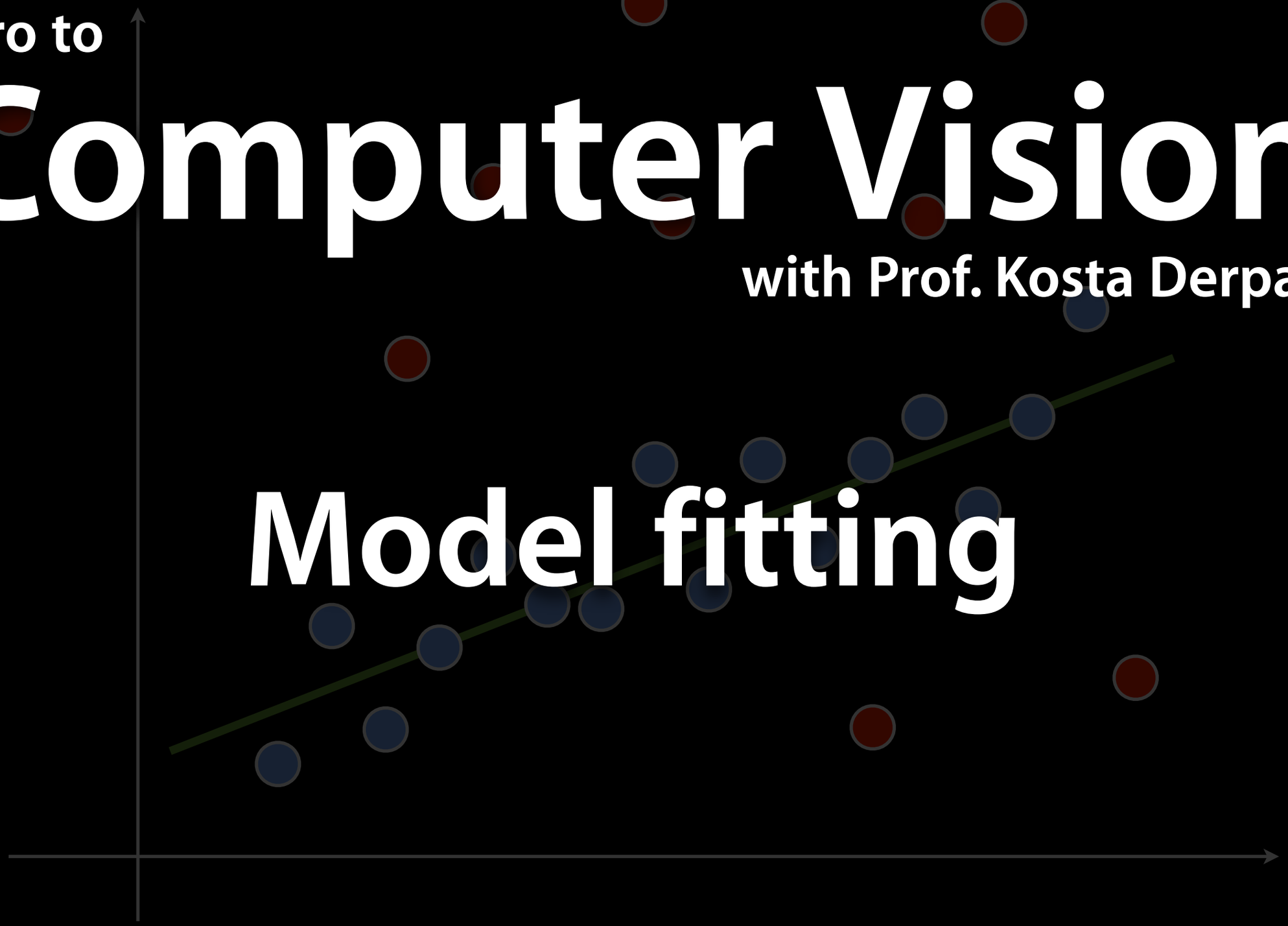
Intro to

Computer Vision

with Prof. Kosta Derpanis



Model fitting

LECTURE TOPICS

Least-squares fitting

Hough transform

RANSAC

Image stitching

Want to associate a model with observed features

Model could be a line

Model could be a circle

Model could be an arbitrary shape

Model could be a transformation

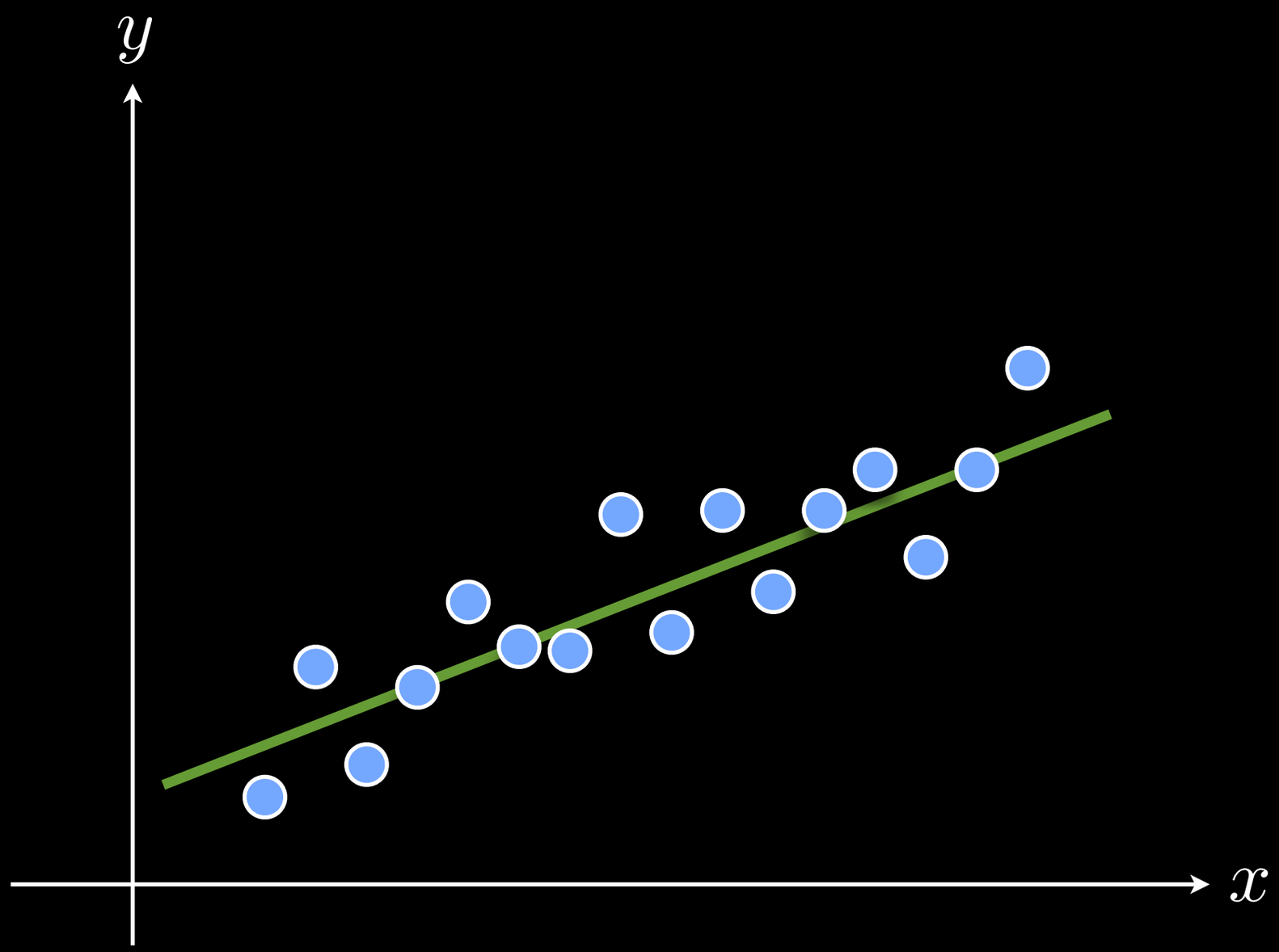
Main Questions

What model(s) represent the set of features best?

