{\text{domain}} \underbrace{G_{\sigma_r}(|I_{\mathbf{p}} - I_{\mathbf{q}}|)}_{\text{range}} \underbrace{I_{\mathbf{q}}}_{\text{input}}$$



reproduced from [Durand 02]

Space and Range Parameters

$$BF[I]_{\mathbf{p}} = \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q} \in S} G_{\sigma_s}(\|\mathbf{p} - \mathbf{q}\|)\, G_{\sigma_r}(|I_{\mathbf{p}} - I_{\mathbf{q}}|) I_{\mathbf{q}}$$

- space σ_s : spatial extent of the kernel, size of the considered neighborhood.

- range σ_r : “minimum” amplitude of an edge

Exploring the Parameter Space

input

$\sigma_r = 0.1$

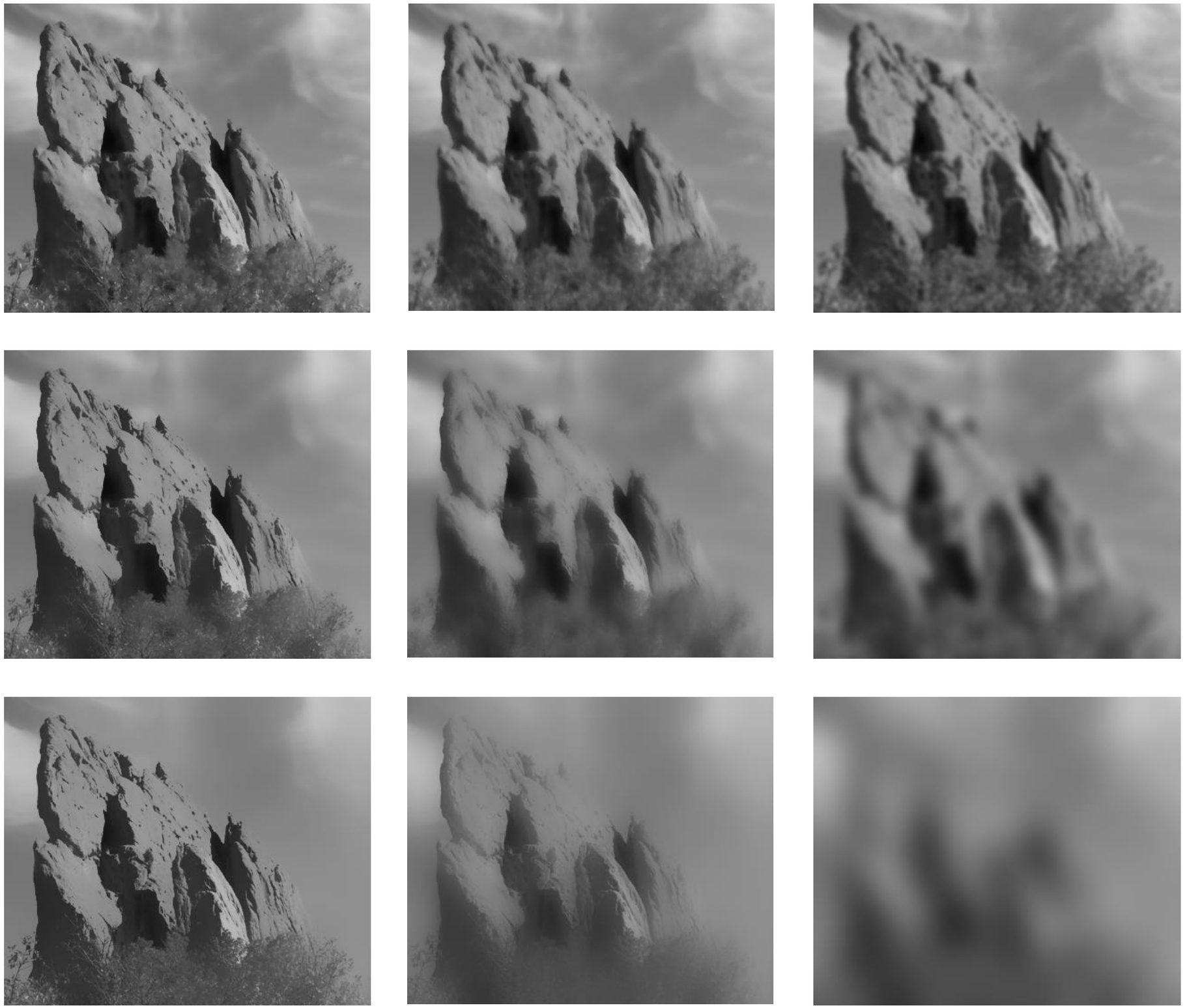
$\sigma_r = 0.25$

$\sigma_r = \infty$
(Gaussian blur)

$\sigma_s = 2$

$\sigma_s = 6$

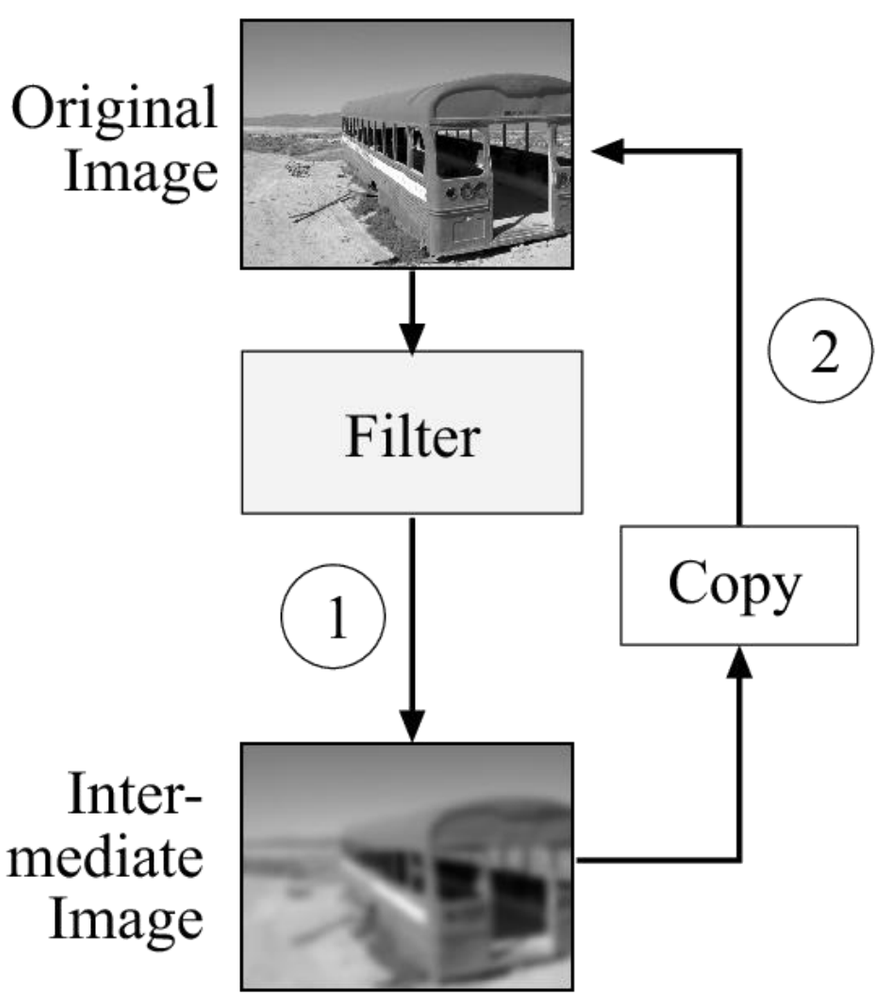
$\sigma_s = 18$



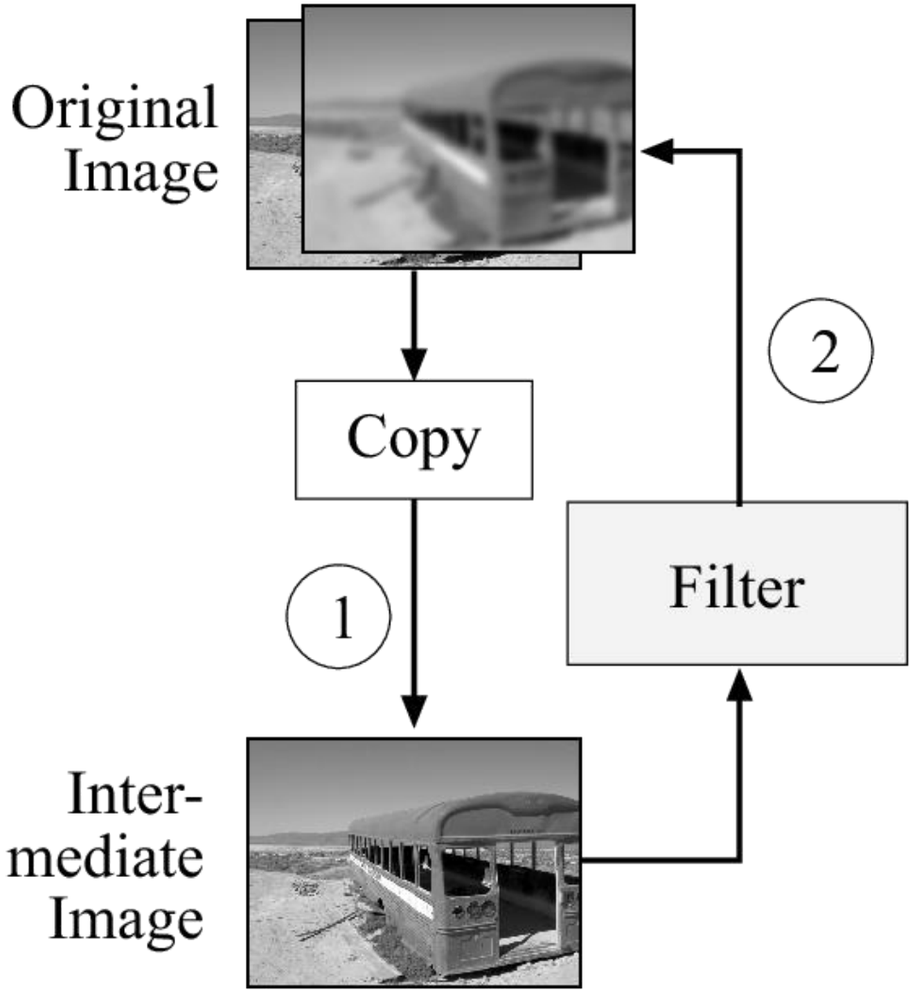
Practical Issue in Filtering

- Making temp copy of input image
- Handling image boundary pixels

Filtering: Using Temp Copy of Input



Version A

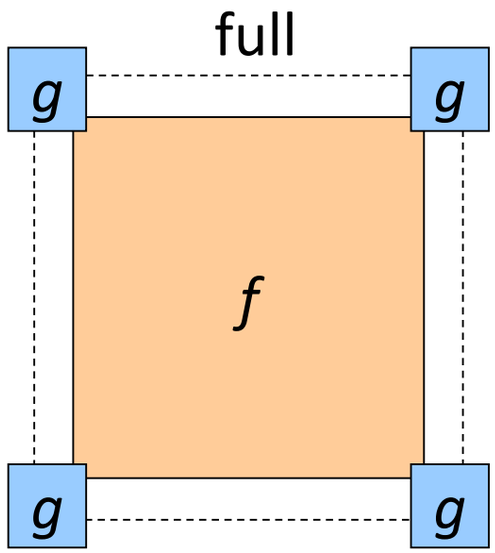


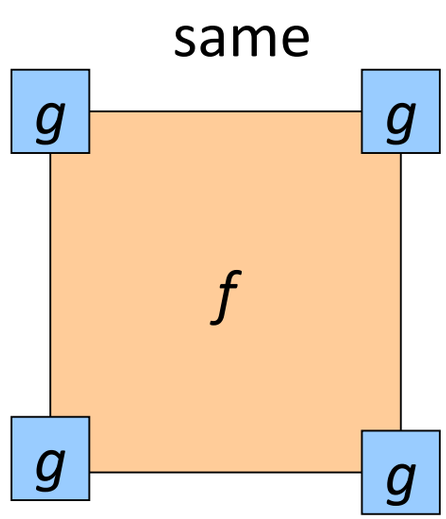
Version B

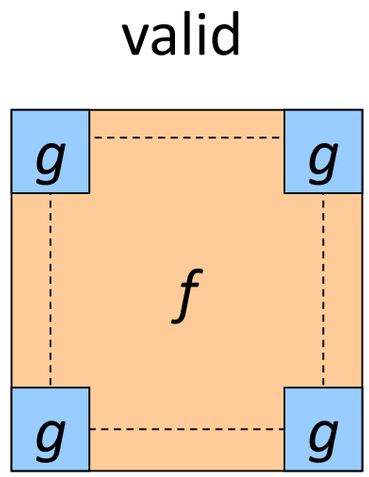
Filtering: Boundary Issues

- What is the size of the output filtered image?
- MATLAB: output size / “shape” options
  - *shape* = ‘full’: output size is sum of sizes of f and g
  - *shape* = ‘same’: output size is same as f
  - *shape* = ‘valid’: output size is difference of sizes of f and g

- OpenCV: only ‘same’ mode supported
  - Use copyMakeBorder() for ‘full’ mode
  - Set ROI for ‘valid’ mode







Source: S. Lazebnik

Filtering: Boundary Issues

- What about near the edge?
  - the filter window falls off the edge of the image
  - need to extrapolate
  - methods:
    - clip filter (black)
      - borderType = BORDER_CONSTANT
    - wrap around
      - borderType = BORDER_WRAP
    - copy edge
      - borderType = BORDER_REPLICATE
    - reflect across edge
      - borderType = BORDER_REFLECT_101

Source: S. Marschner

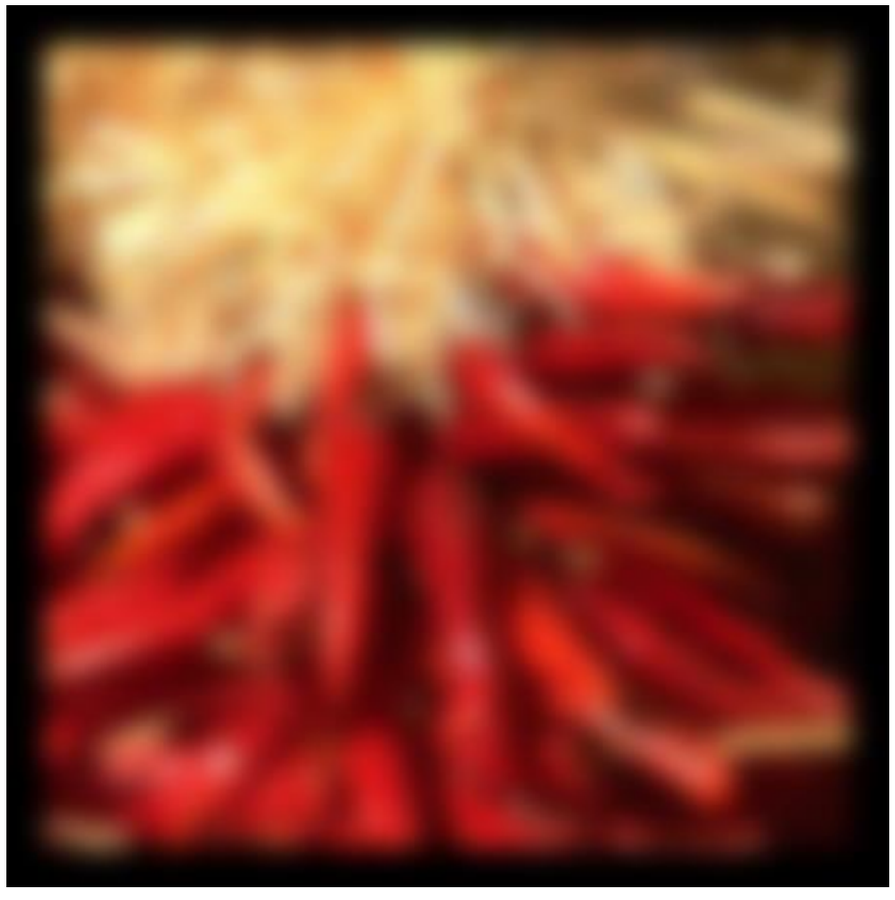
Filtering: Boundary Issues

- What about near the edge?
  - the filter window falls off the edge of the image
  - need to extrapolate
  - methods:
    - **clip filter (black)**
      - borderType = BORDER_CONSTANT
    - wrap around
      - borderType = BORDER_WRAP
    - copy edge
      - borderType = BORDER_REPLICATE
    - reflect across edge
      - borderType = BORDER_REFLECT_101

Source: S. Marschner

Filtering: Boundary Issues

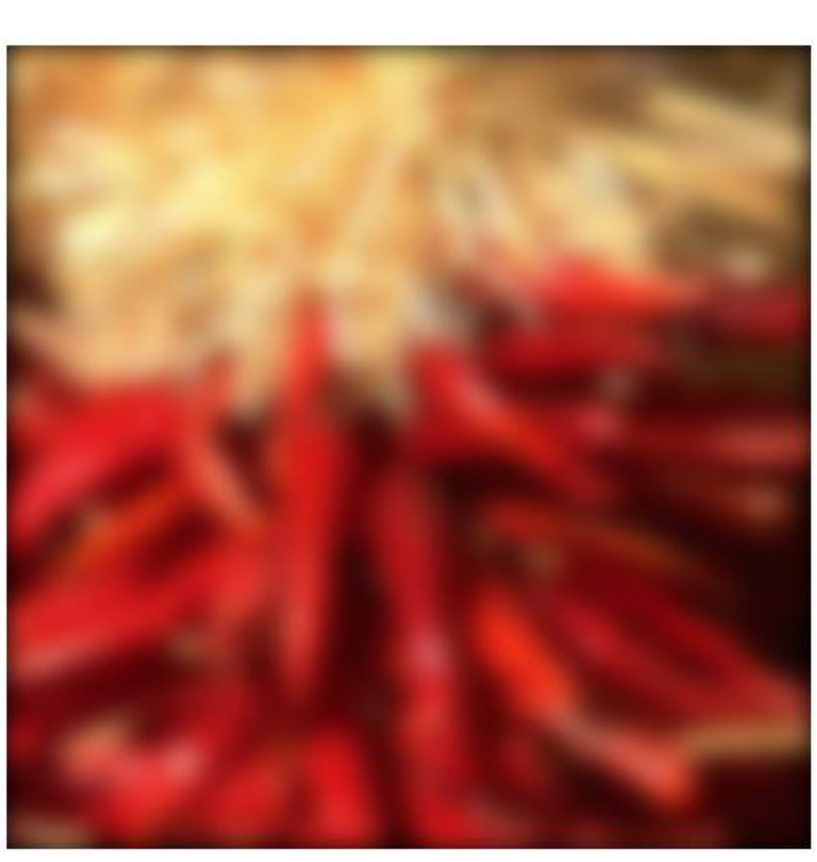
- What about near the edge?
  - the filter window falls off the edge of the image
  - need to extrapolate
  - methods:
    - **clip filter (black)**
      - borderType = BORDER_CONSTANT
    - wrap around
      - borderType = BORDER_WRAP
    - copy edge
      - borderType = BORDER_REPLICATE
    - reflect across edge
      - borderType = BORDER_REFLECT_101



Source: S. Marschner

Filtering: Boundary Issues

- What about near the edge?
  - the filter window falls off the edge of the image
  - need to extrapolate
  - methods:
    - **clip filter (black)**
      - borderType = BORDER_CONSTANT
    - wrap around
      - borderType = BORDER_WRAP
    - copy edge
      - borderType = BORDER_REPLICATE
    - reflect across edge
      - borderType = BORDER_REFLECT_101



Source: S. Marschner

Filtering: Boundary Issues

- What about near the edge?
  - the filter window falls off the edge of the image
  - need to extrapolate
  - methods:
    - clip filter (black)
      - borderType = BORDER_CONSTANT
    - **wrap around**
      - borderType = BORDER_WRAP
    - copy edge
      - borderType = BORDER_REPLICATE
    - reflect across edge
      - borderType = BORDER_REFLECT_101

Source: S. Marschner

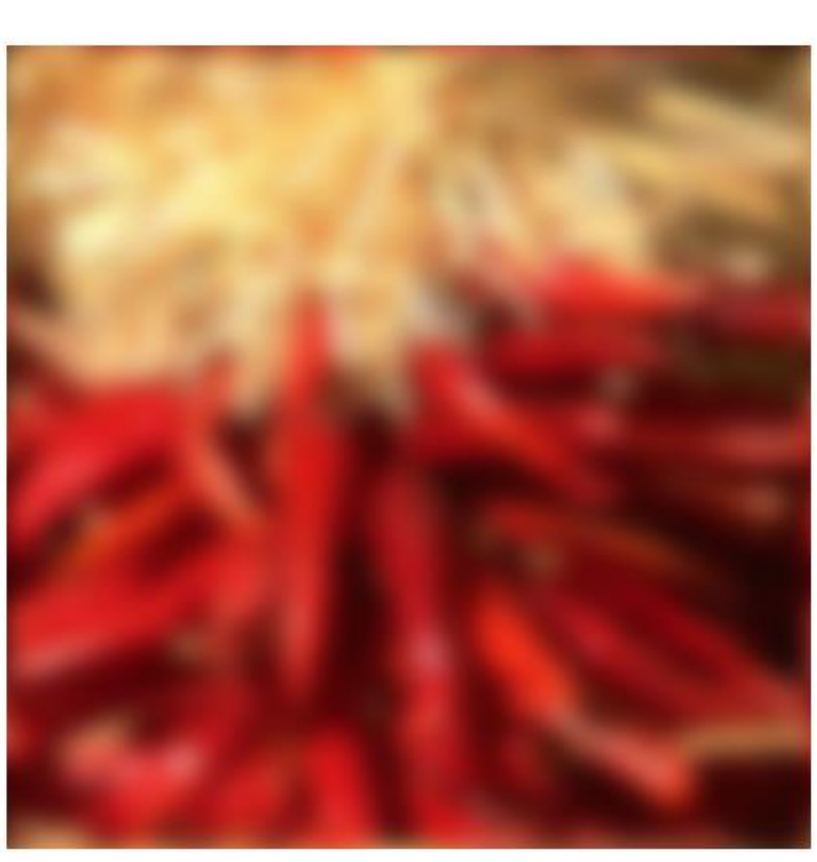
Filtering: Boundary Issues

- What about near the edge?
  - the filter window falls off the edge of the image
  - need to extrapolate
  - methods:
    - clip filter (black)
      - borderType = BORDER_CONSTANT
    - **wrap around**
      - borderType = BORDER_WRAP
    - copy edge
      - borderType = BORDER_REPLICATE
    - reflect across edge
      - borderType = BORDER_REFLECT_101

Source: S. Marschner

Filtering: Boundary Issues

- What about near the edge?
  - the filter window falls off the edge of the image
  - need to extrapolate
  - methods:
    - clip filter (black)
      - borderType = BORDER_CONSTANT
    - **wrap around**
      - borderType = BORDER_WRAP
    - copy edge
      - borderType = BORDER_REPLICATE
    - reflect across edge
      - borderType = BORDER_REFLECT_101



Source: S. Marschner

Filtering: Boundary Issues

- What about near the edge?
  - the filter window falls off the edge of the image
  - need to extrapolate
  - methods:
    - clip filter (black)
      - borderType = BORDER_CONSTANT
    - wrap around
      - borderType = BORDER_WRAP
    - **copy edge**
      - borderType = BORDER_REPLICATE
    - reflect across edge
      - borderType = BORDER_REFLECT_101

Source: S. Marschner

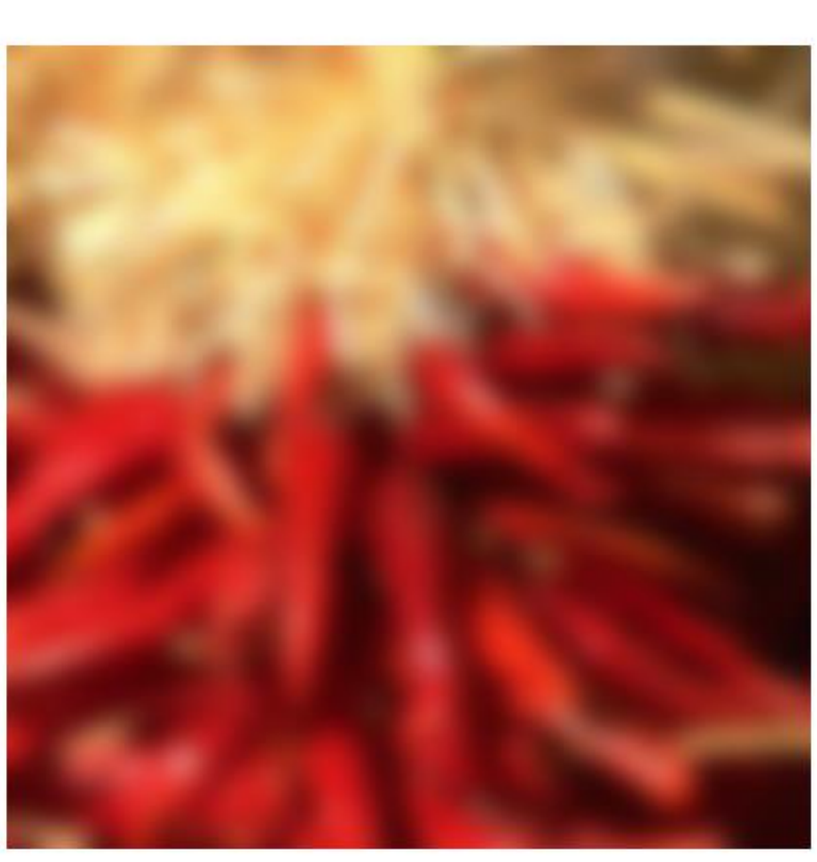
Filtering: Boundary Issues

- What about near the edge?
  - the filter window falls off the edge of the image
  - need to extrapolate
  - methods:
    - clip filter (black)
      - borderType = BORDER_CONSTANT
    - wrap around
      - borderType = BORDER_WRAP
    - **copy edge**
      - borderType = BORDER_REPLICATE
    - reflect across edge
      - borderType = BORDER_REFLECT_101

Source: S. Marschner

Filtering: Boundary Issues

- What about near the edge?
  - the filter window falls off the edge of the image
  - need to extrapolate
  - methods:
    - clip filter (black)
      - borderType = BORDER_CONSTANT
    - wrap around
      - borderType = BORDER_WRAP
    - **copy edge**
      - borderType = BORDER_REPLICATE
    - reflect across edge
      - borderType = BORDER_REFLECT_101



Source: S. Marschner

Filtering: Boundary Issues

- What about near the edge?
  - the filter window falls off the edge of the image
  - need to extrapolate
  - methods:
    - clip filter (black)
      - borderType = BORDER_CONSTANT
    - wrap around
      - borderType = BORDER_WRAP
    - copy edge
      - borderType = BORDER_REPLICATE
    - **reflect across edge**
      - borderType = BORDER_REFLECT_101

Source: S. Marschner

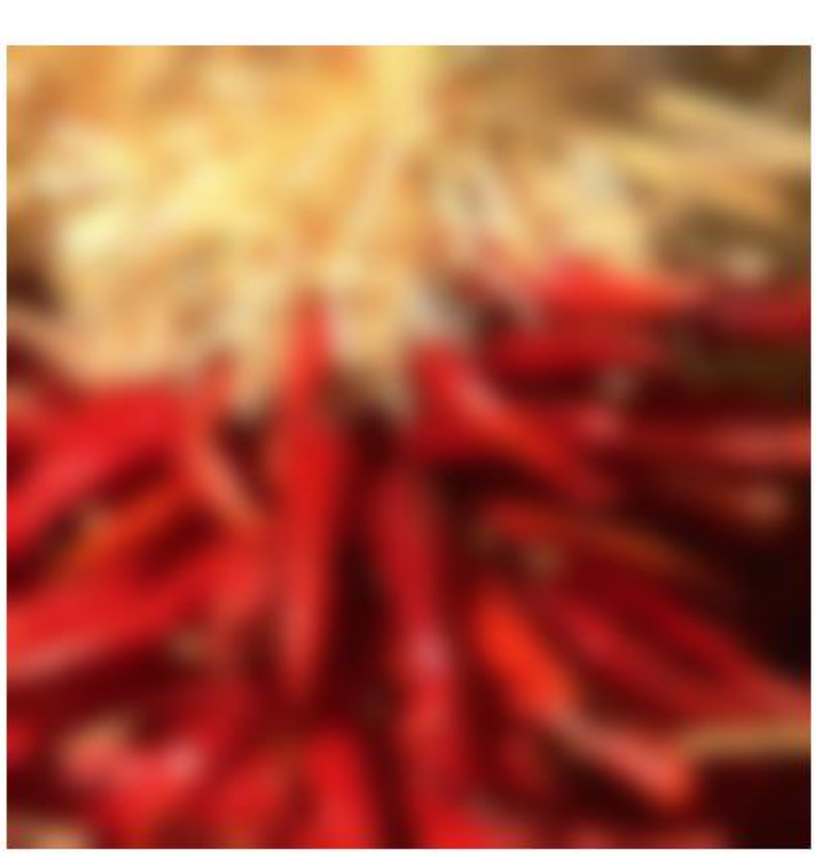
Filtering: Boundary Issues

- What about near the edge?
  - the filter window falls off the edge of the image
  - need to extrapolate
  - methods:
    - clip filter (black)
      - borderType = BORDER_CONSTANT
    - wrap around
      - borderType = BORDER_WRAP
    - copy edge
      - borderType = BORDER_REPLICATE
    - **reflect across edge**
      - borderType = BORDER_REFLECT_101

Source: S. Marschner

Filtering: Boundary Issues

- What about near the edge?
  - the filter window falls off the edge of the image
  - need to extrapolate
  - methods:
    - clip filter (black)
      - borderType = BORDER_CONSTANT
    - wrap around
      - borderType = BORDER_WRAP
    - copy edge
      - borderType = BORDER_REPLICATE
    - **reflect across edge**
      - borderType = BORDER_REFLECT_101



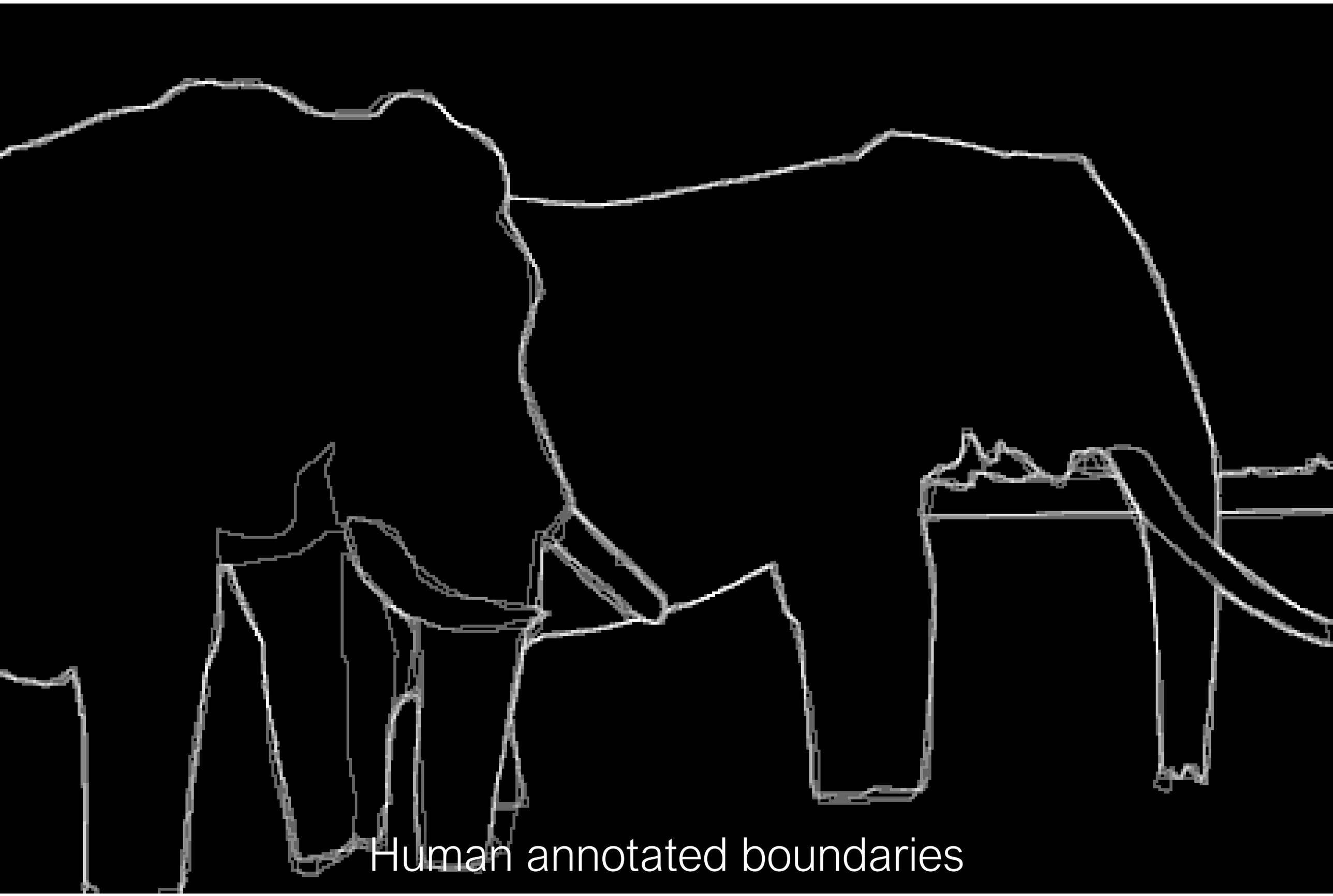
Source: S. Marschner

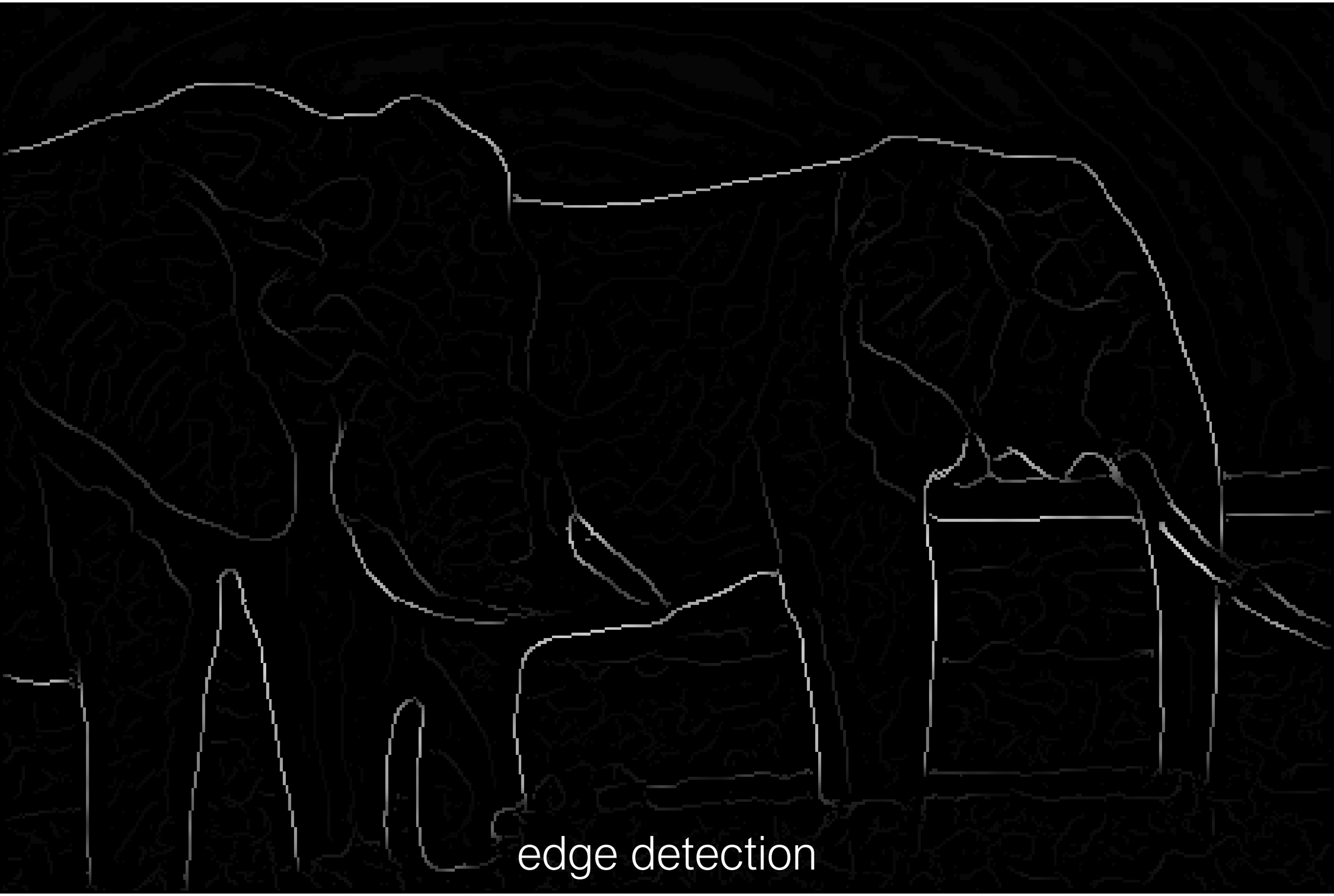
Summary

- Linear Filtering
  - Smoothing filters
    - Box filters
    - Gaussian filters
  - Difference filters
    - Sharpening filter
  - Relationship with convolution
  - Separability
- Non-linear filtering
  - Median filter
  - Weighted median filter
  - Bilateral filtering
- Practical Issues

Finding boundaries

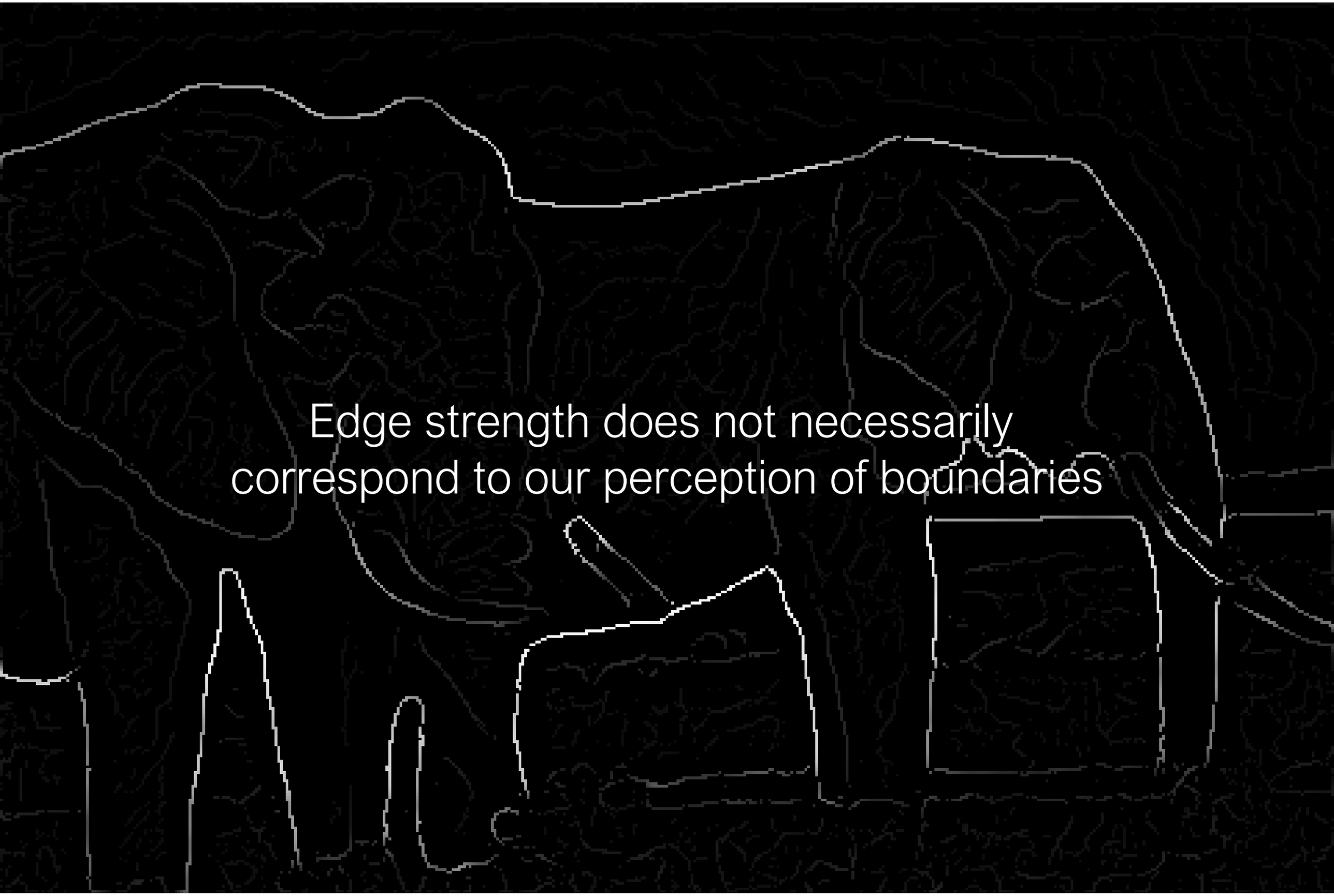
Where are the object boundaries?


Human annotated boundaries

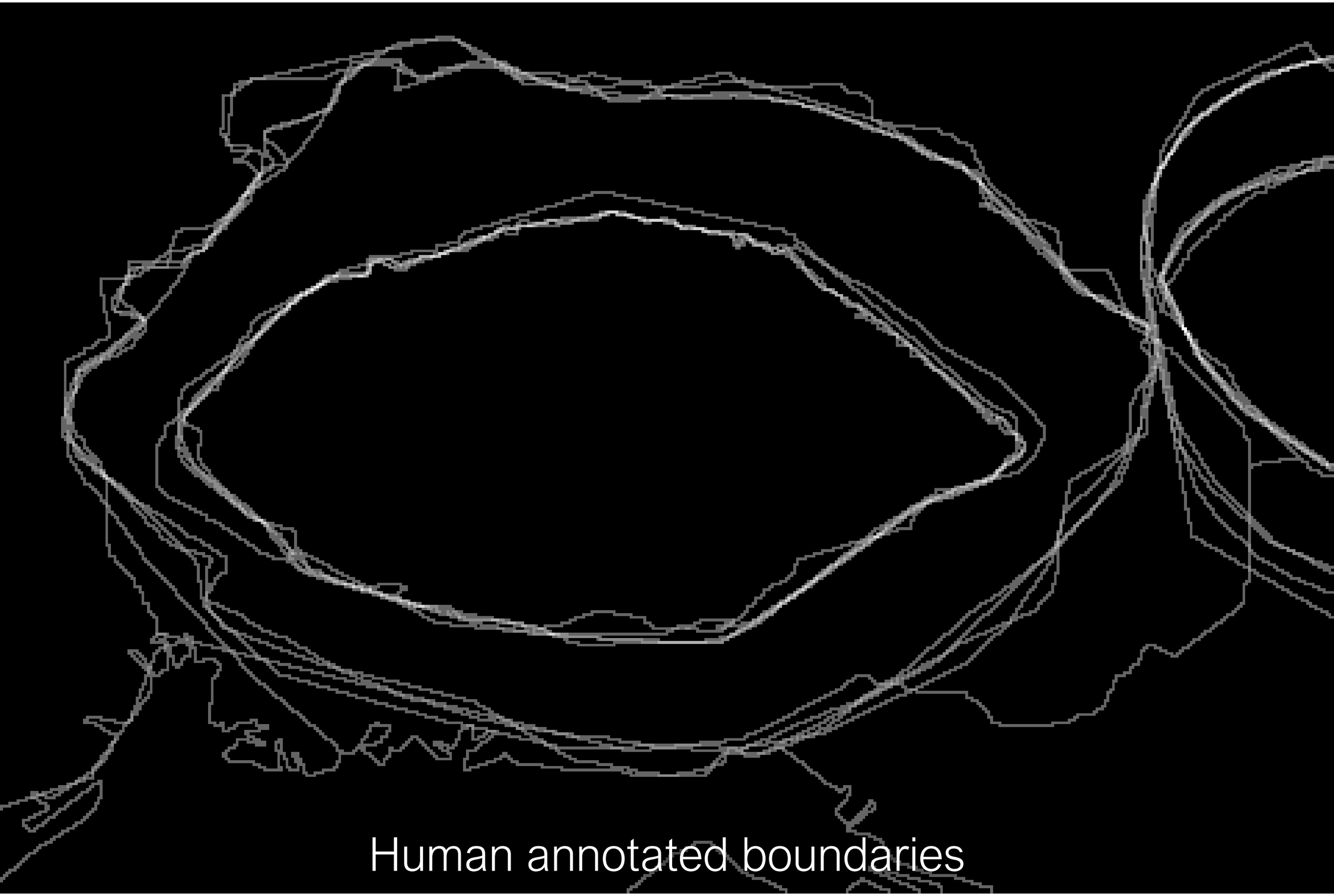


edge detection

Multi-scale edge detection


Edge strength does not necessarily
correspond to our perception of boundaries

Where are the object boundaries?



Human annotated boundaries

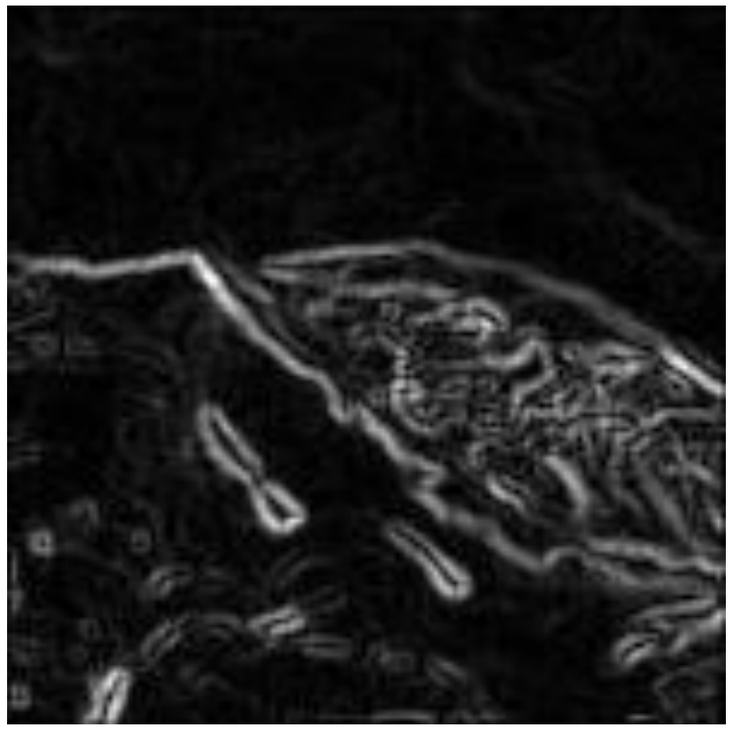

edge detection


Defining boundaries are hard for us too

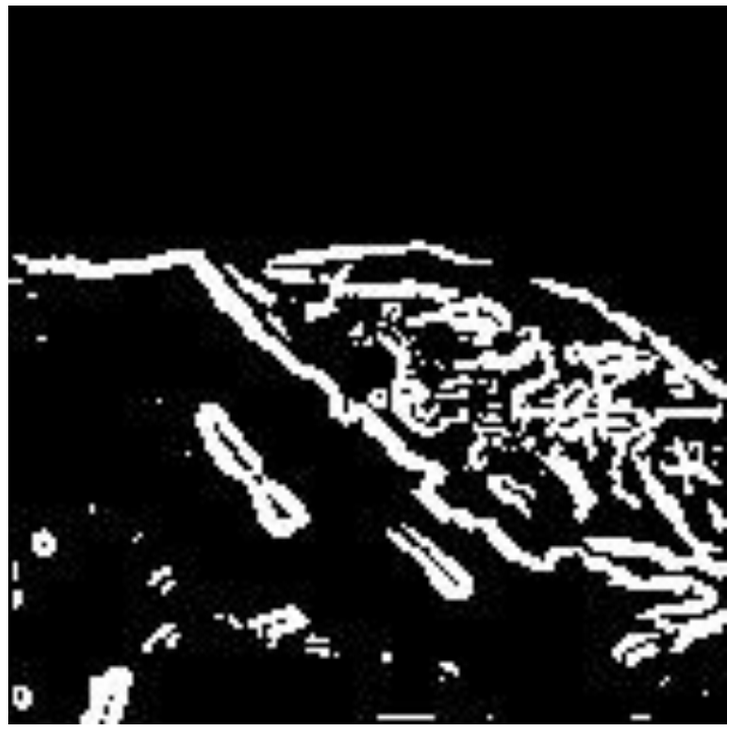
Where is the boundary of the mountain top?