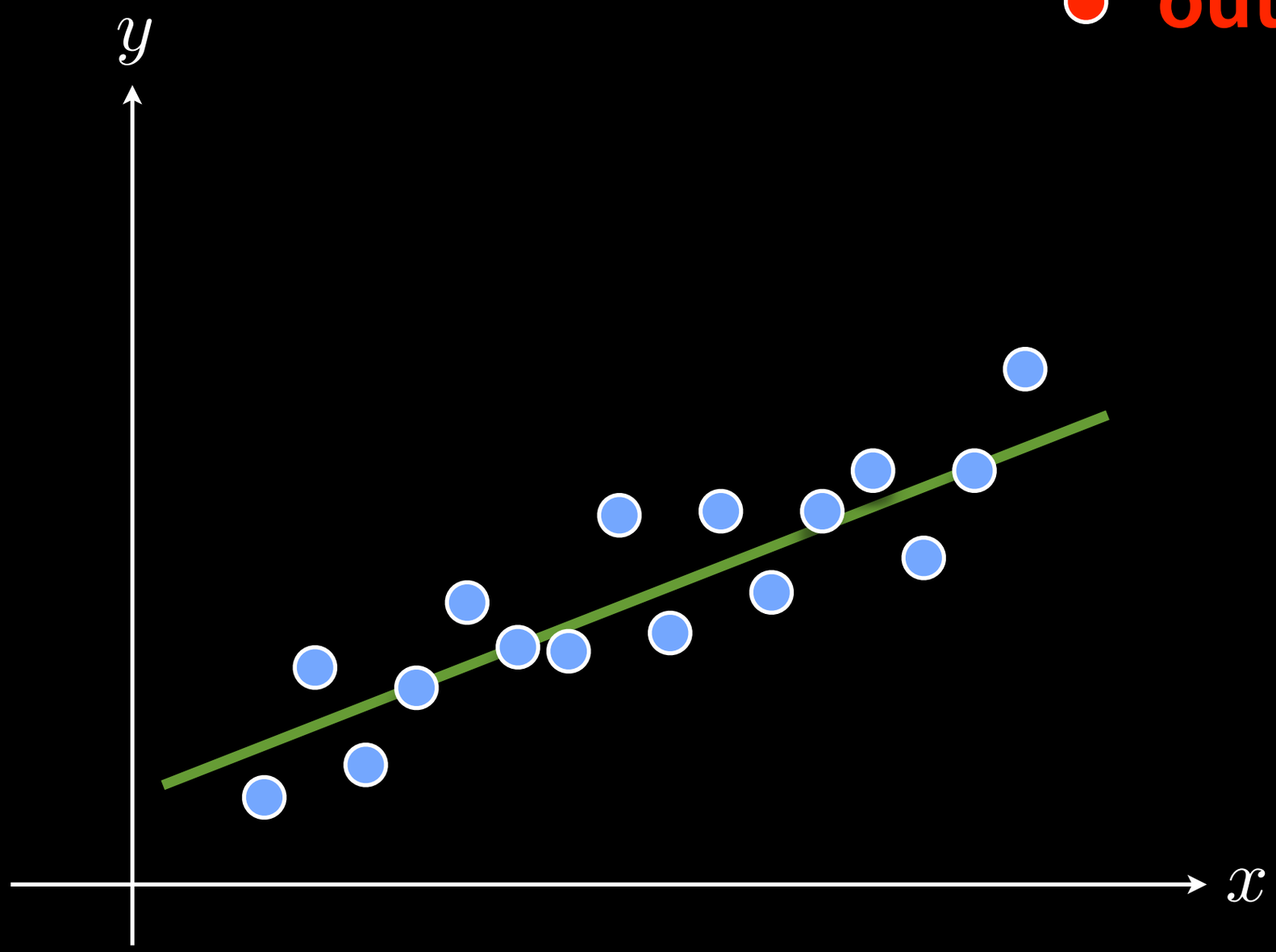
How many model instances are there?

Which of several model instances gets which feature?