Lines are hard to find

Original image



Edge detection

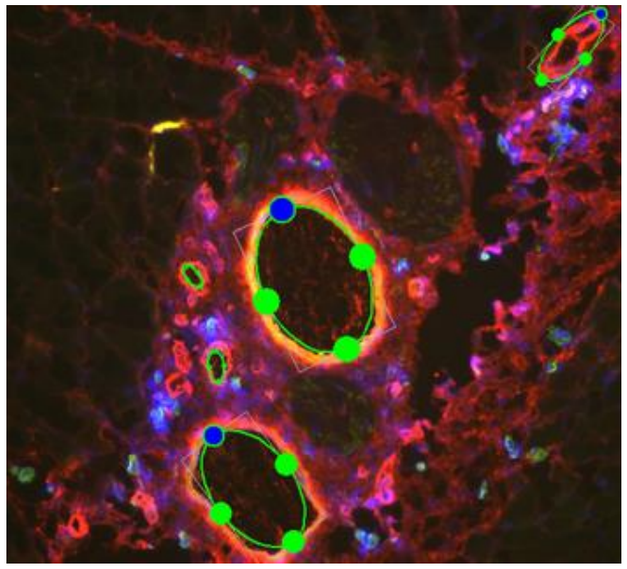


Thresholding

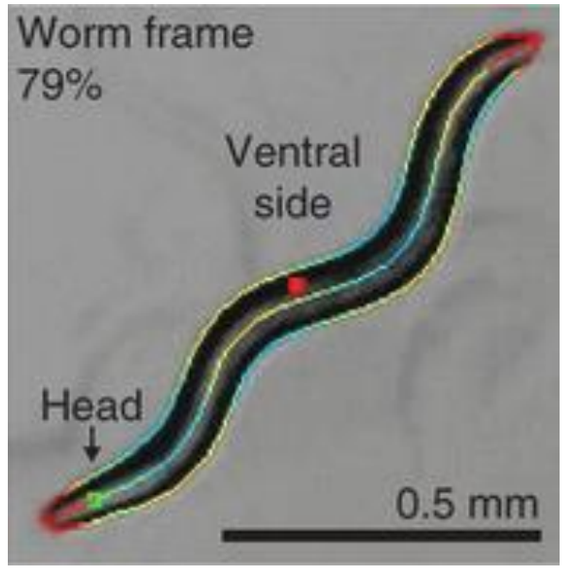
Noisy edge image
Incomplete boundaries

Applications

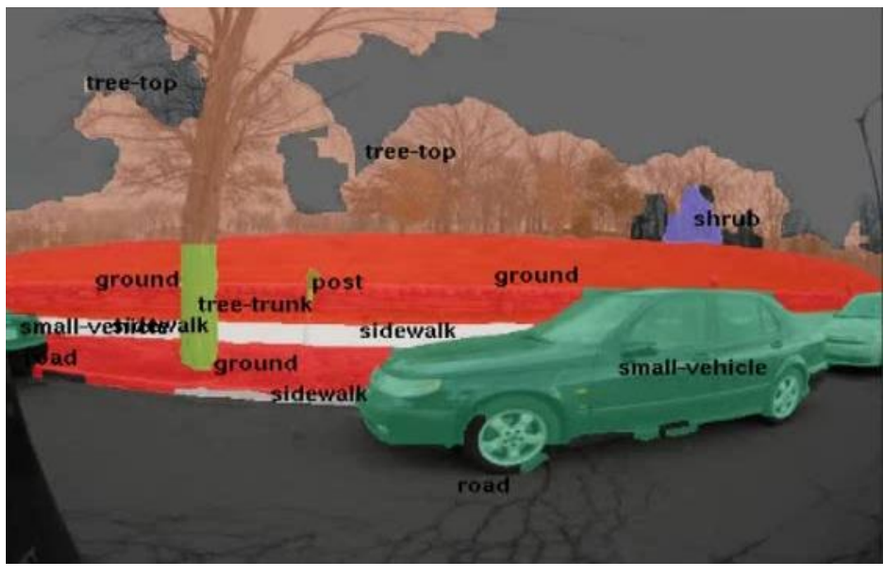
Autonomous Vehicles
(lane line detection)



tissue engineering
(blood vessel counting)

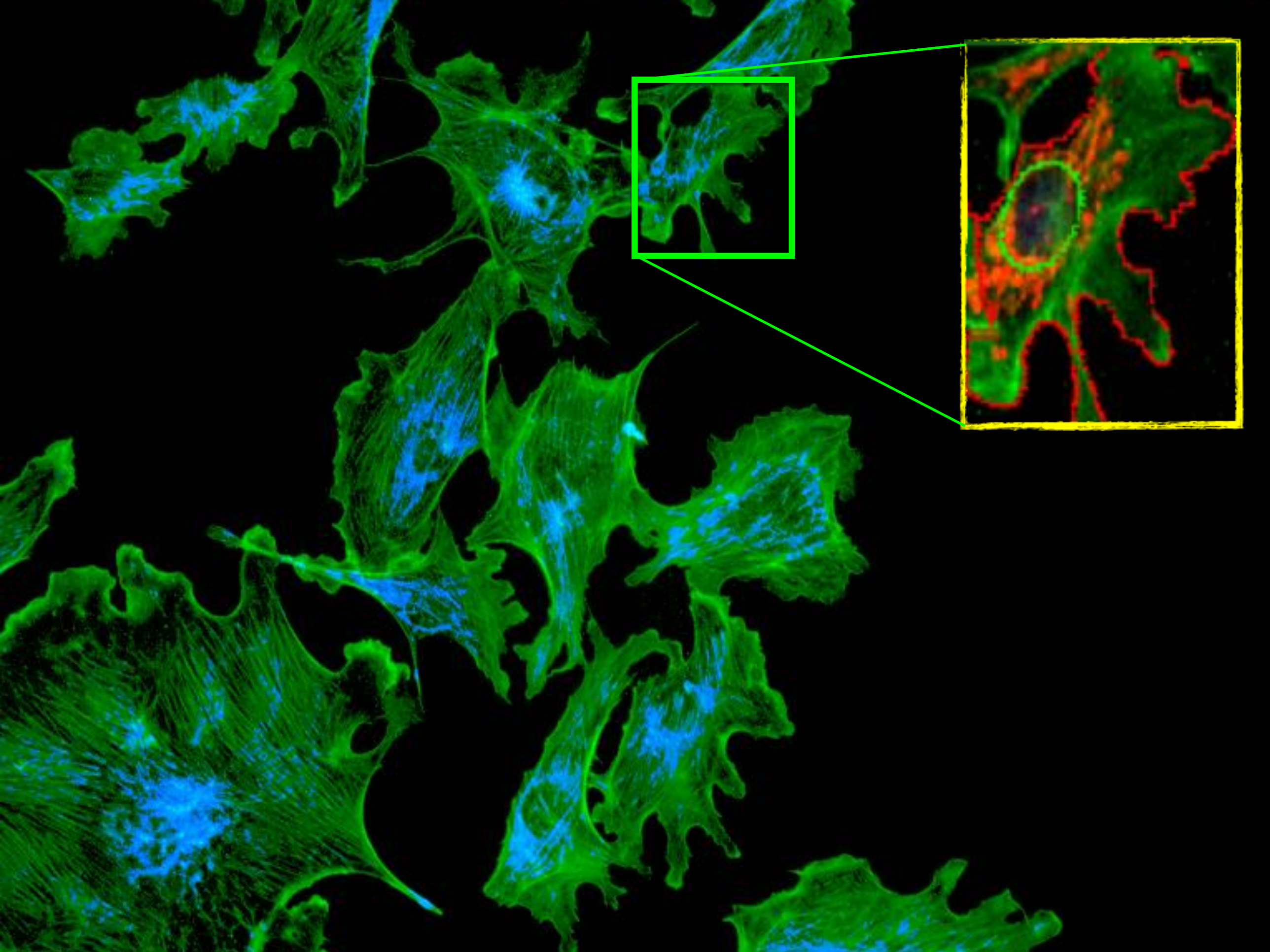


behavioral genetics
(earthworm contours)



Autonomous Vehicles
(semantic scene segmentation)

Computational Photography
(image inpainting)



Line fitting

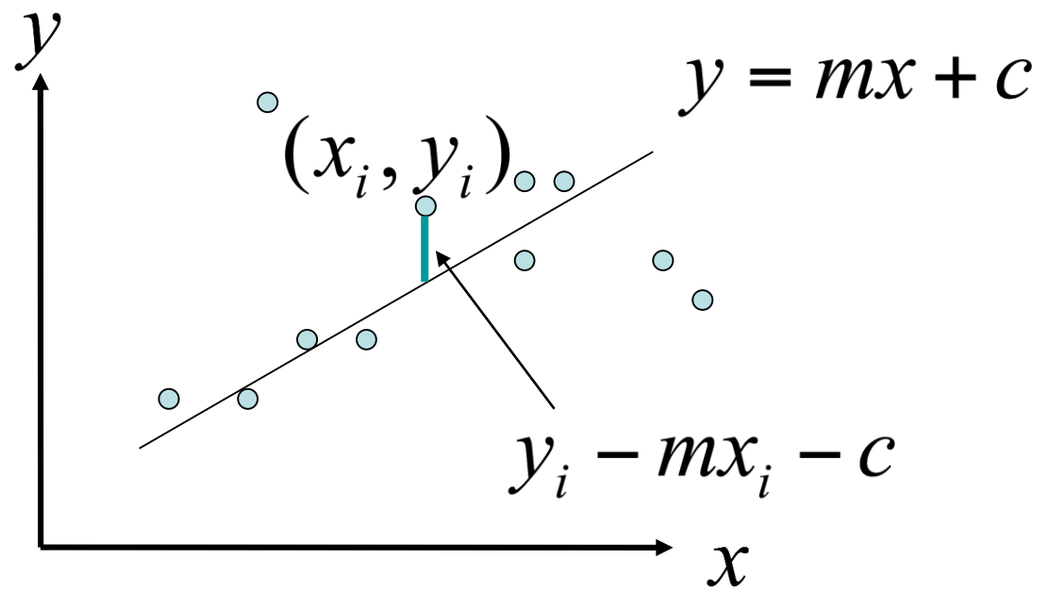
Line fitting

Given: Many (x_i, y_i) pairs

Find: Parameters (m, c)

Minimize: Average square distance:

$$E = \sum_i \frac{(y_i - mx_i - c)^2}{N}$$



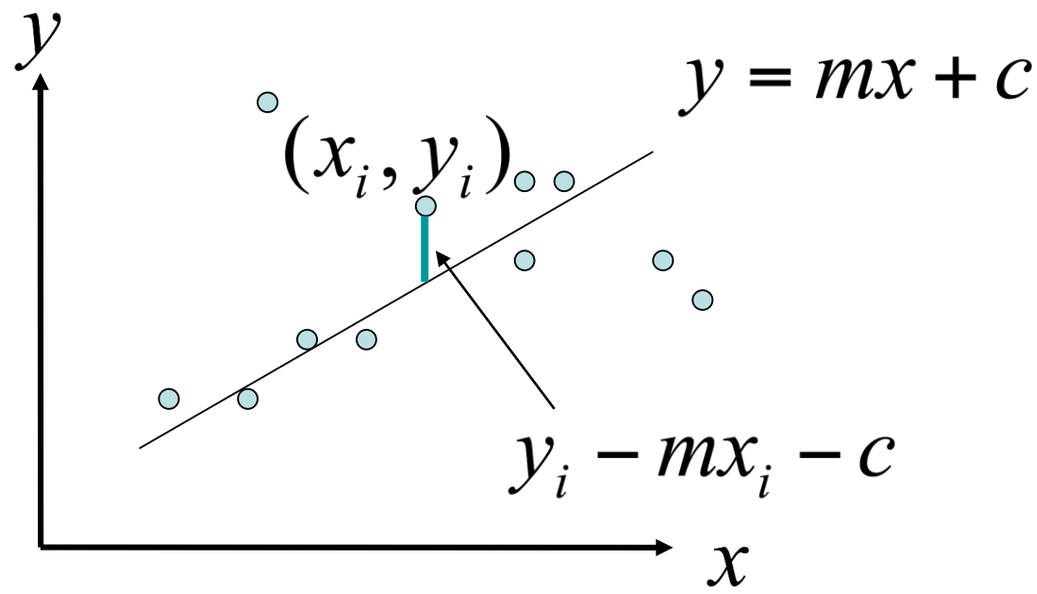
Line fitting

Given: Many (x_i, y_i) pairs

Find: Parameters (m, c)

Minimize: Average square distance:

$$E = \sum_i \frac{(y_i - mx_i - c)^2}{N}$$



Using:

$$\frac{\partial E}{\partial m} = 0 \quad \& \quad \frac{\partial E}{\partial c} = 0$$

Note:

$$\bar{y} = \frac{\sum_i y_i}{N} \qquad \bar{x} = \frac{\sum_i x_i}{N}$$

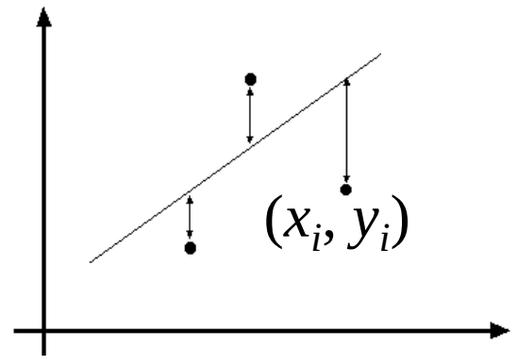
$$c = \bar{y} - m\,\bar{x}$$

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

What are some problems with the approach?

Data: $(x_1, y_1), \ldots, (x_n, y_n)$

Line equation: $y_i = m\,x_i + b$



Find (m, b) to minimize $E = \sum_{i=1}^{n} (y_i - mx_i - b)^2$

$$Y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \qquad X = \begin{bmatrix} x_1 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{bmatrix} \qquad B = \begin{bmatrix} m \\ b \end{bmatrix}$$

$$E = \|Y - XB\|^2 = (Y - XB)^T (Y - XB) = Y^T Y - 2(XB)^T Y + (XB)^T (XB)$$

$$\frac{dE}{dB} = 2X^T XB - 2X^T Y = 0$$

$$X^T XB = X^T Y$$

Normal equations: least squares solution to $XB=Y$

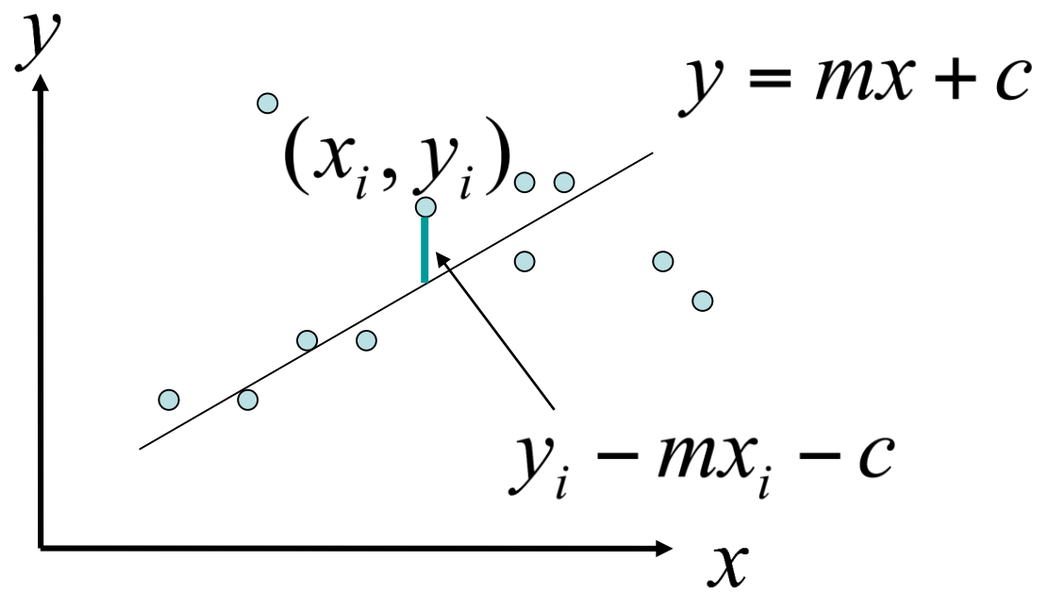
Line fitting

Given: Many (x_i, y_i) pairs

Find: Parameters (m, c)

Minimize: Average square distance:

$$E = \sum_i \frac{(y_i - mx_i - c)^2}{N}$$

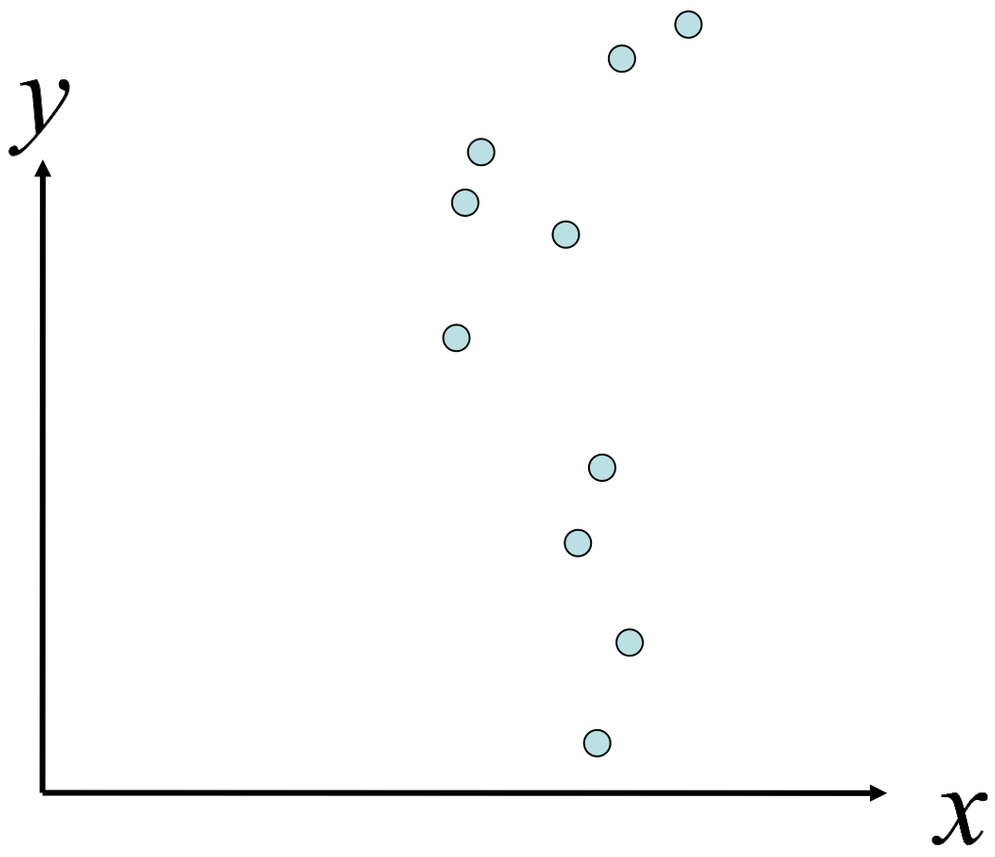


How can we solve this minimization?

Problems with parameterizations

Where is the line that minimizes E?

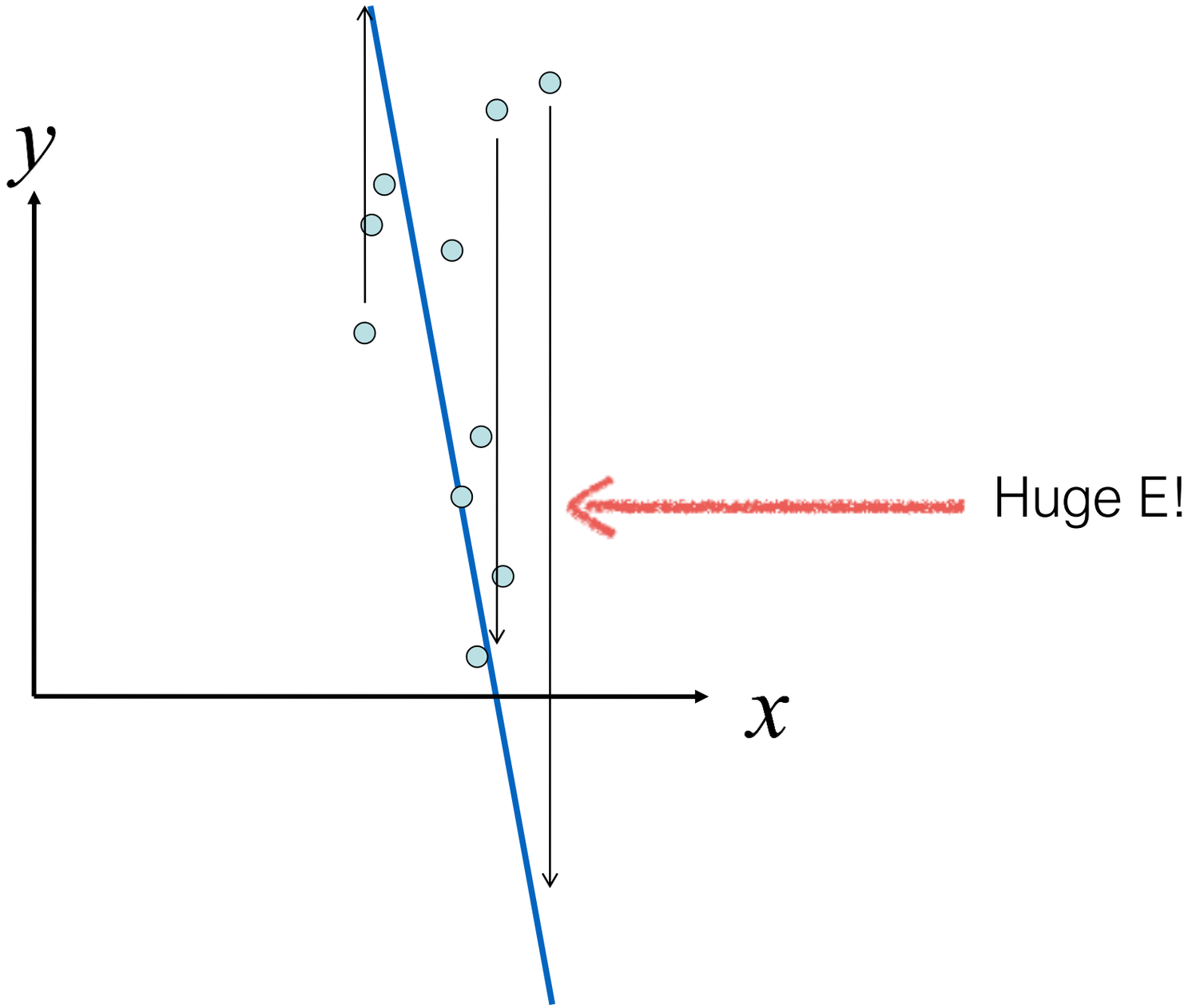
$$E = \sum_{i=1}^{n} (y_i - mx_i - b)^2$$



Problems with parameterizations

Where is the line that minimizes E?

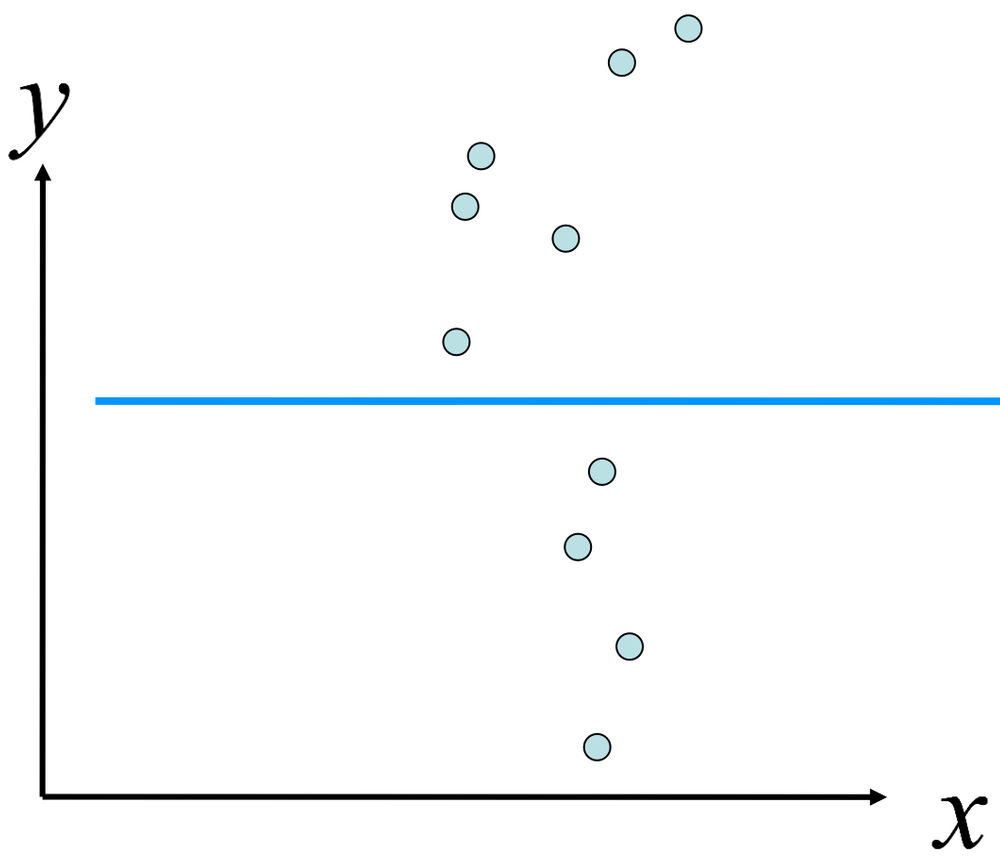
$$E = \sum_{i=1}^{n} (y_i - mx_i - b)^2$$



Problems with parameterizations

Where is the line that minimizes E?

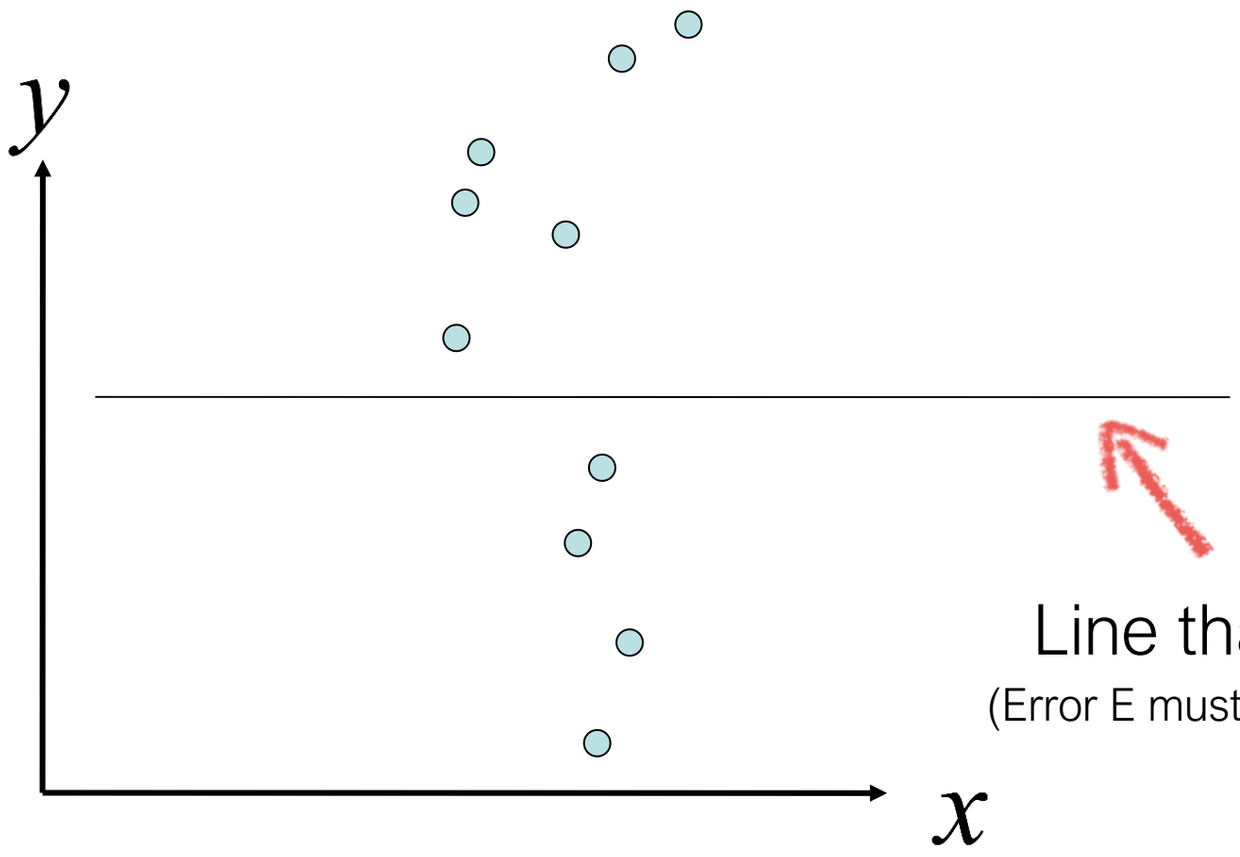
$$E = \sum_{i=1}^{n} (y_i - mx_i - b)^2$$



Line that minimizes E!!

Problems with parameterizations

Where is the line that minimizes E?

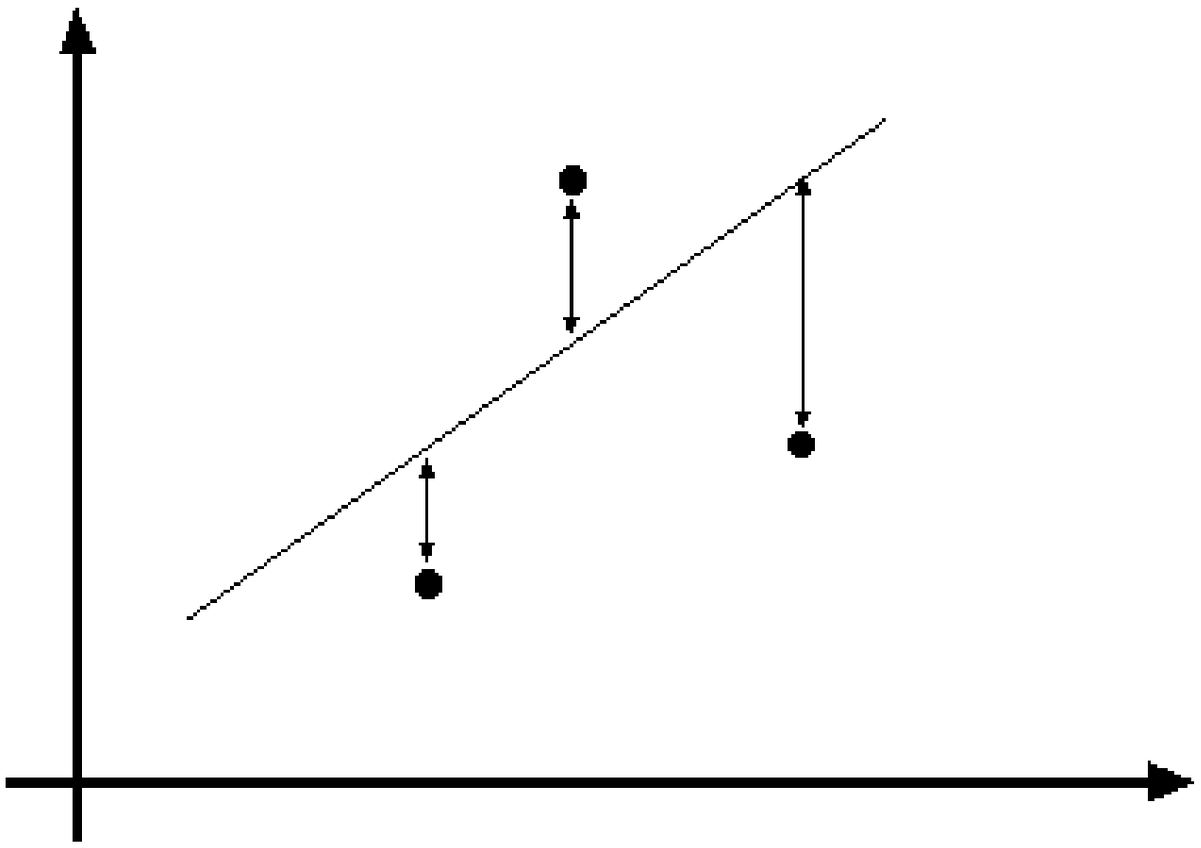


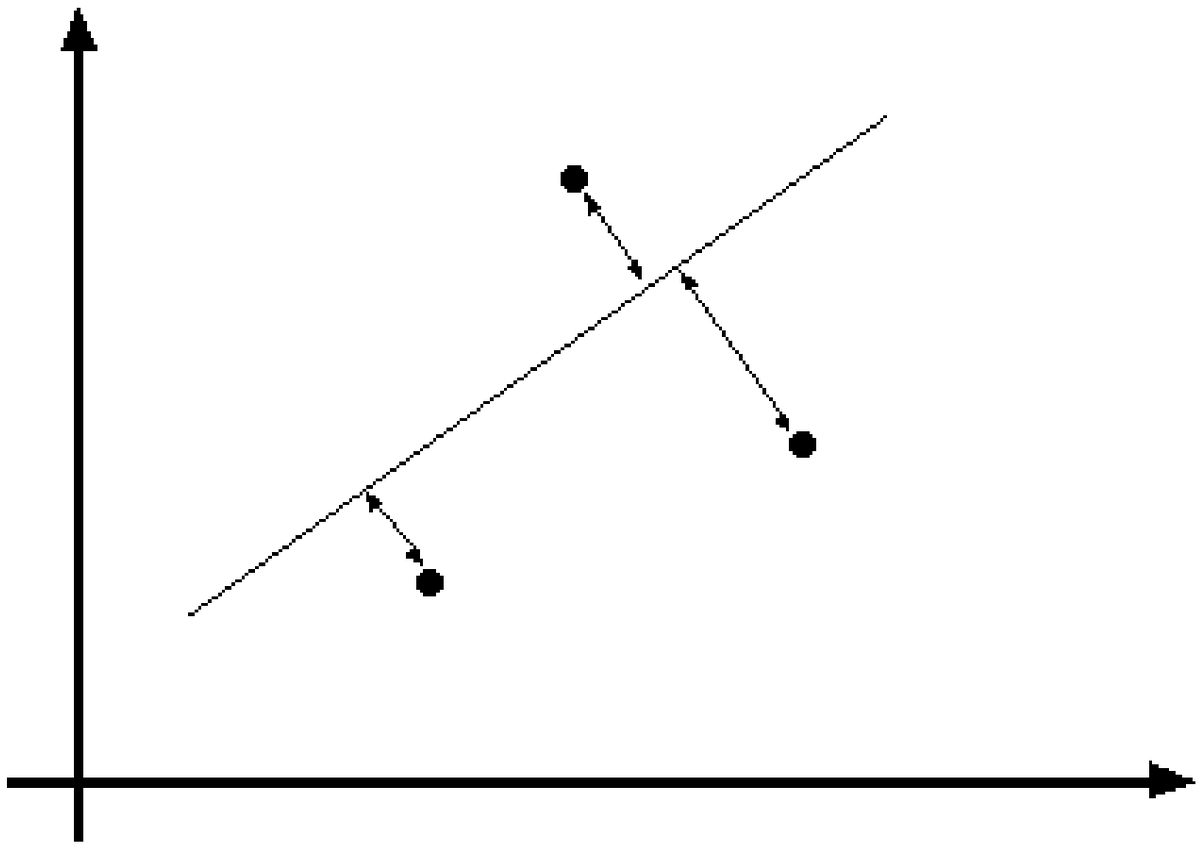
Line that minimizes E!!

(Error E must be formulated carefully!)

How can we deal with this?

Line fitting is easily setup as a maximum likelihood problem
… but choice of model is important

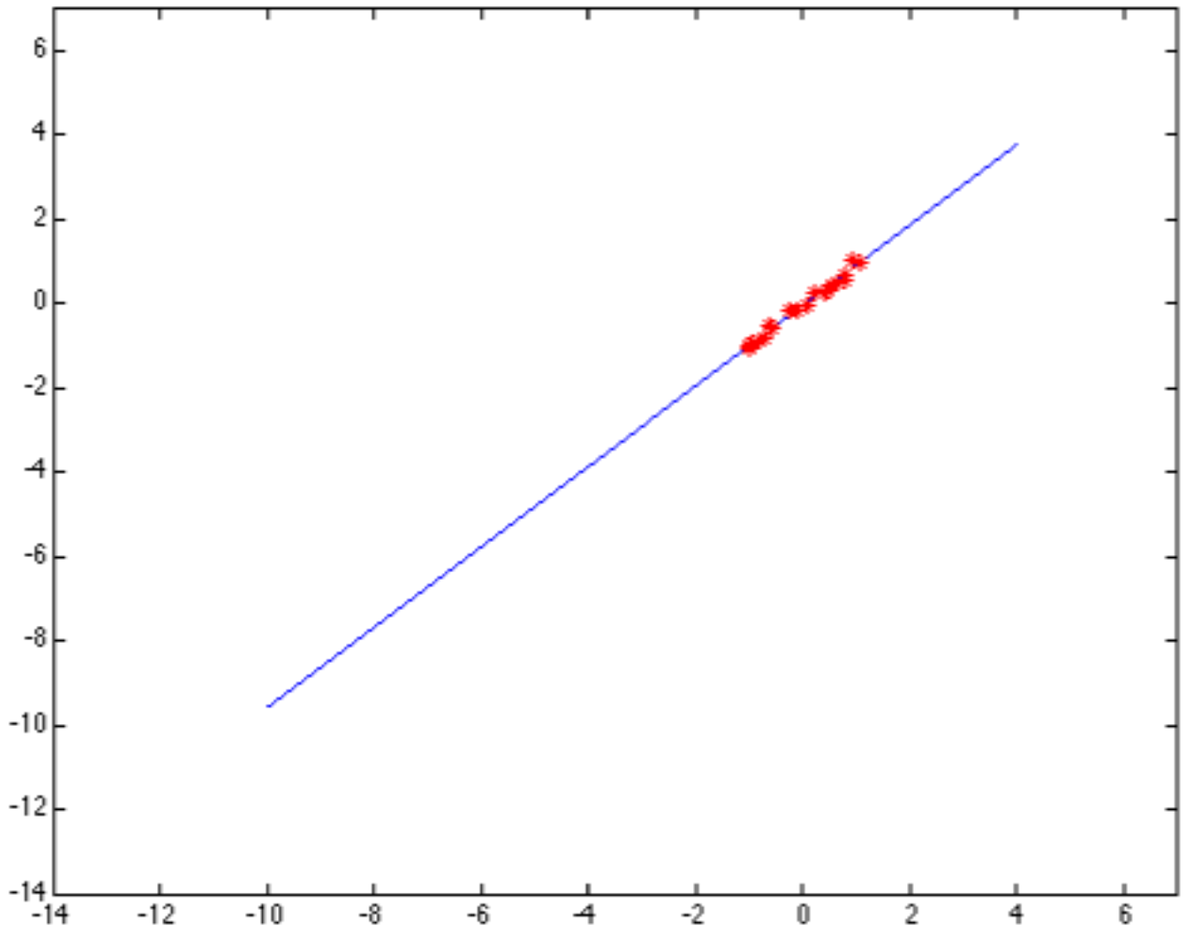




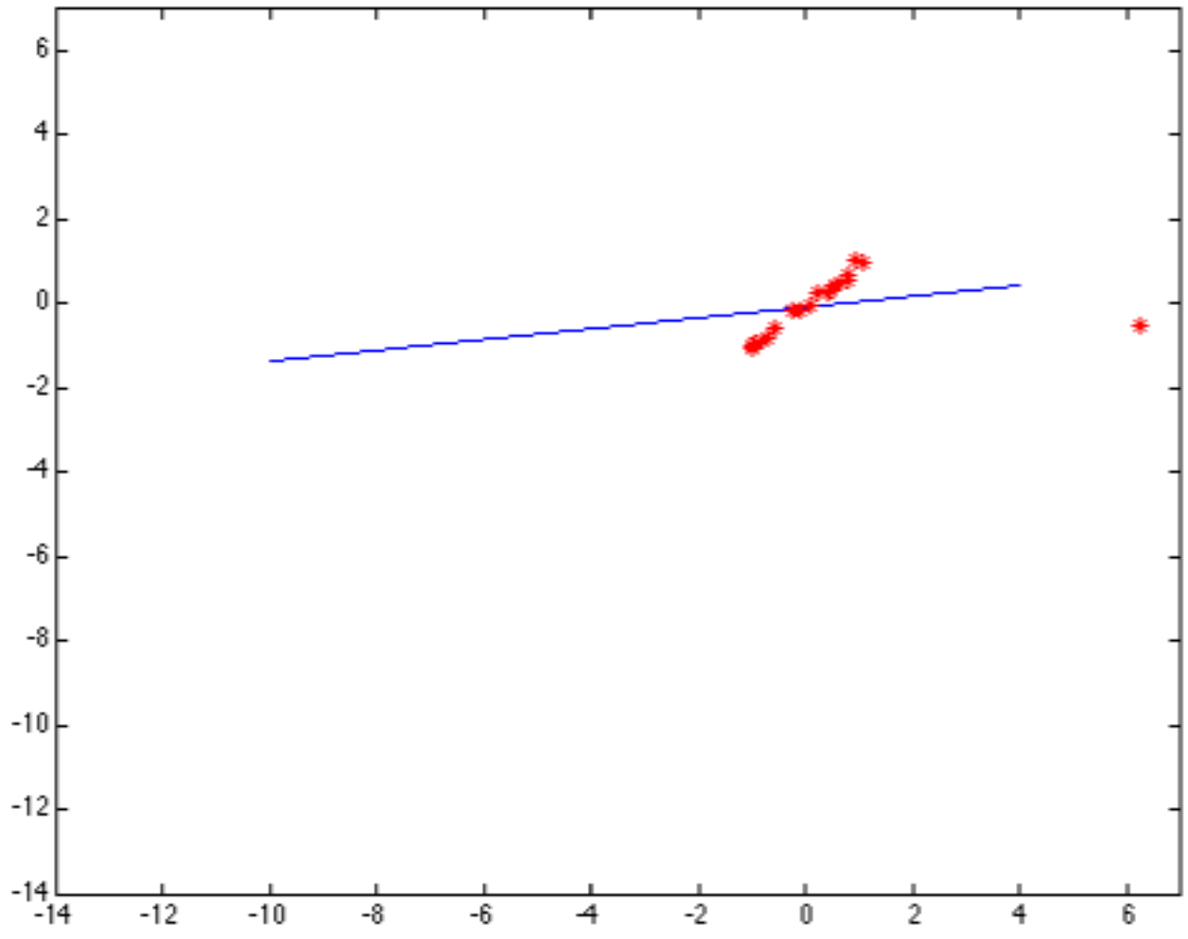
$$E = \sum_{i=1}^{n} (y_i - mx_i - b)^2$$

What optimization are we solving here?

Problems with noise

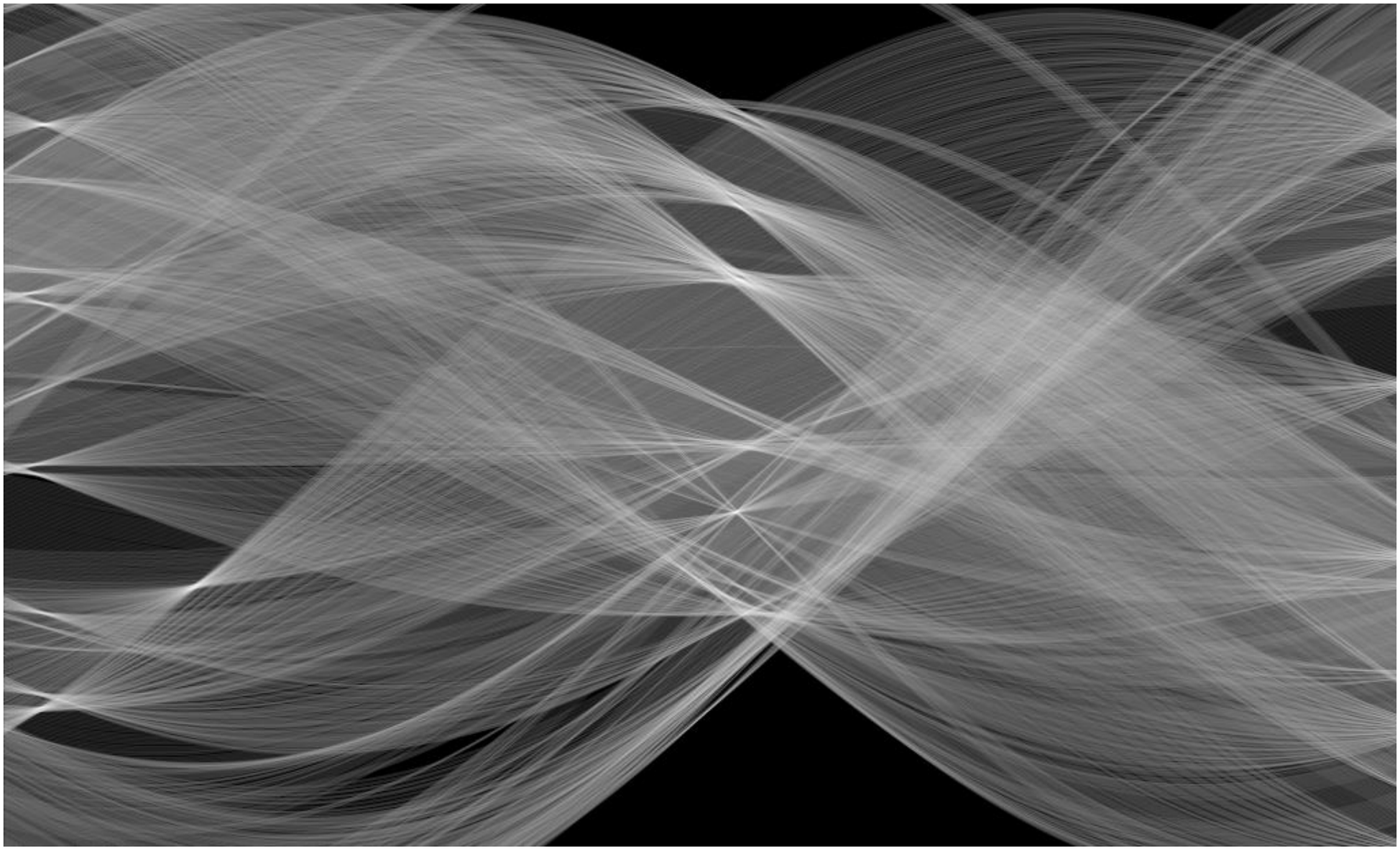


Least-squares error fit



Squared error heavily penalizes outliers

Hough transform



Model fitting is difficult because…

- **Extraneous data:** clutter or multiple models
  - We do not know what is part of the model?
  - Can we pull out models with a few parts from much larger amounts of background clutter?
- **Missing data:** only some parts of model are present
- **Noise**

- Cost:
  - It is not feasible to check all combinations of features by fitting a model to each possible subset

So what can we do?

Line parameterizations

Slope intercept form

$$y = mx + b$$

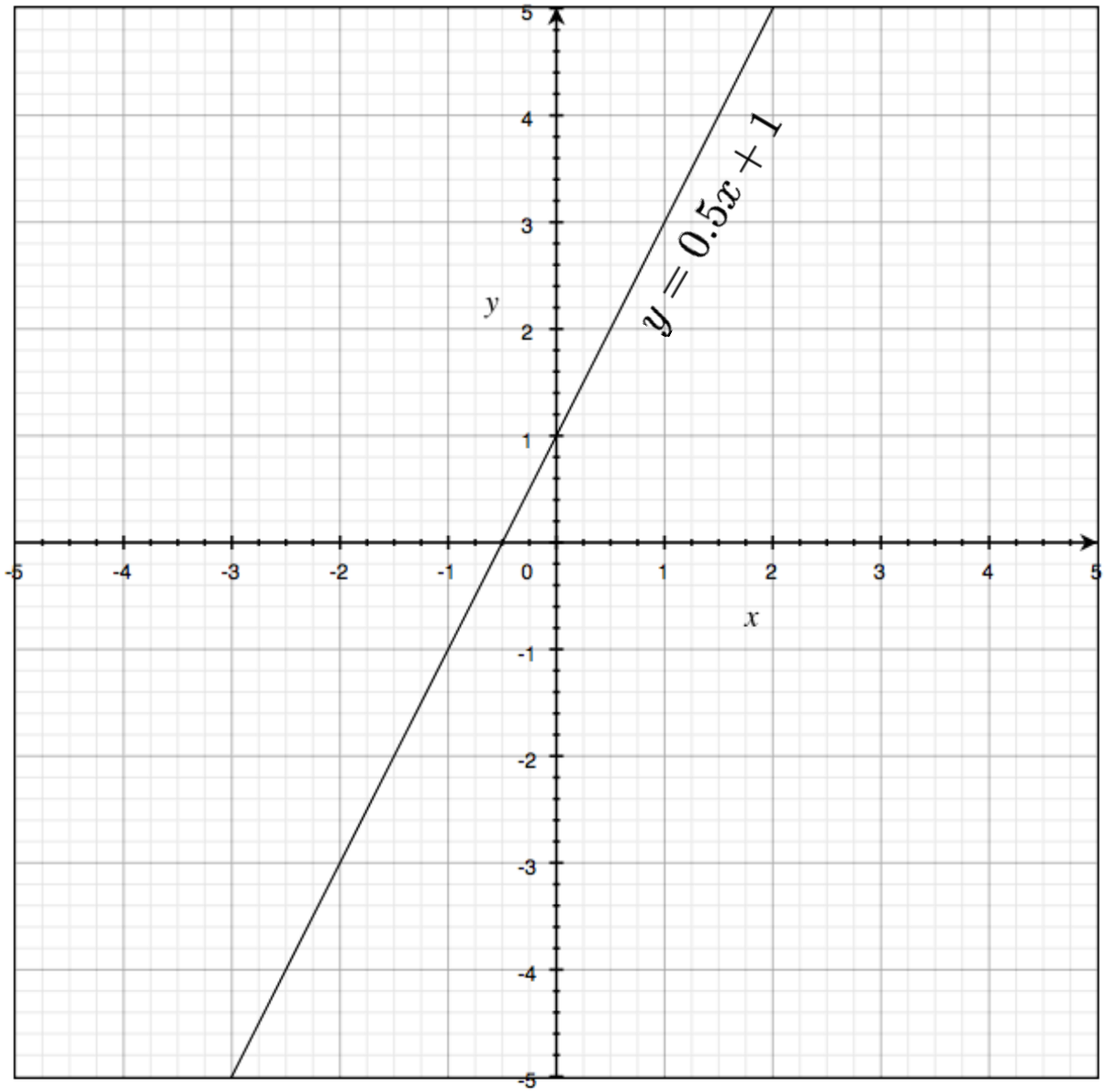
slope y-intercept

What are m and b?

Slope intercept form

$$y = mx + b$$

slope y-intercept



Double intercept form

$$\frac{x}{a} + \frac{y}{b} = 1$$

x-intercept

y-intercept

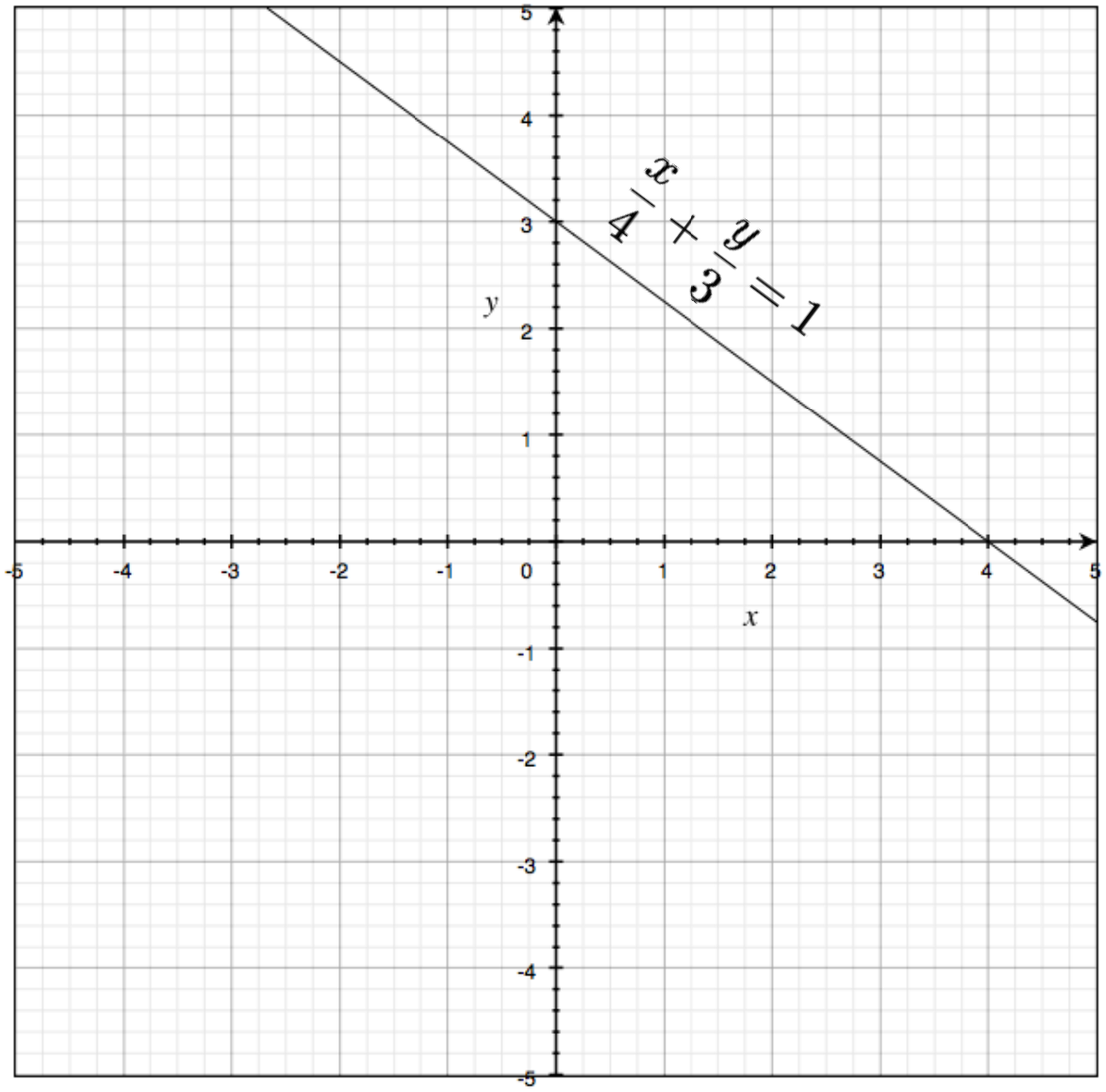
What are x and y?