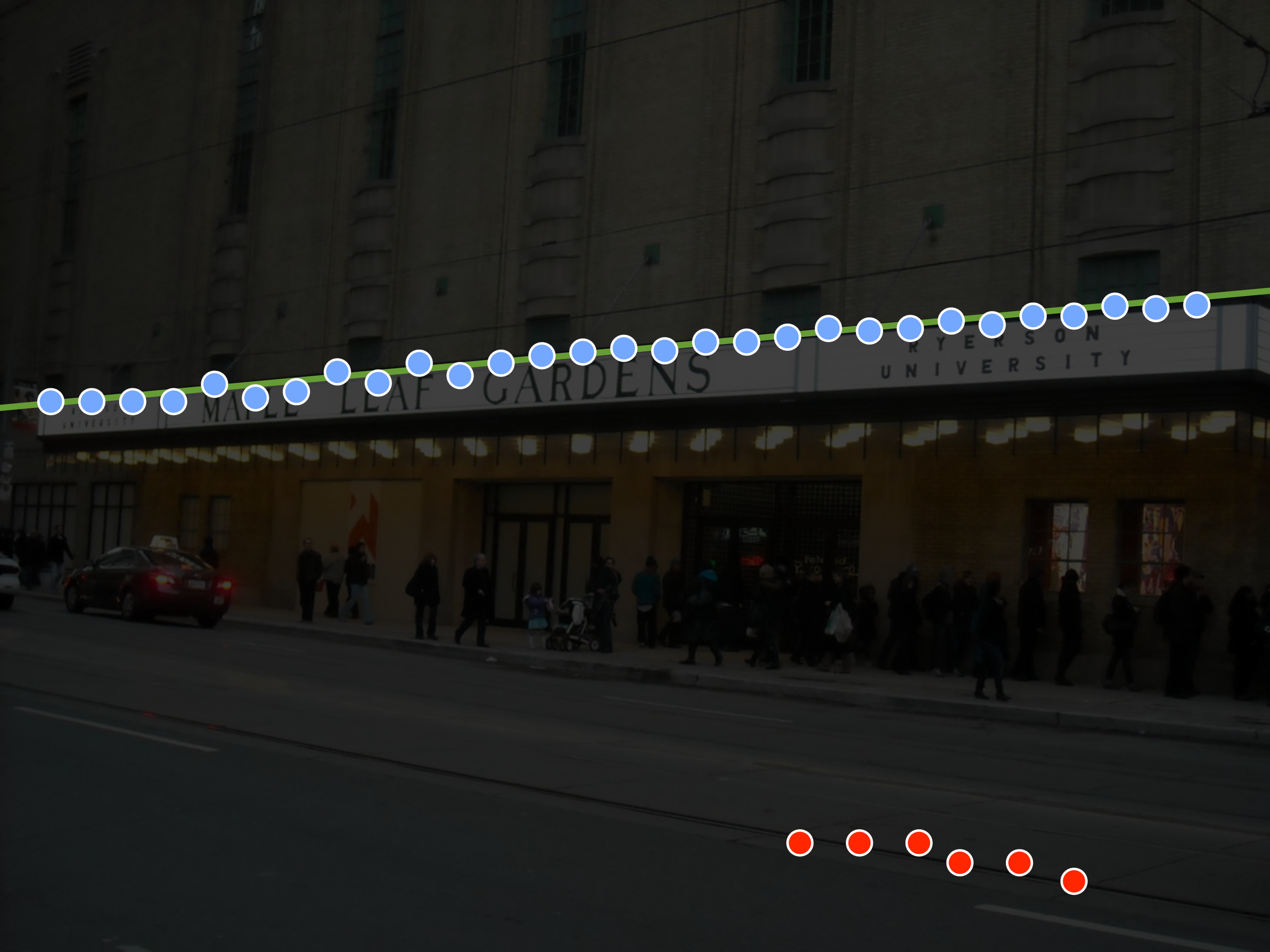
Noisy data



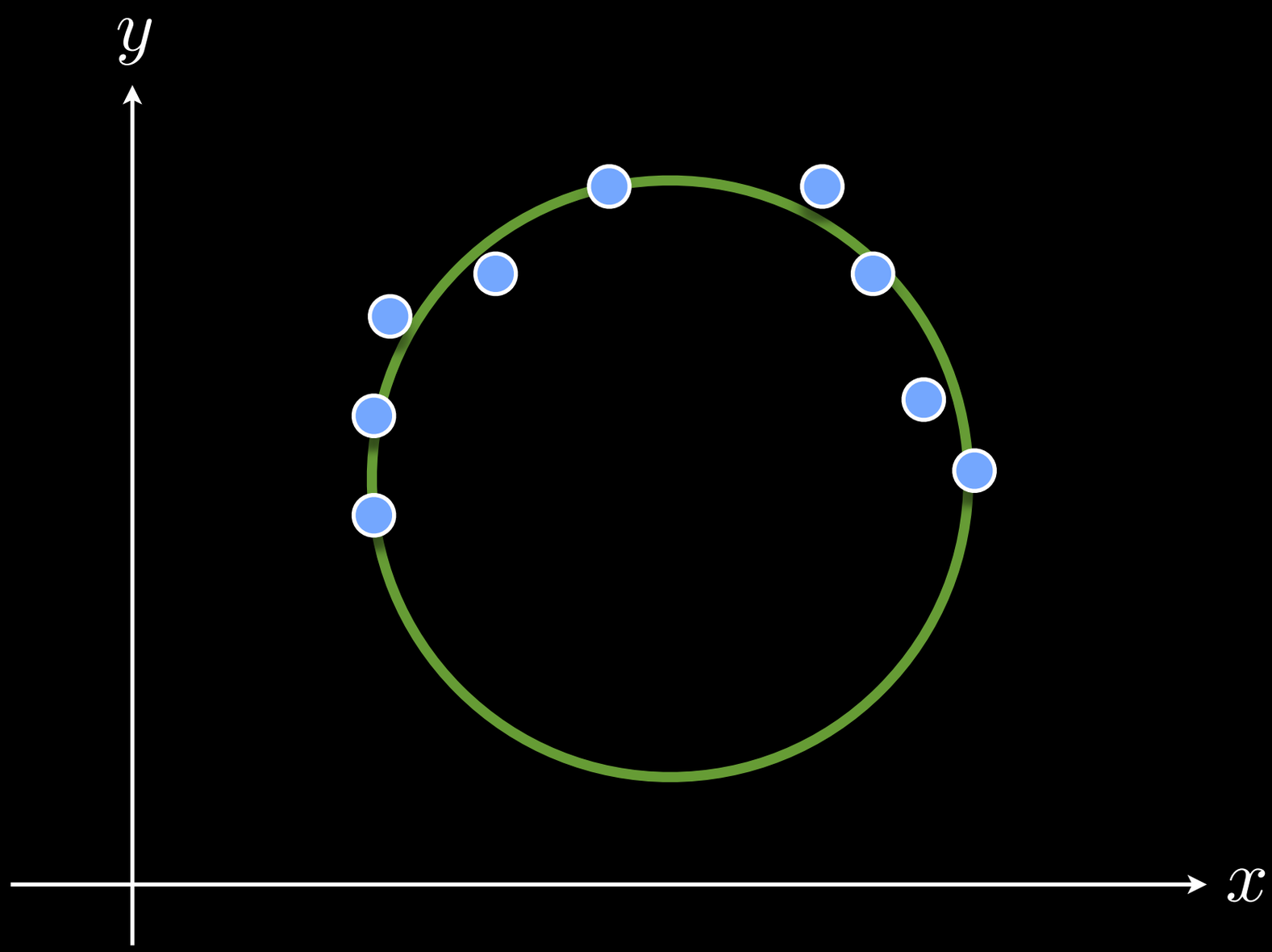