Double intercept form

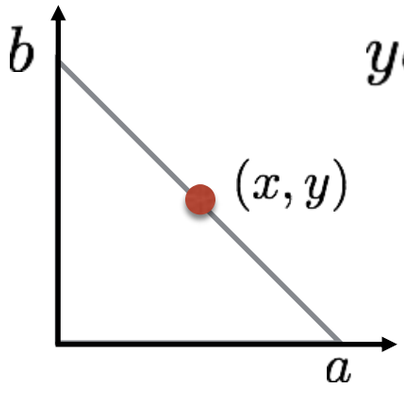
$$\frac{x}{a} + \frac{y}{b} = 1$$

x-intercept

y-intercept



Derivation:



(Similar slope) $\frac{y-b}{x-0} = \frac{0-y}{a-x}$

$$ya + yx - ba + bx = -yx$$

$$ya + bx = ba$$

$$\frac{y}{b} + \frac{x}{a} = 1$$

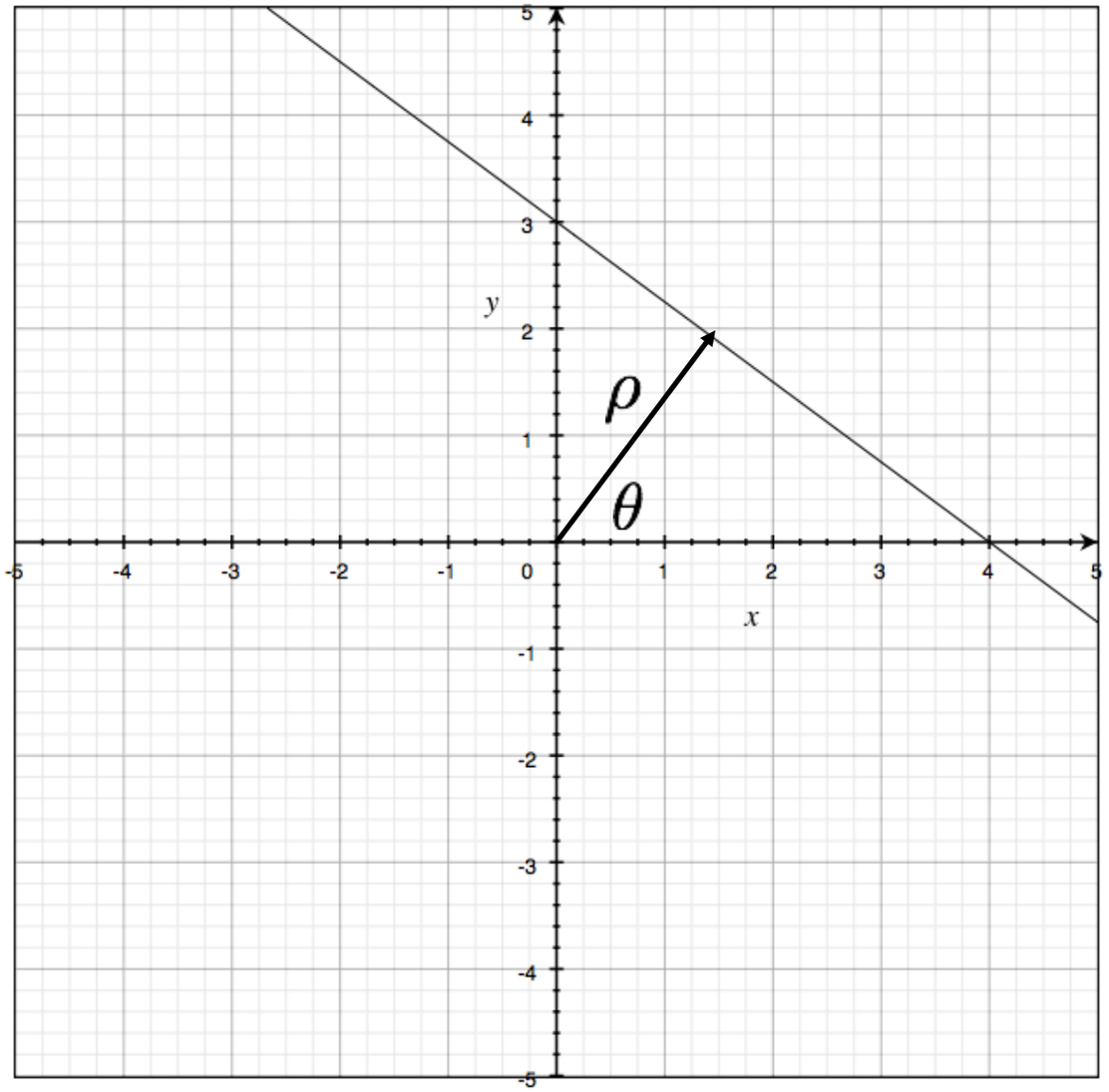
Normal Form

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

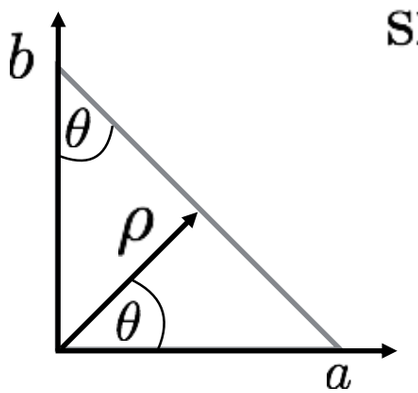
What are rho and theta?

Normal Form

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



Derivation:



$$\cos\theta = \frac{\rho}{a} \rightarrow a = \frac{\rho}{\cos\theta}$$

$$\sin\theta = \frac{\rho}{b} \rightarrow b = \frac{\rho}{\sin\theta}$$

plug into: $\frac{x}{a} + \frac{y}{b} = 1$

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

Hough transform

- 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

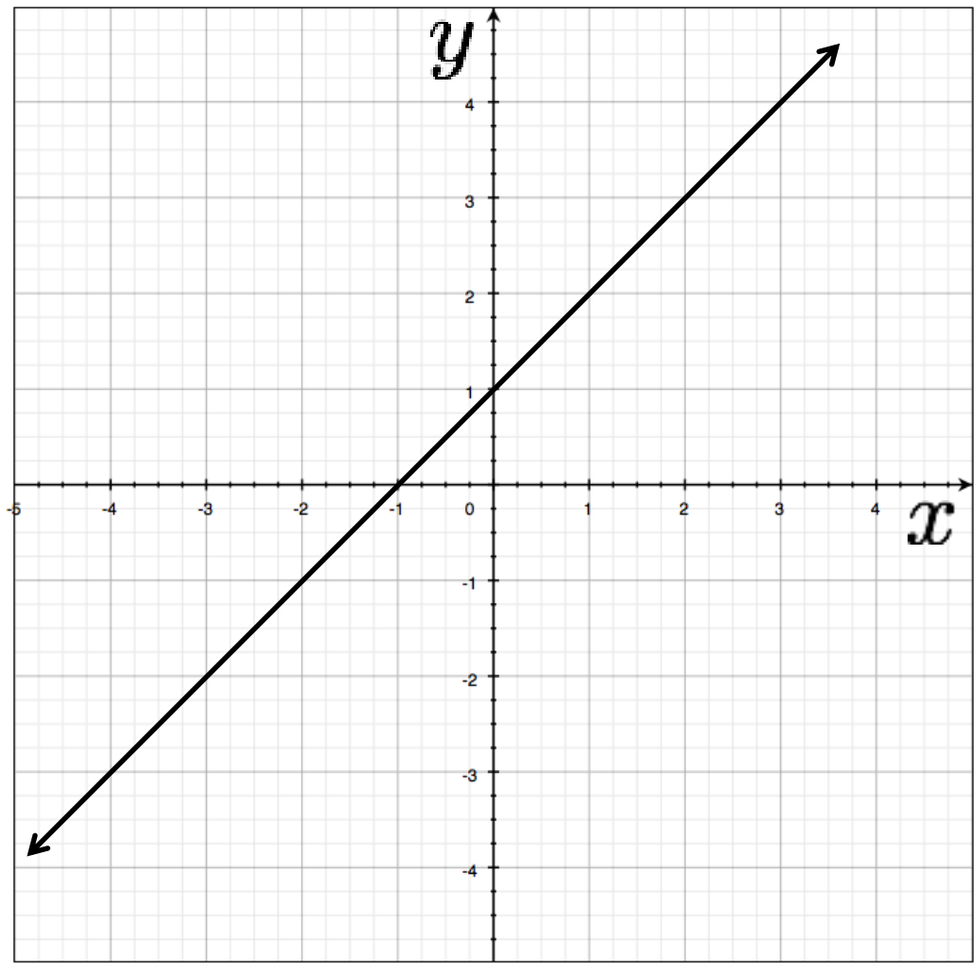


Image space

Image and parameter space

variables

$$y = mx + b$$

parameters

variables

$$y - mx = b$$

parameters

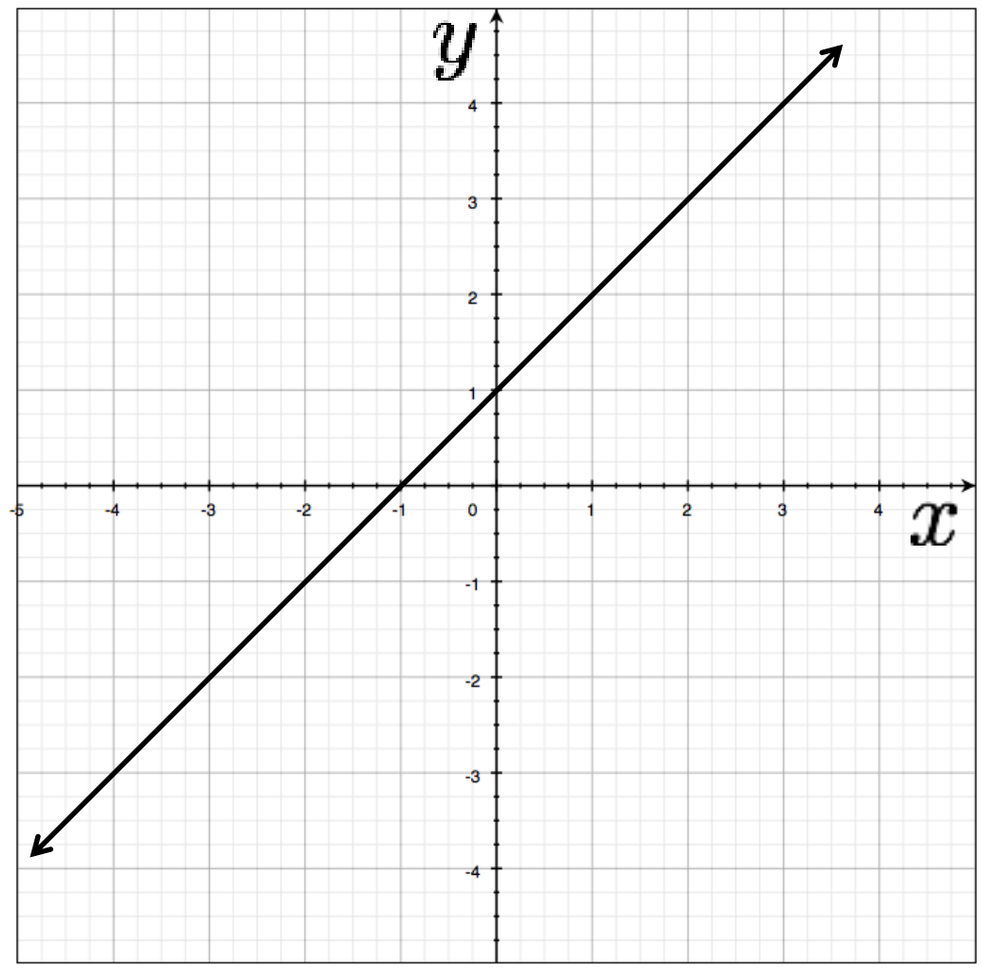
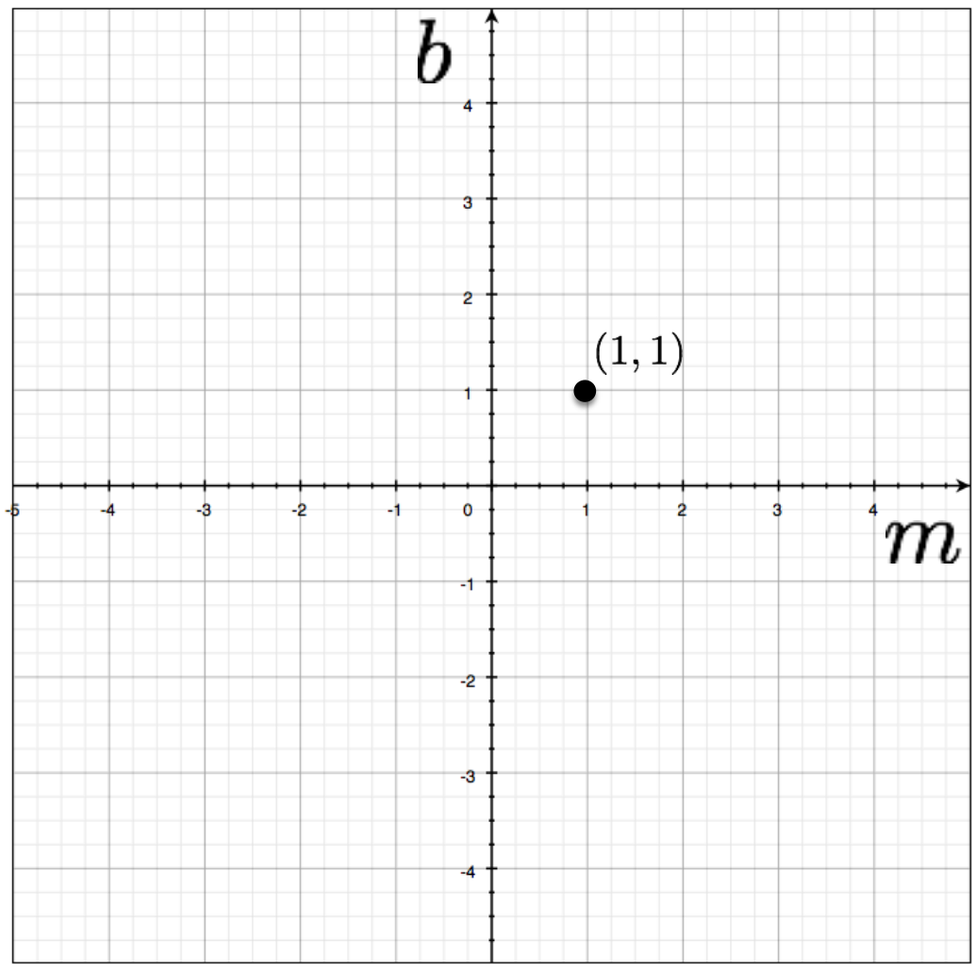


Image space

a line becomes a point



Parameter space

Image and parameter space

variables

$$y = mx + b$$

parameters

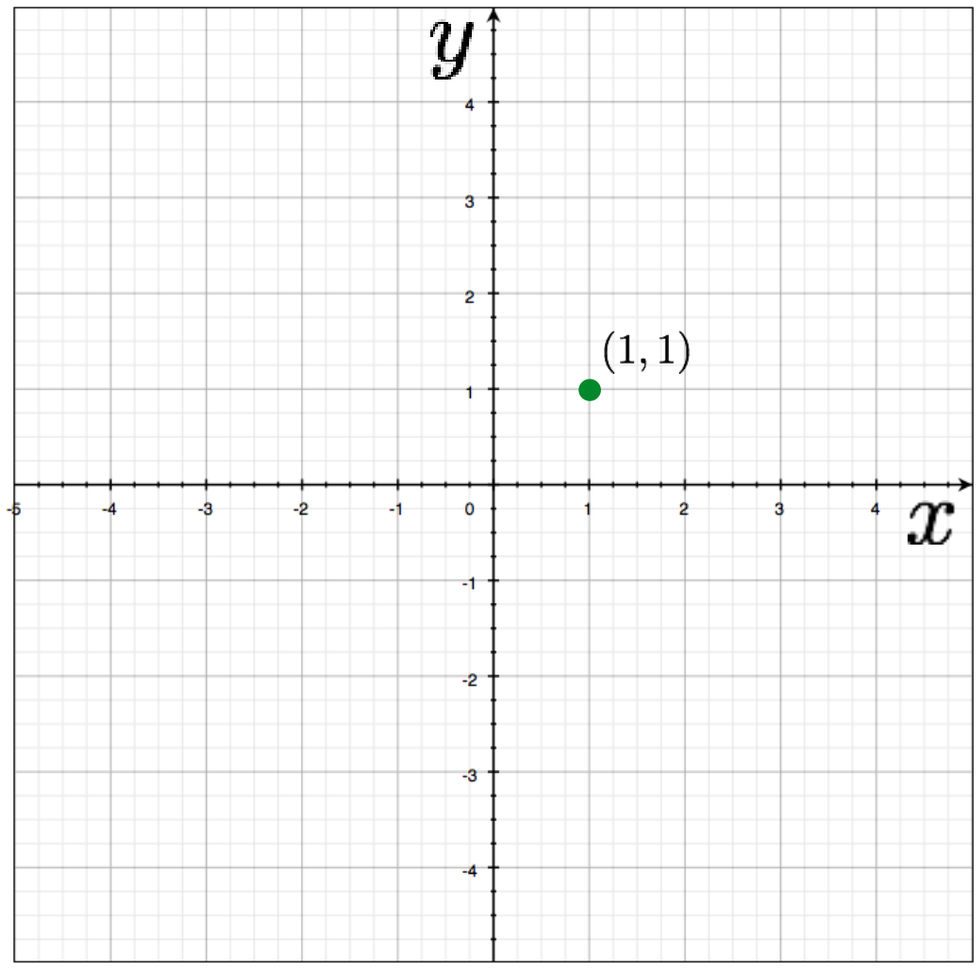


Image space

What would a point in image space become in parameter space?

Image and parameter space

variables

$$y = mx + b$$

parameters

variables

$$y - mx = b$$

parameters

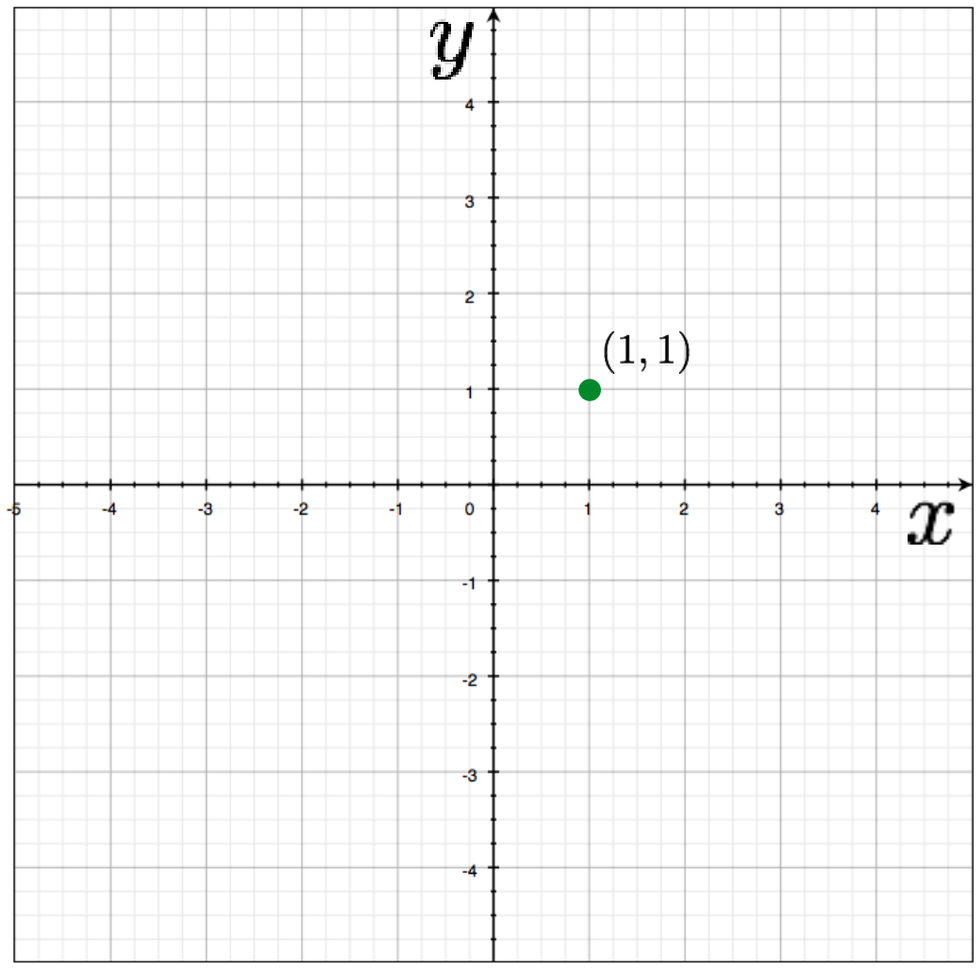
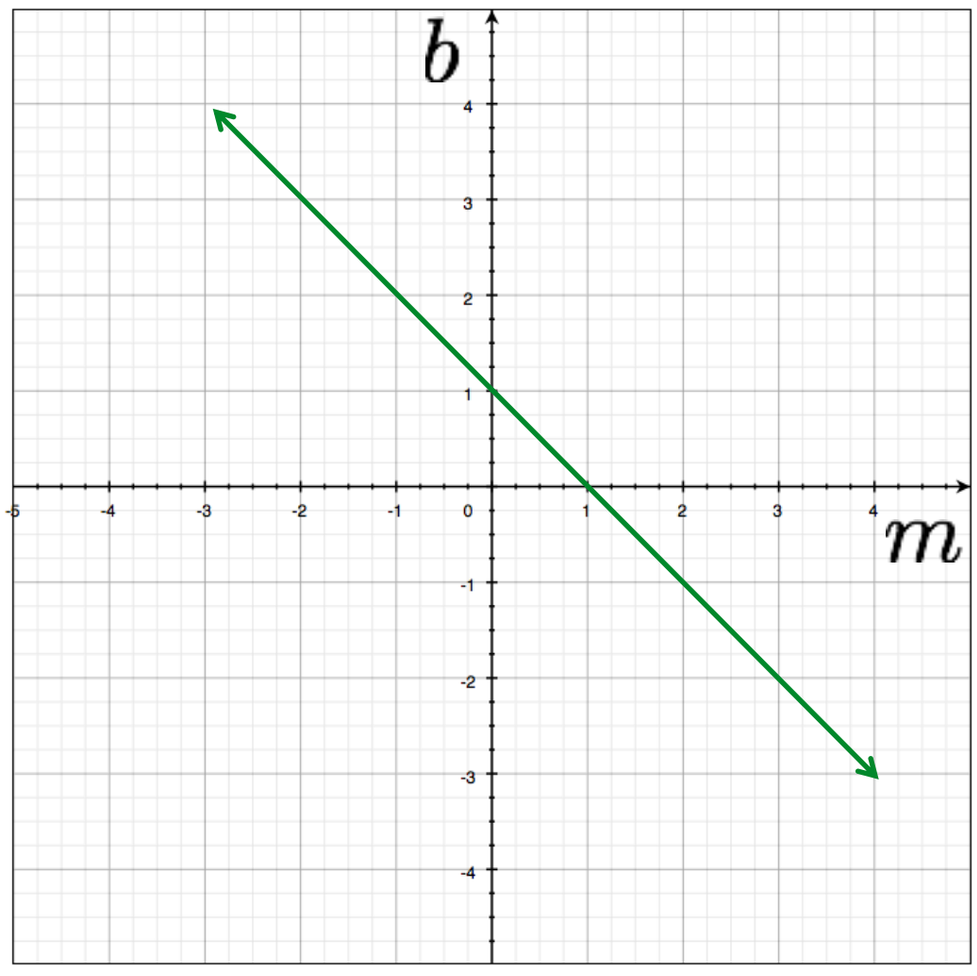


Image space

a point becomes a line



Parameter space

Image and parameter space

variables

$$y = mx + b$$

parameters

variables

$$y - mx = b$$

parameters

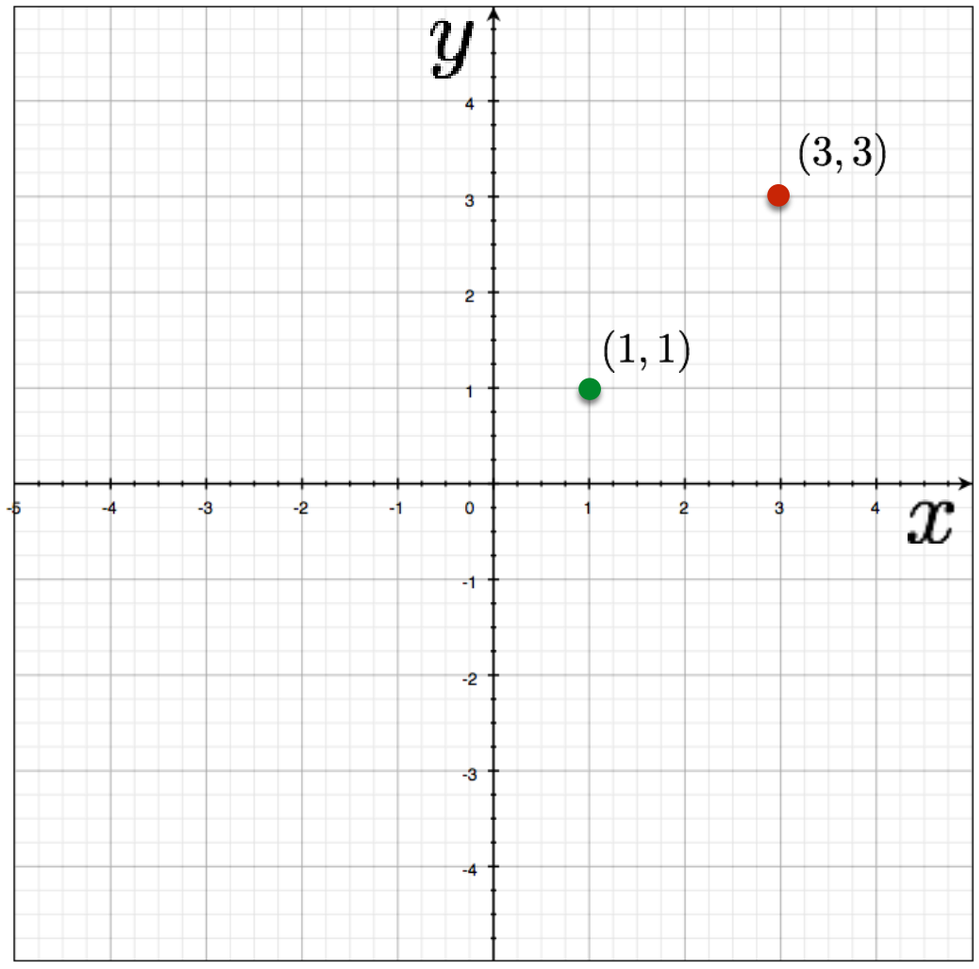
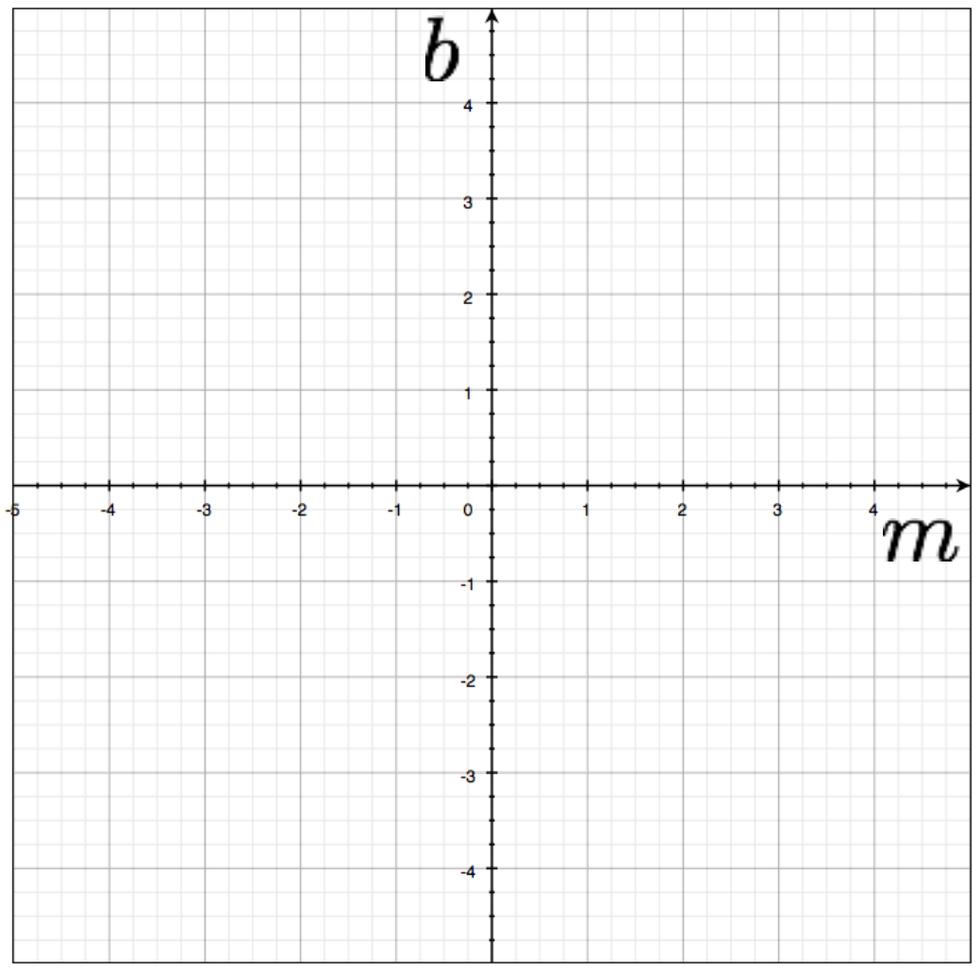


Image space

two points become ?



Parameter space

Image and parameter space

variables

$$y = mx + b$$

parameters

variables

$$y - mx = b$$

parameters

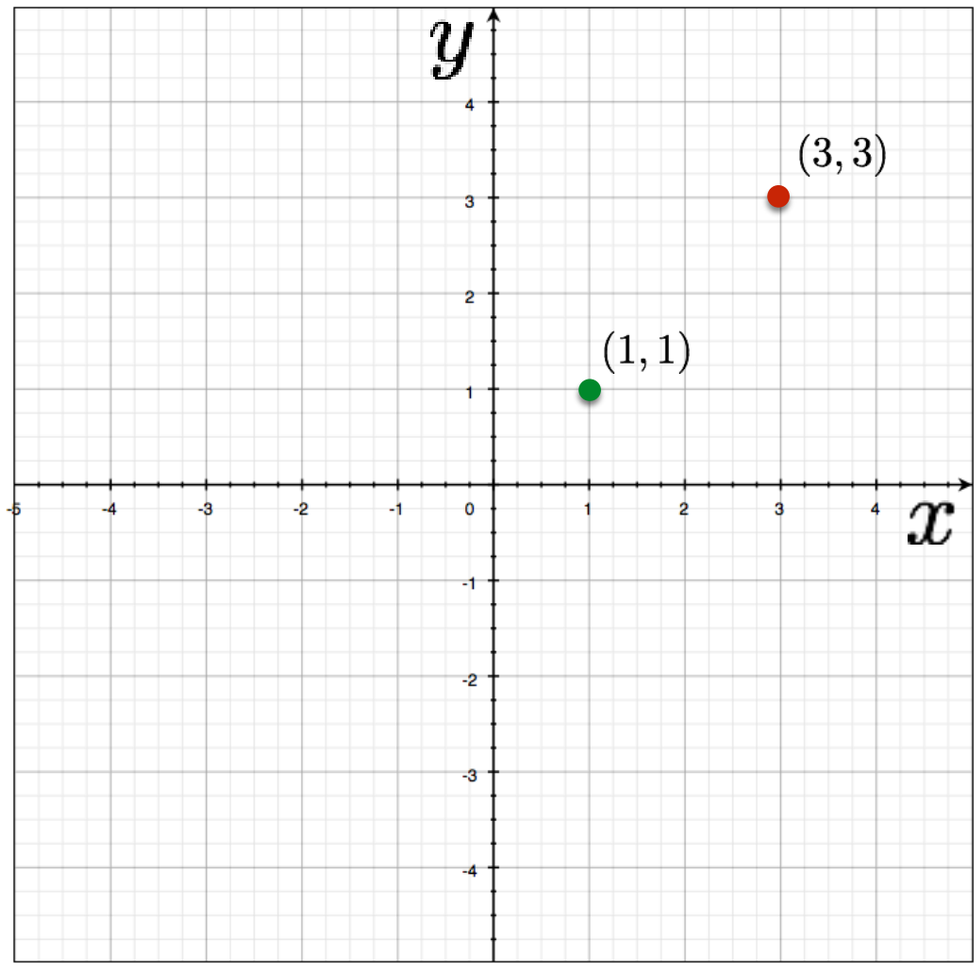
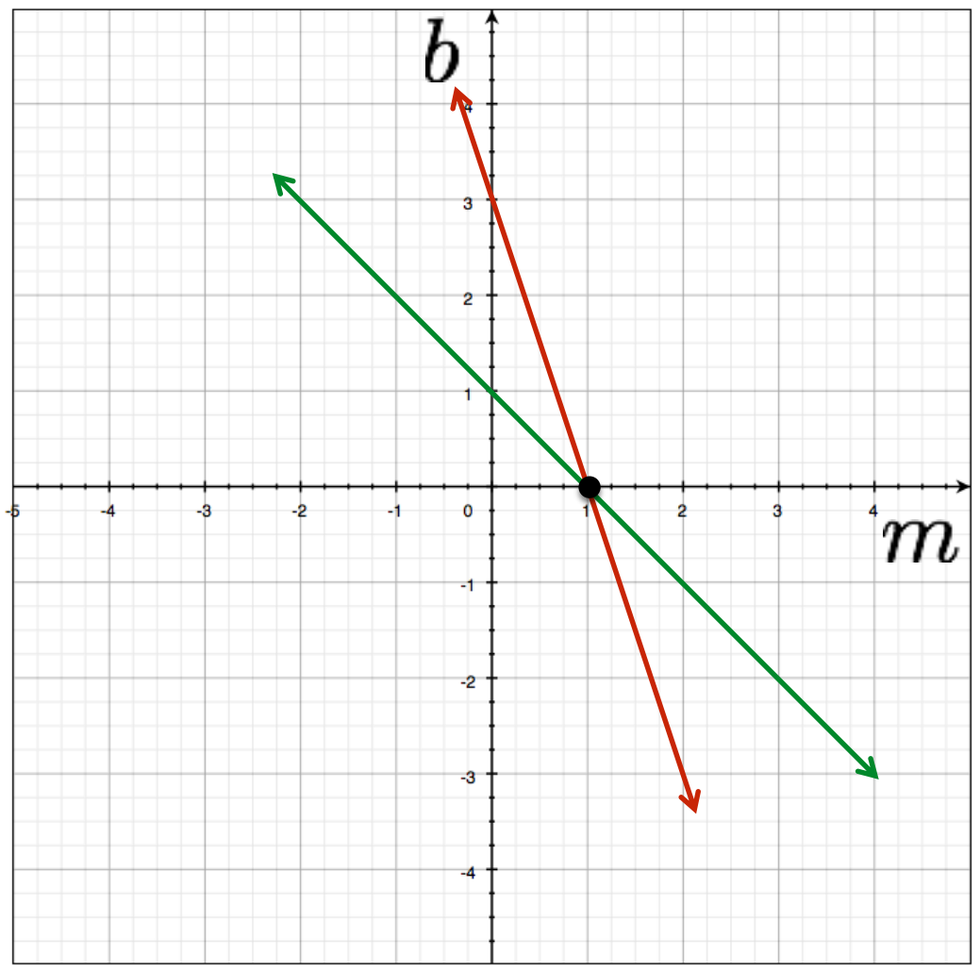


Image space

two points become ?



Parameter space

Image and parameter space

variables

$$y = mx + b$$

parameters

variables

$$y - mx = b$$

parameters

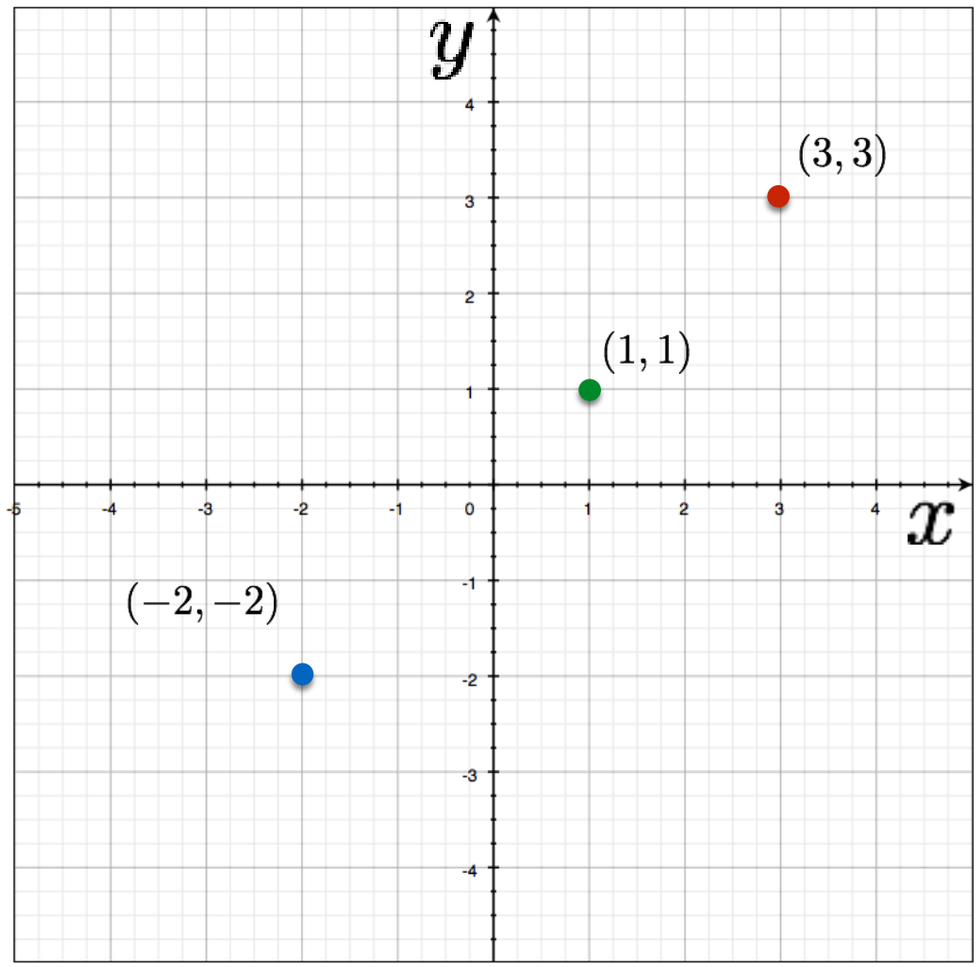
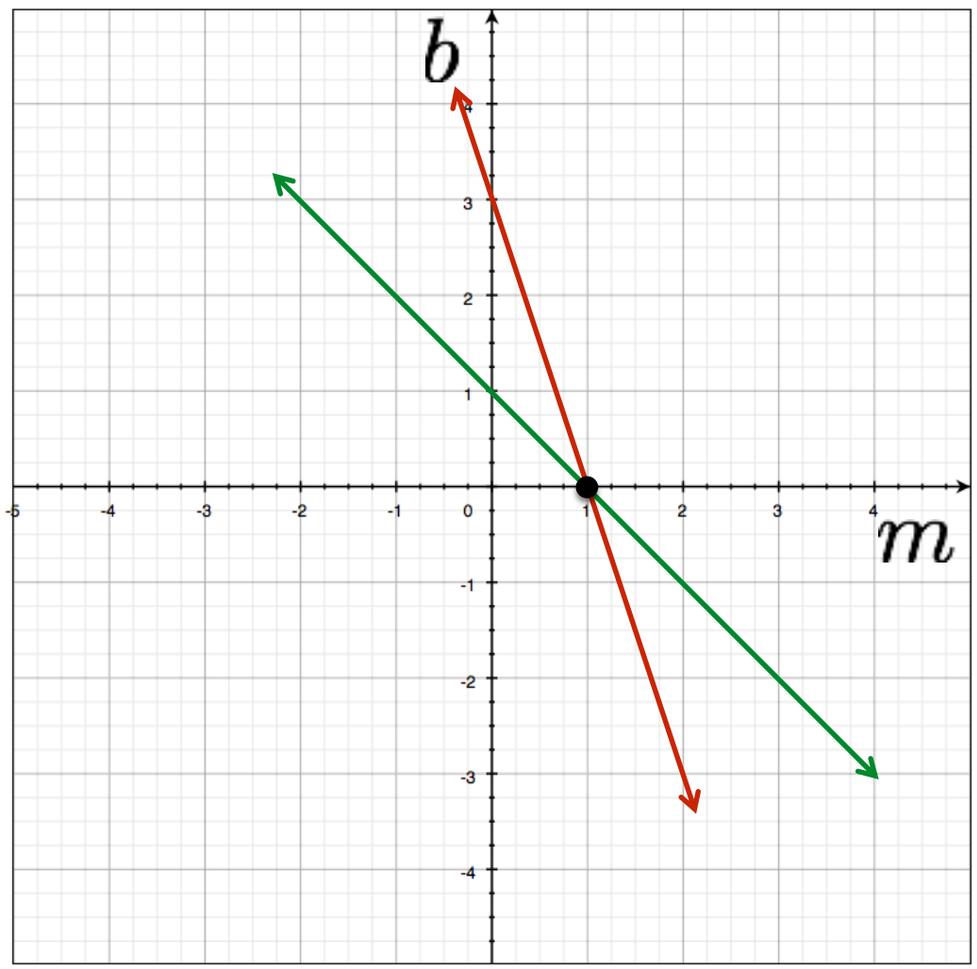


Image space

three points become ?



Parameter space

Image and parameter space

variables

$$y = mx + b$$

parameters

variables

$$y - mx = b$$

parameters

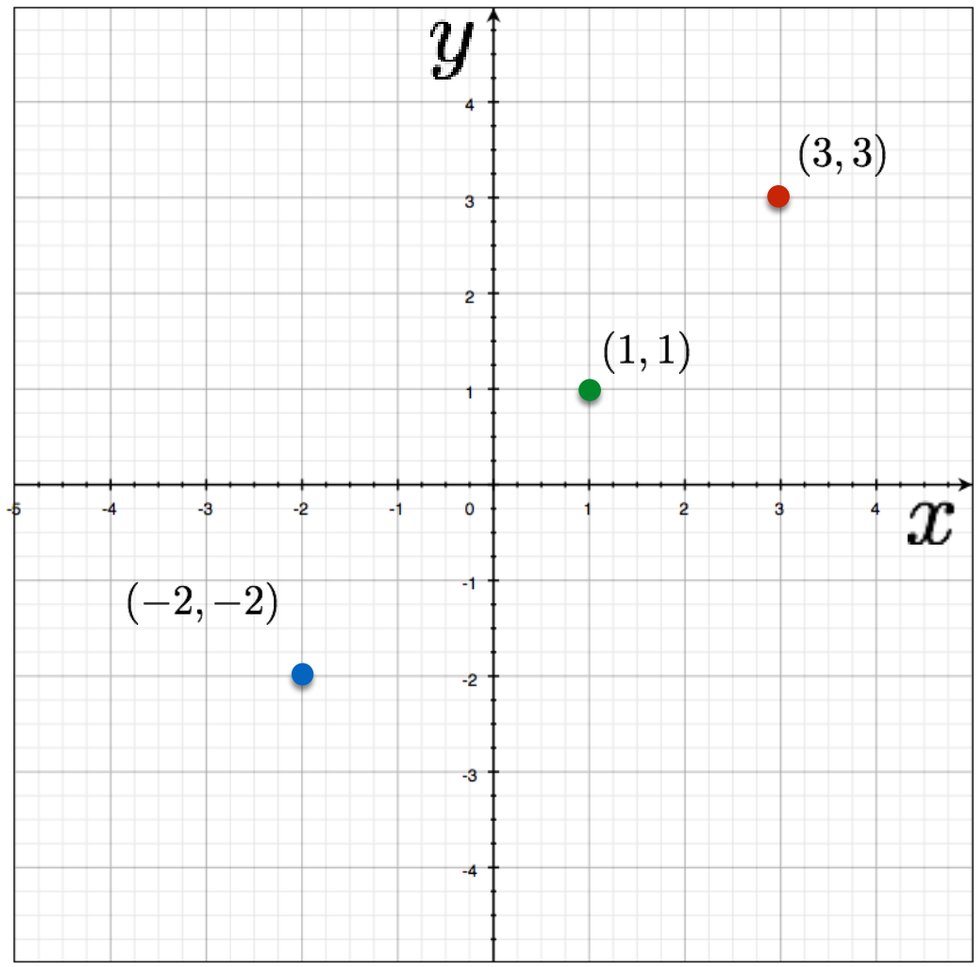
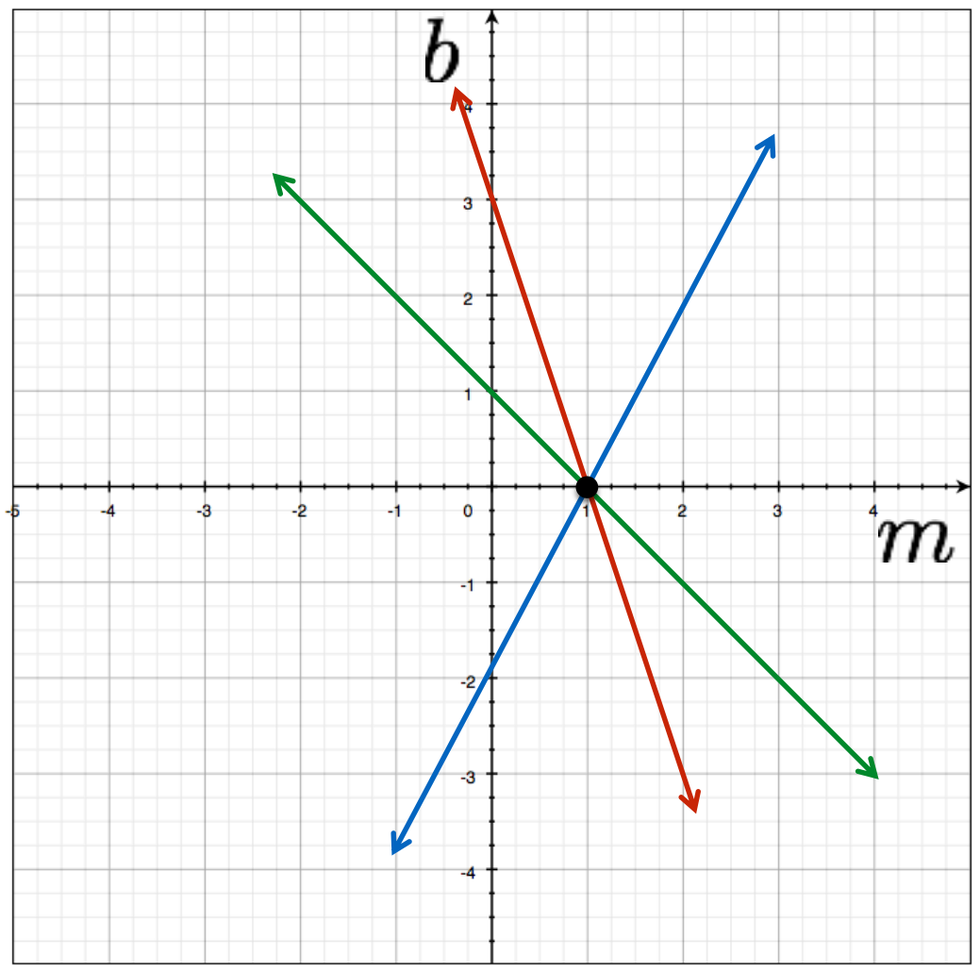


Image space

three points become ?



Parameter space

Image and parameter space

variables

$$y = mx + b$$

parameters

variables

$$y - mx = b$$

parameters

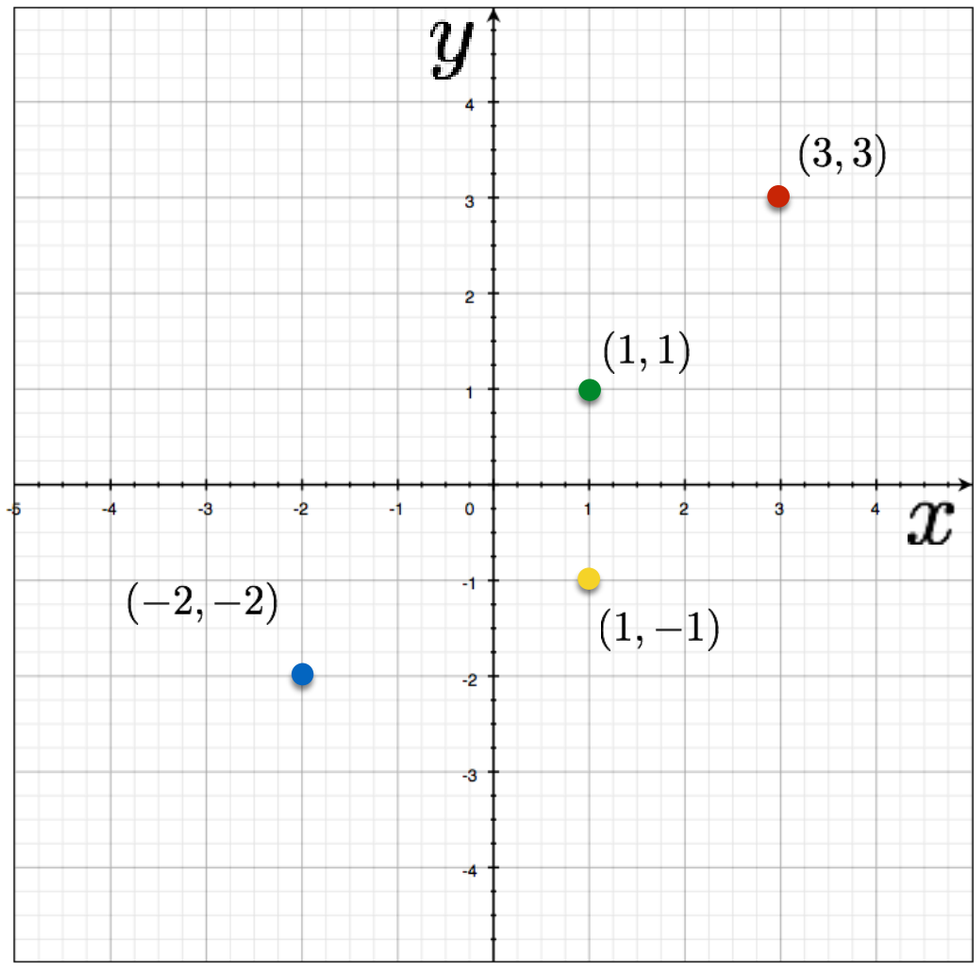
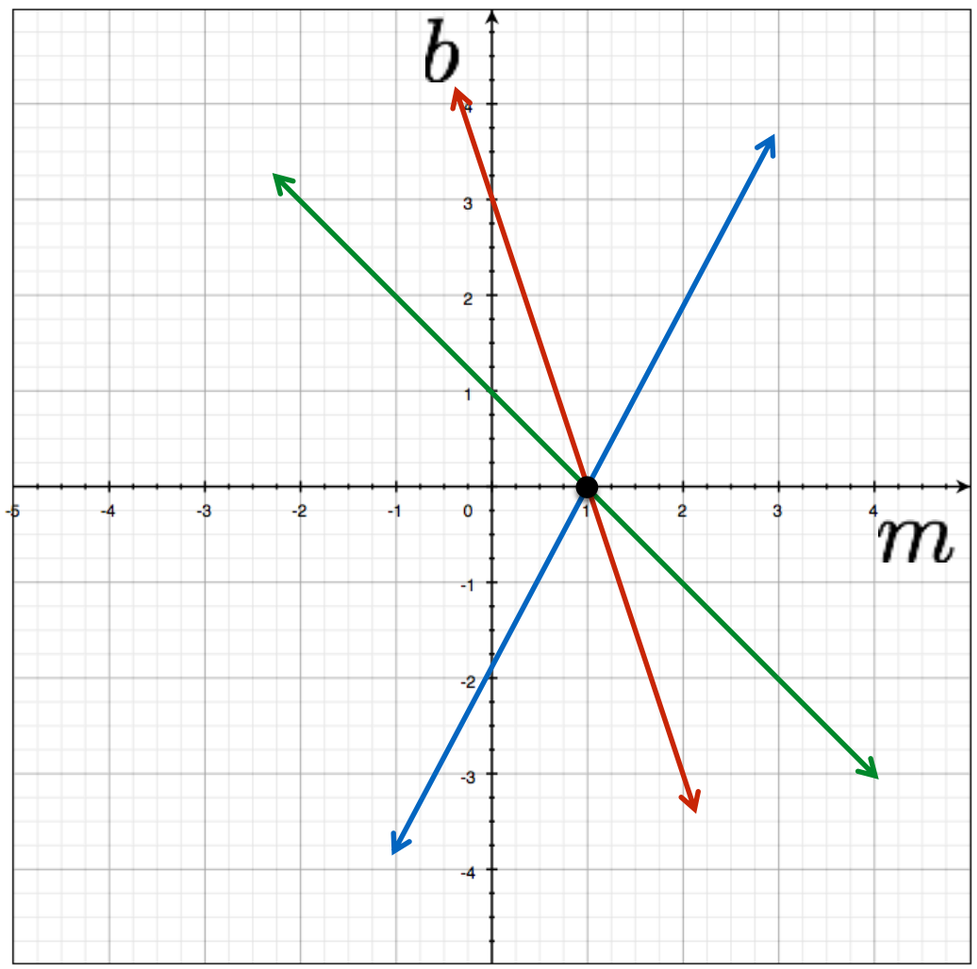


Image space

four points become ?