Outliers in data



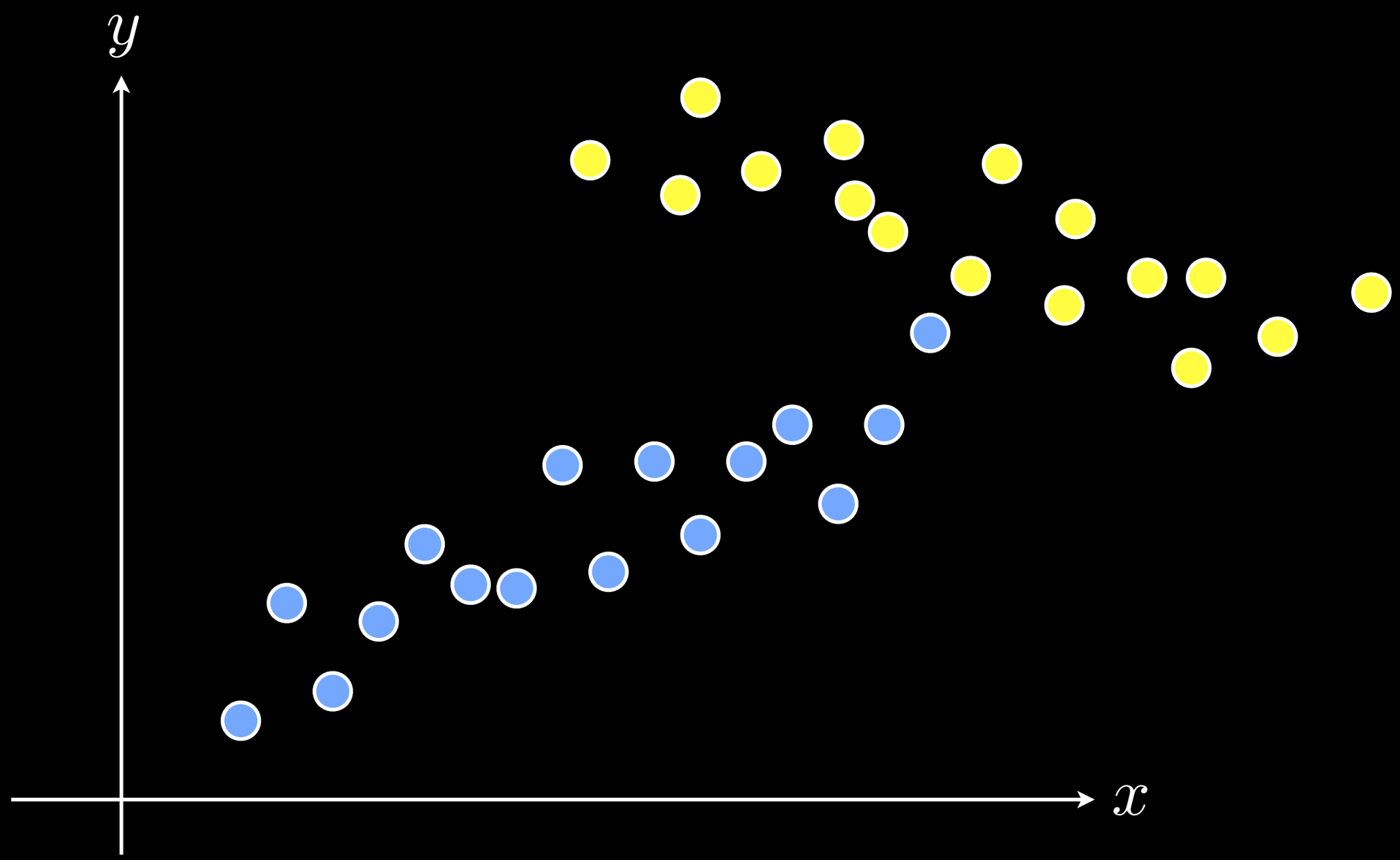
outlier