Parameter space

Image and parameter space

variables

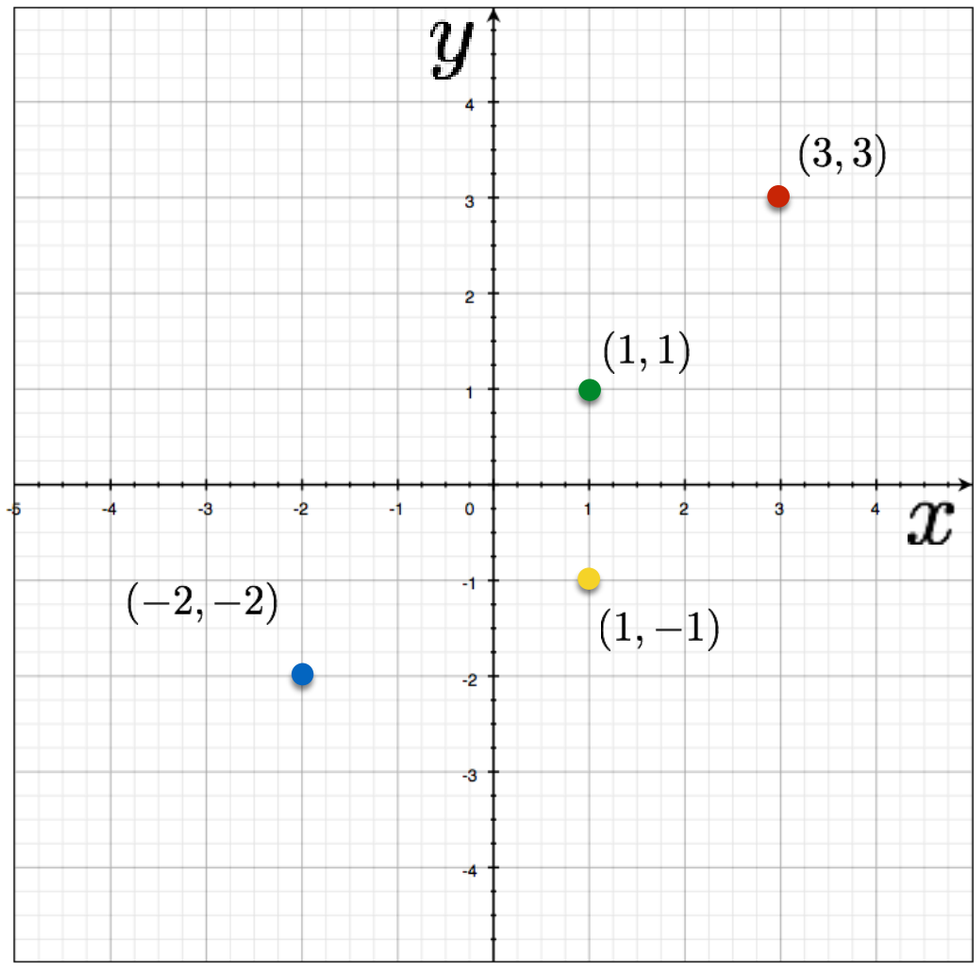
$$y = mx + b$$

parameters

variables

$$y - mx = b$$

parameters



four points become ?

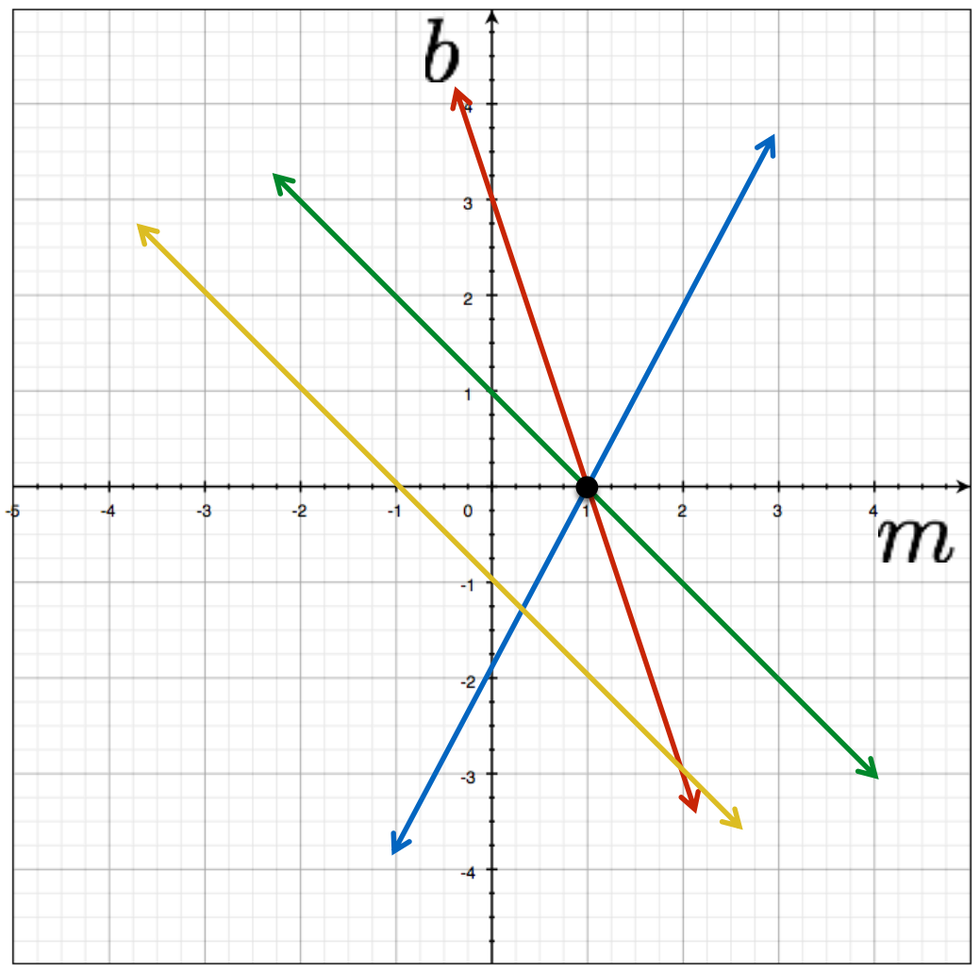


Image space

Parameter space

How would you find the best fitting line?

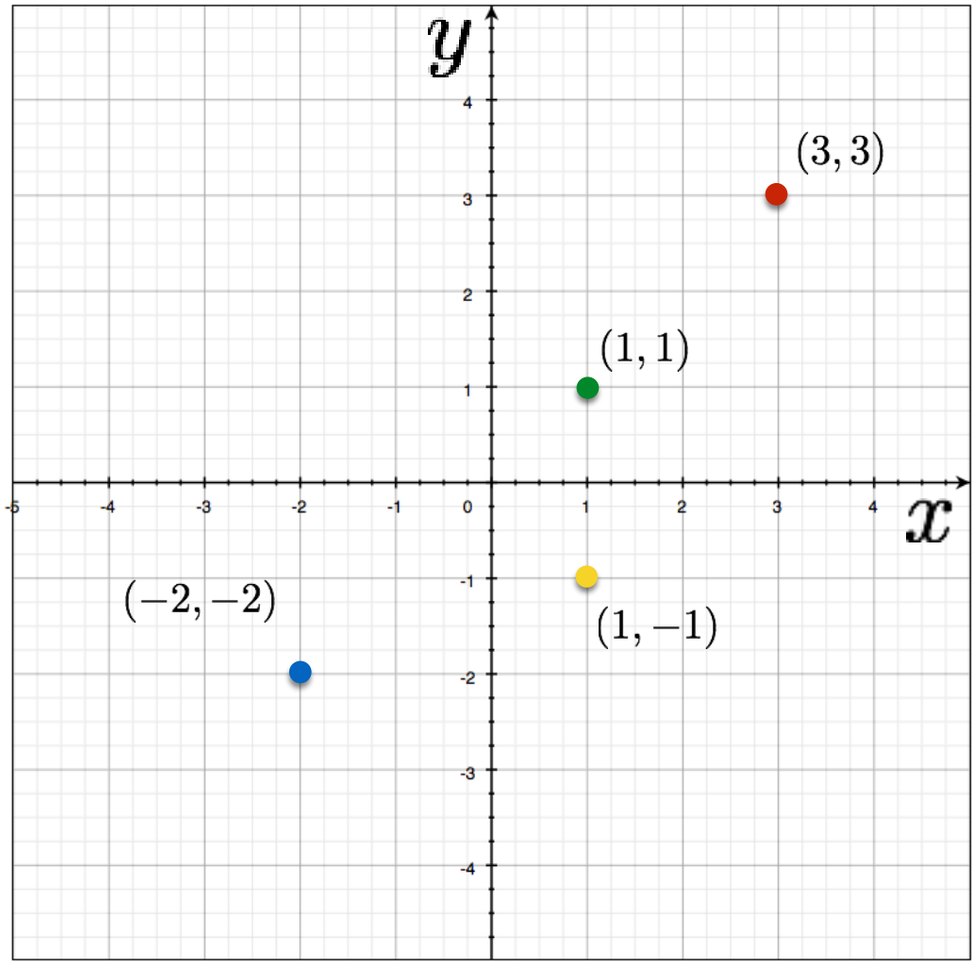
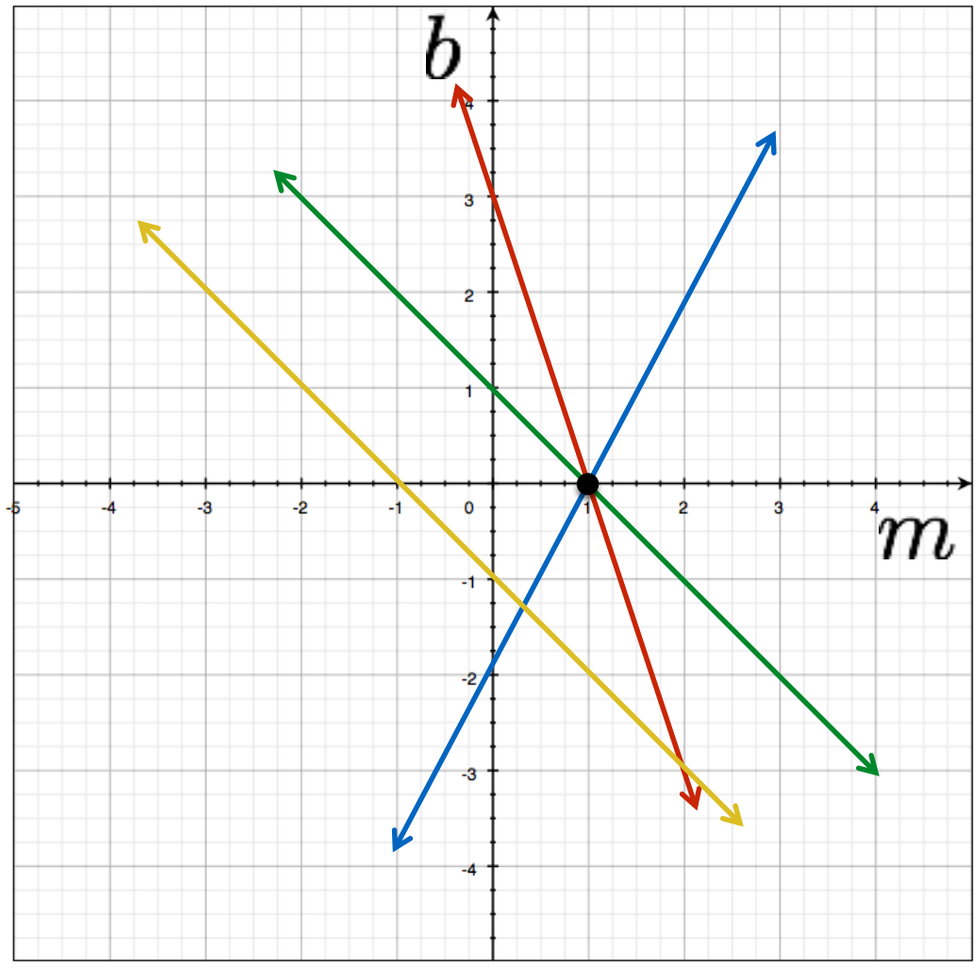


Image space



Parameter space

Is this method robust to measurement noise?

Is this method robust to outliers?

Line Detection by Hough Transform

Algorithm:

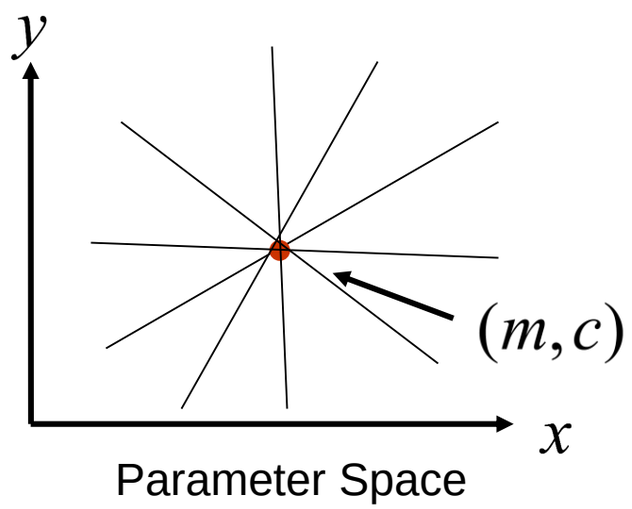
1. Quantize Parameter Space (m,c)

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

$A(m,c)$

	1						1		
		1				1			
			1		1				
				2					
			1		1				
		1				1			
	1						1		

Problems with parameterization

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

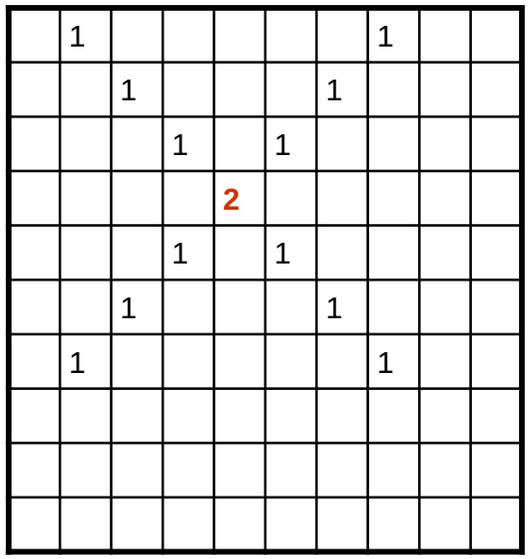
$A(m, c)$

	1						1		
		1				1			
			1		1				
				2					
			1		1				
		1				1			
	1						1		

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!

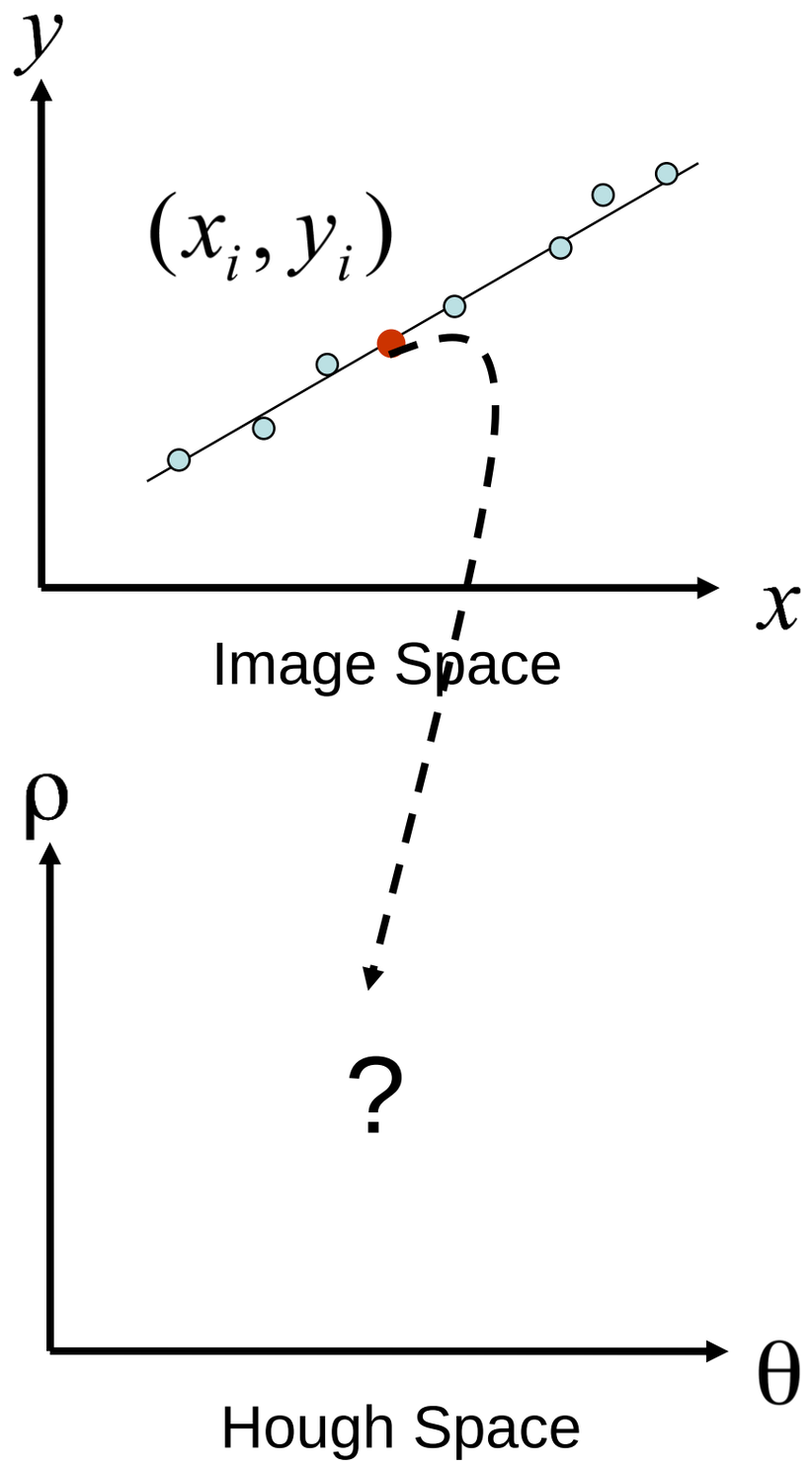
$$-\infty \leq c \leq \infty$$

Better Parameterization

Use normal form:

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

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)

Image and parameter space

variables

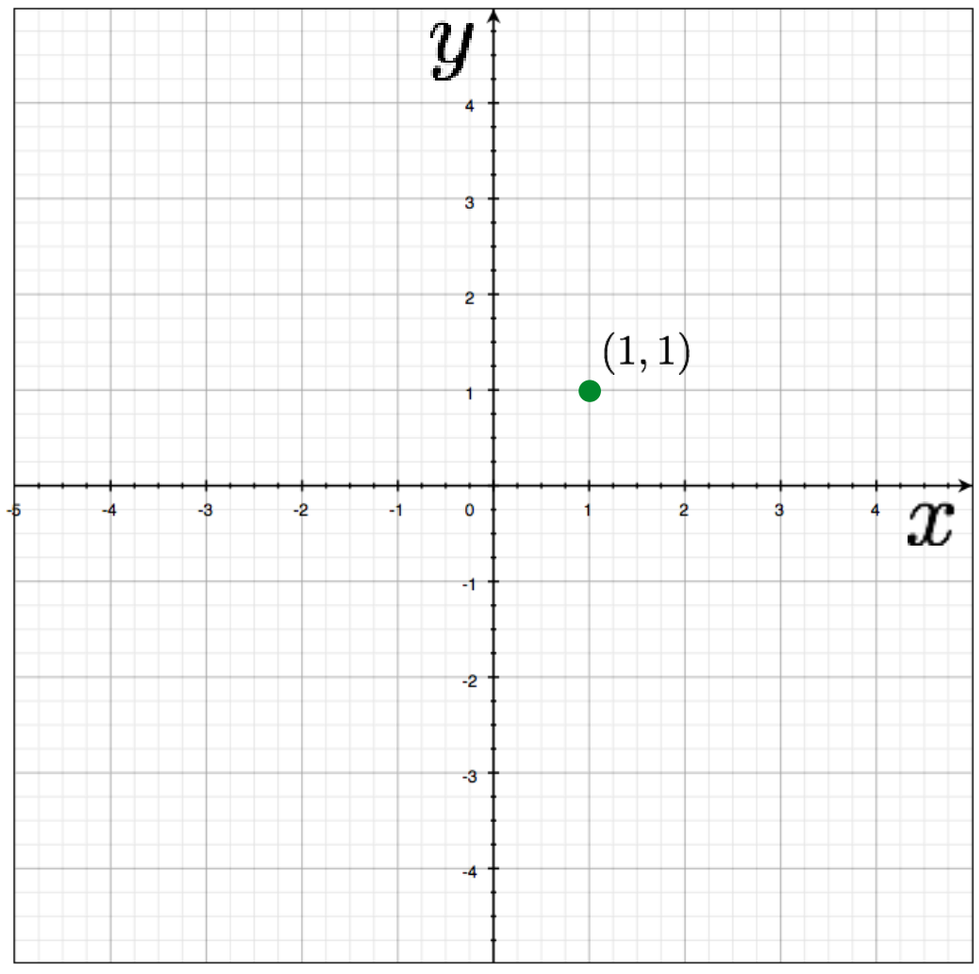
$$y = mx + b$$

parameters

parameters

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

variables



a point becomes?

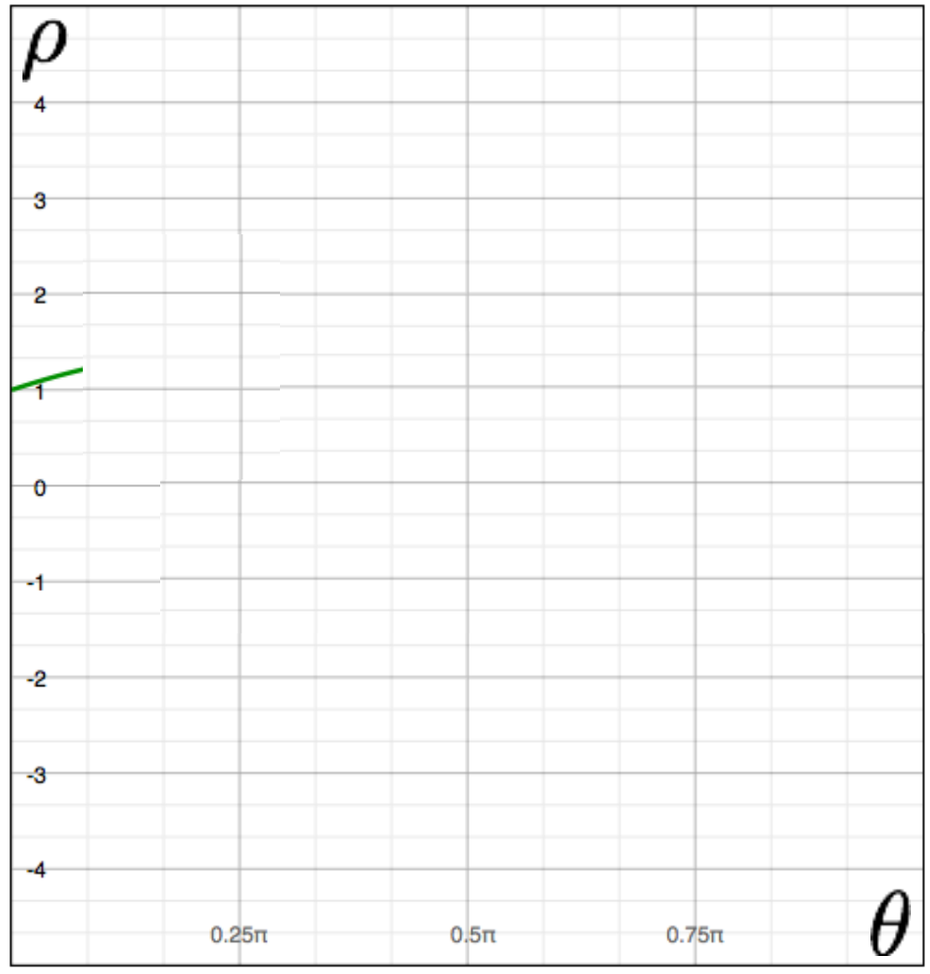


Image space

Parameter space

Image and parameter space

variables

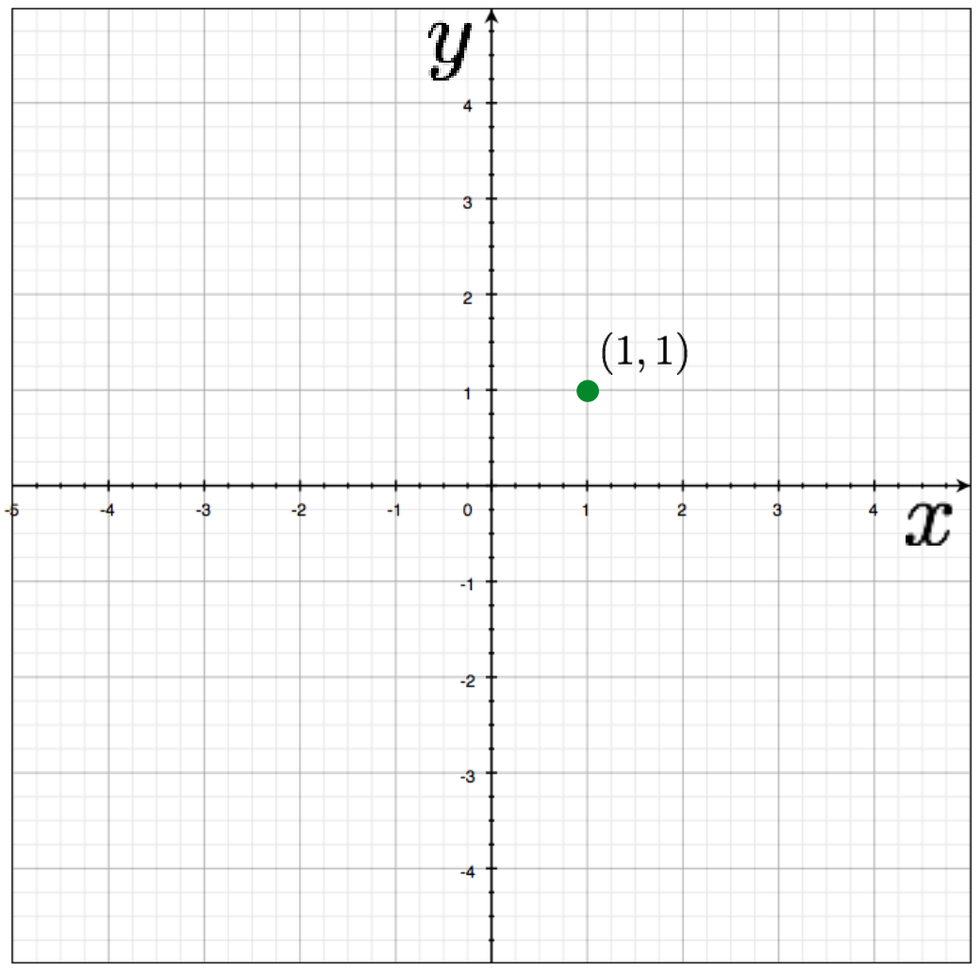
$$y = mx + b$$

parameters

parameters

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

variables



a point becomes a wave

Image space

Parameter space

Image and parameter space

variables

$$y = mx + b$$

parameters

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

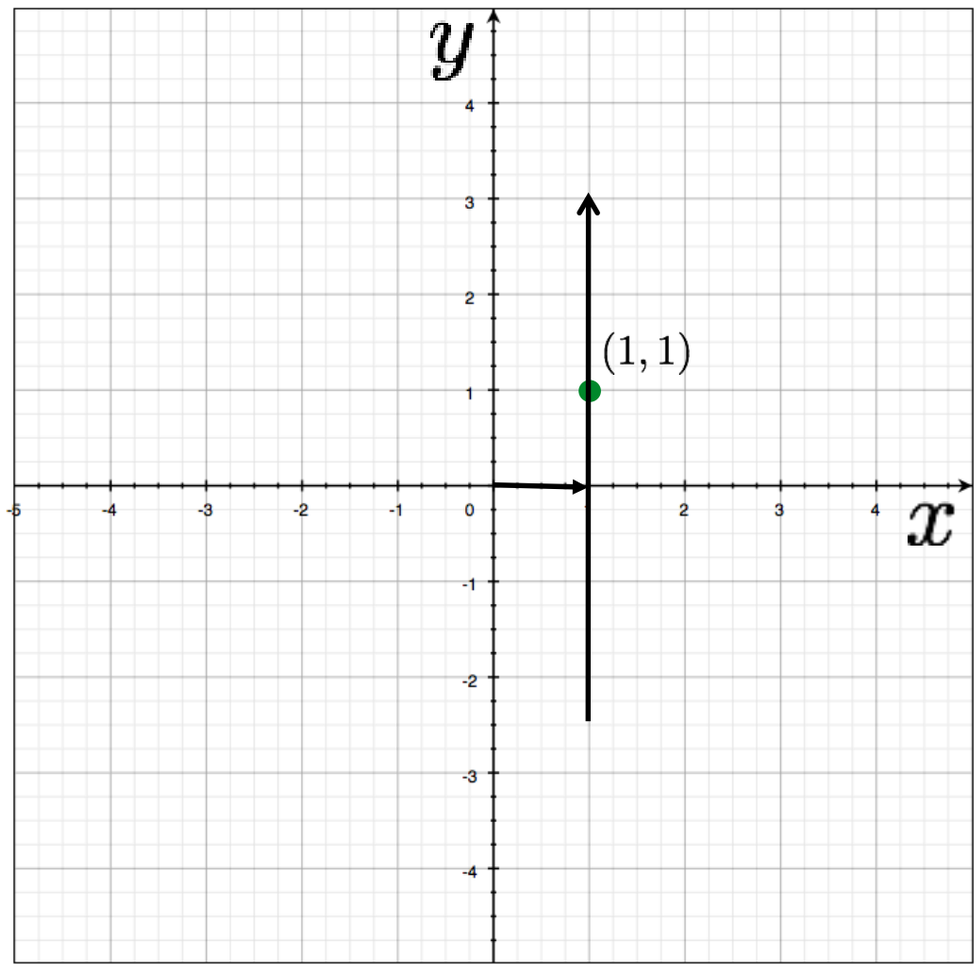
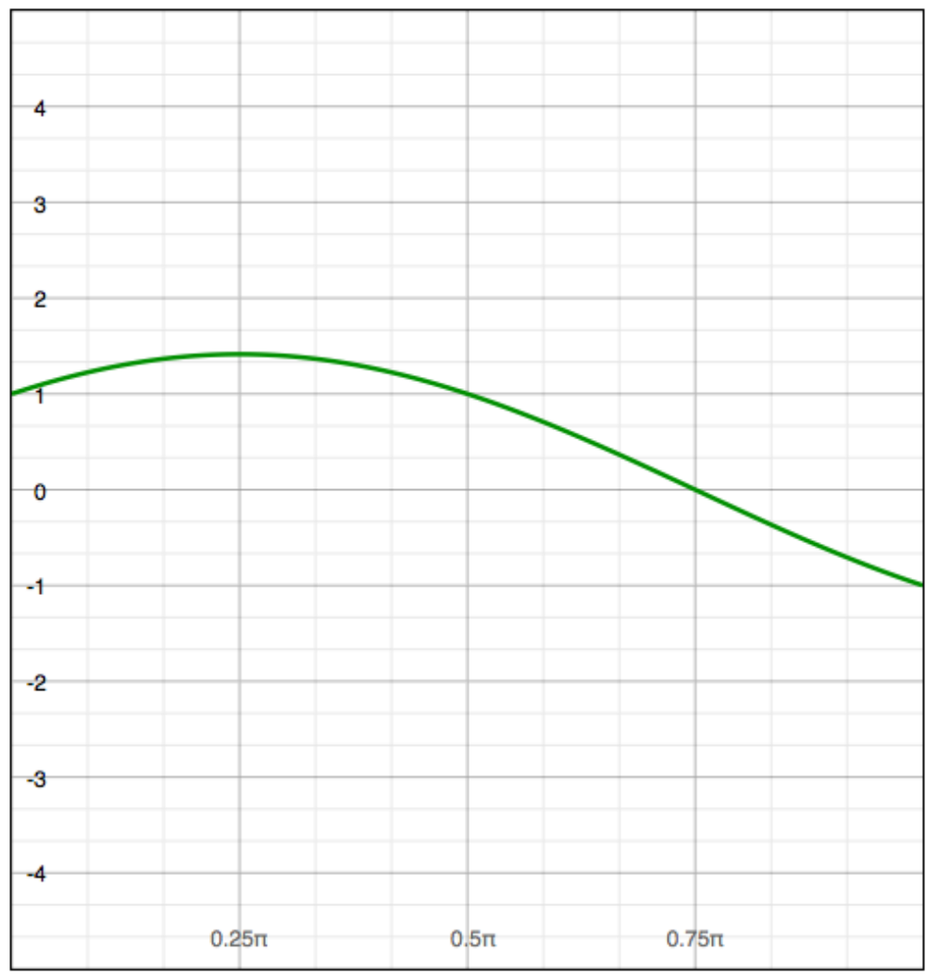


Image space

a line becomes?



Parameter space

Image and parameter space

variables

$$y = mx + b$$

parameters

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

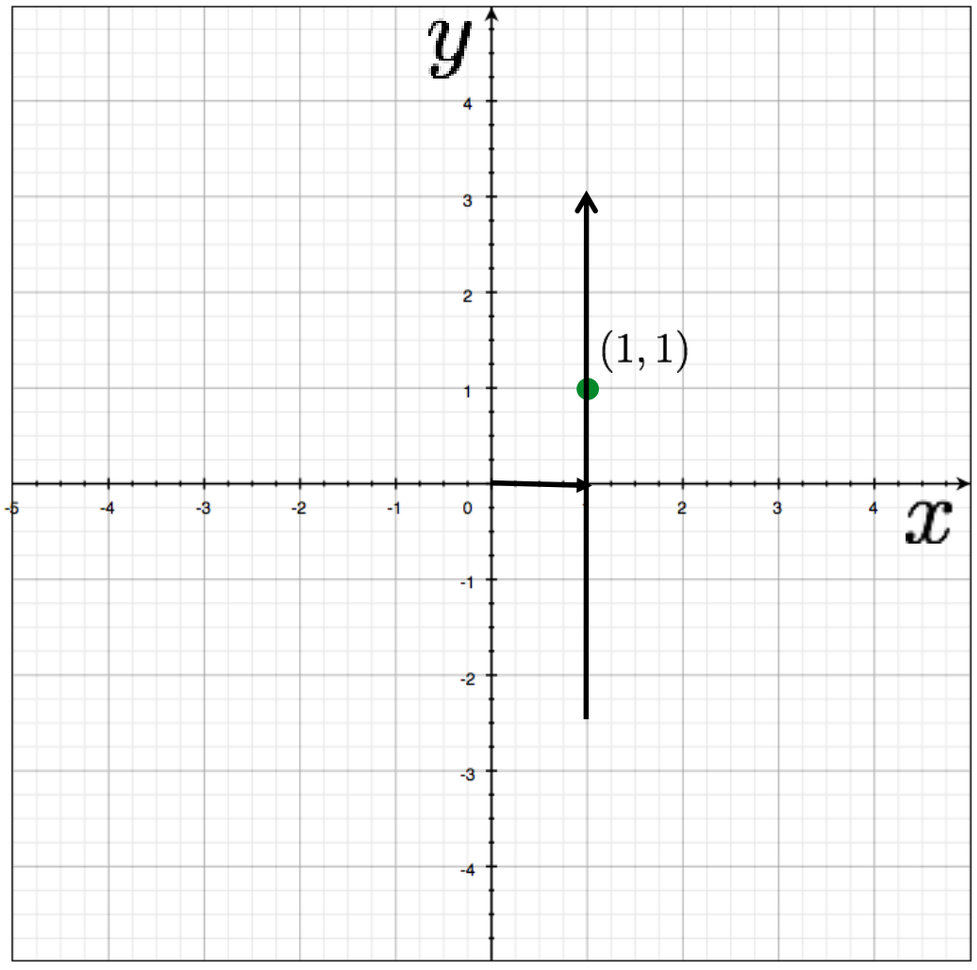
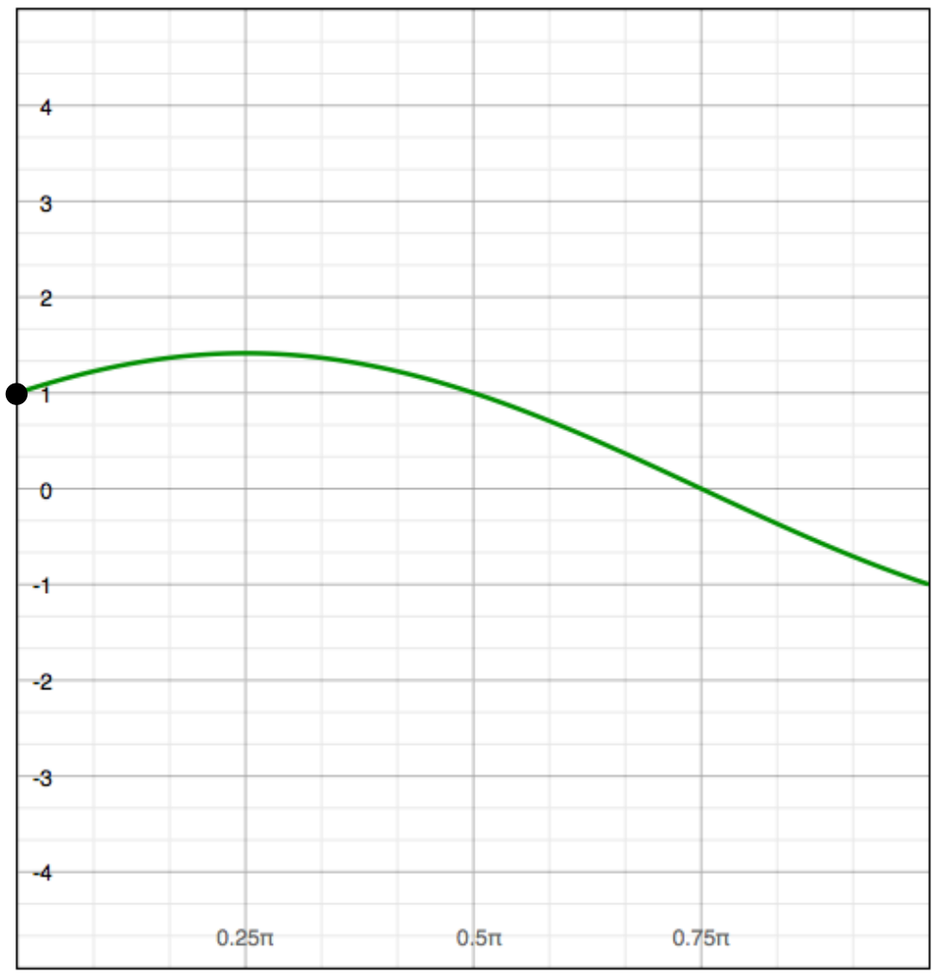


Image space

a line becomes a point



Parameter space

Image and parameter space

variables

$$y = mx + b$$

parameters

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

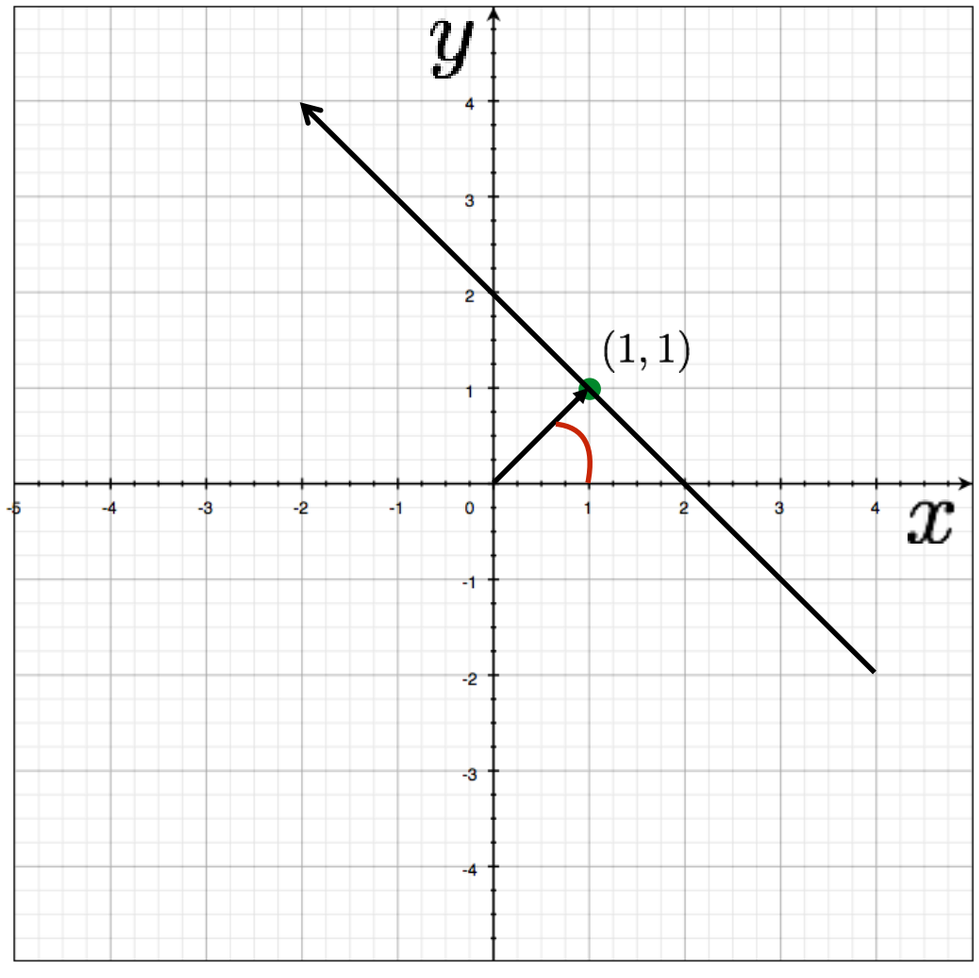
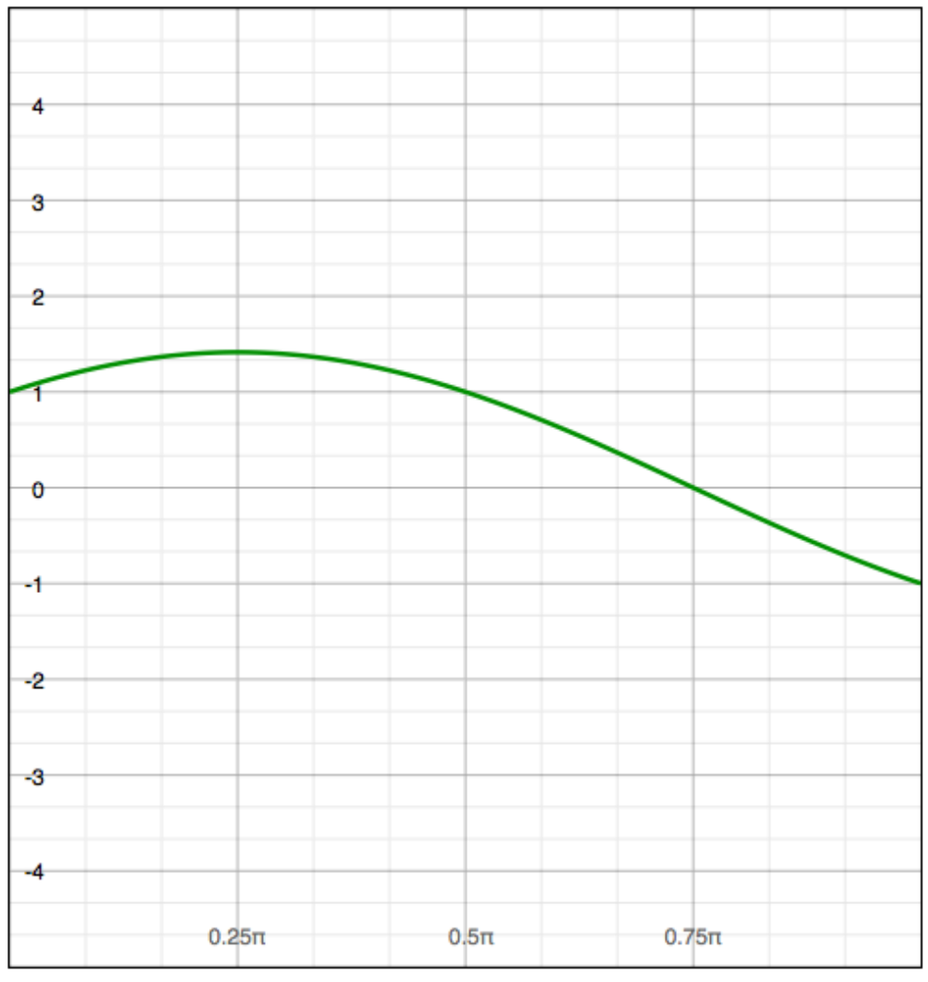


Image space

a line becomes?



Parameter space

Image and parameter space

variables

$$y = mx + b$$

parameters

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

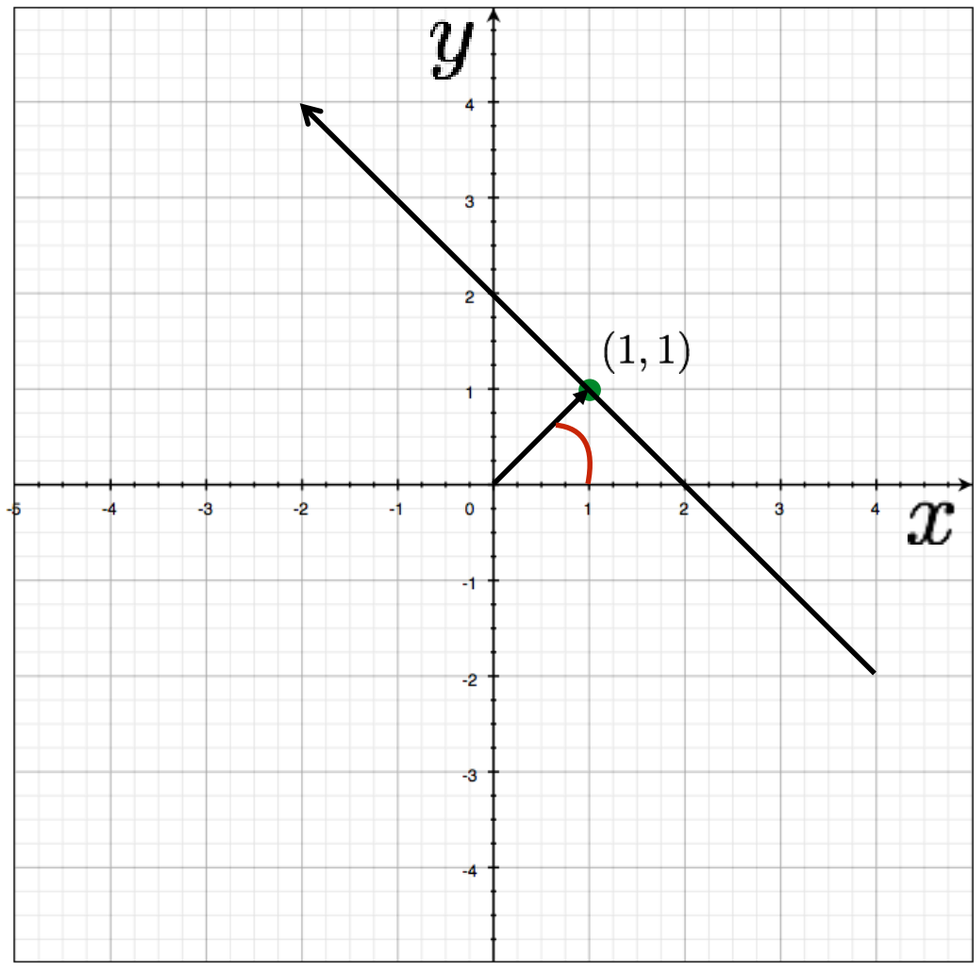
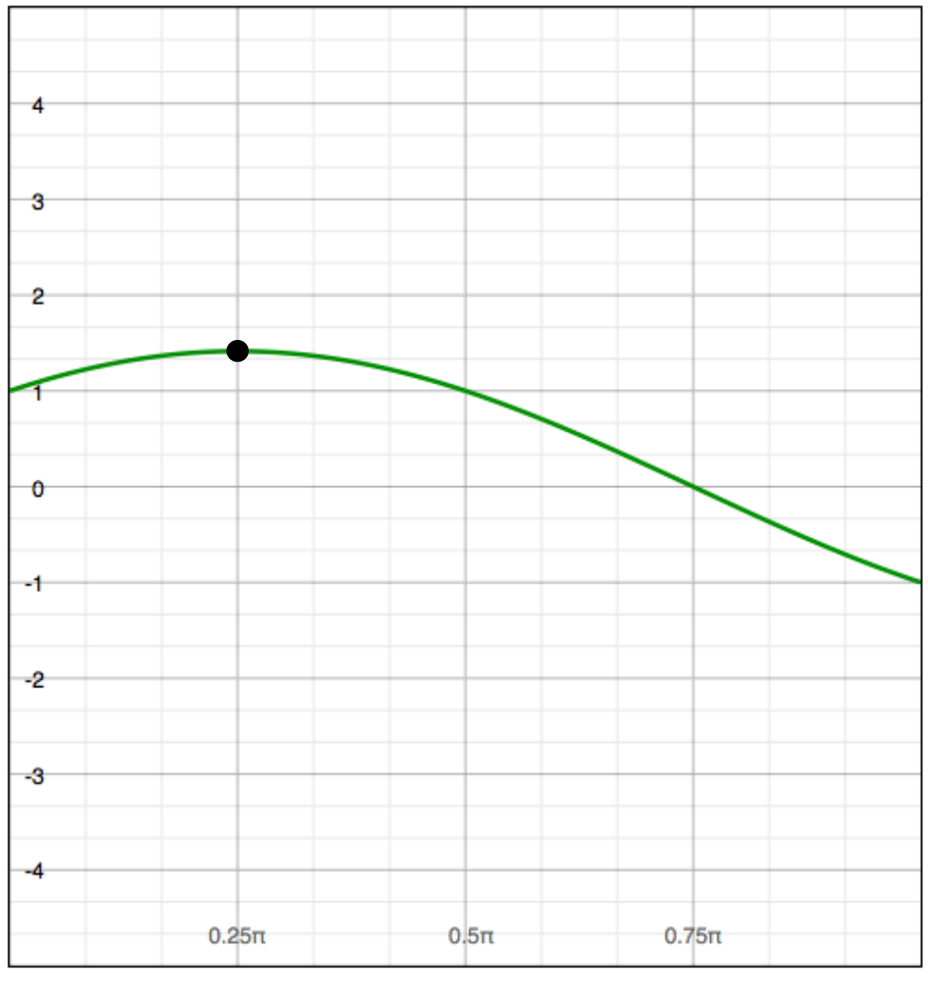