MAPLE LEAF GARDENS
RYERSON
UNIVERSITY

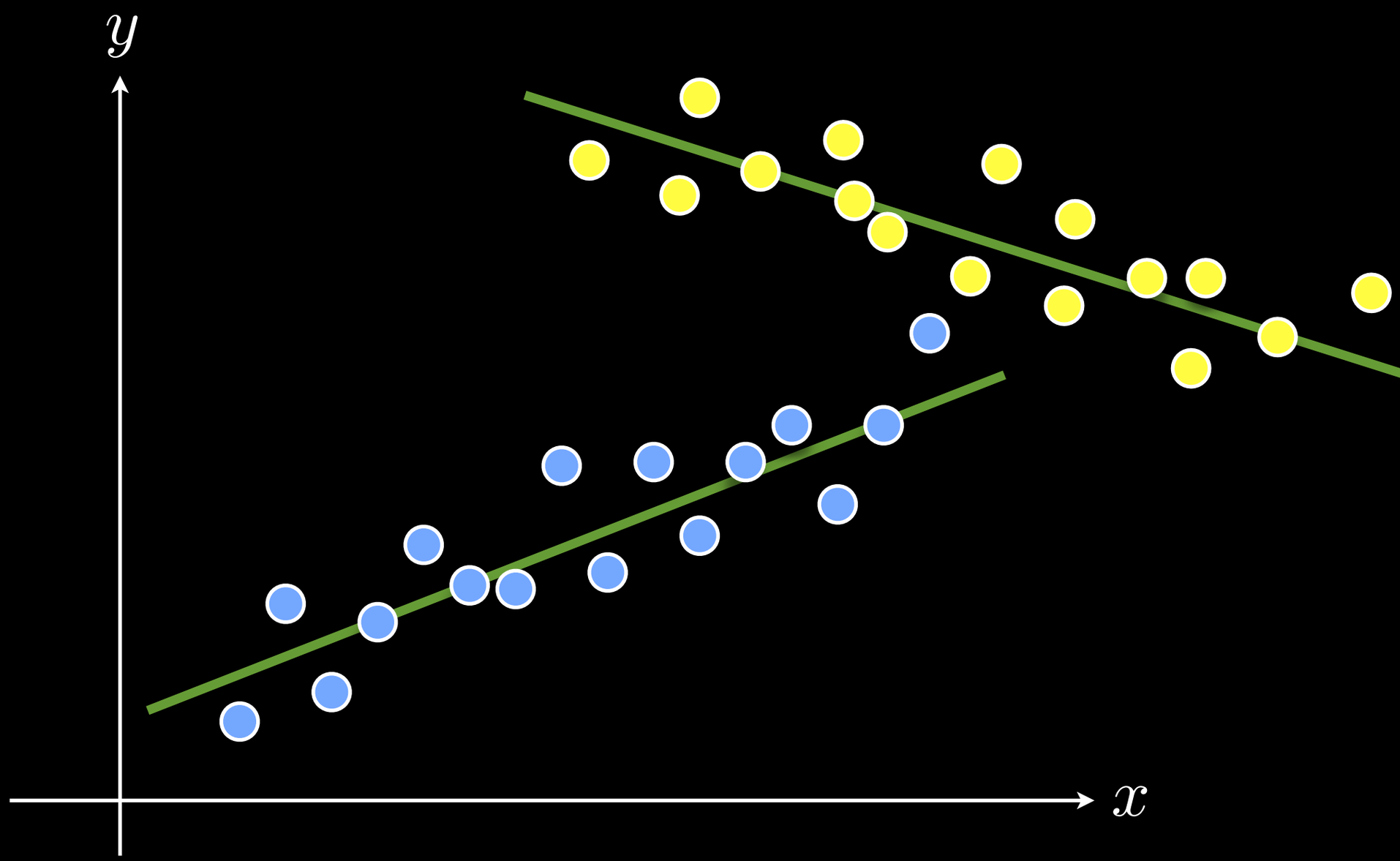