Image space

a line becomes a point



Parameter space

Image and parameter space

variables

$$y = mx + b$$

parameters

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

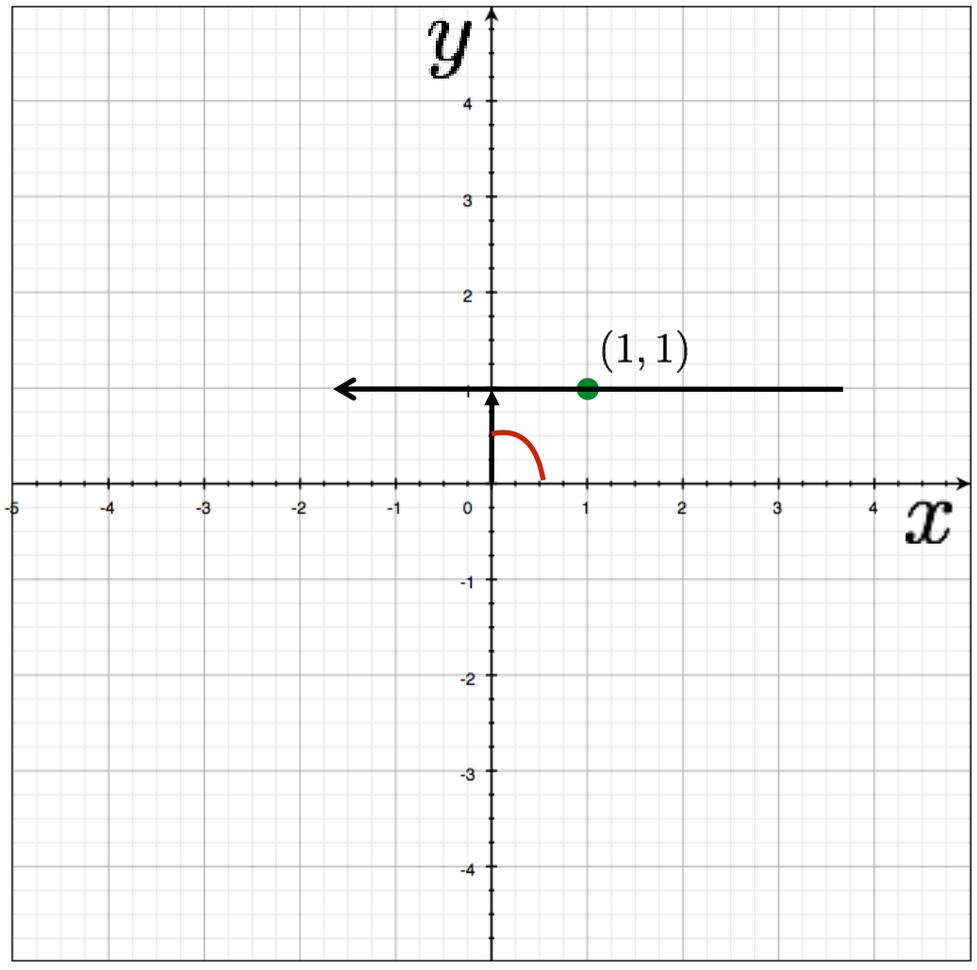
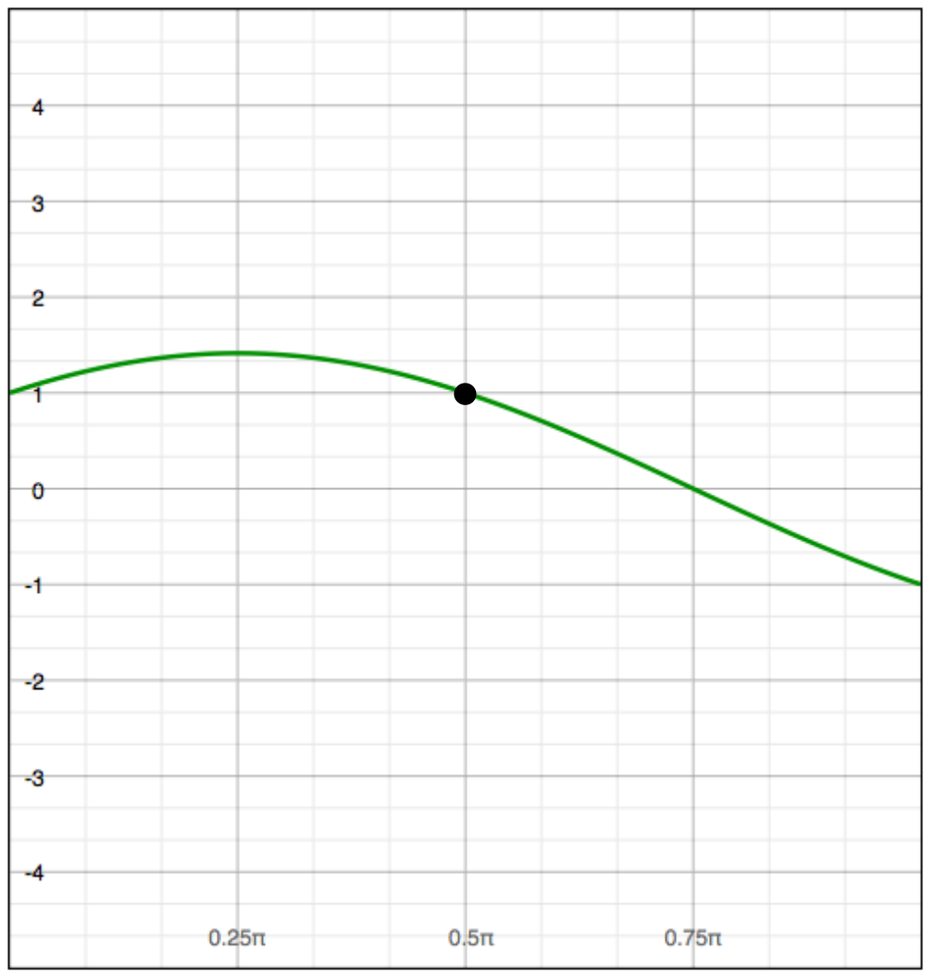


Image space

a line becomes a point



Parameter space

Image and parameter space

variables

$$y = mx + b$$

parameters

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

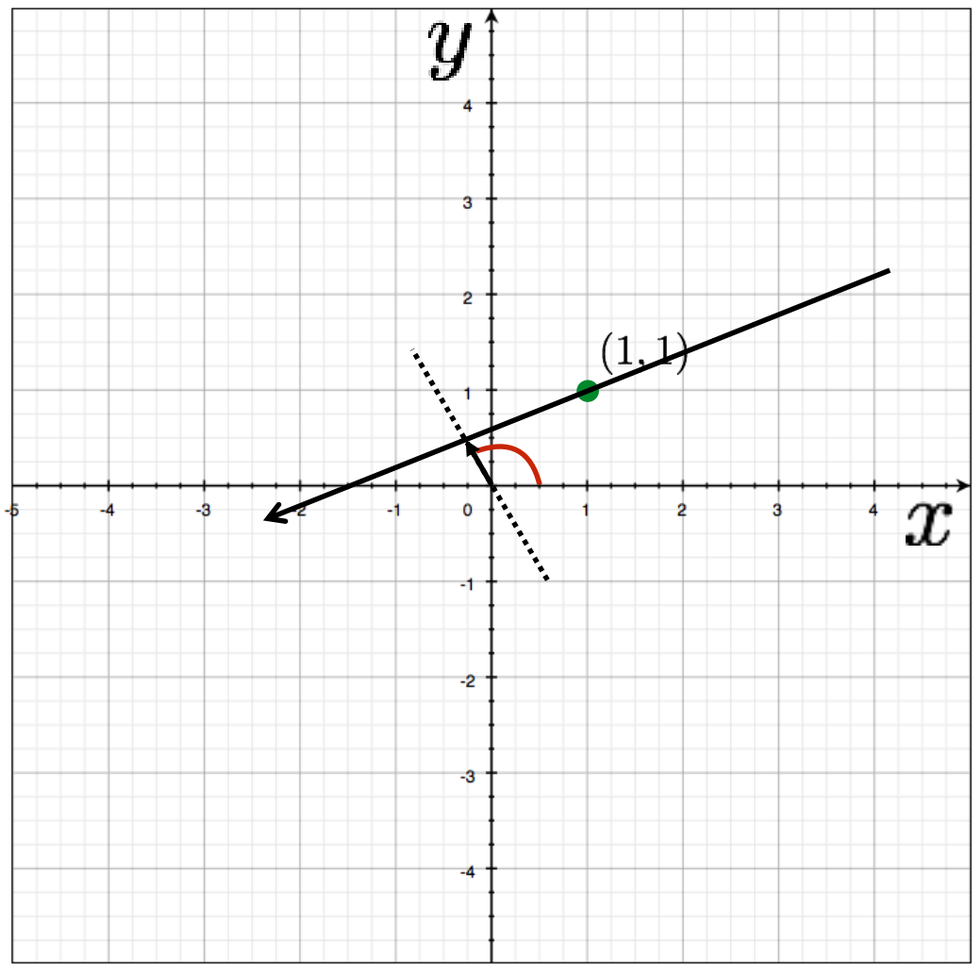
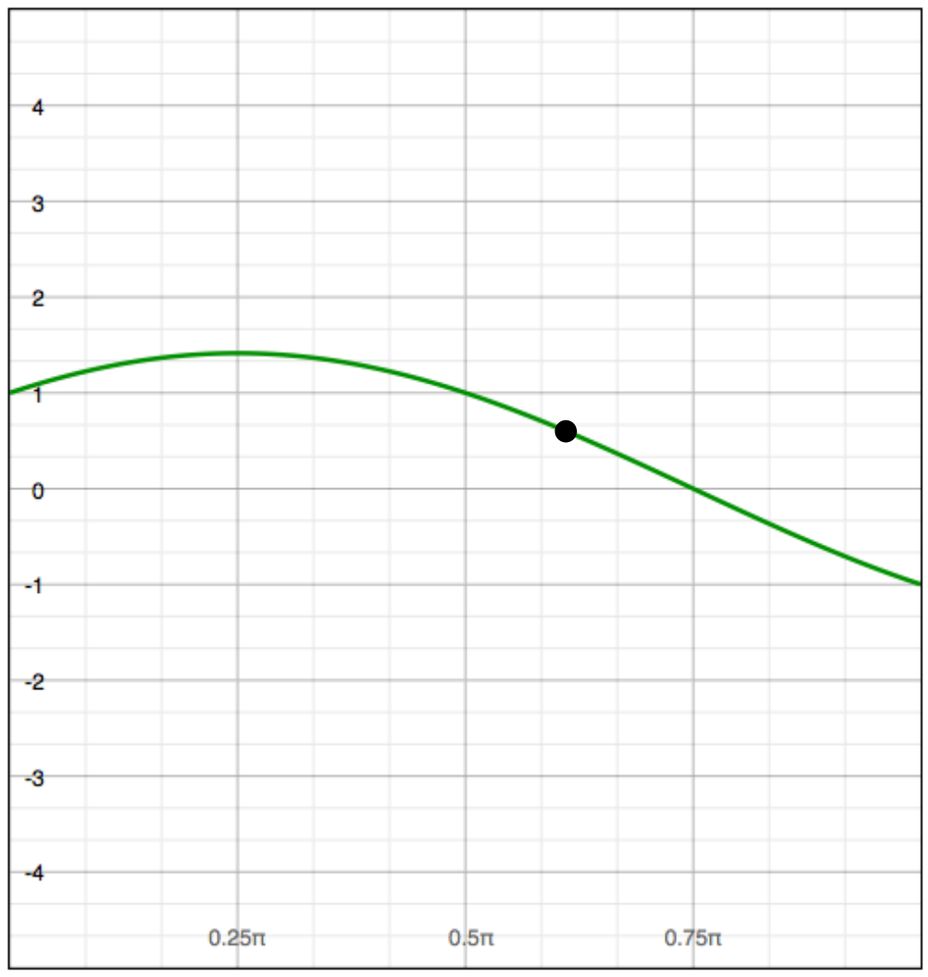


Image space

a line becomes a point



Parameter space

Image and parameter space

variables

$$y = mx + b$$

parameters

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

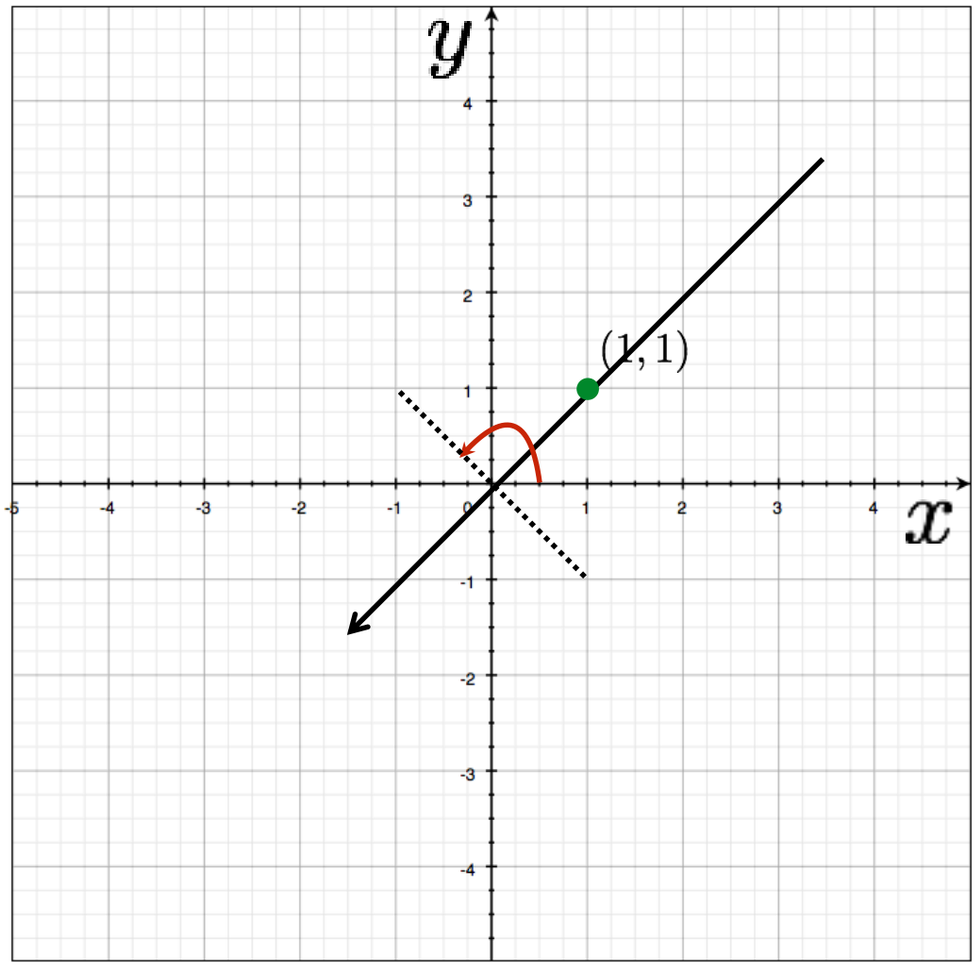
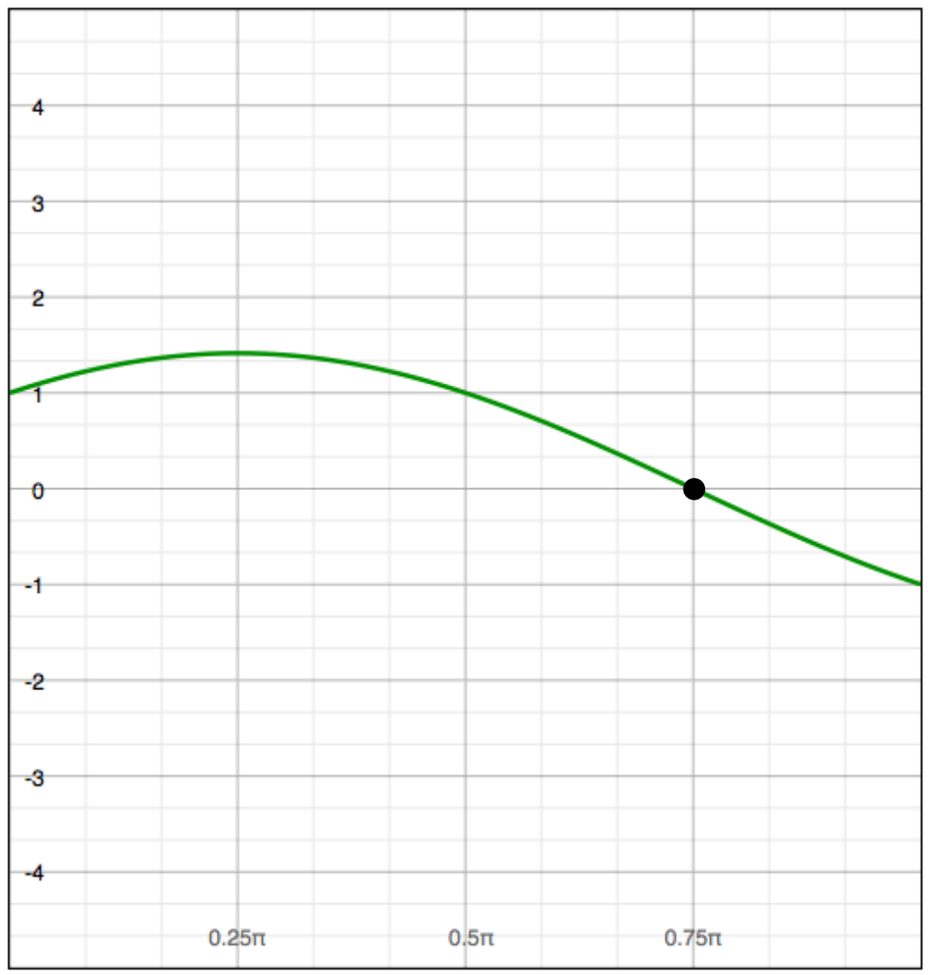


Image space

a line becomes a point



Parameter space

Image and parameter space

variables

$$y = mx + b$$

parameters

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

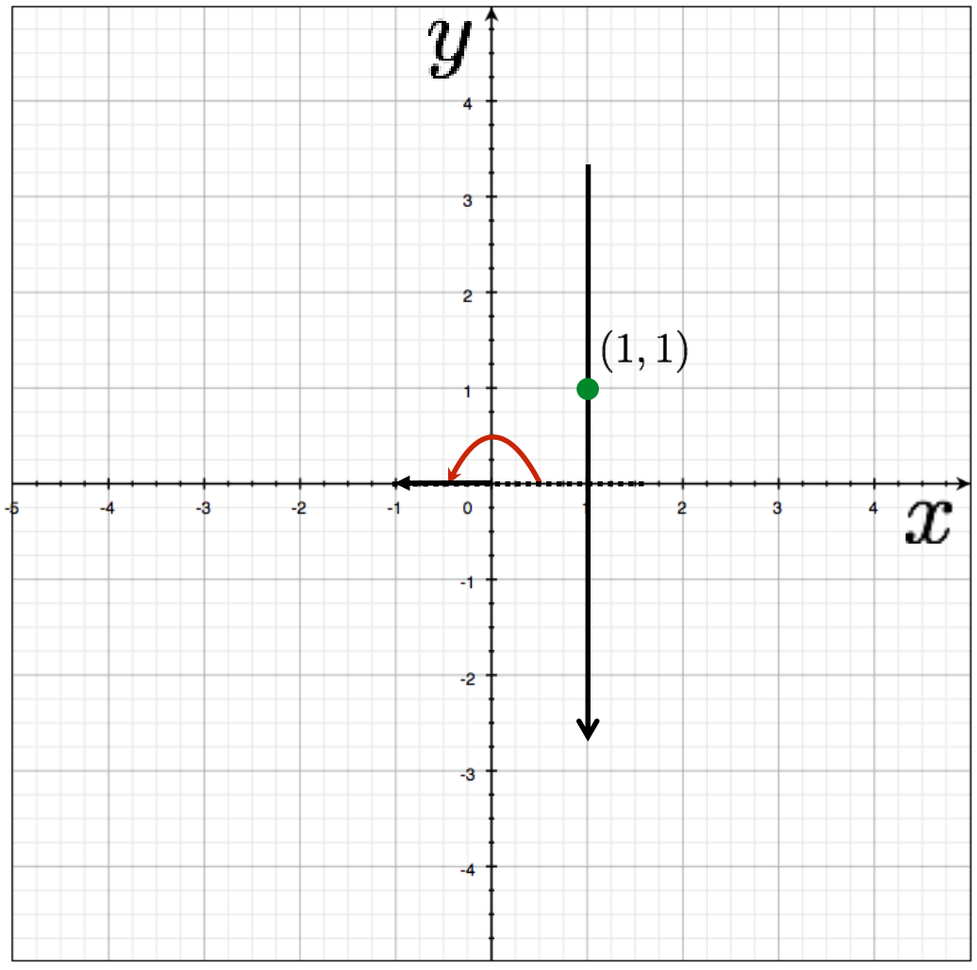
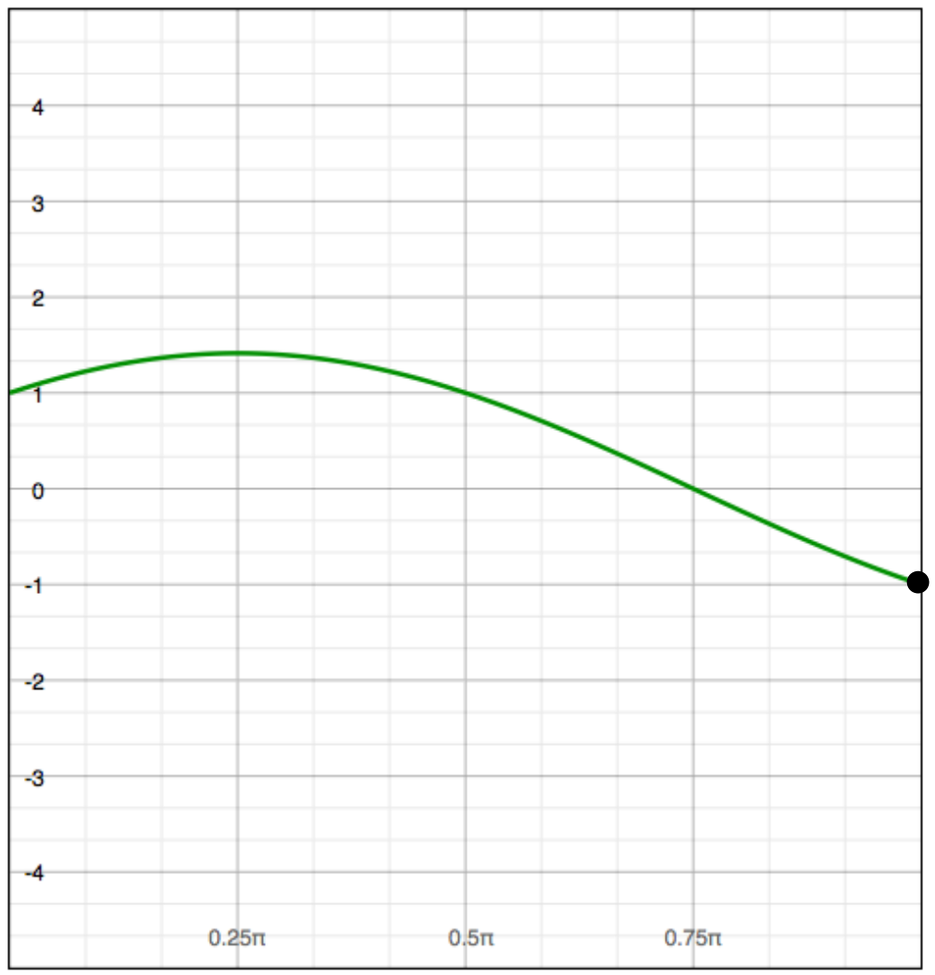


Image space

a line becomes a point



Parameter space

Image and parameter space

variables

$$y = mx + b$$

parameters

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

Wait …why is rho negative?

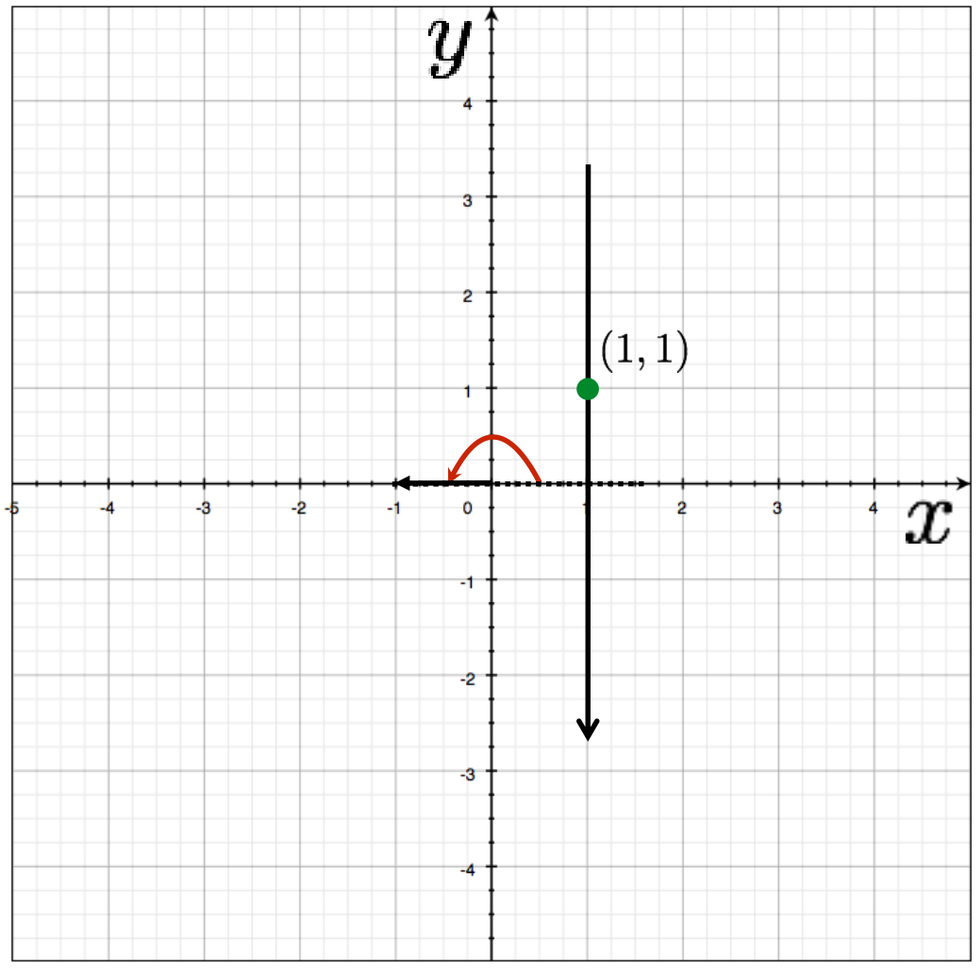
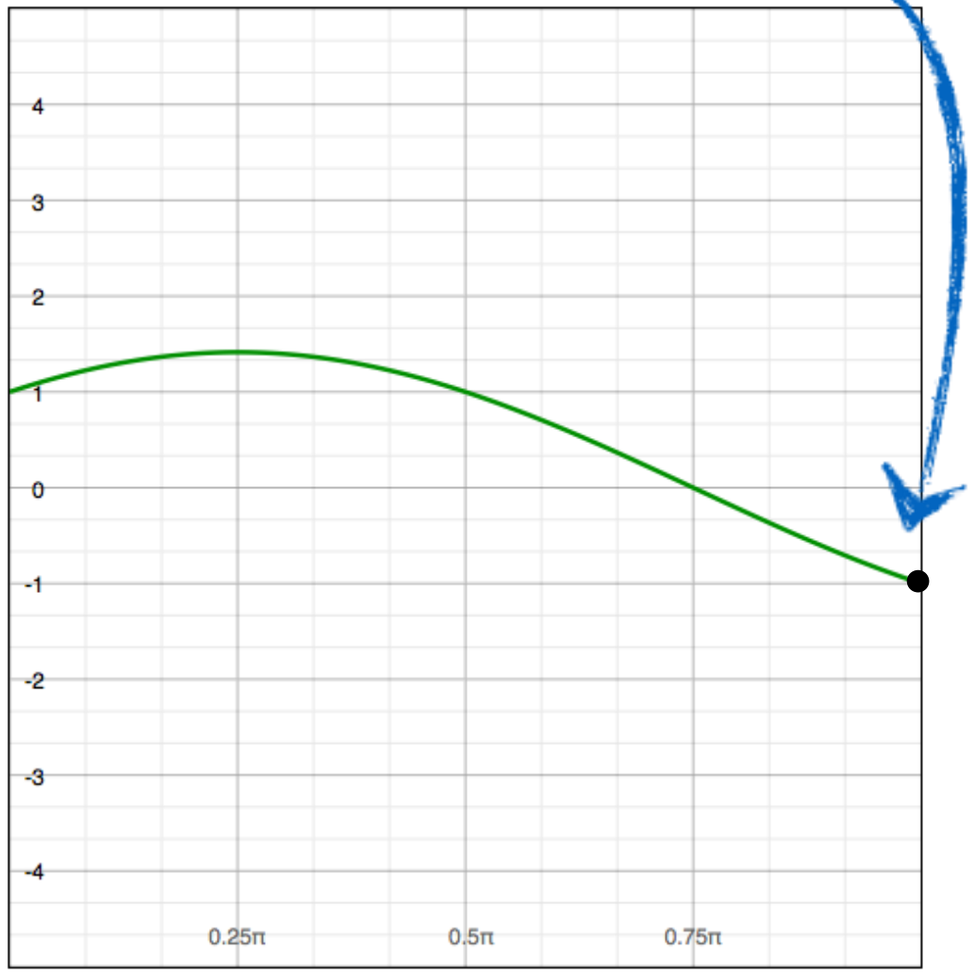


Image space

a line becomes a point



Parameter space

Image and parameter space

variables

$$y = mx + b$$

parameters

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

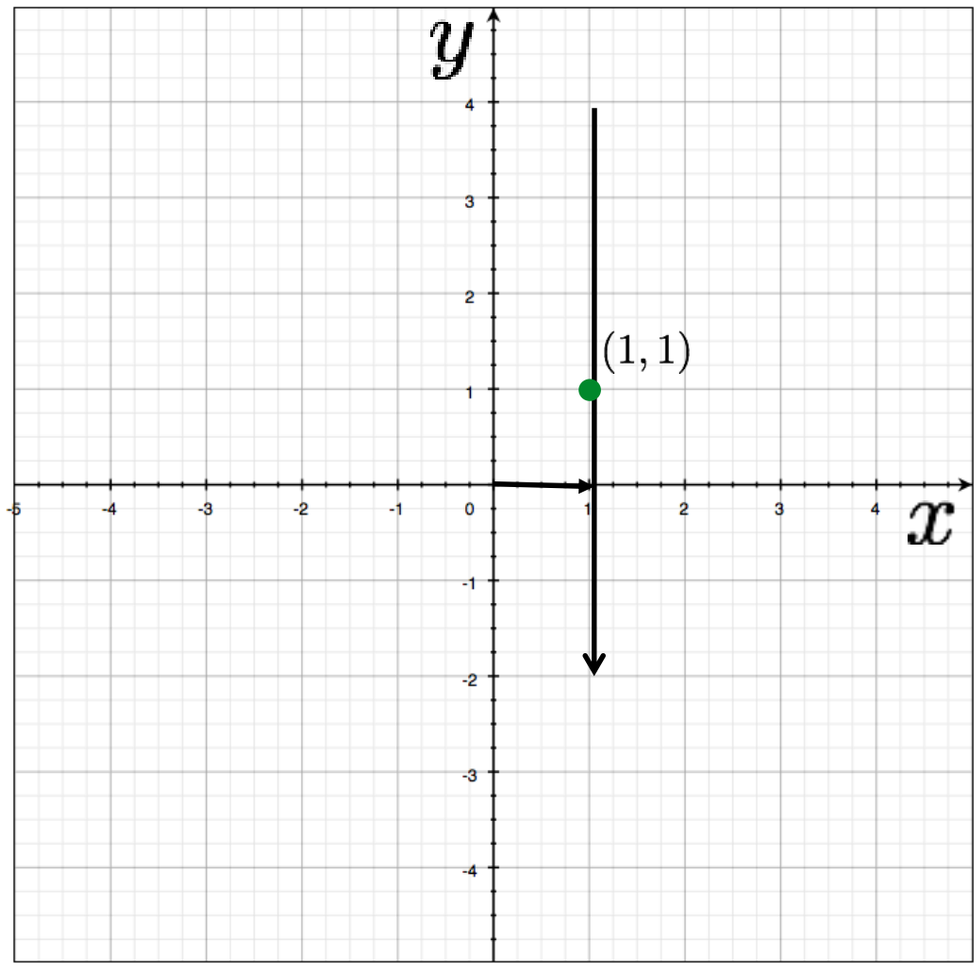
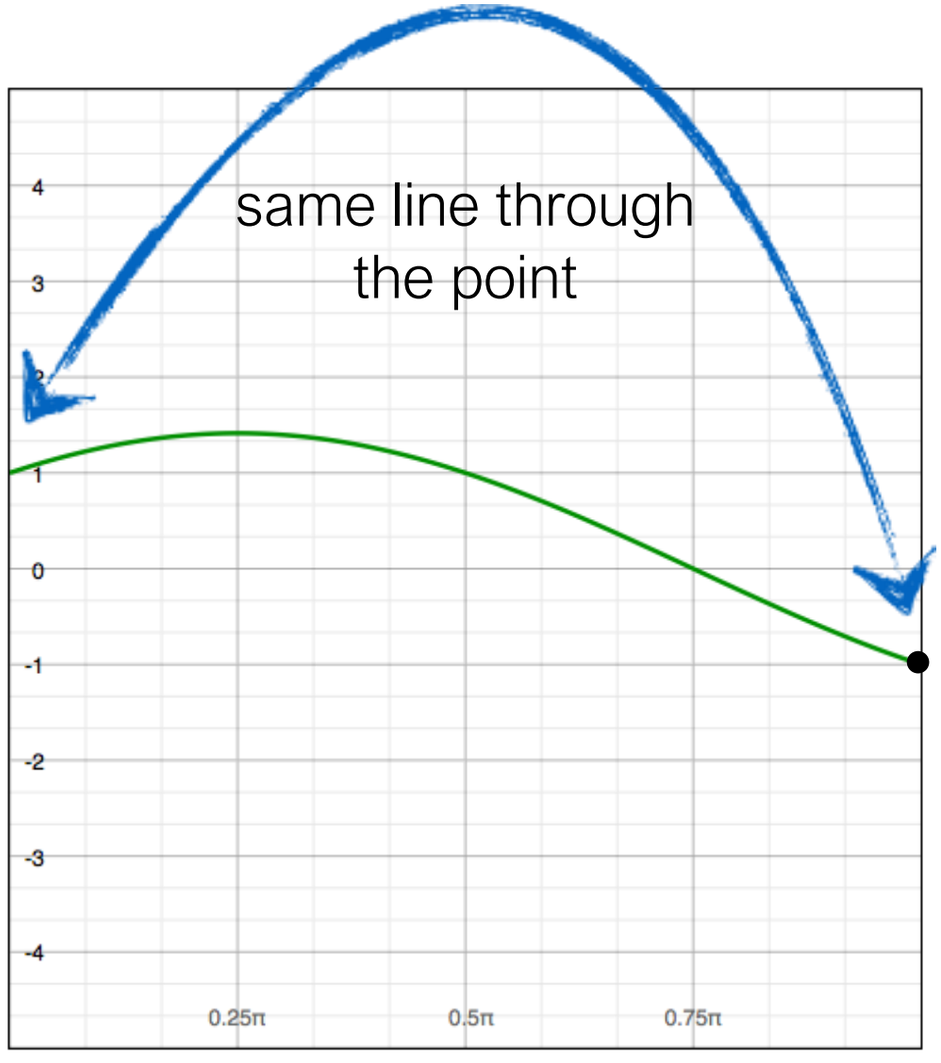


Image space

a line becomes a point

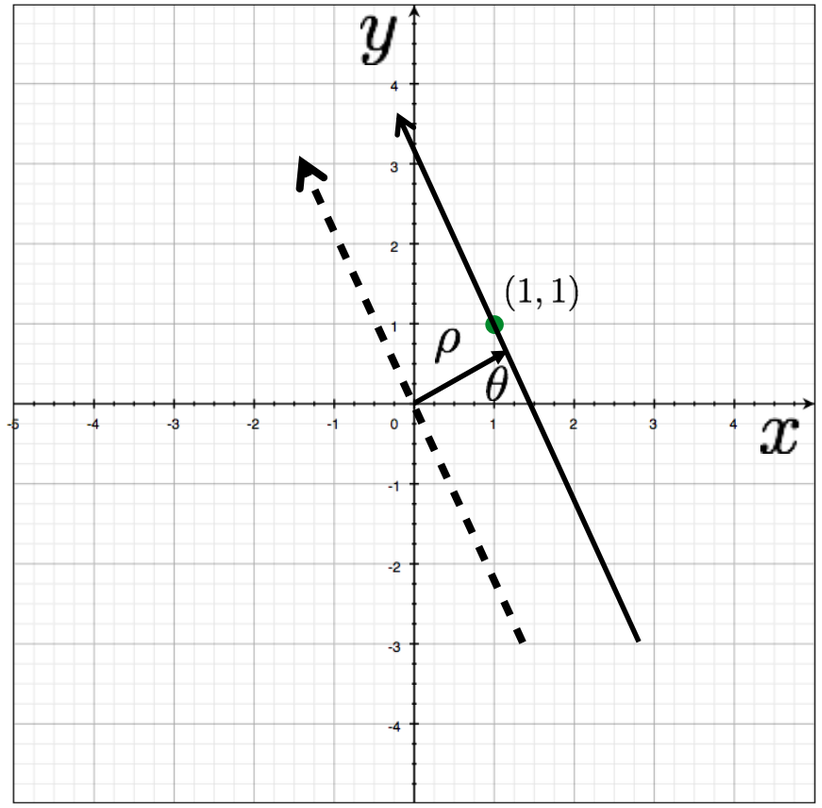


Parameter space

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:

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

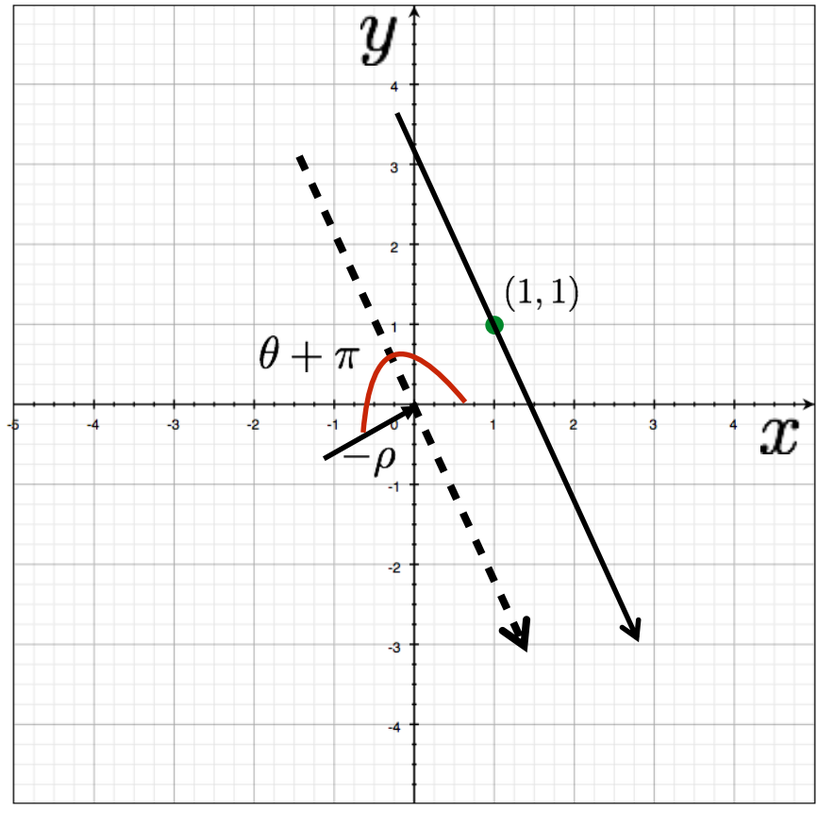


Image and parameter space

variables

$$y = mx + b$$

parameters

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

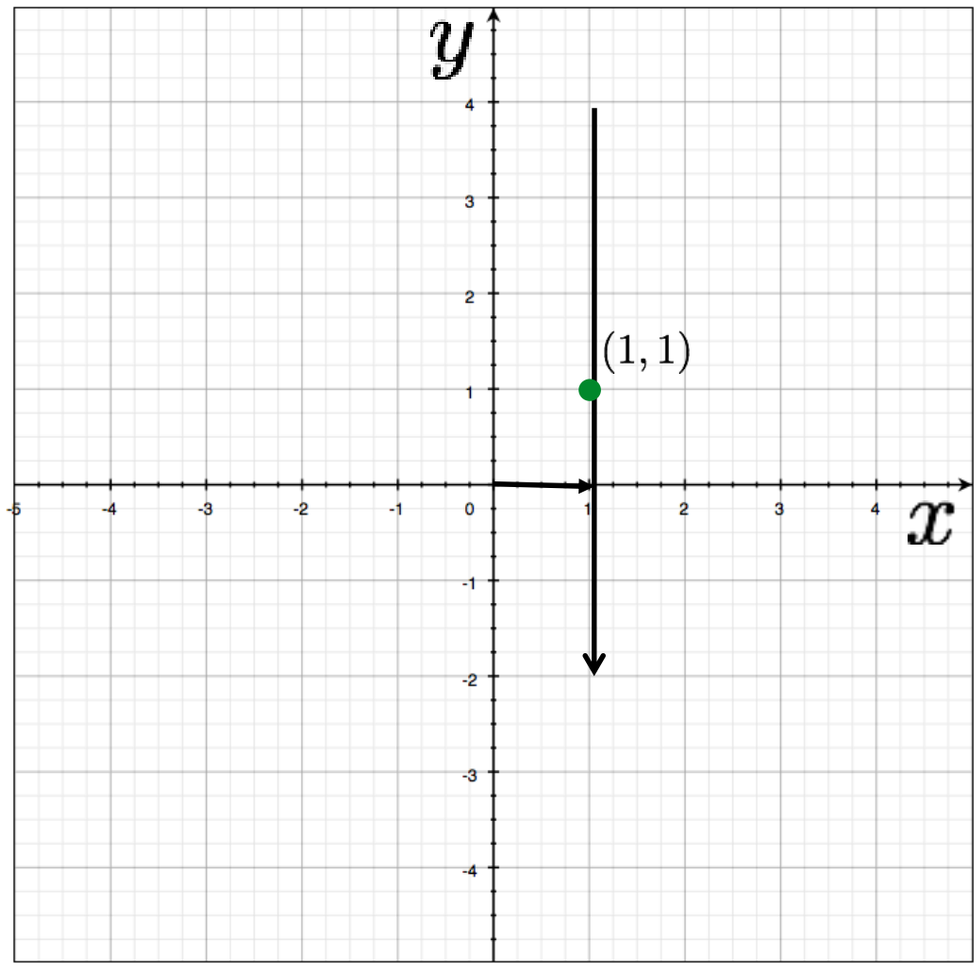
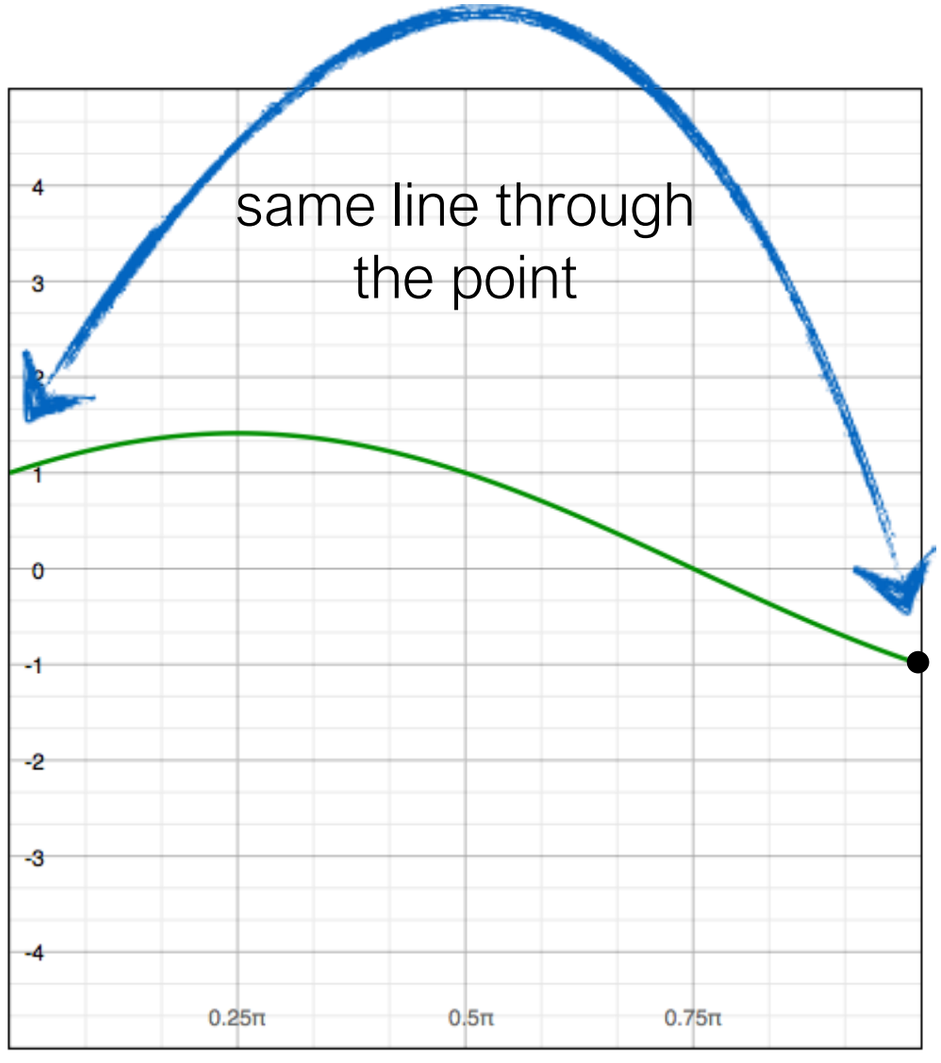


Image space

a line becomes a point



Parameter space

Image and parameter space

variables

$$y = mx + b$$

parameters

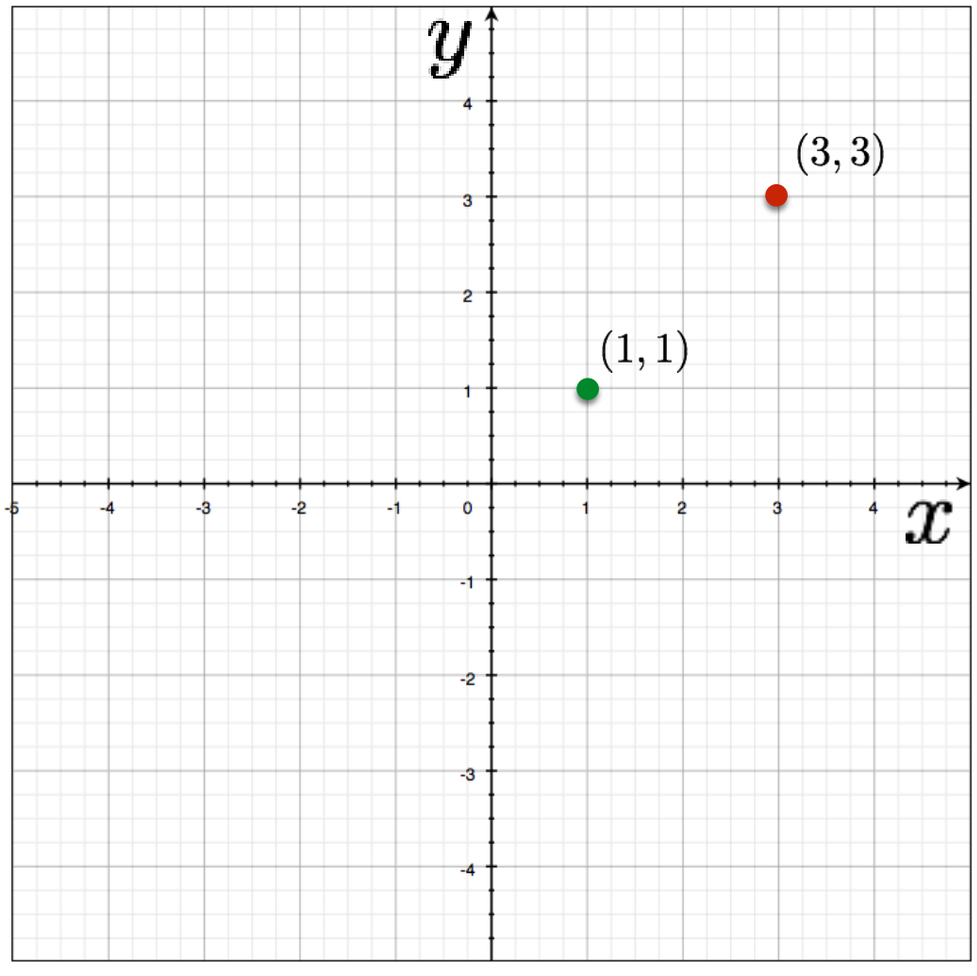
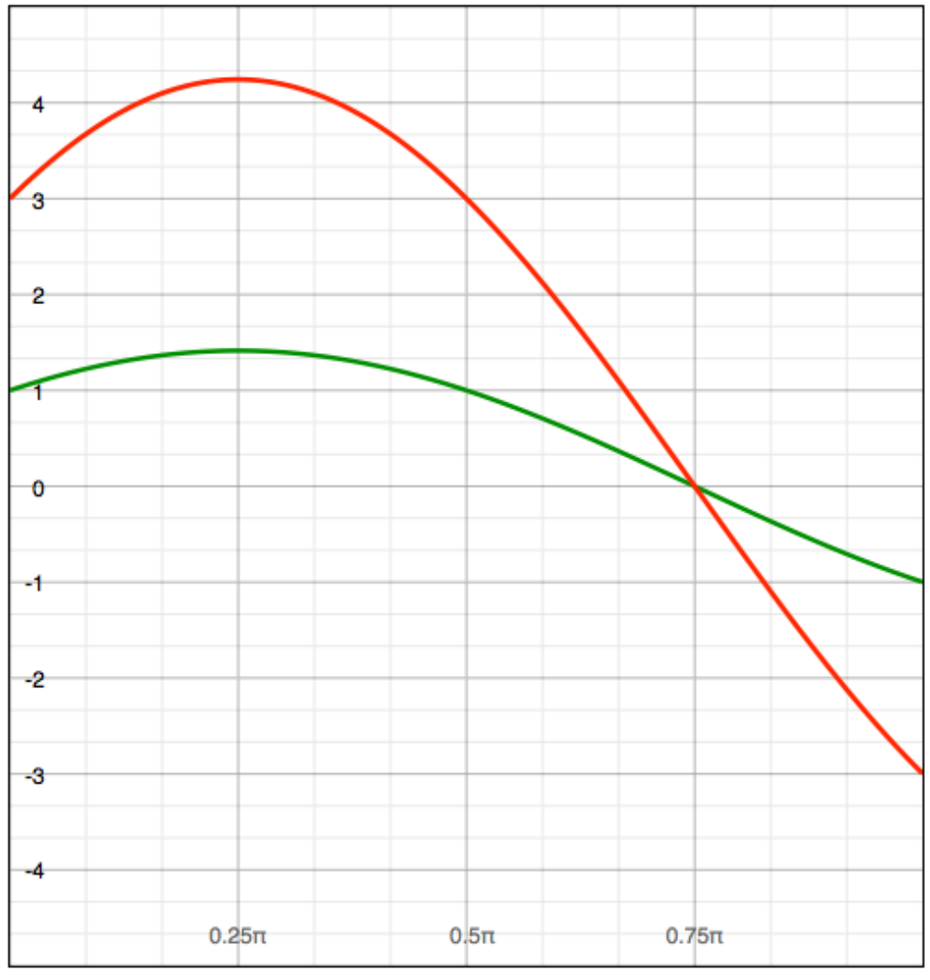


Image space

two points become ?