Missing data

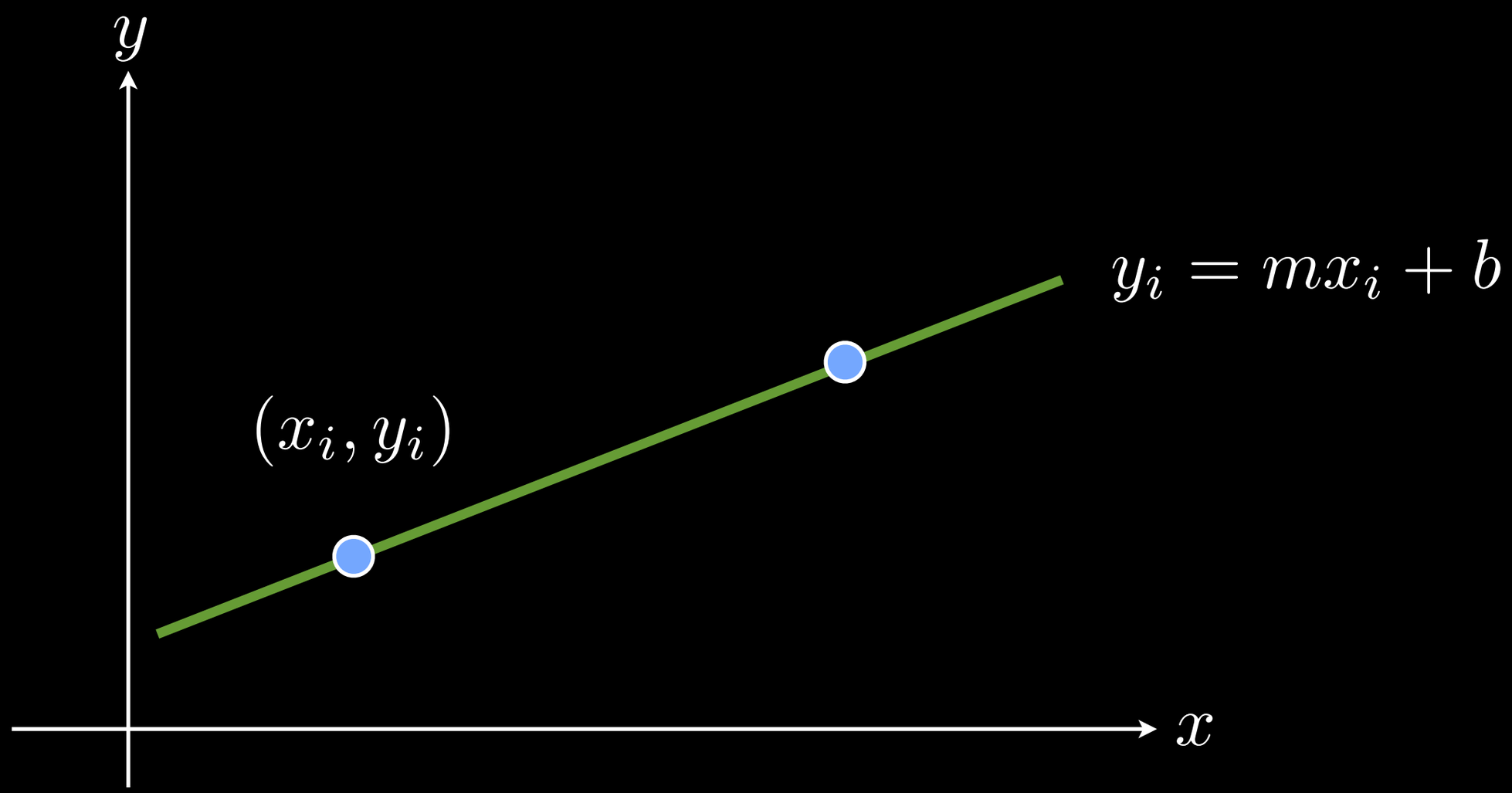


Feature binding