Parameter space

Image and parameter space

variables

$$y = mx + b$$

parameters

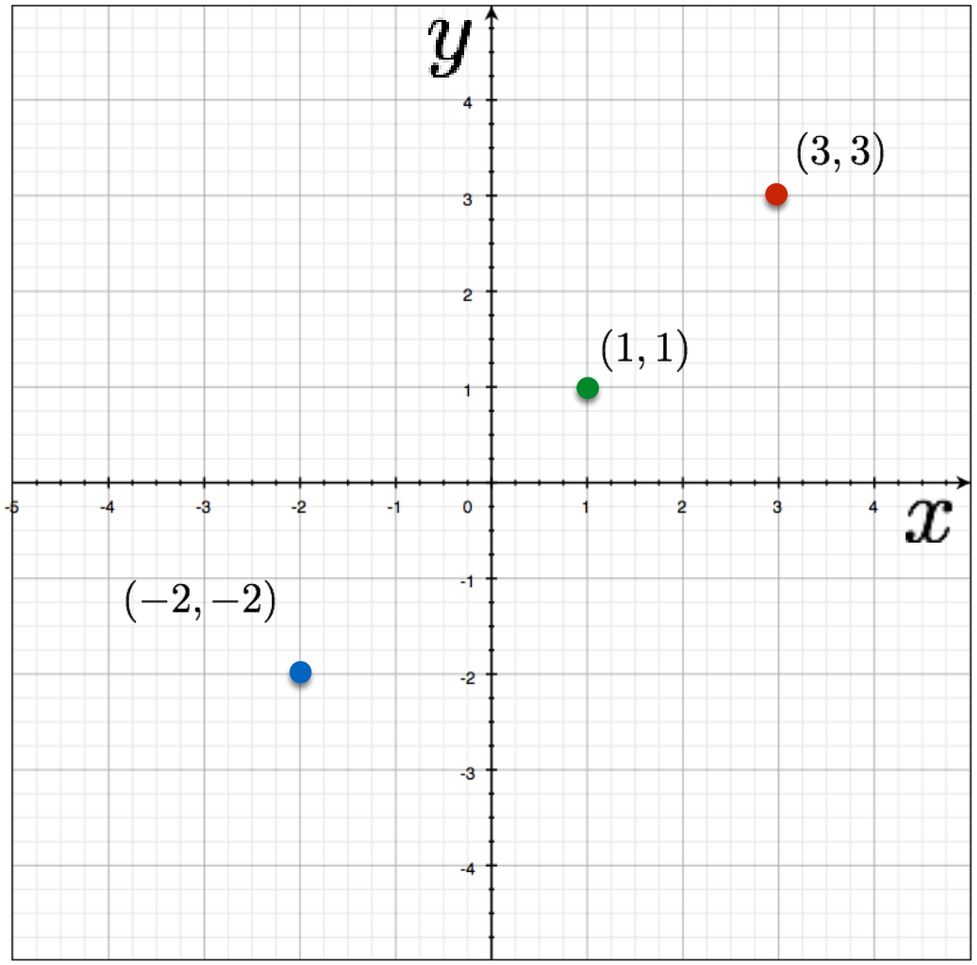
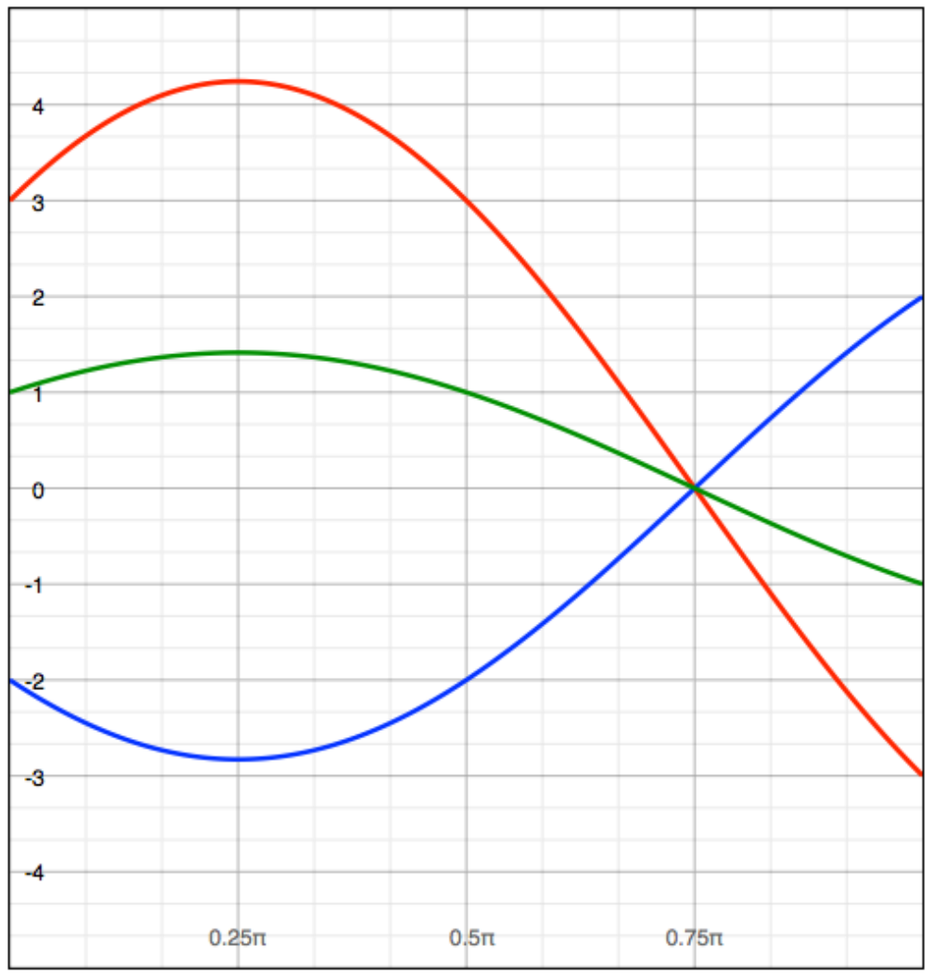


Image space

three points become ?



Parameter space

Image and parameter space

variables

$$y = mx + b$$

parameters

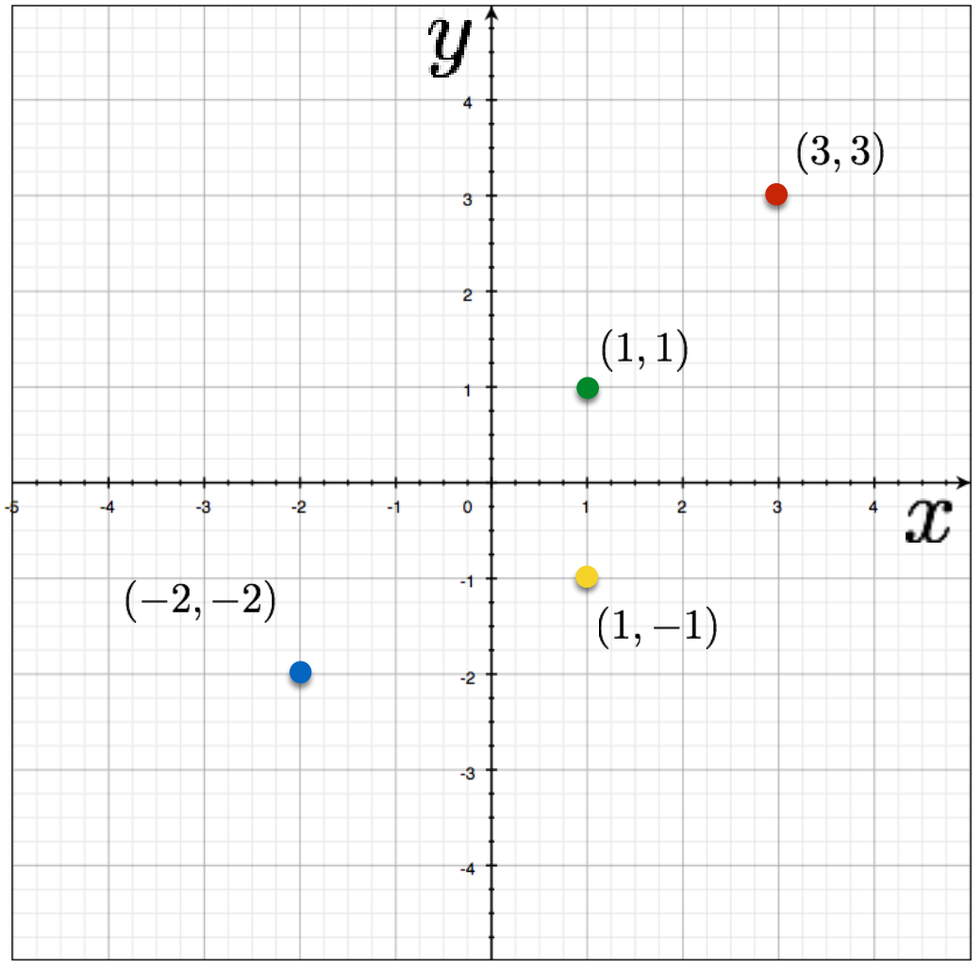
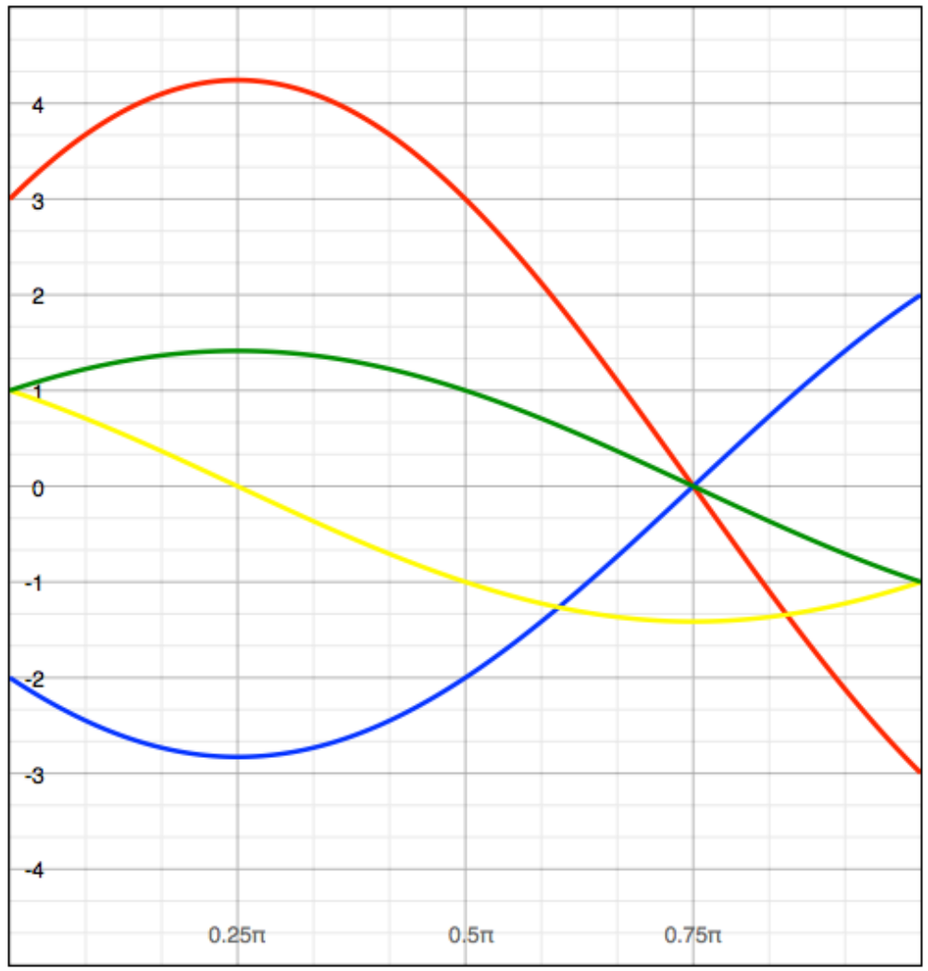


Image space

four points become ?



Parameter space

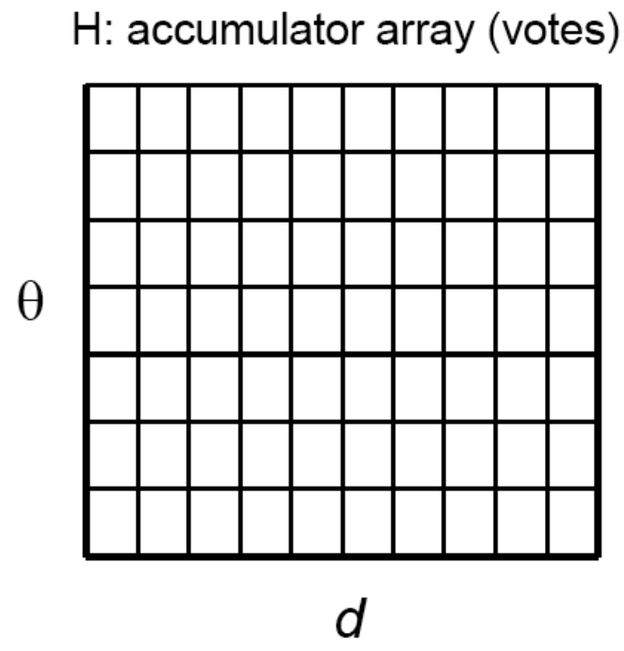
Implementation

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!

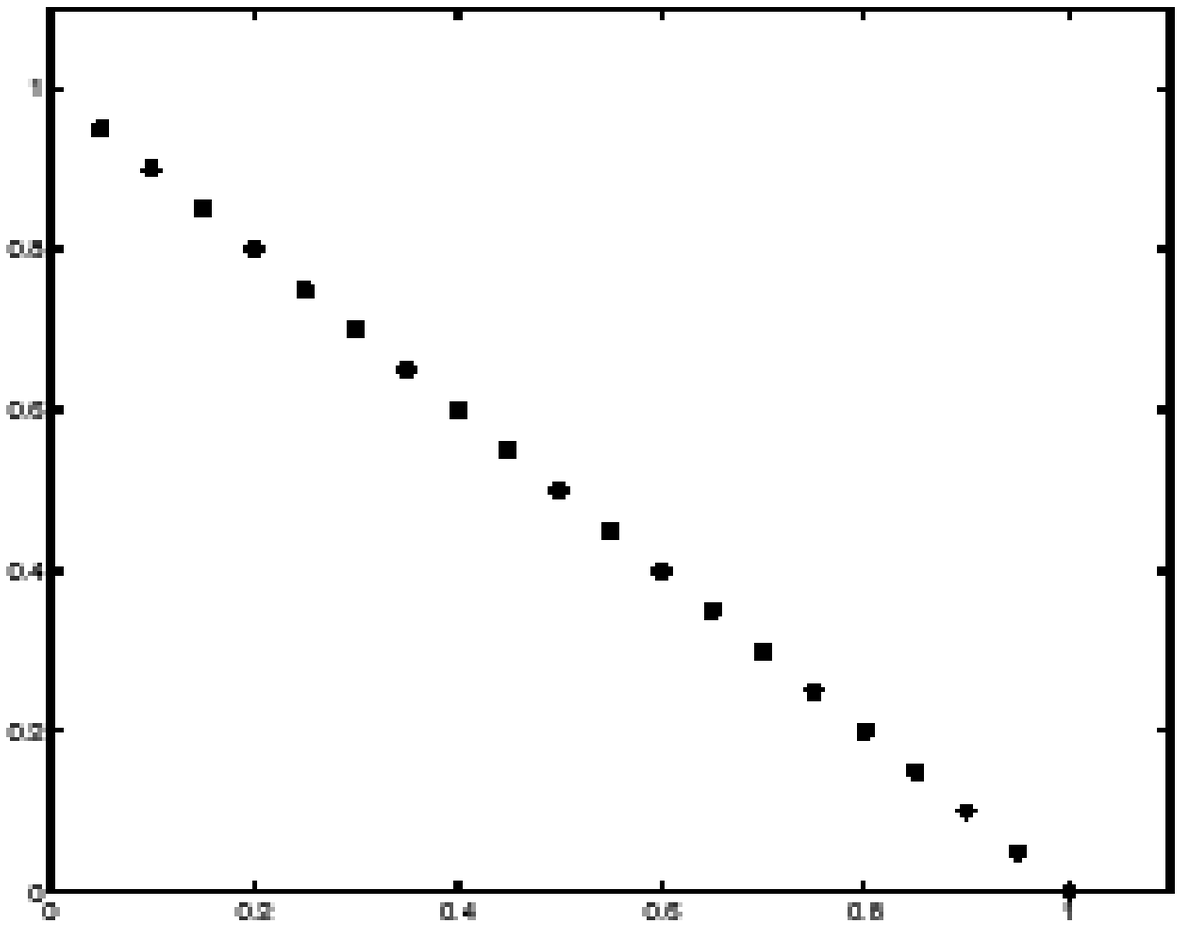
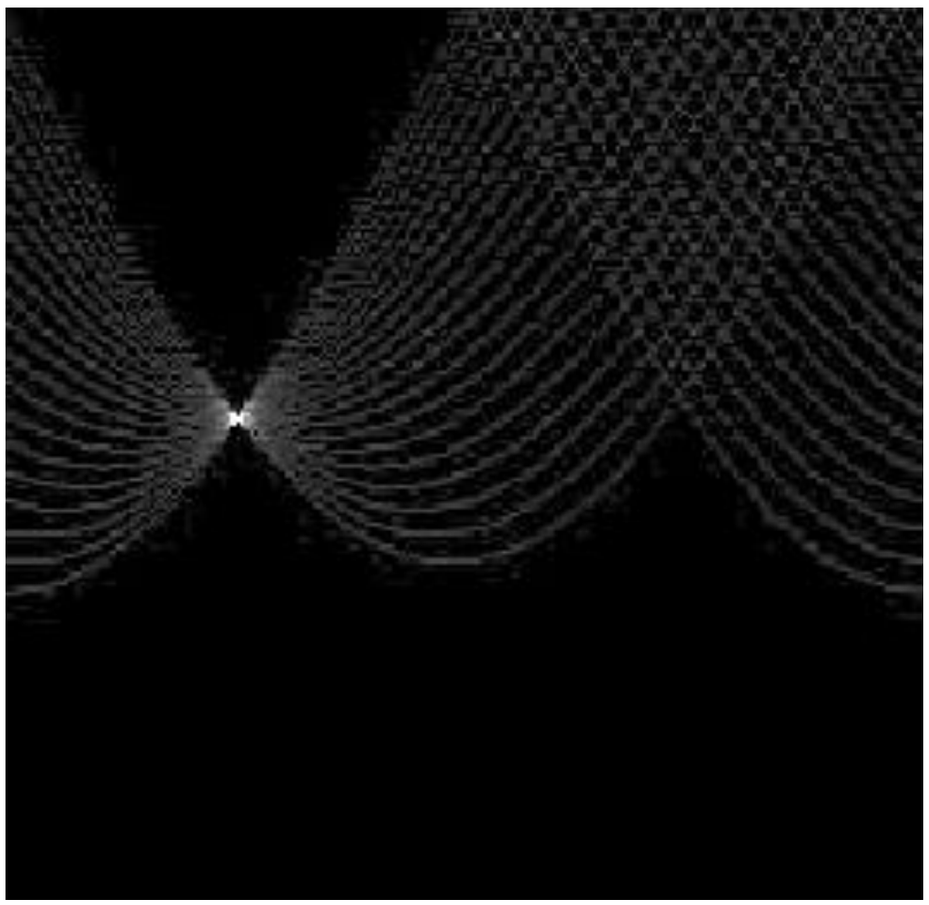


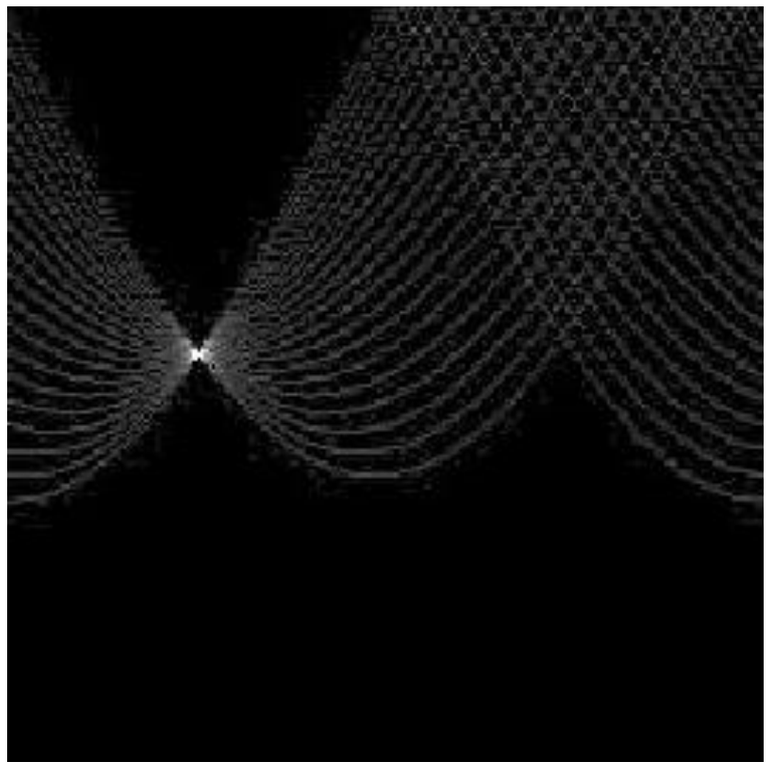
Image space

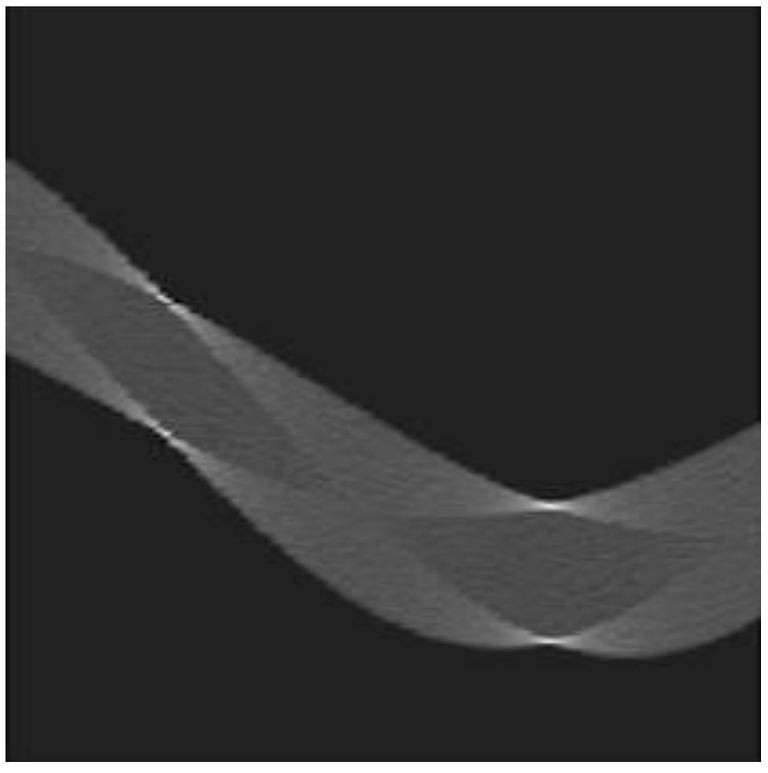


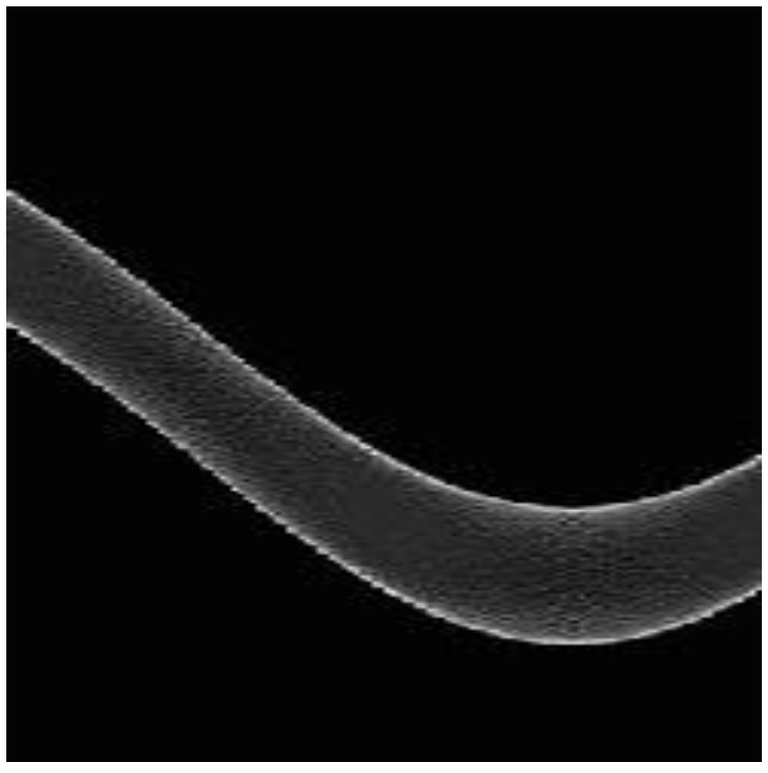
Votes

Basic shapes

(in parameter space)



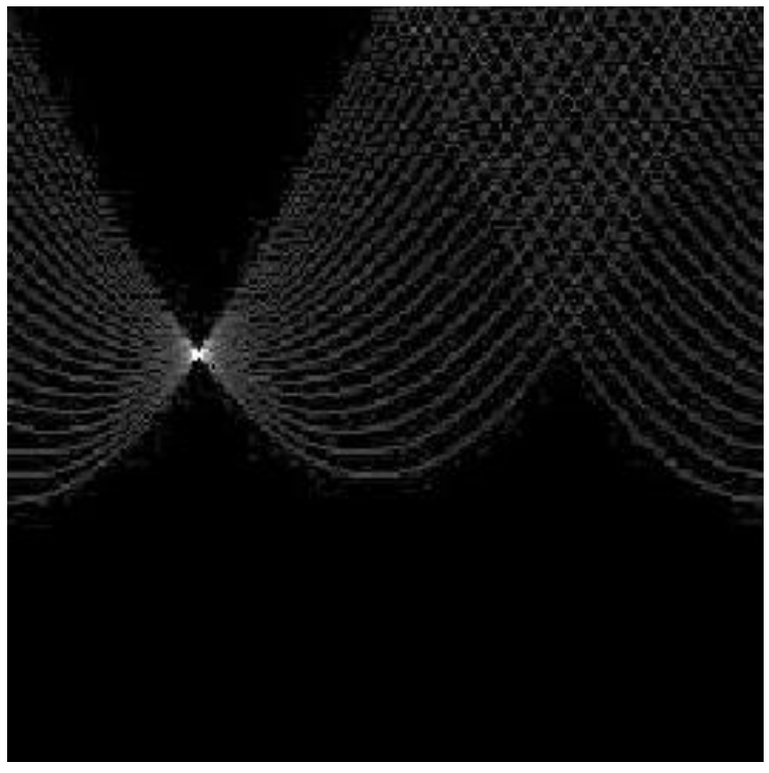




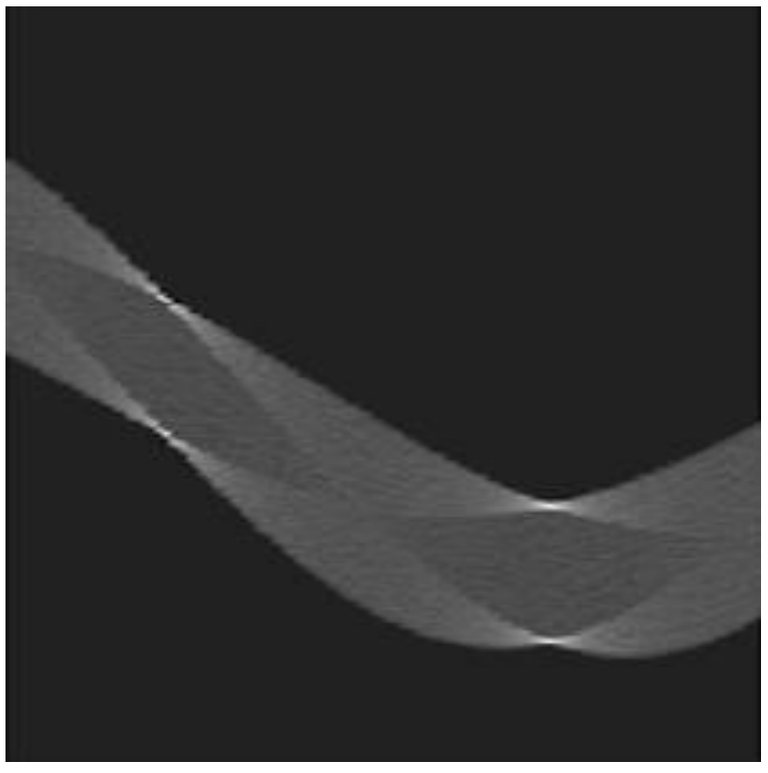
can you guess the shape?

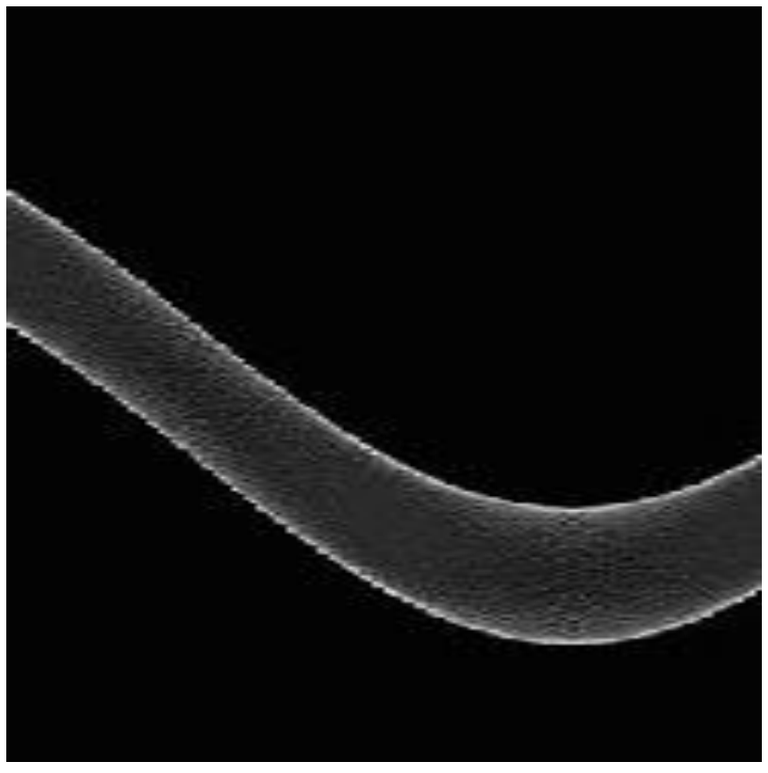
Basic shapes

(in parameter space)



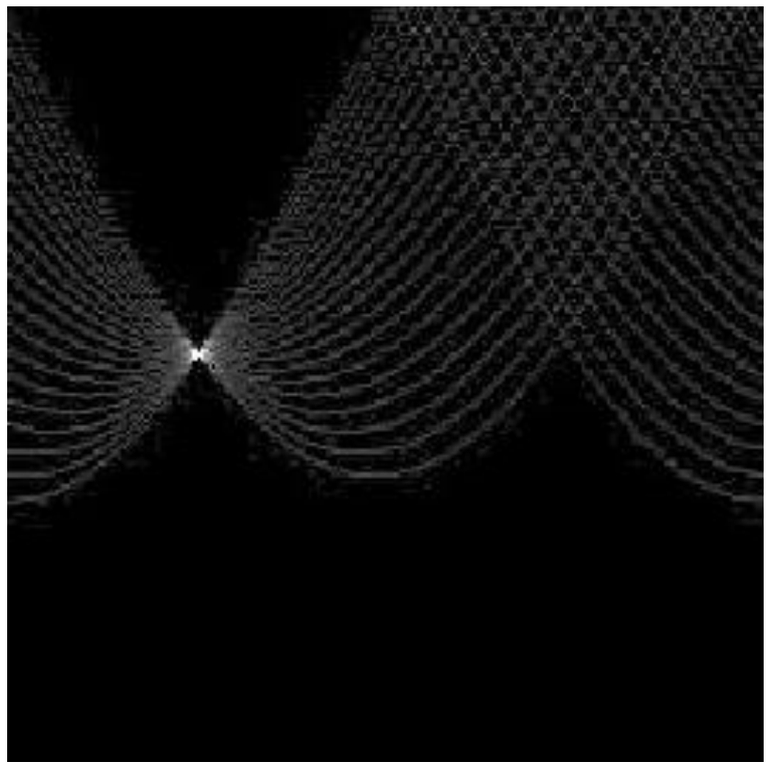
line



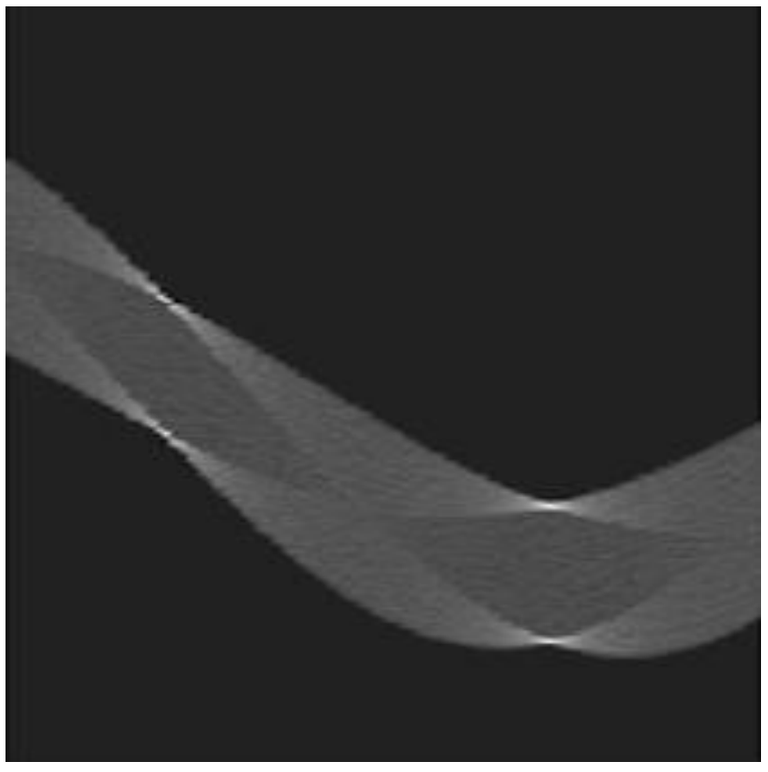


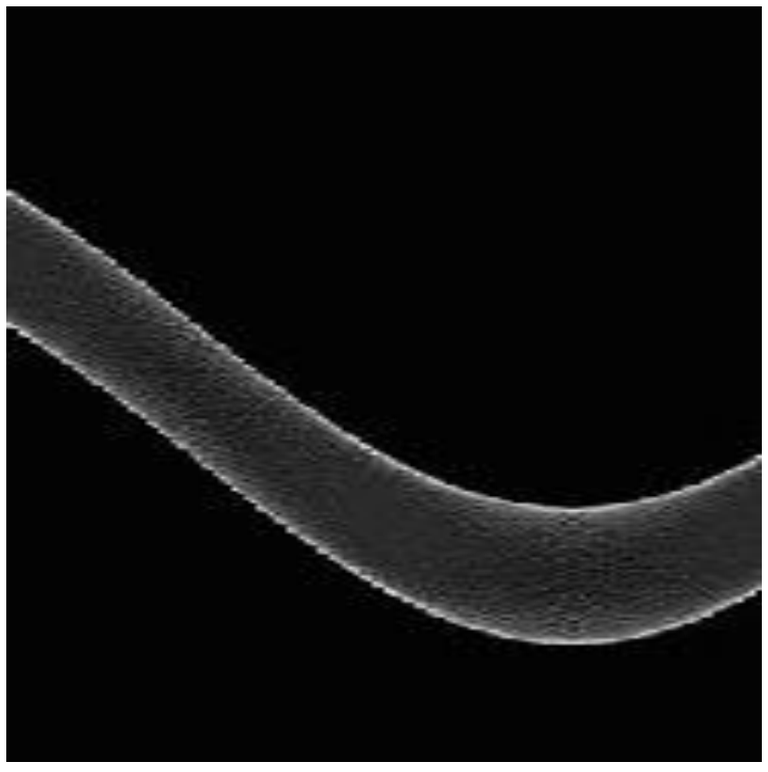
Basic shapes

(in parameter space)



line

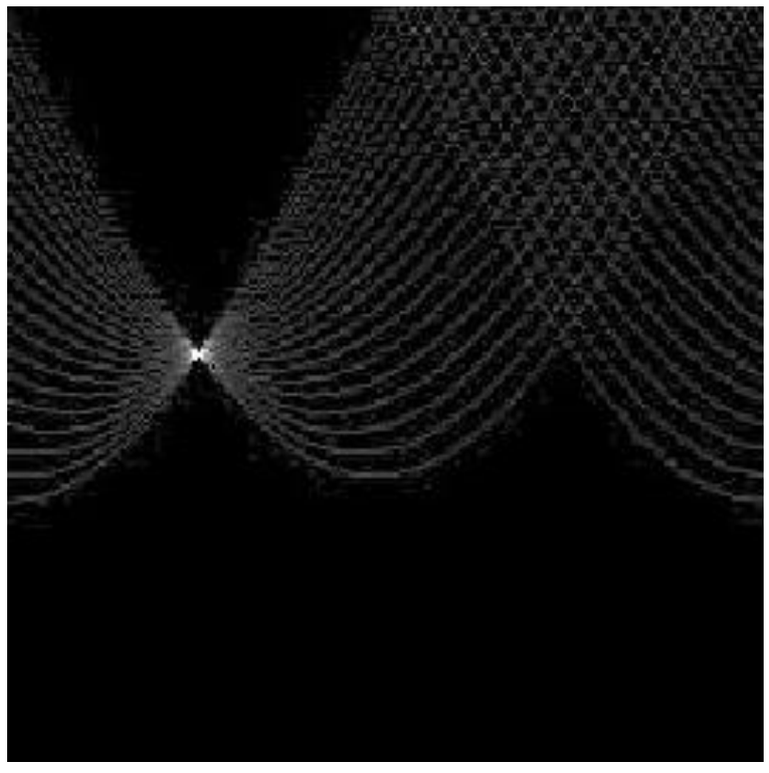




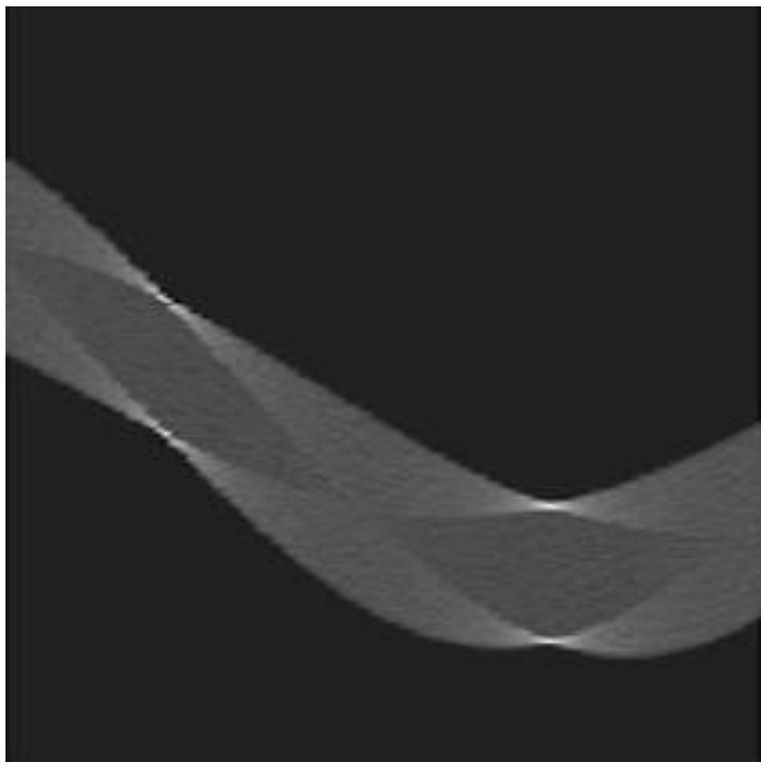
rectangle

Basic shapes

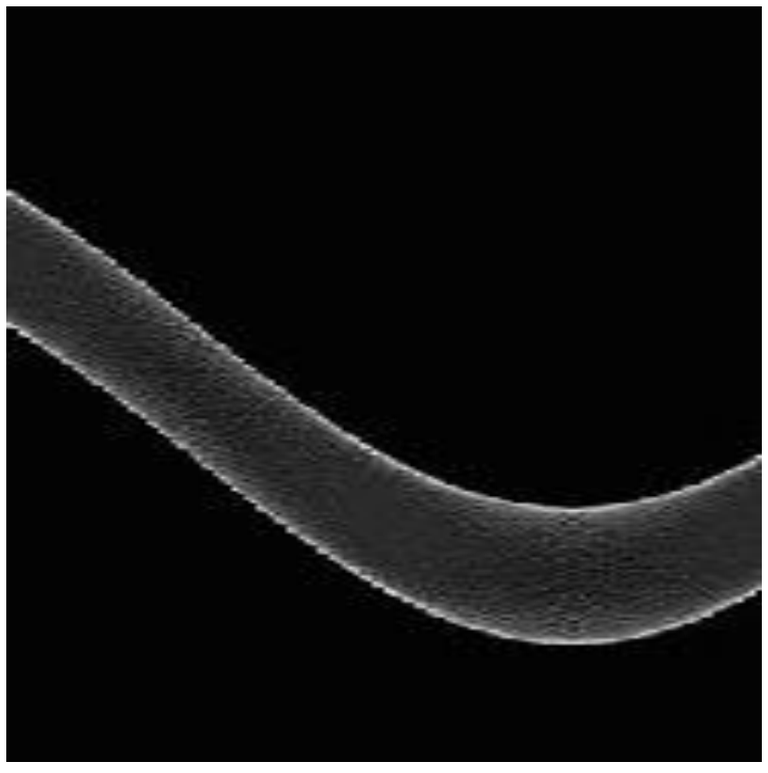
(in parameter space)



line

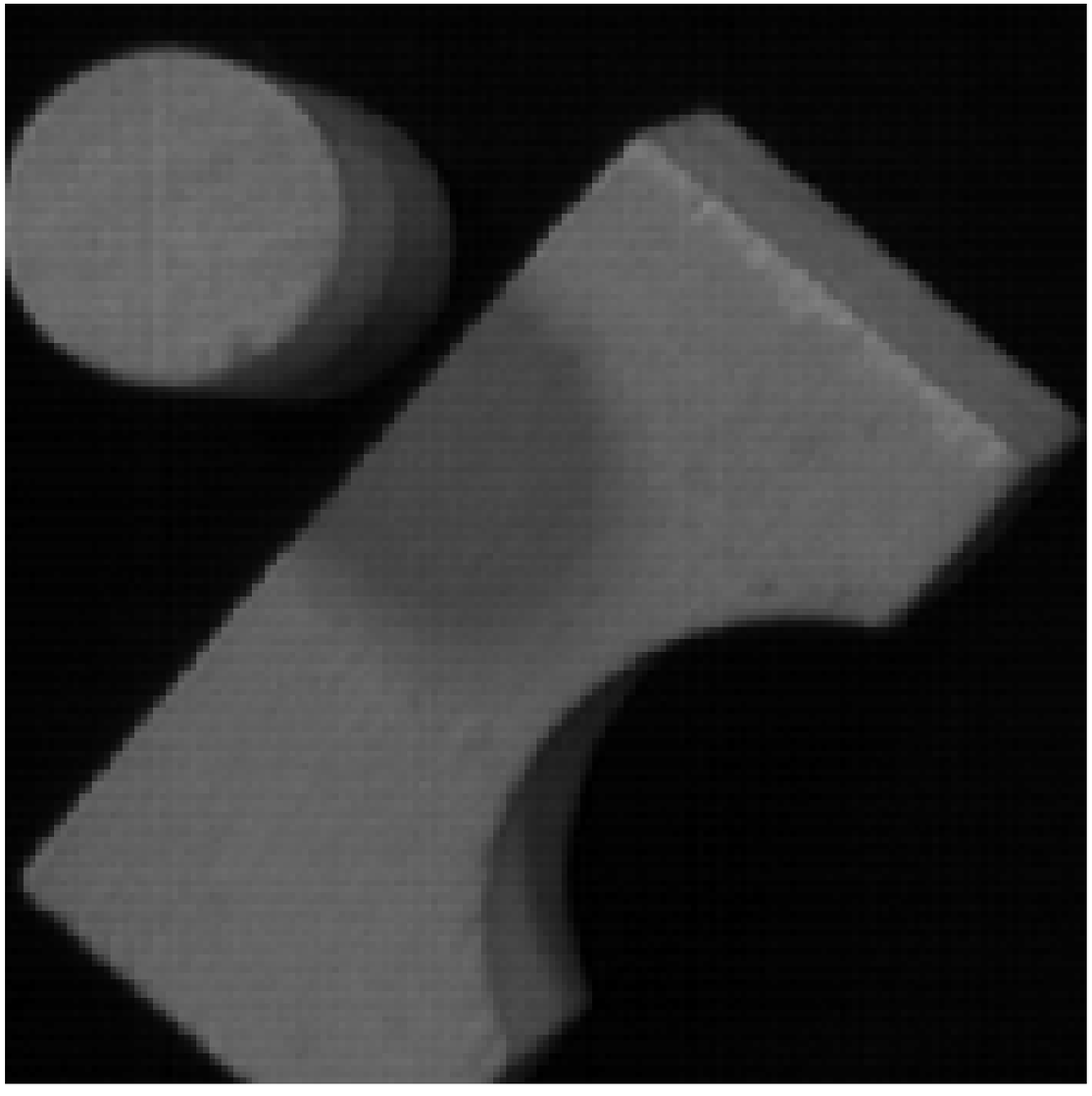


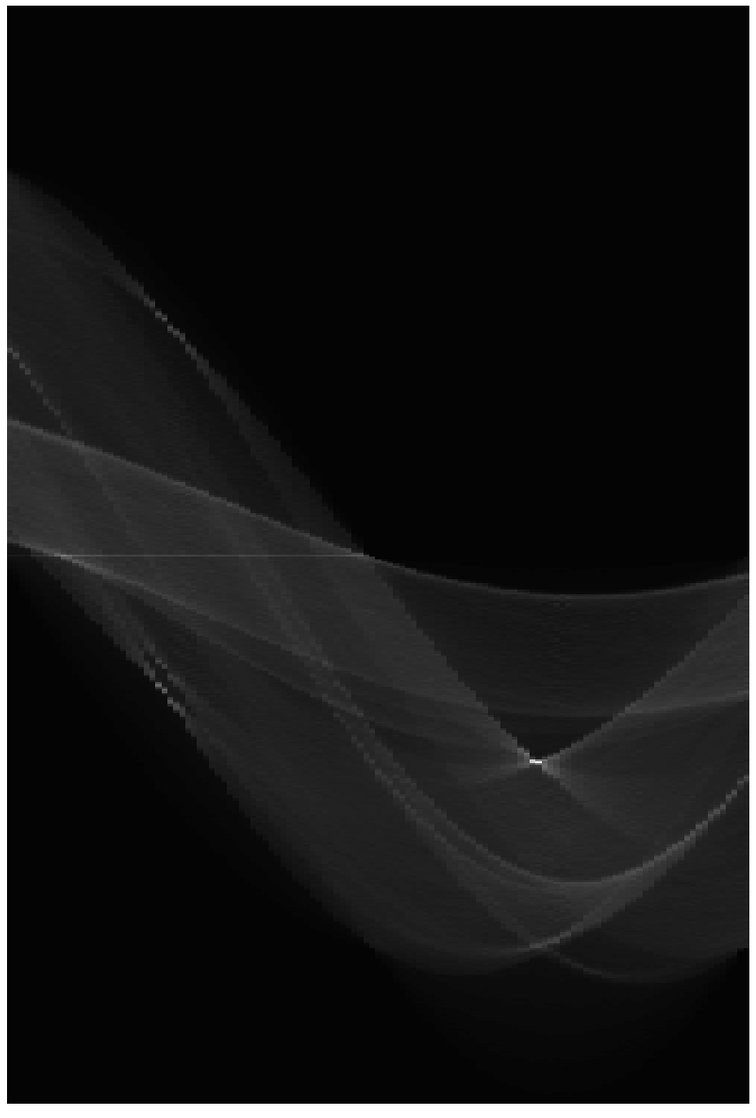
rectangle



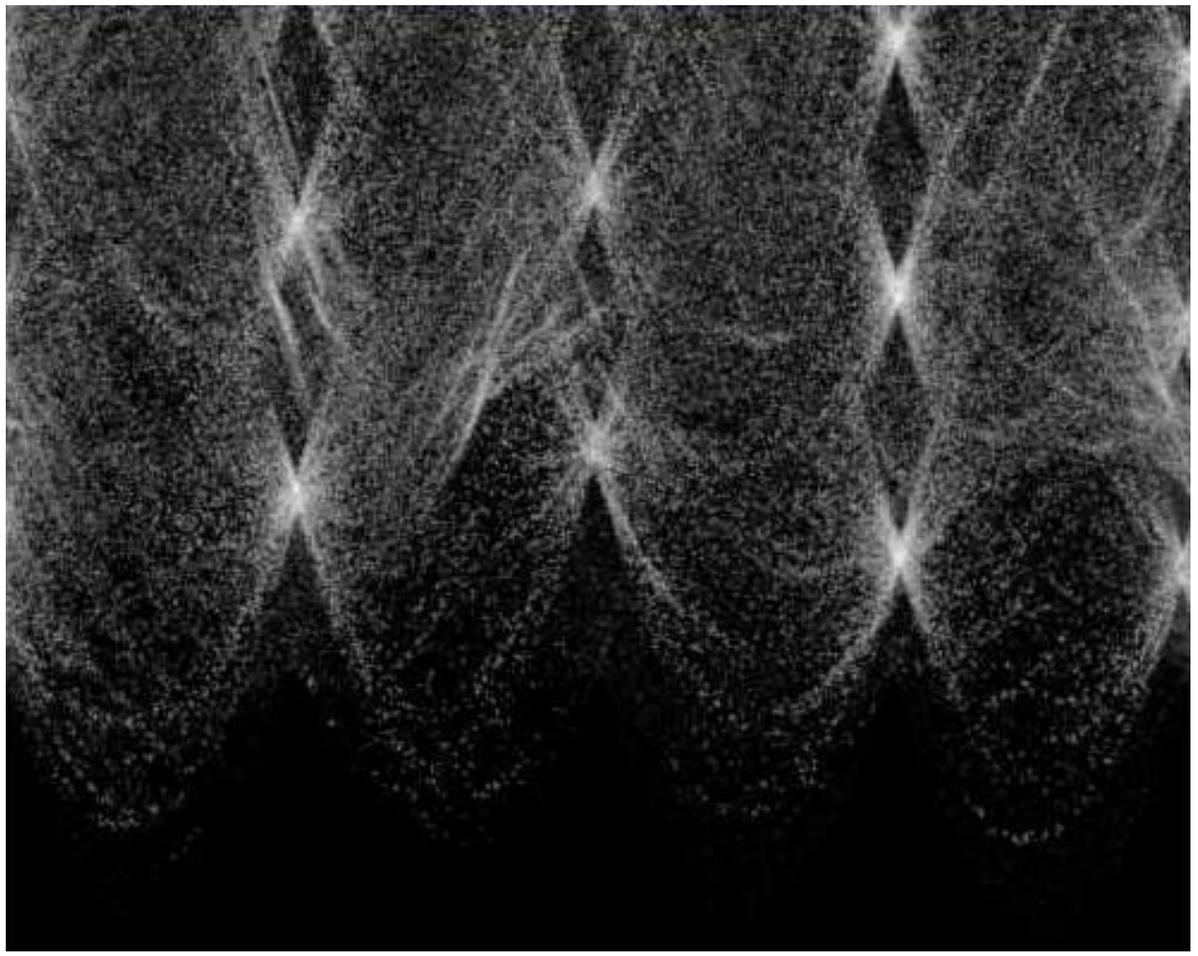
circle

Basic Shapes





More complex image



In practice, measurements are noisy…

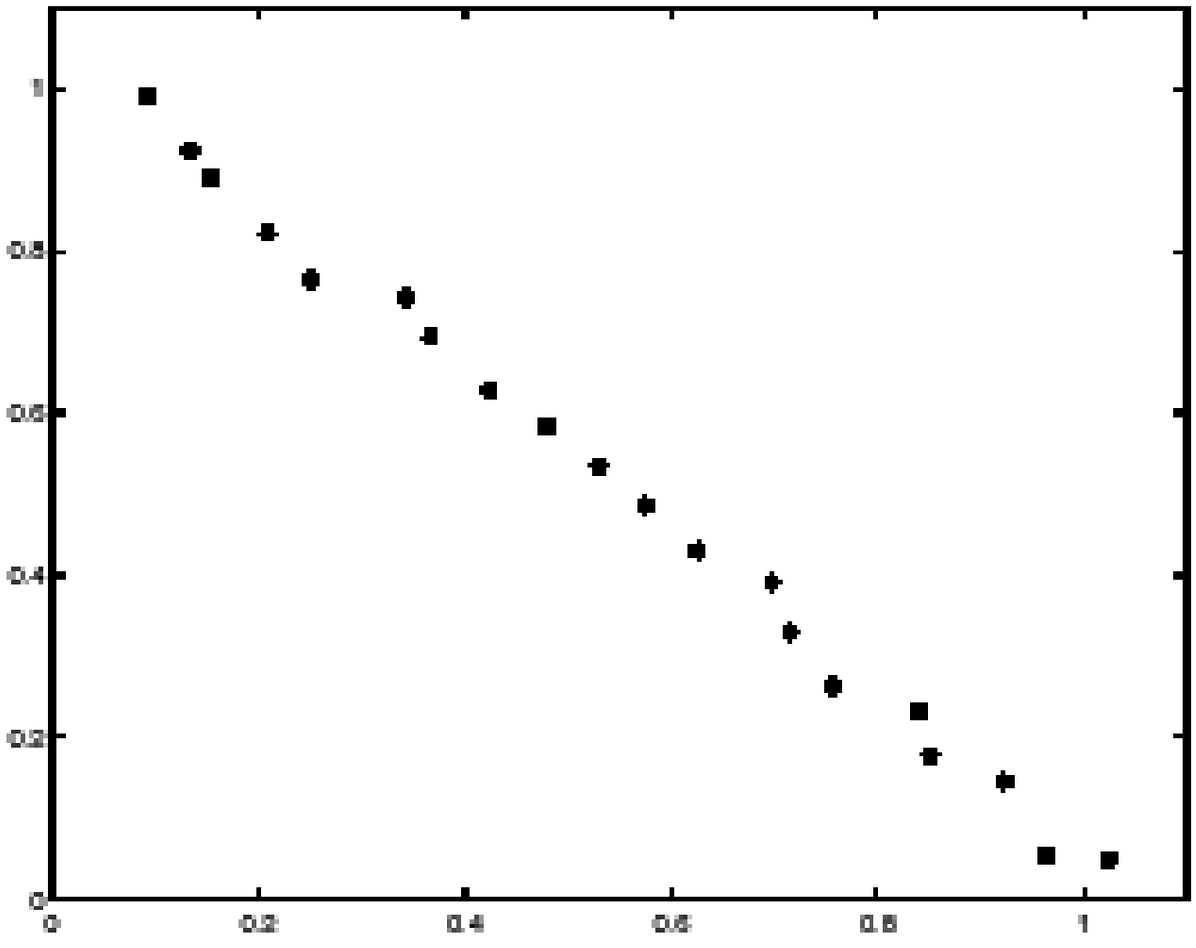
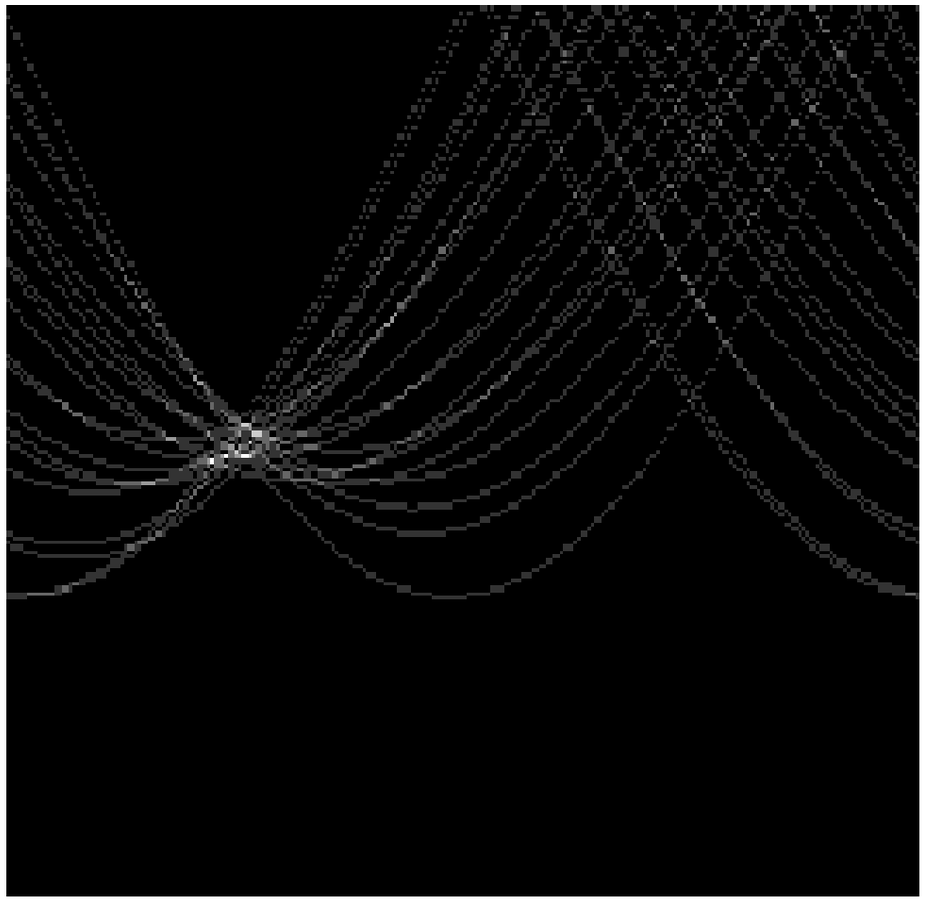