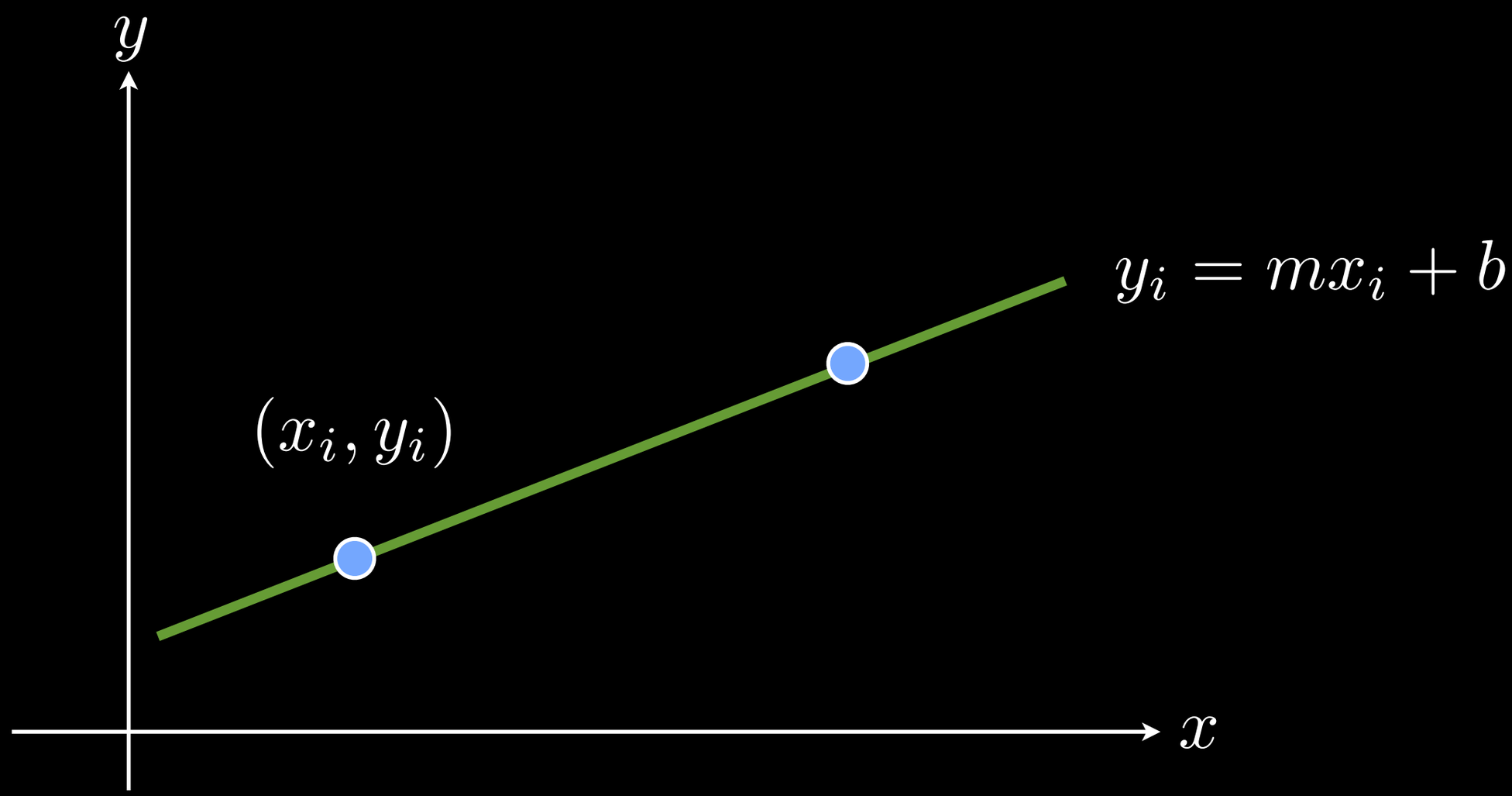
Feature binding





How do we find the line?