Image space



Votes

Too much noise …

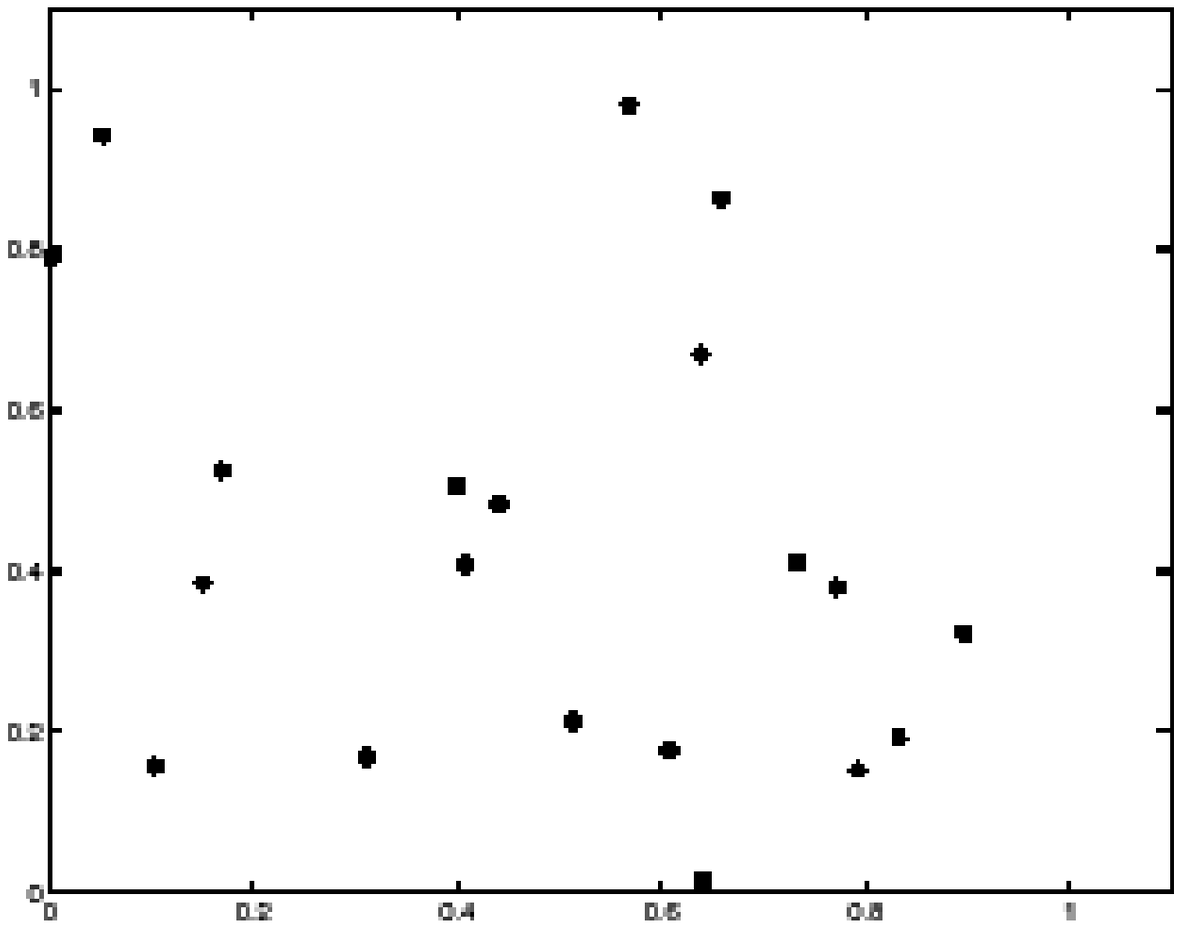
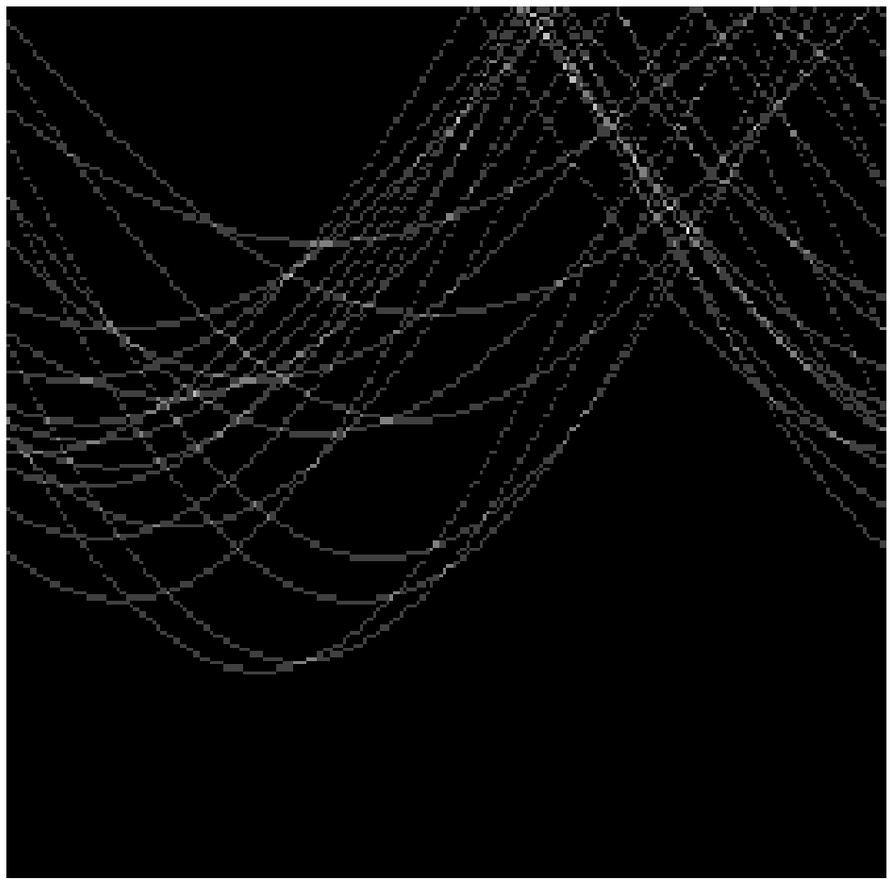
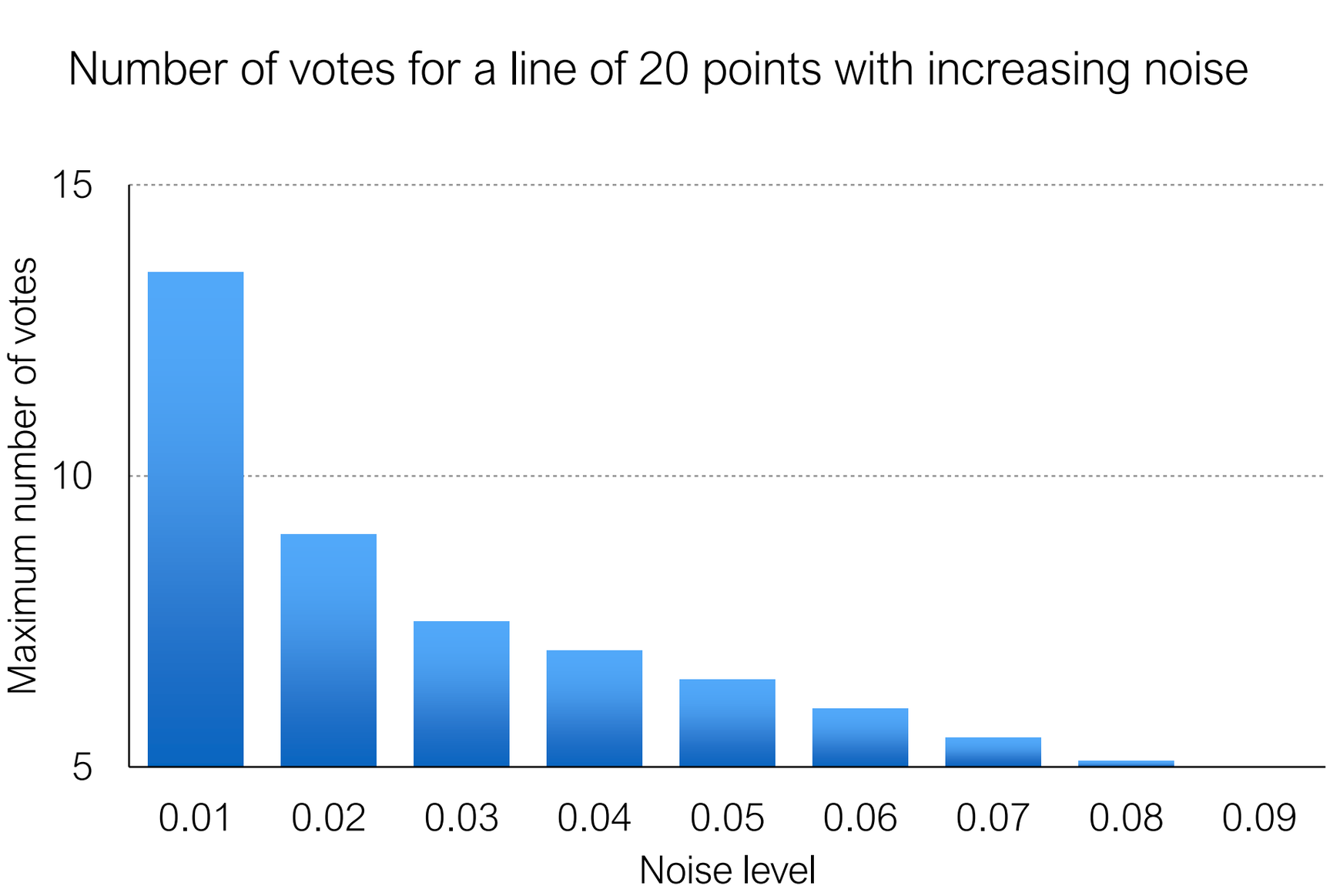


Image space



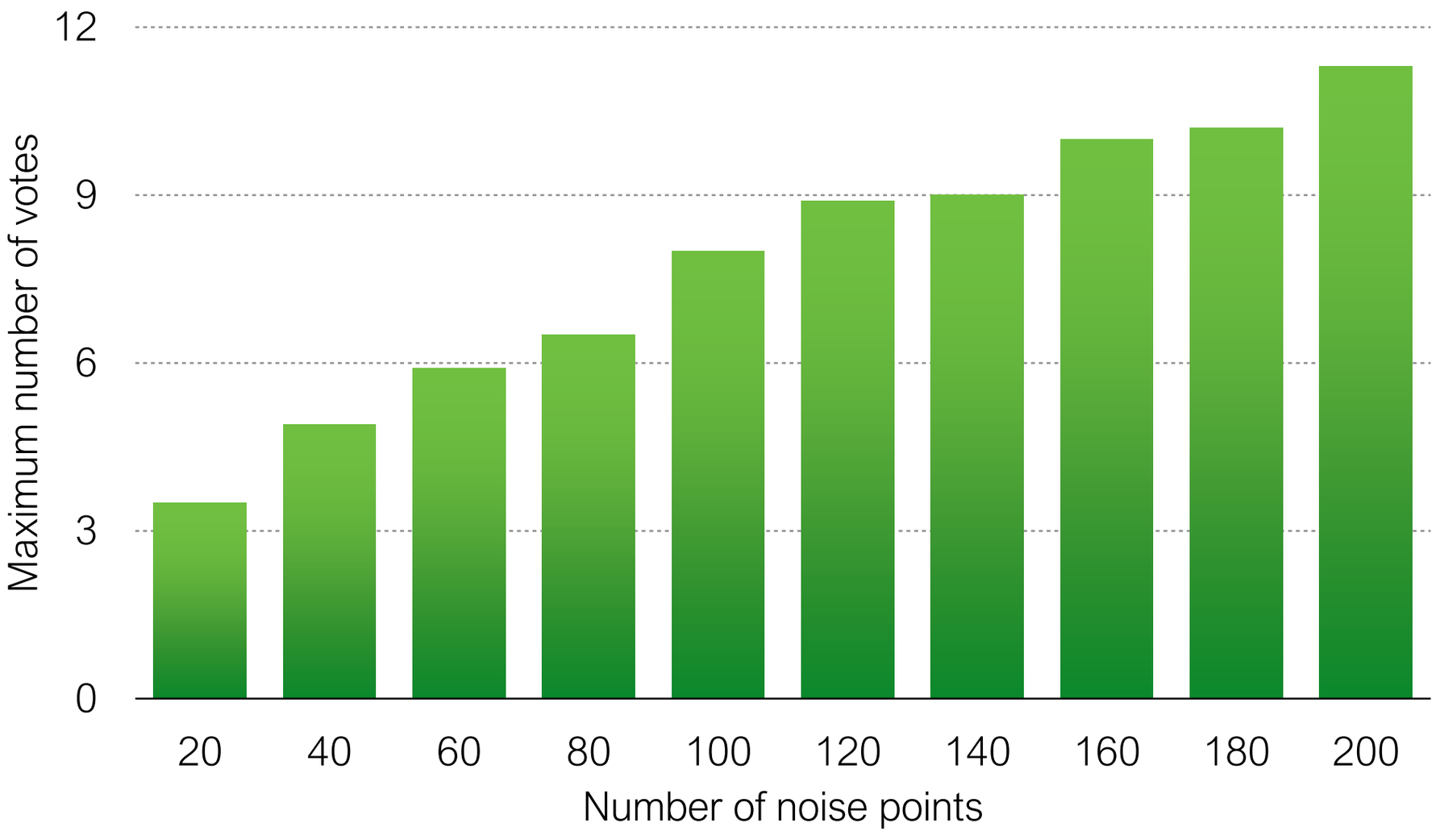
Votes

Effects of noise level



More noise, fewer votes (in the right bin)

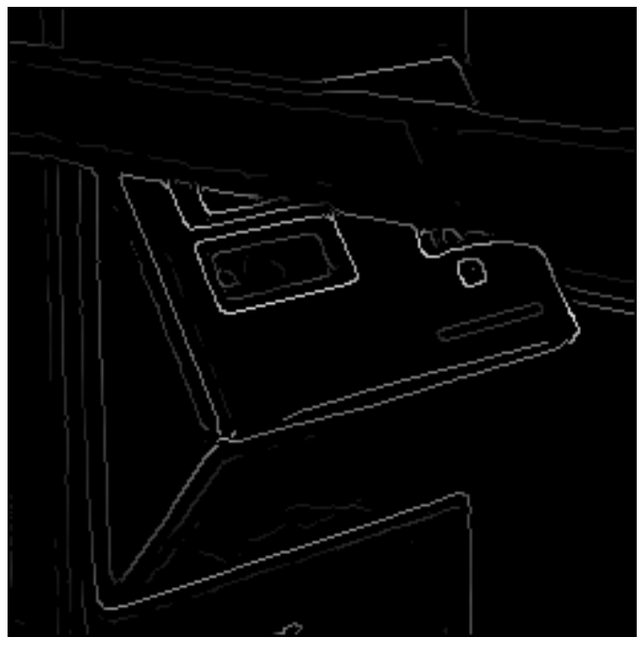
Effect of noise points



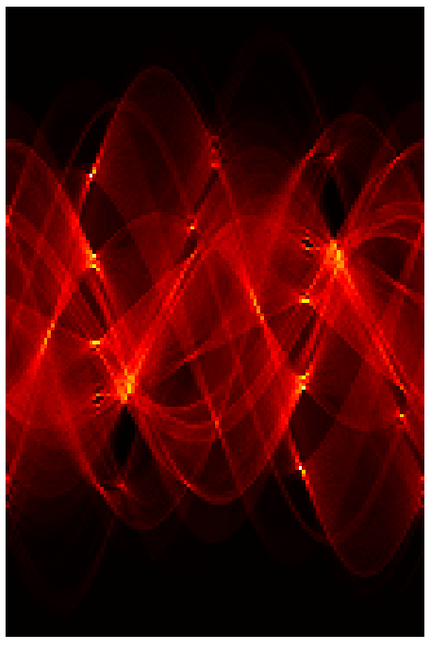
More noise, more votes (in the wrong bin)

Real-world example

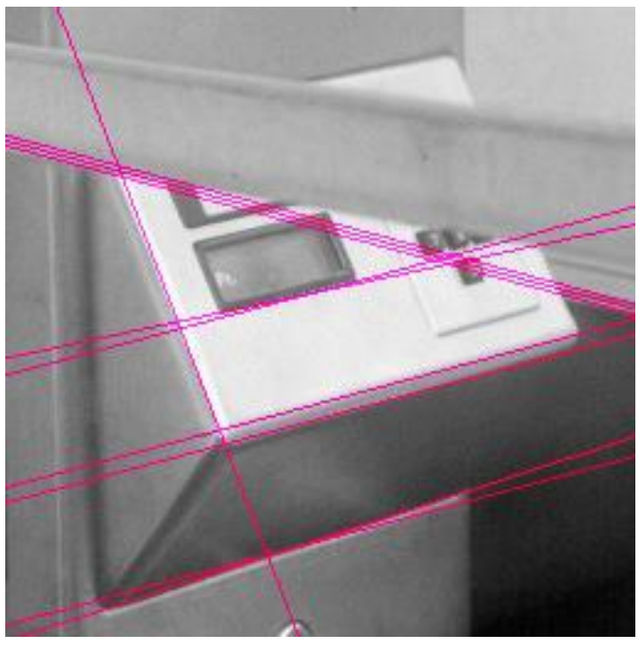
Original



Edges



parameter space



Hough Lines

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

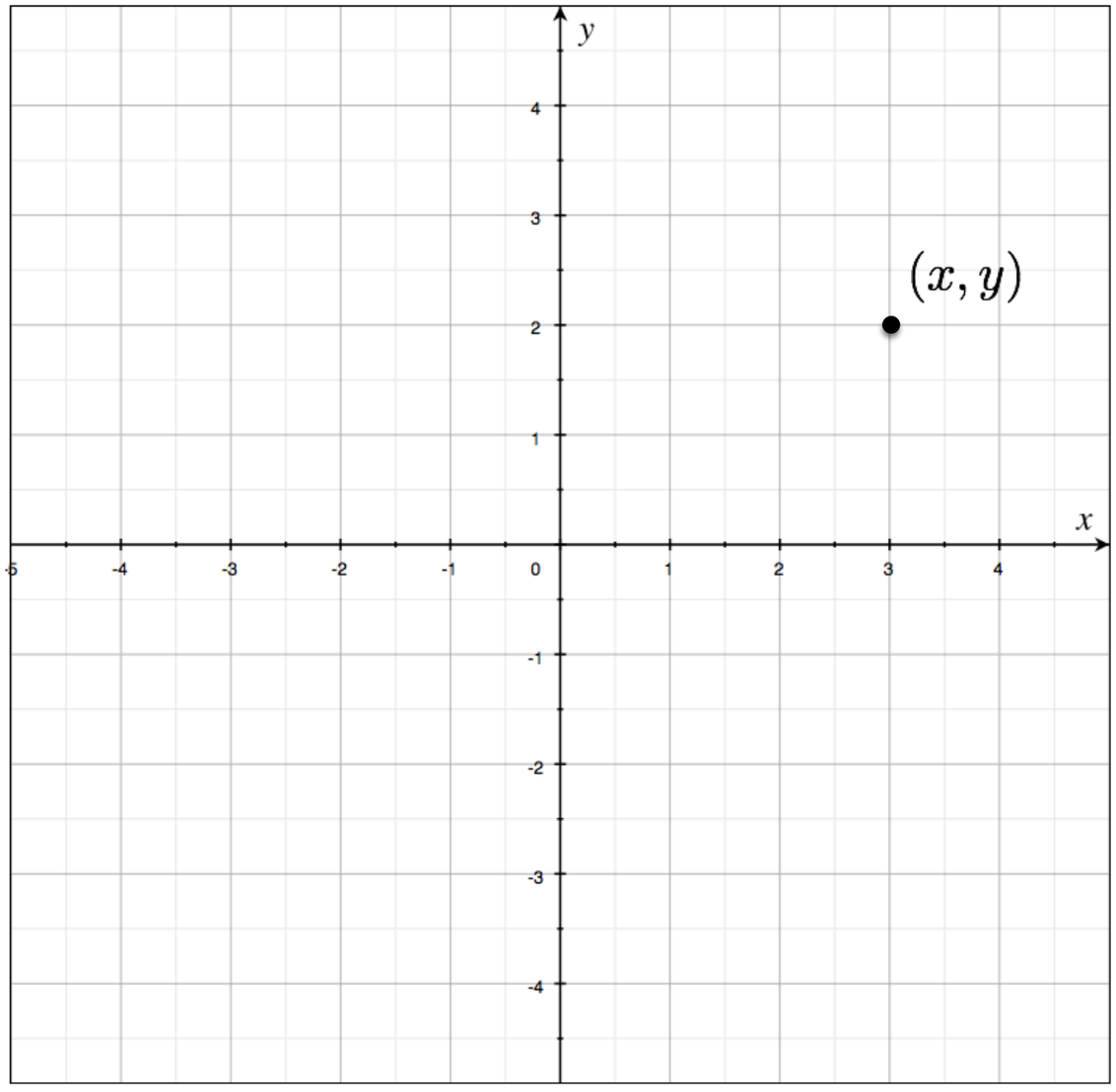
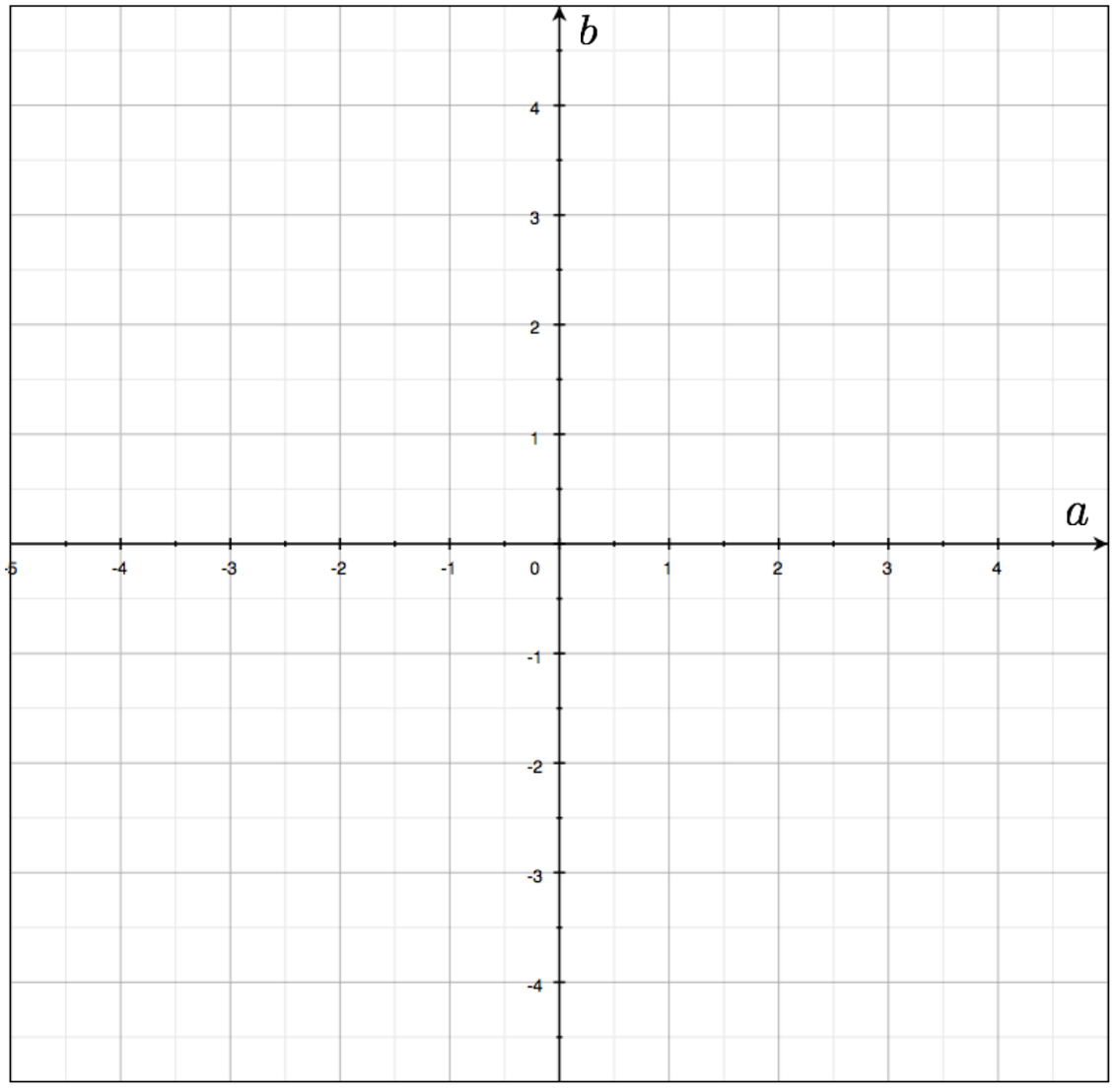


Image space

parameters

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

variables



Parameter space

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

parameters

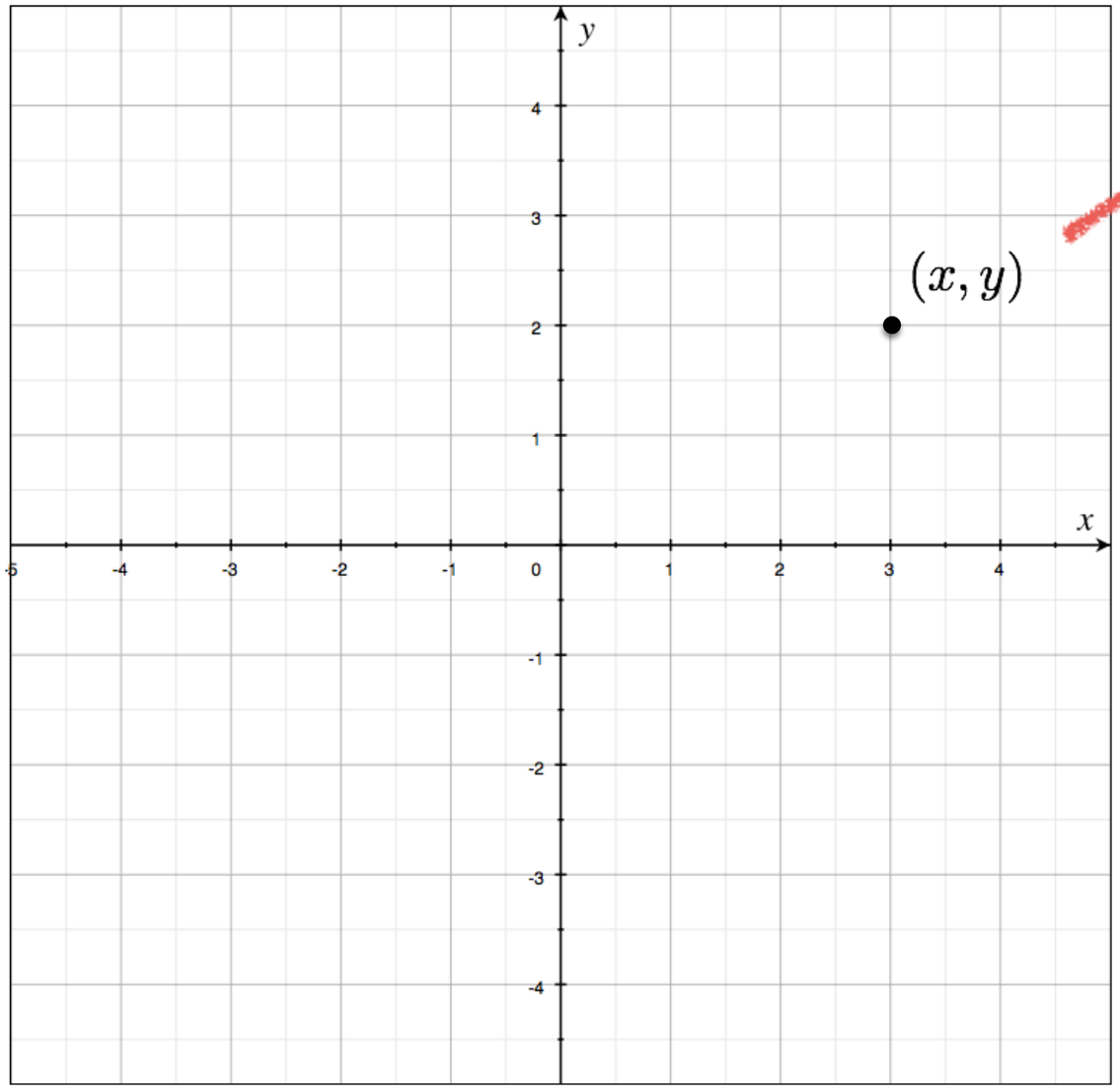
$$(x-a)^2+(y-b)^2=r^2$$

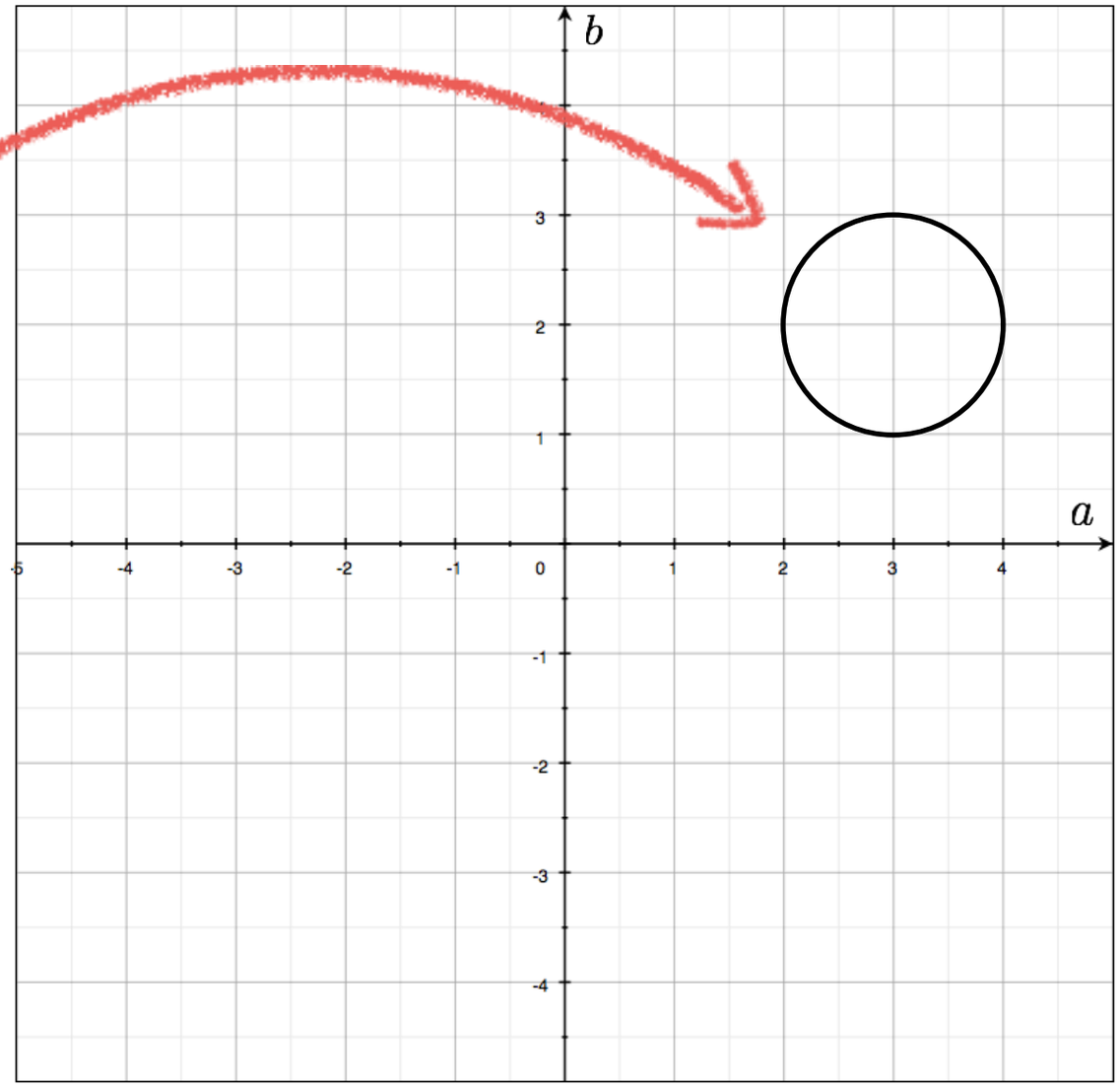
variables

parameters

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

variables

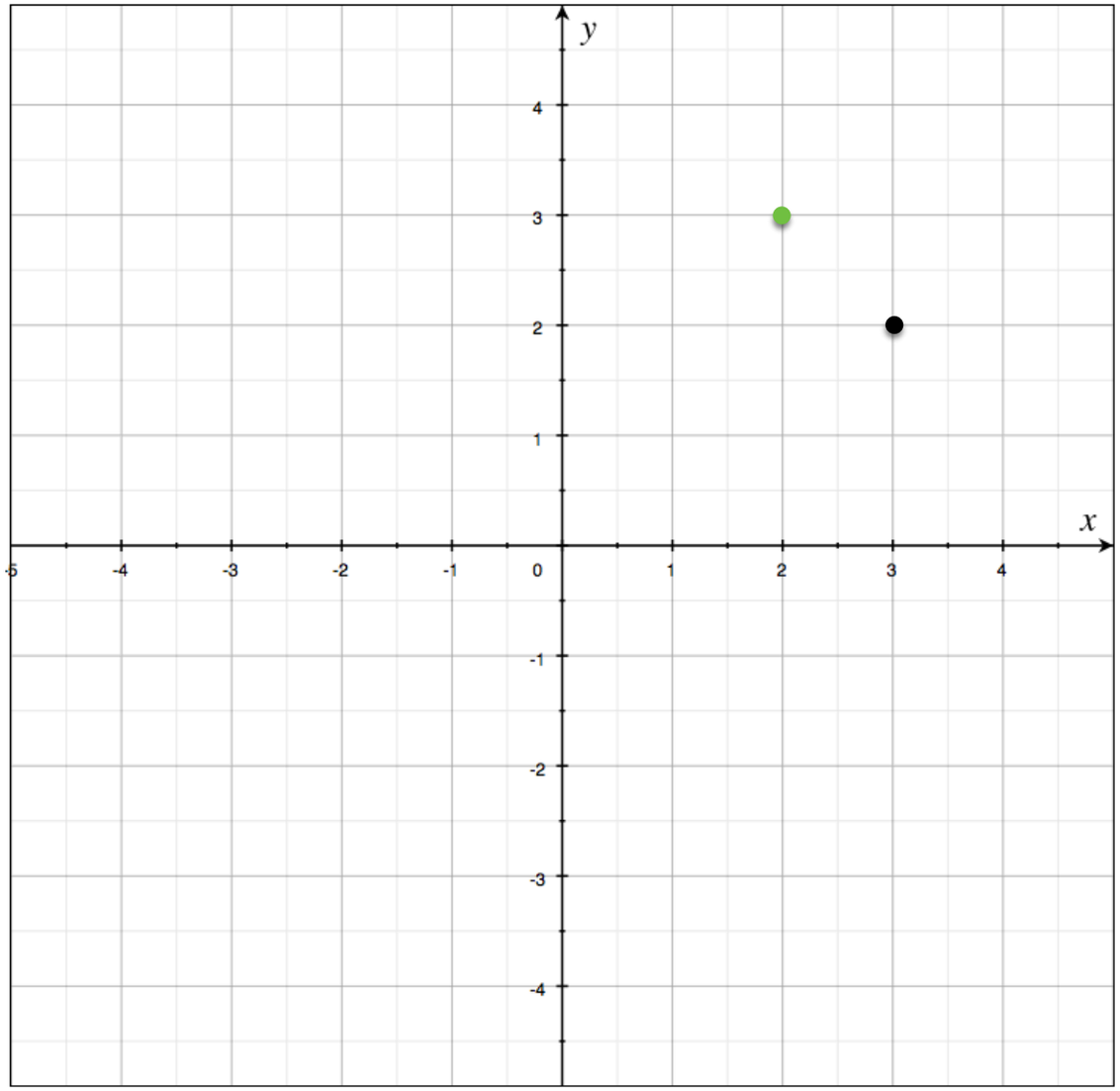




parameters

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

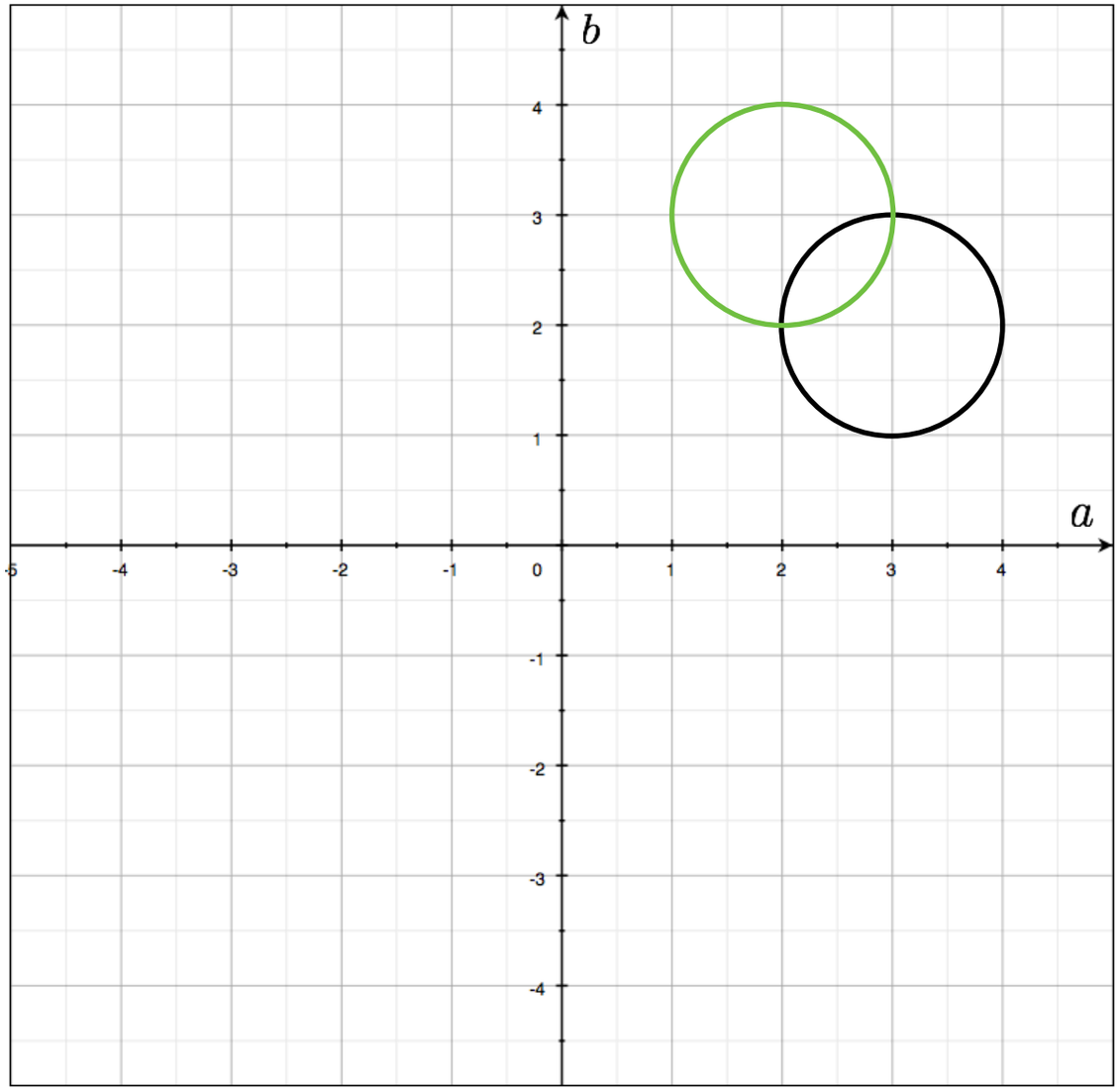
variables



parameters

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

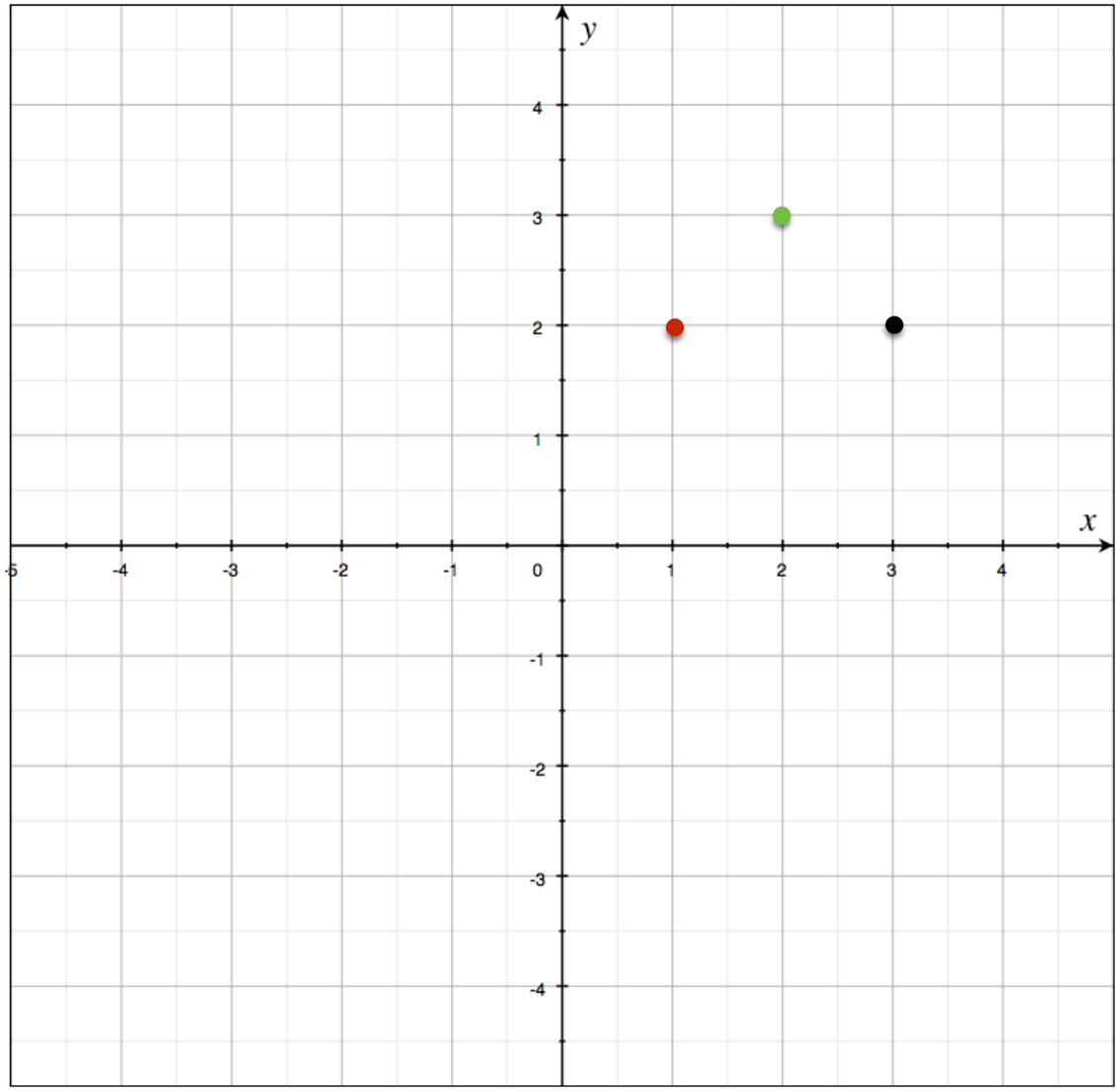
variables



parameters

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

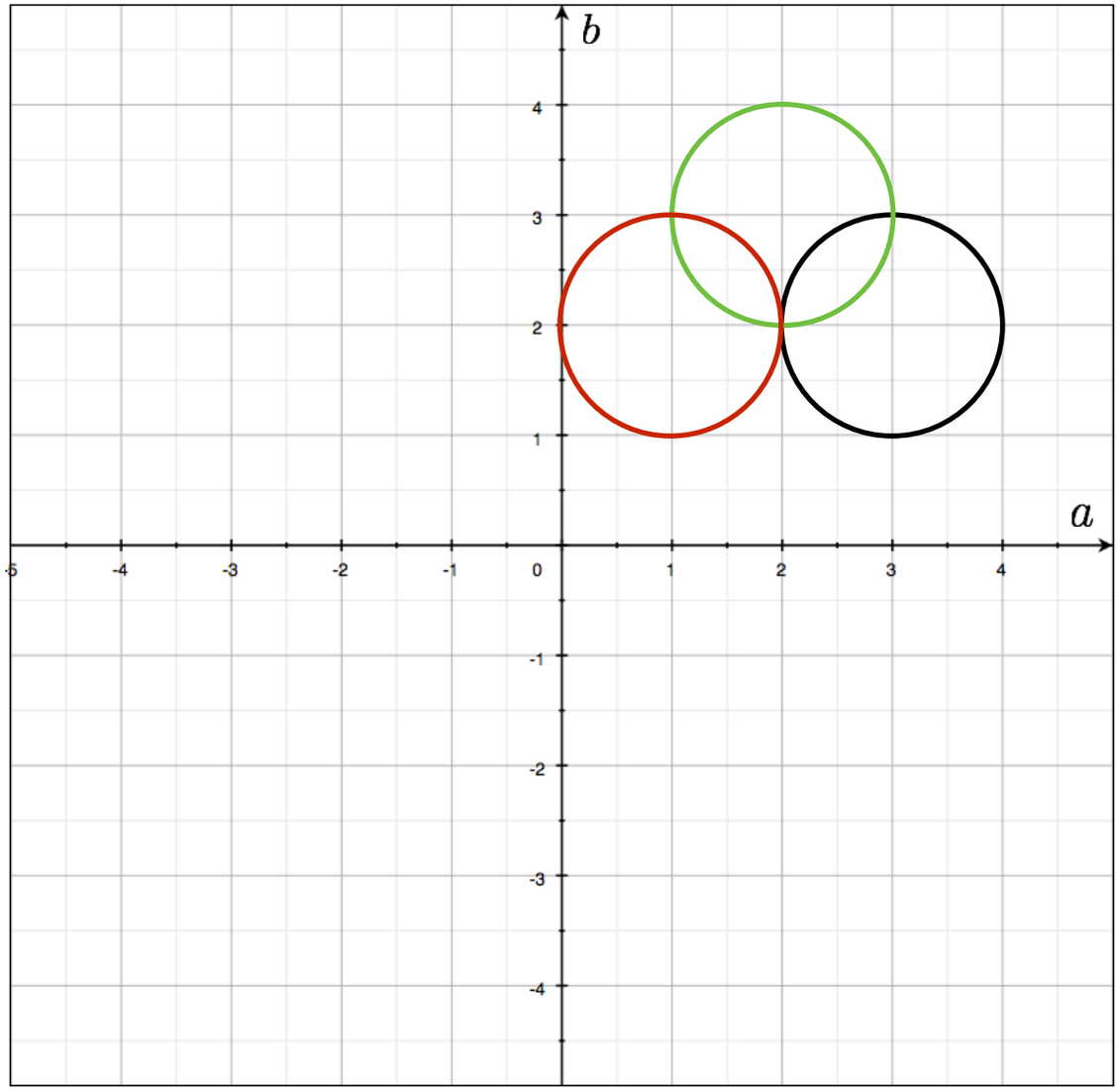
variables



parameters

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

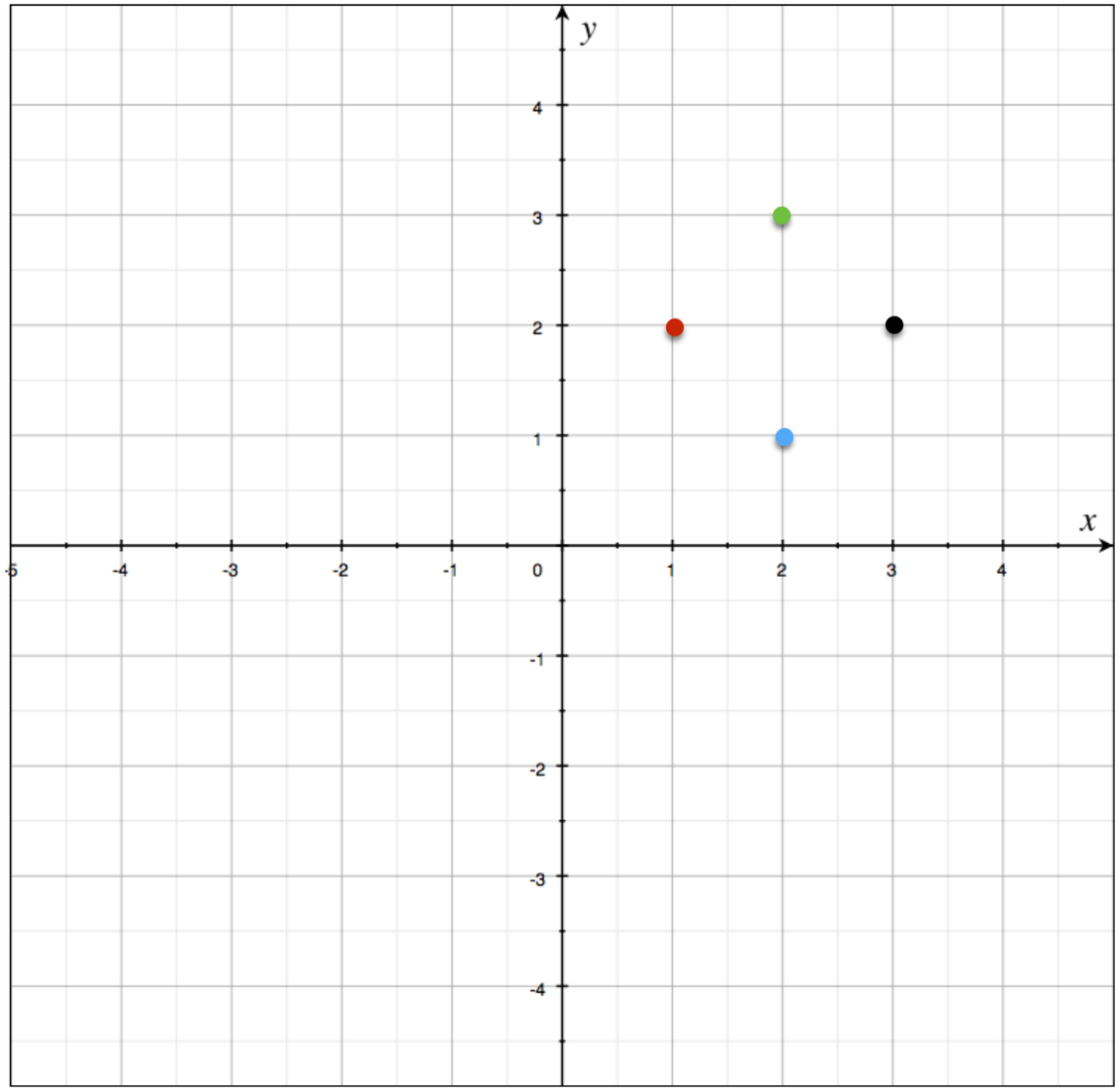
variables



parameters

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

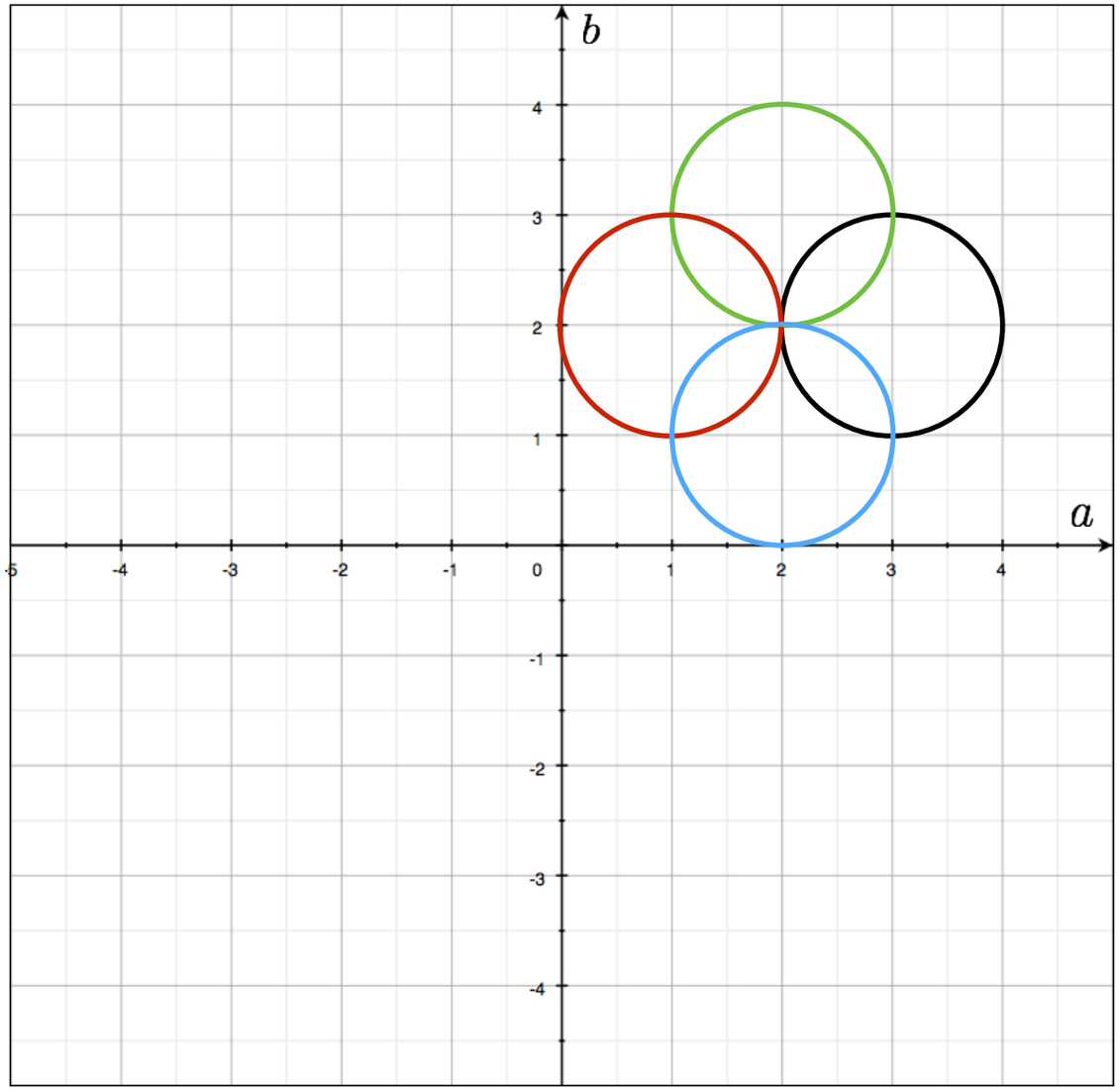
variables



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

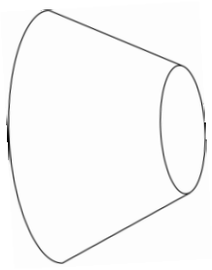
$$(x-a)^2+(y-b)^2=r^2$$

variables

If radius is not known: 3D Hough Space!

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

Surface shape in Hough space is complicated

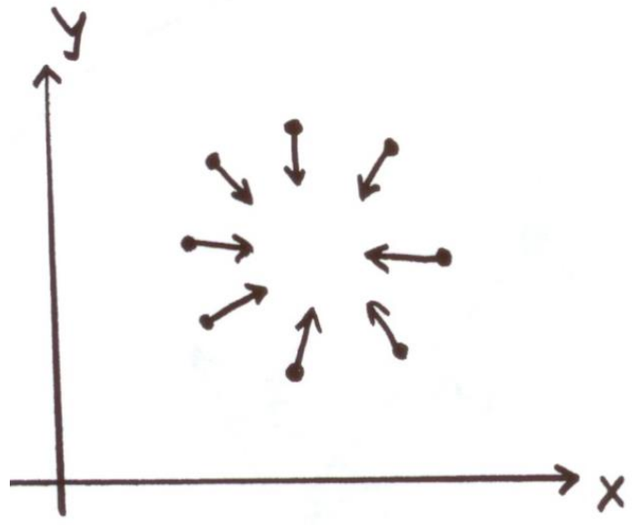


Using Gradient Information

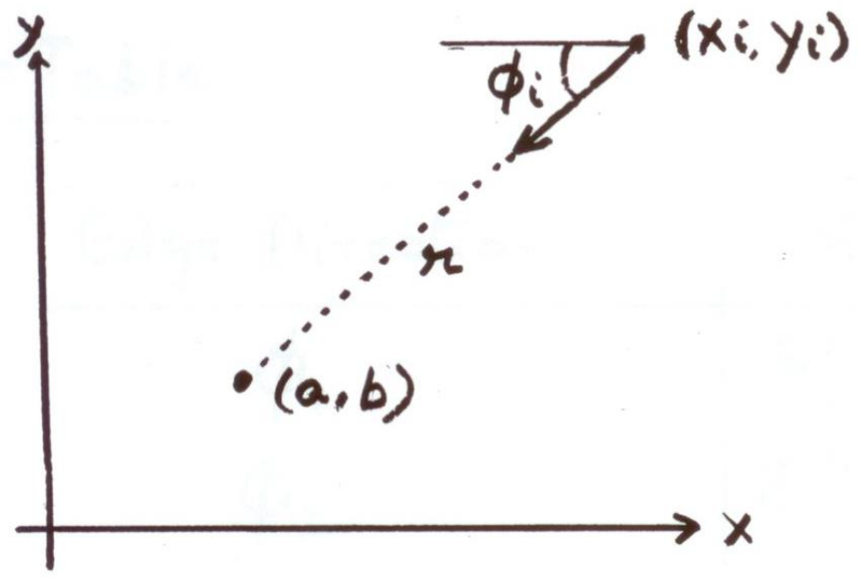
Gradient information can save lot of computation:

Edge Location (x_i, y_i)

Edge Direction ϕ_i



Assume radius is known:



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

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

Need to increment only one point in accumulator!

parameters

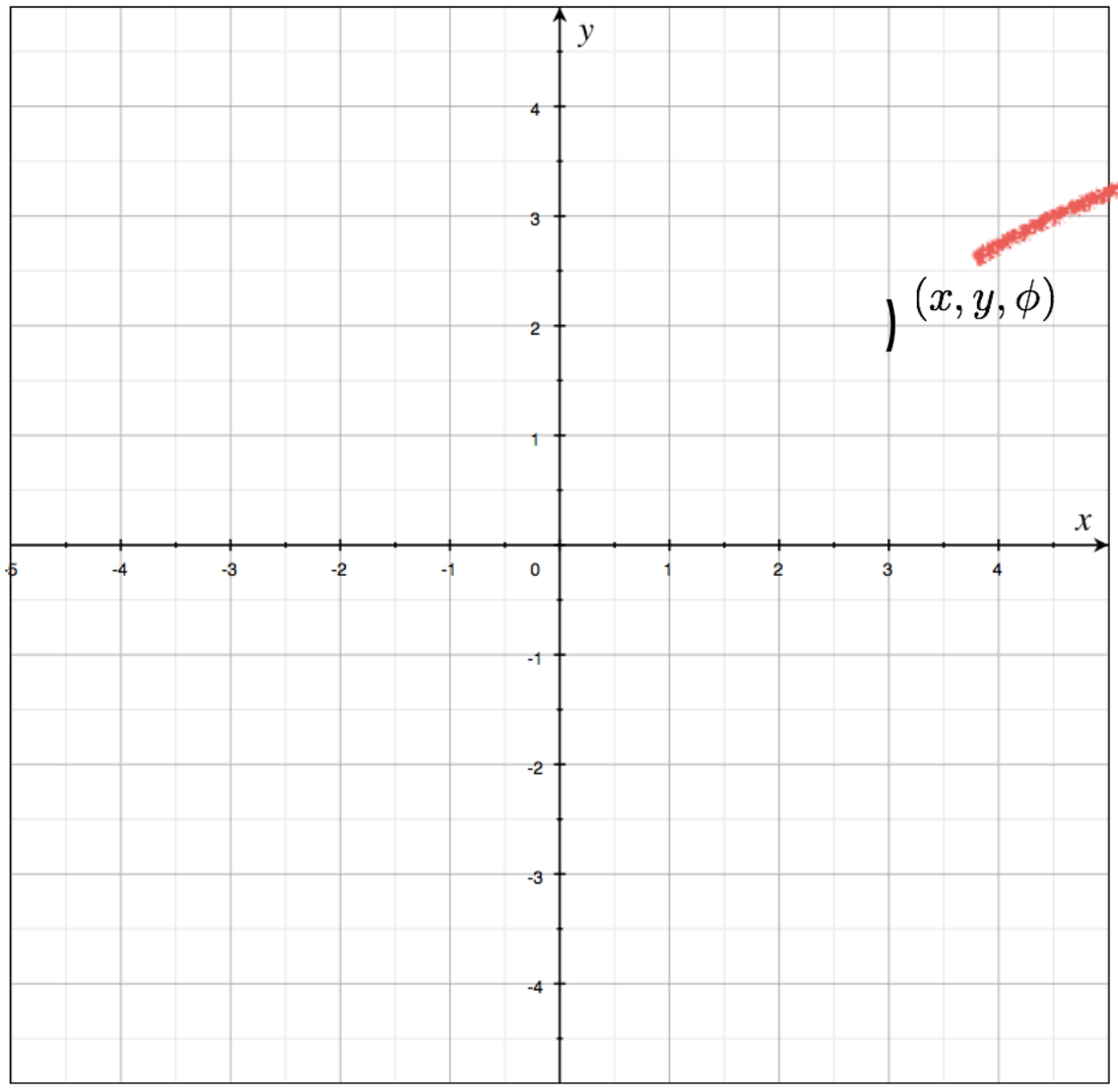
$$(x-a)^2+(y-b)^2=r^2$$

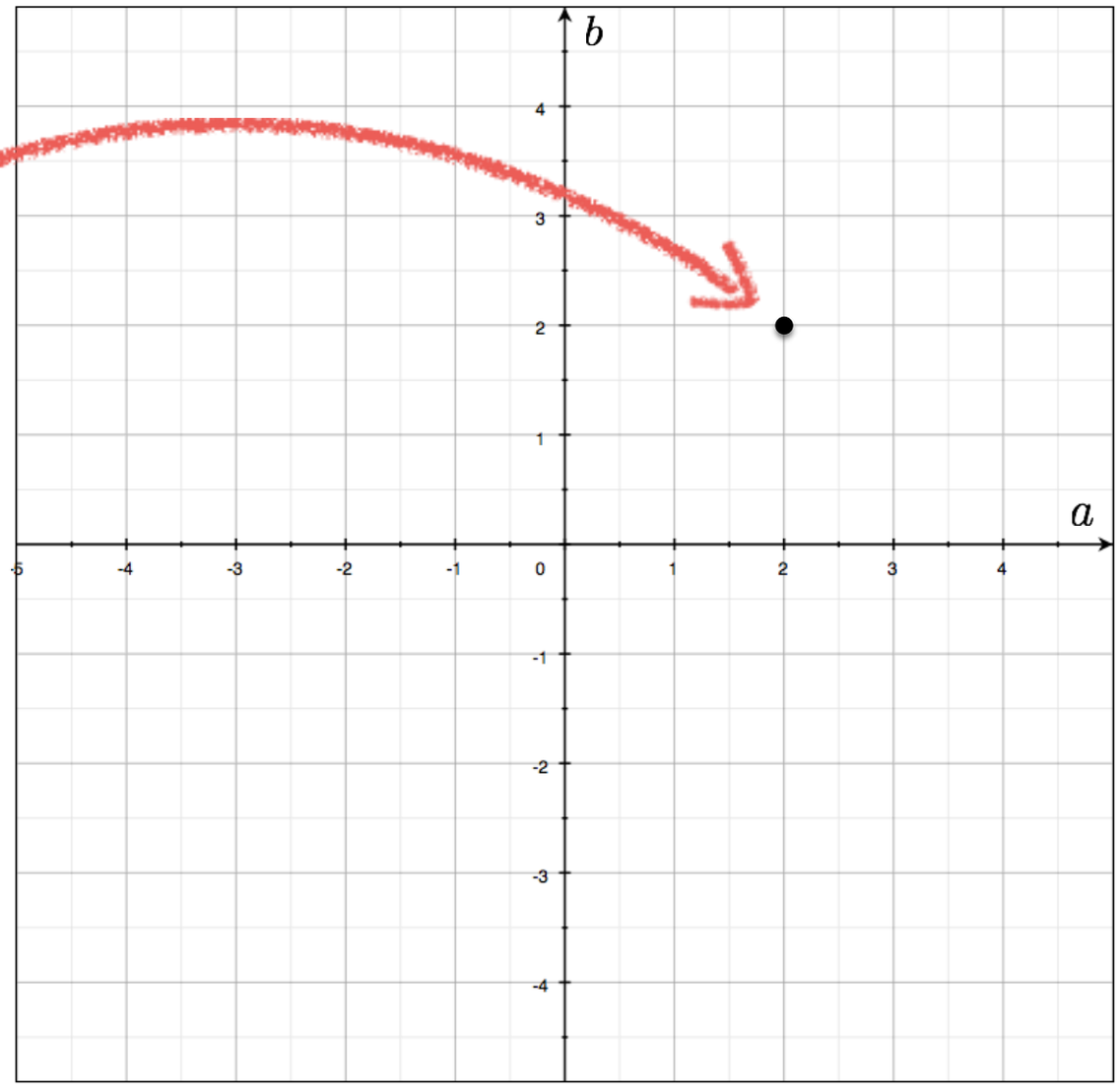
variables

parameters

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

variables

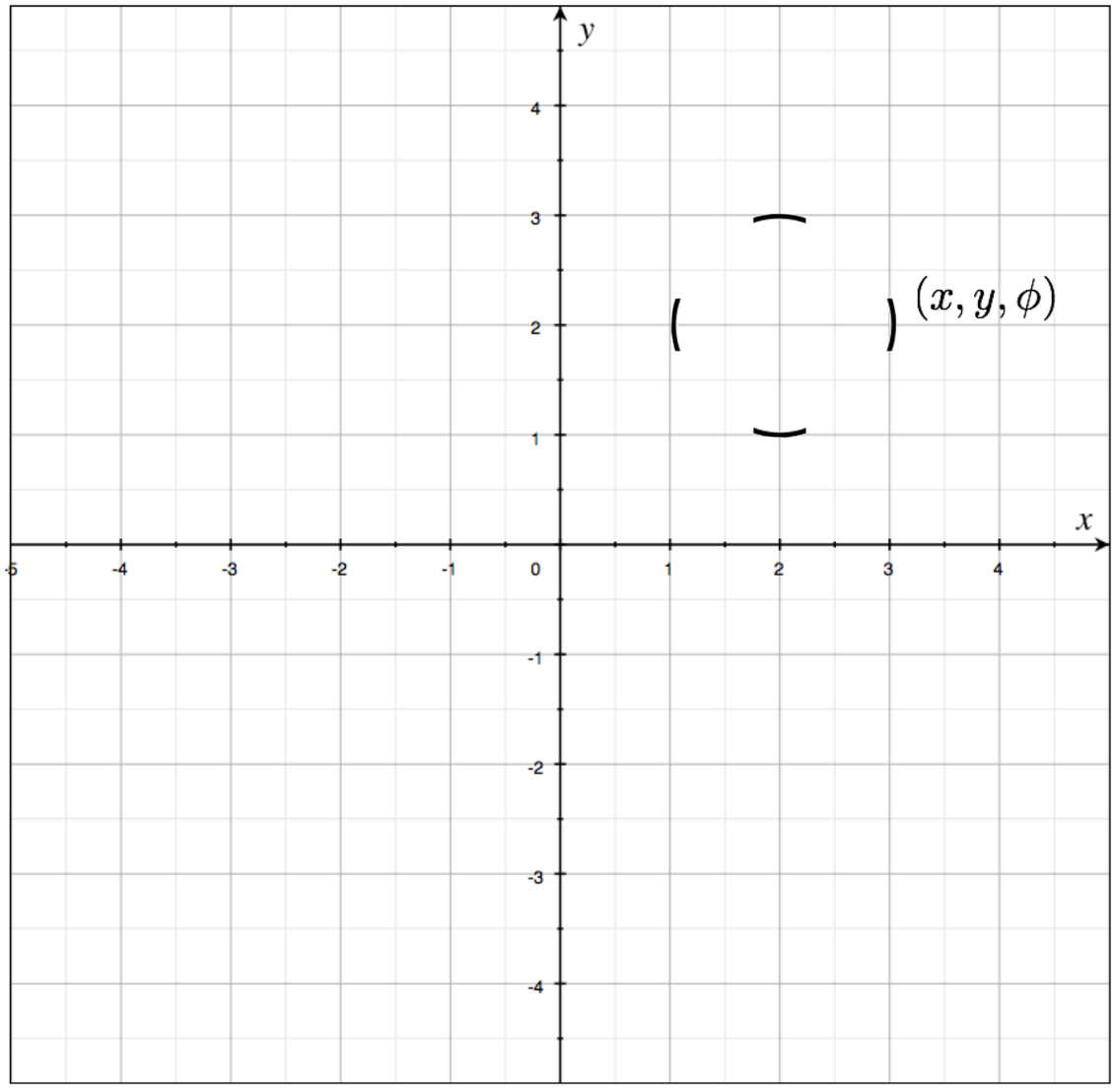




parameters

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

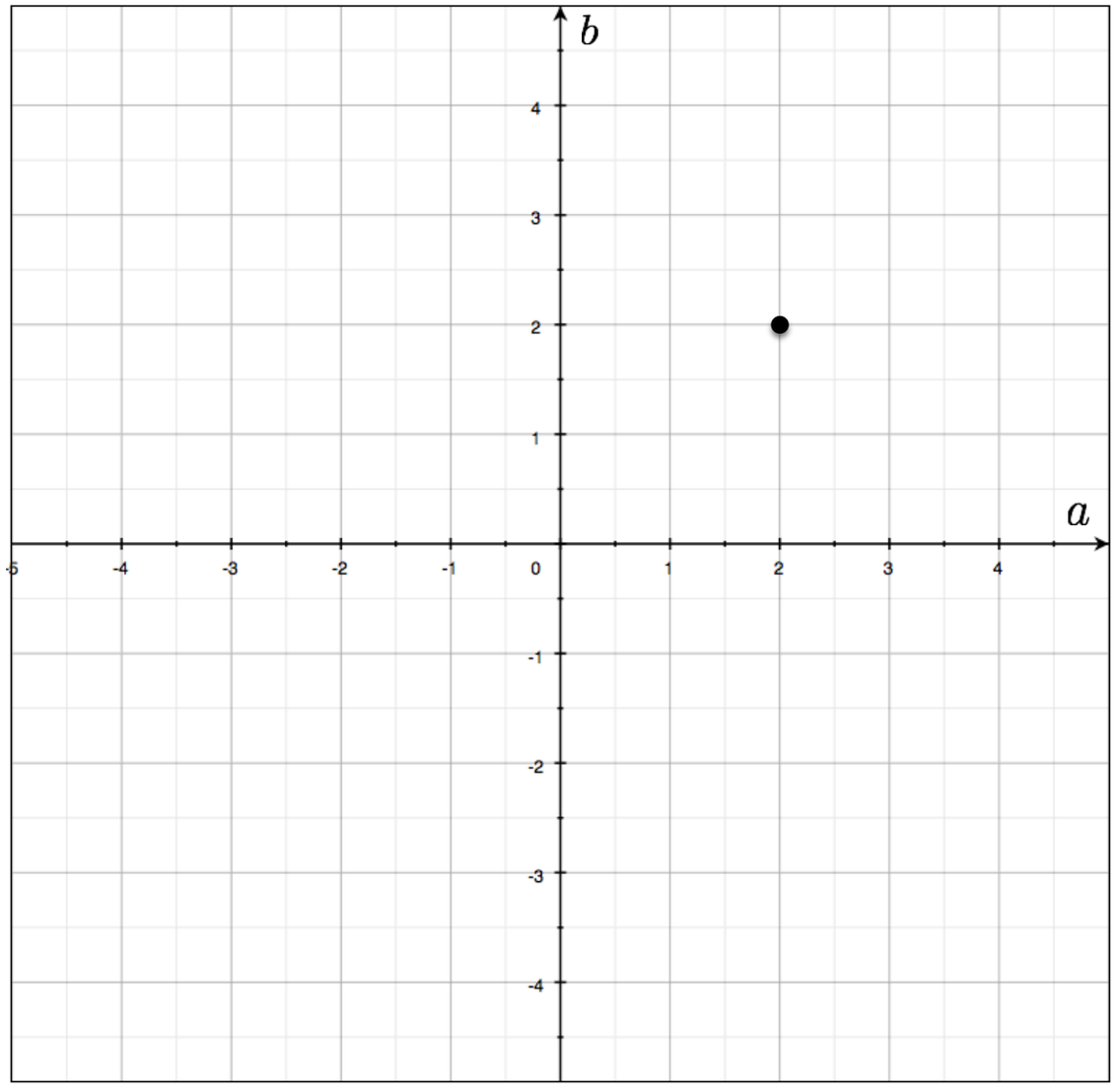
variables

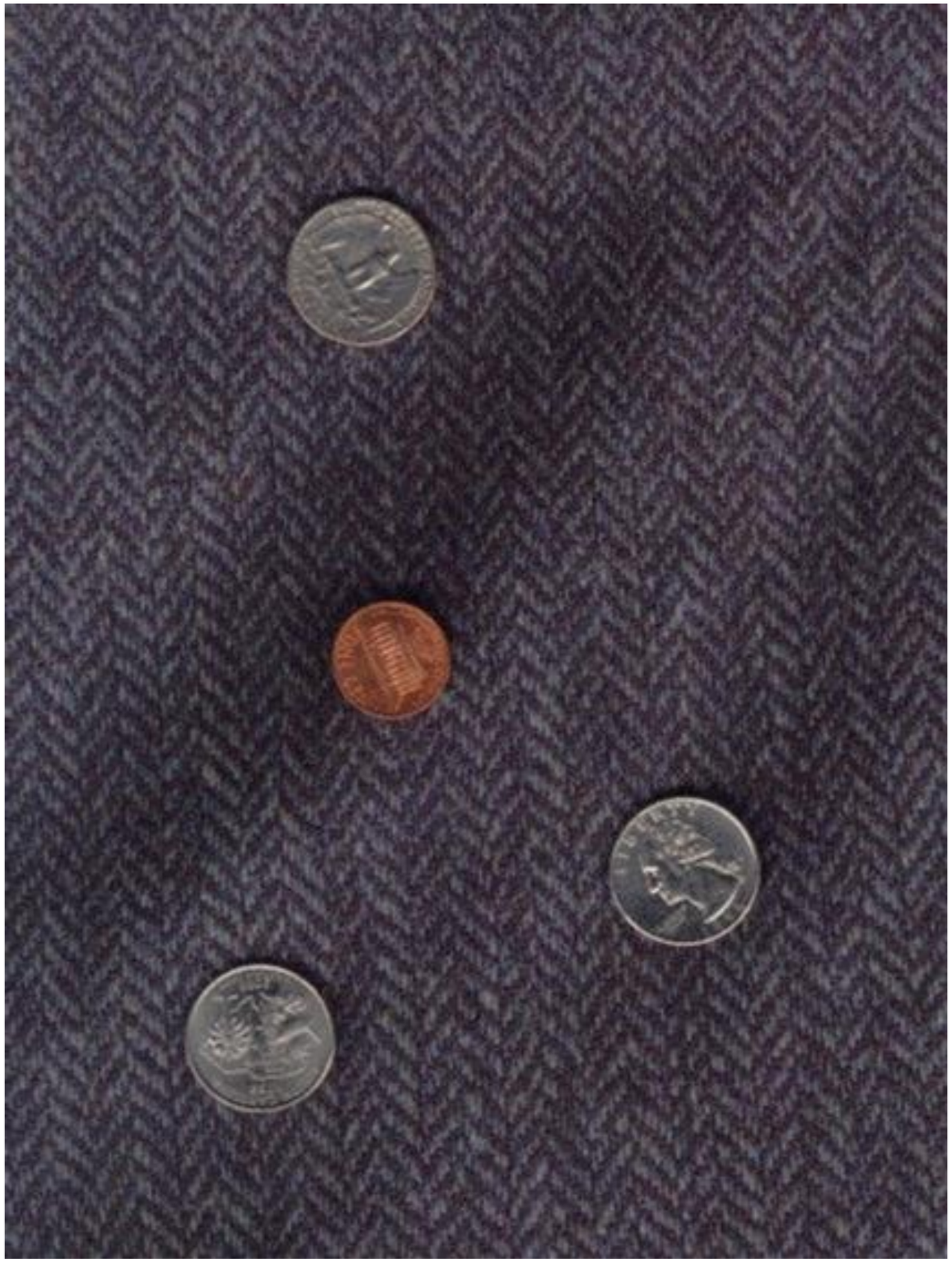


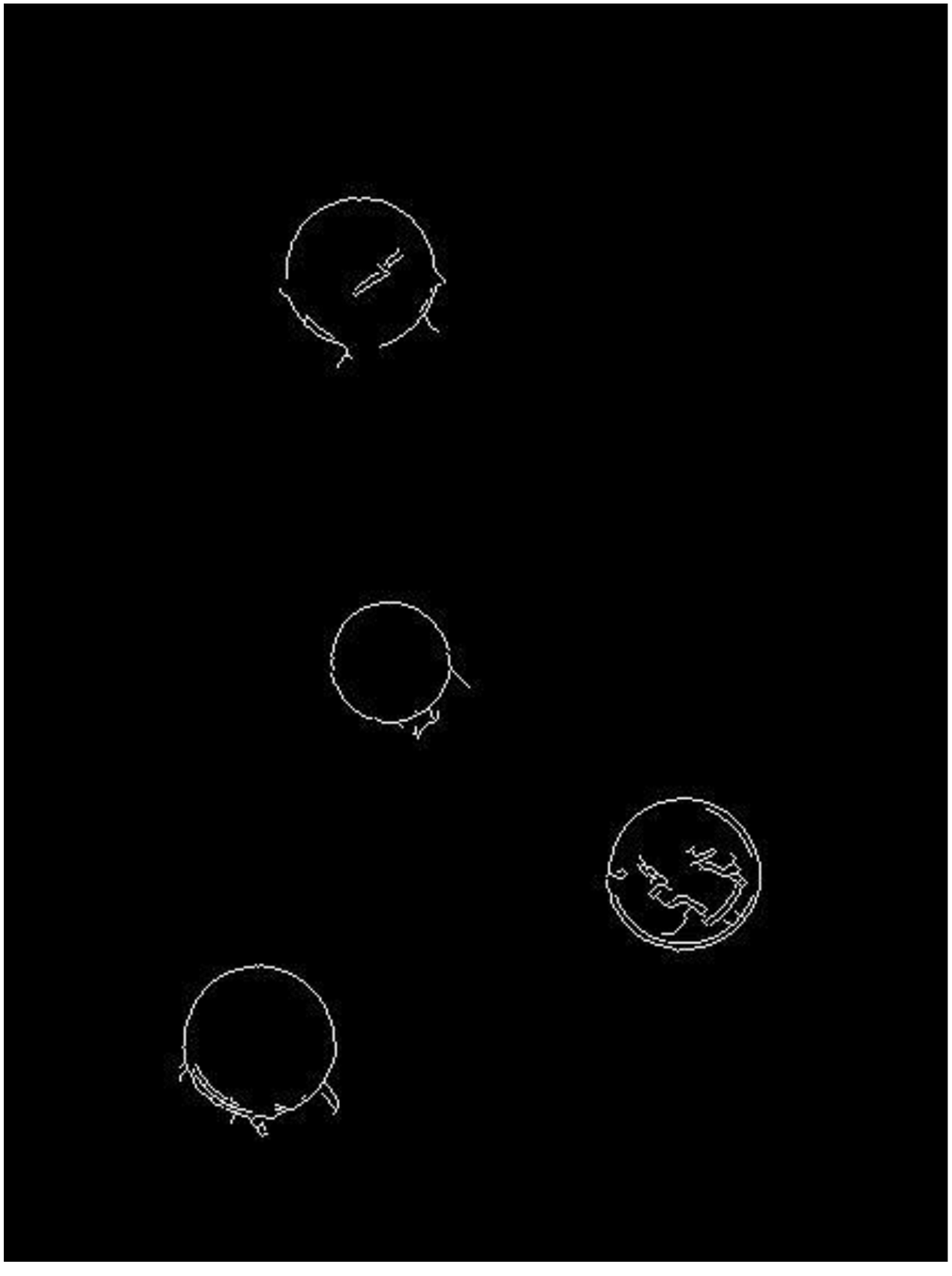
parameters

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

variables



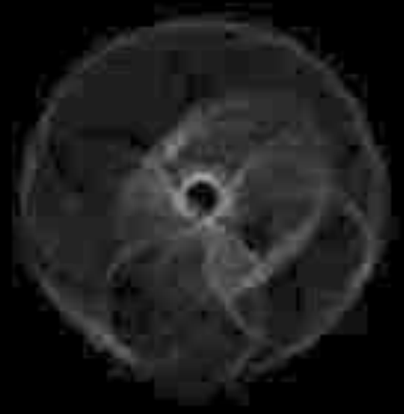




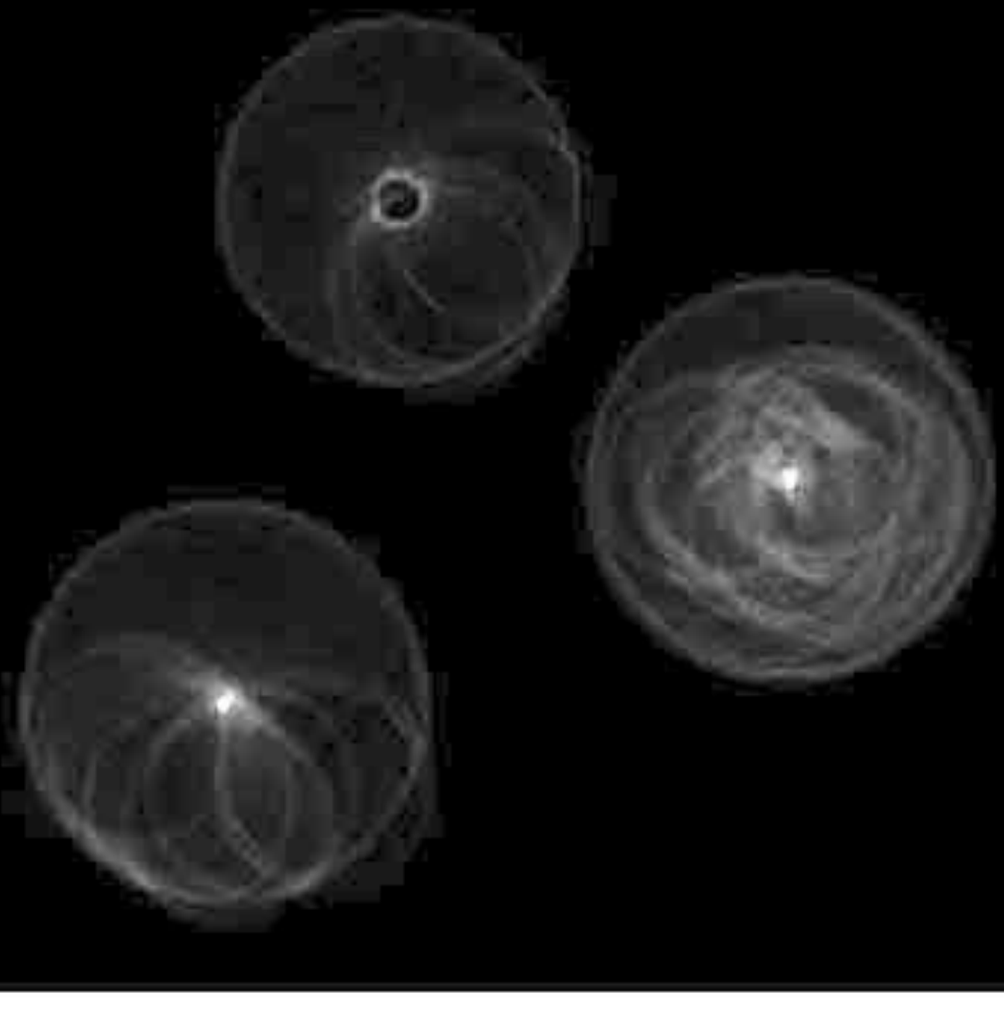

Pennie Hough detector


Quarter Hough detector

Pennie Hough detector



Quarter Hough detector



The Hough transform …

Deals with occlusion well?

Detects multiple instances?

Robust to noise?

Good computational complexity?

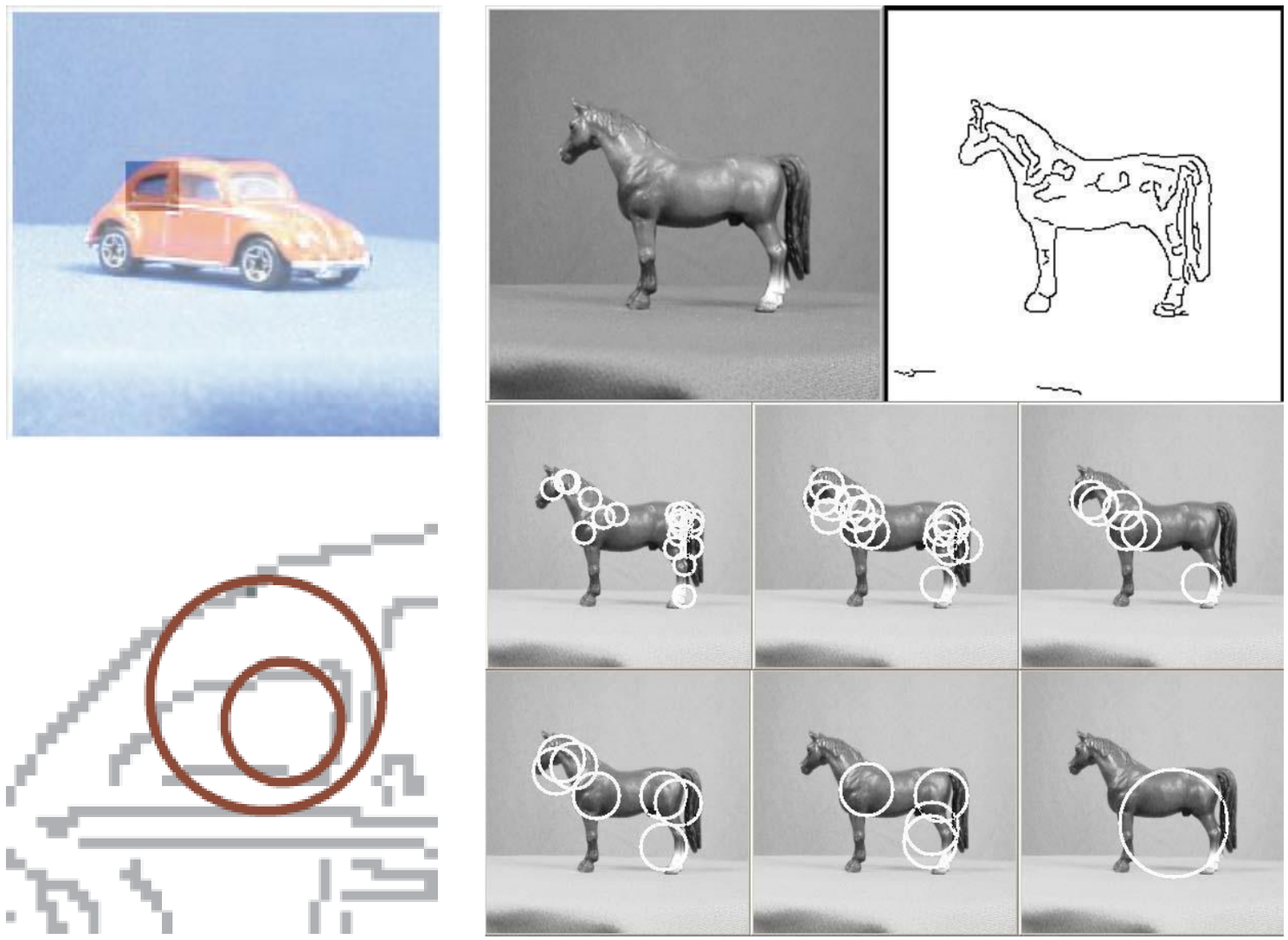
Easy to set parameters?

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

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

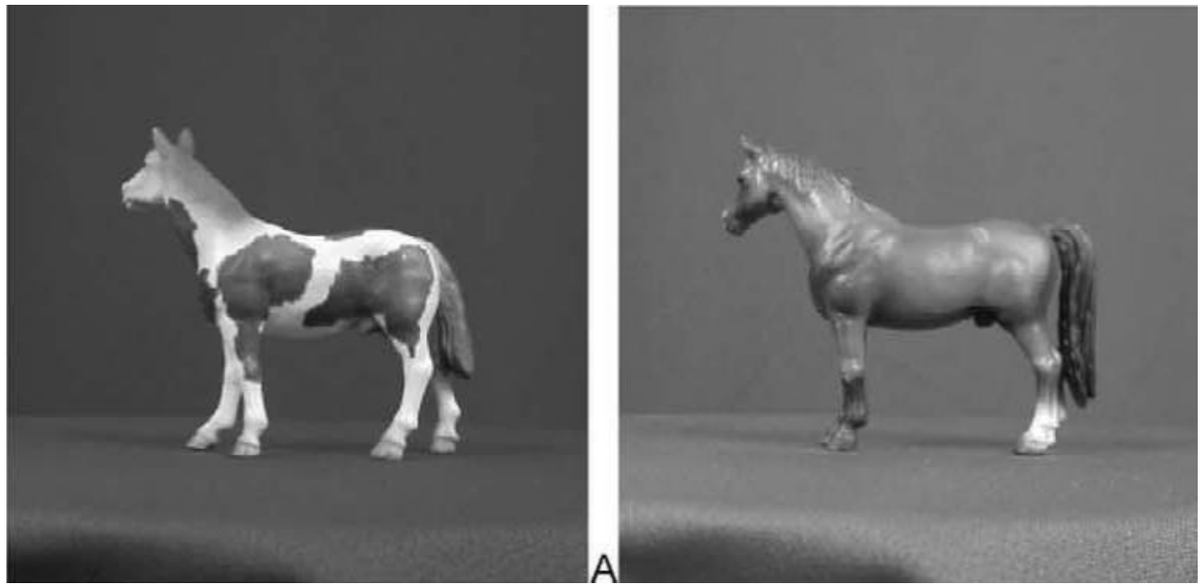
Application of Hough transforms

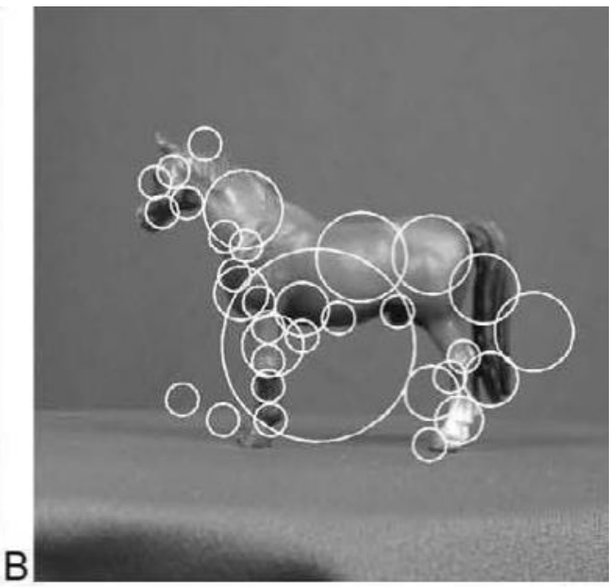
Detecting shape features



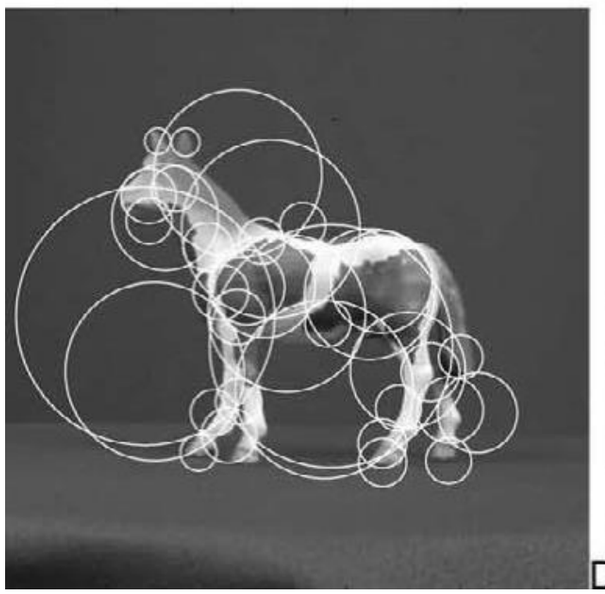
F. Jurie and C. Schmid, Scale-invariant shape features for recognition of object categories, CVPR 2004

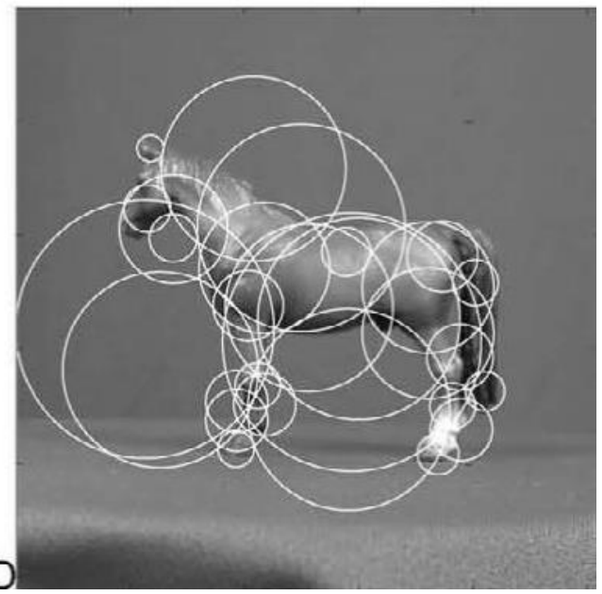
Original images





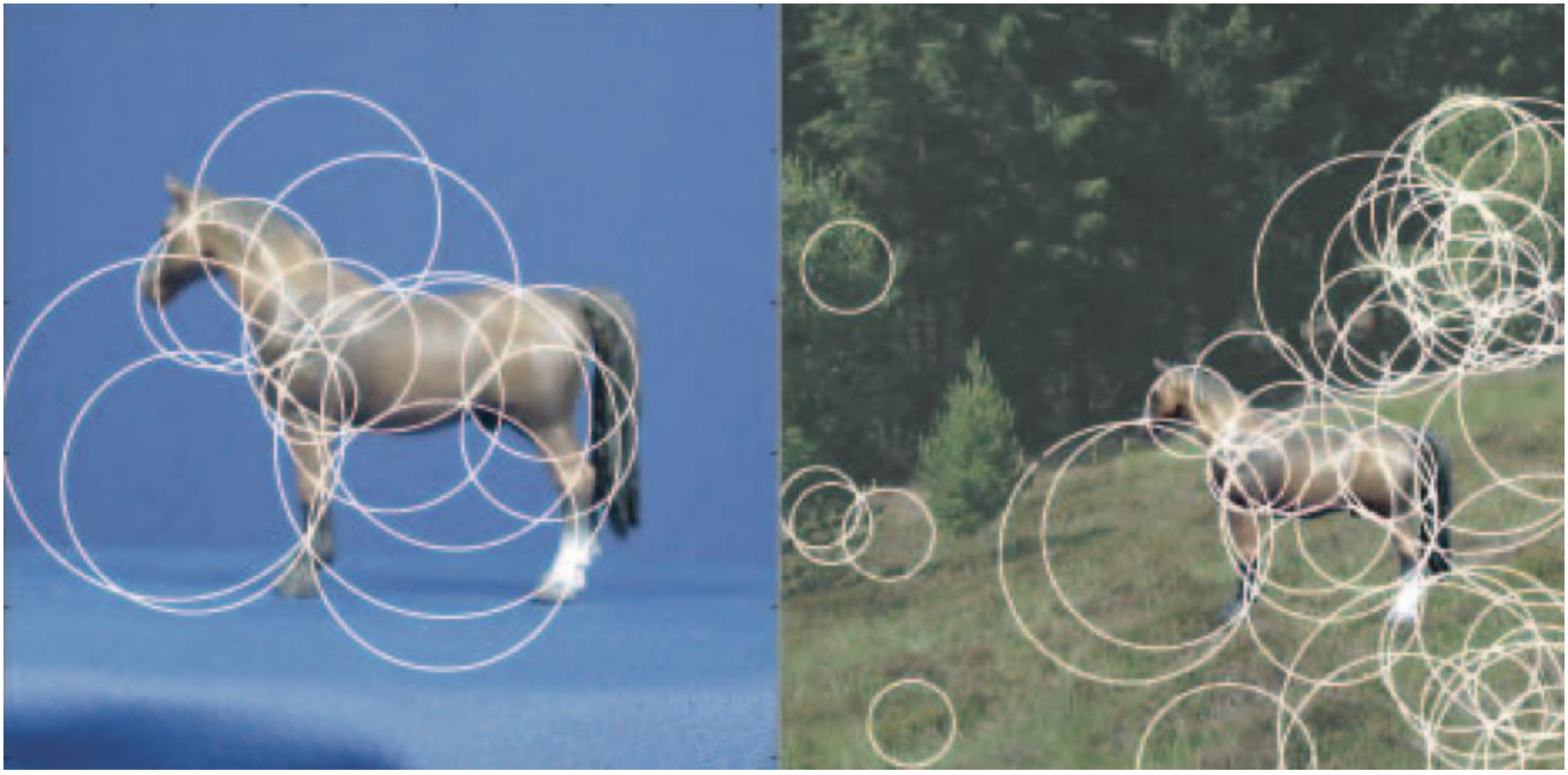
Laplacian circles





Hough-like circles

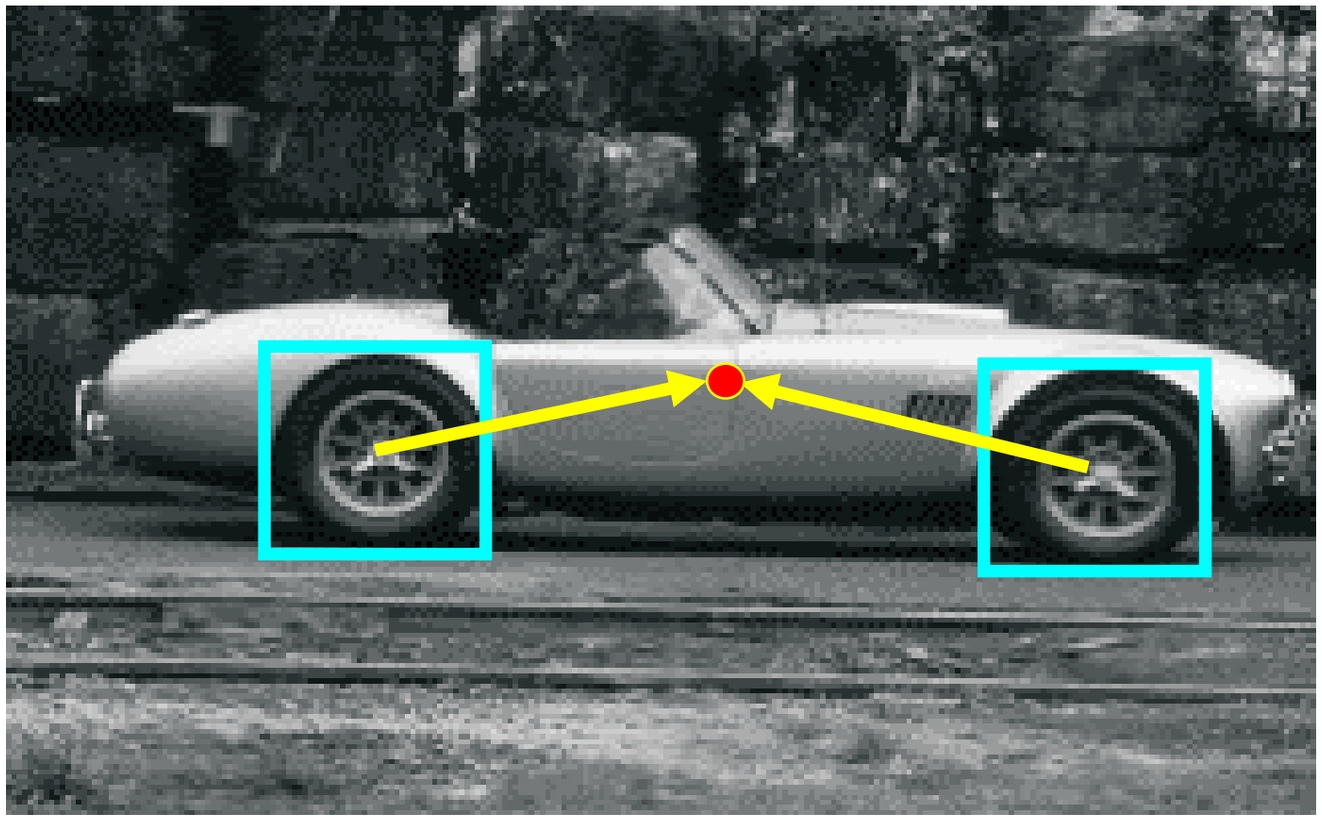
Which feature detector is more consistent?



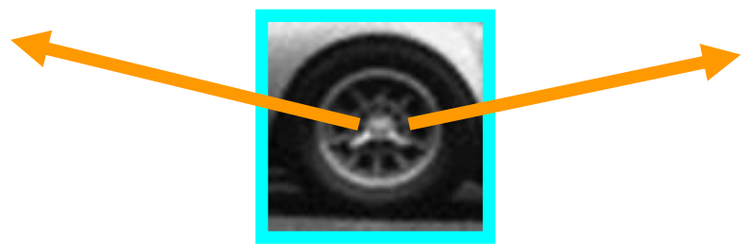
Robustness to scale and clutter

Object detection

Index displacements by “visual codeword”



training image



visual codeword with displacement vectors

B. Leibe, A. Leonardis, and B. Schiele, Combined Object Categorization and Segmentation with an Implicit Shape Model,
ECCV Workshop on Statistical Learning in Computer Vision 2004

References

Basic reading:

- Szeliski textbook, Sections 4.2, 4.3.