$$y_0 = mx_0 + b$$
$$y_1 = mx_1 + b$$



How do we find the line?

$$\begin{pmatrix} x_0 & 1 \\ x_1 & 1 \end{pmatrix} \begin{pmatrix} m \\ b \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \end{pmatrix}$$

How do we find the line?

$$\begin{pmatrix} x_0 & 1 \\ x_1 & 1 \end{pmatrix} \begin{pmatrix} m \\ b \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \end{pmatrix}$$

$$\mathbf{A} \qquad \mathbf{p} \qquad \mathbf{b}$$

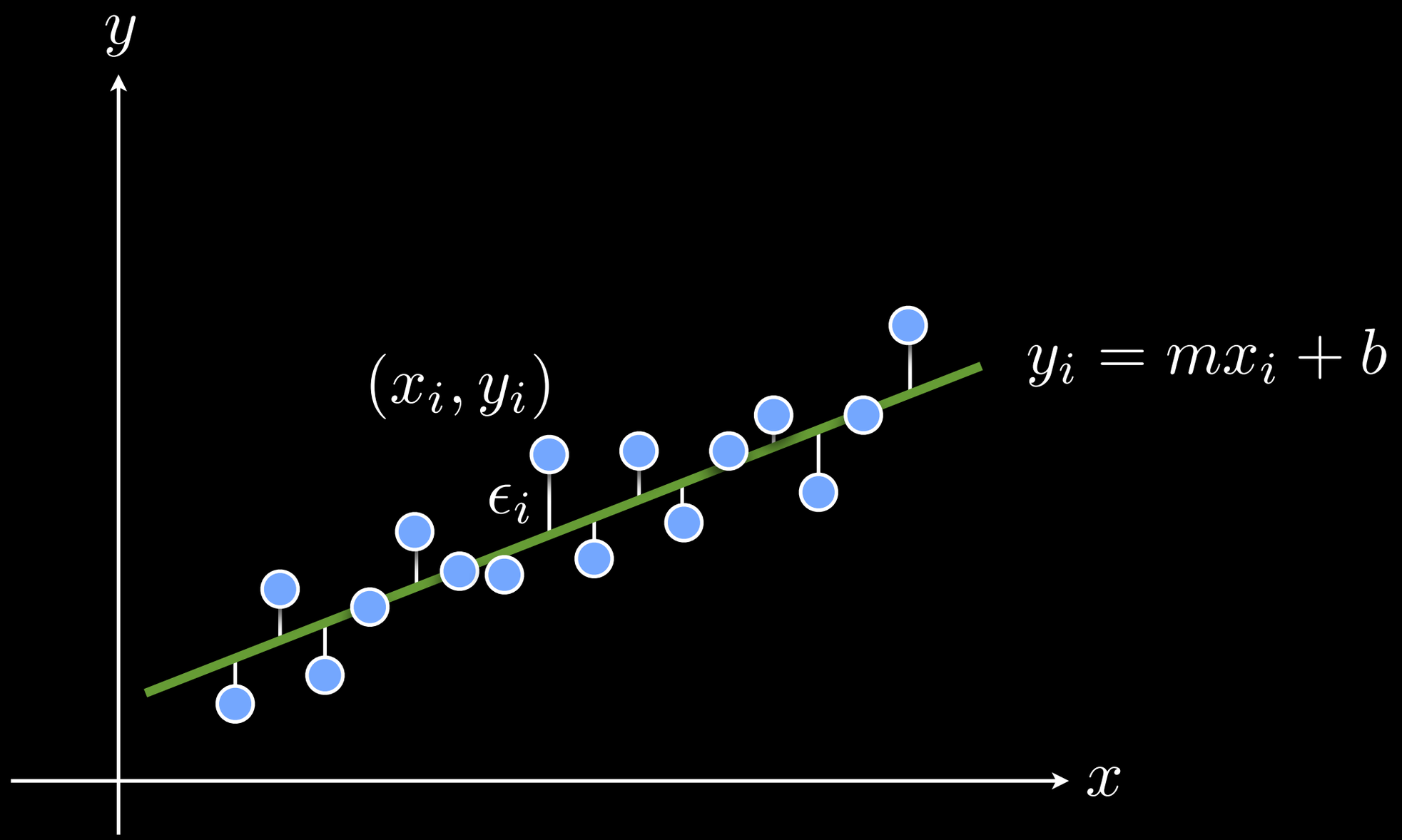
What is the solution for the line parameters?

How do we find the line?

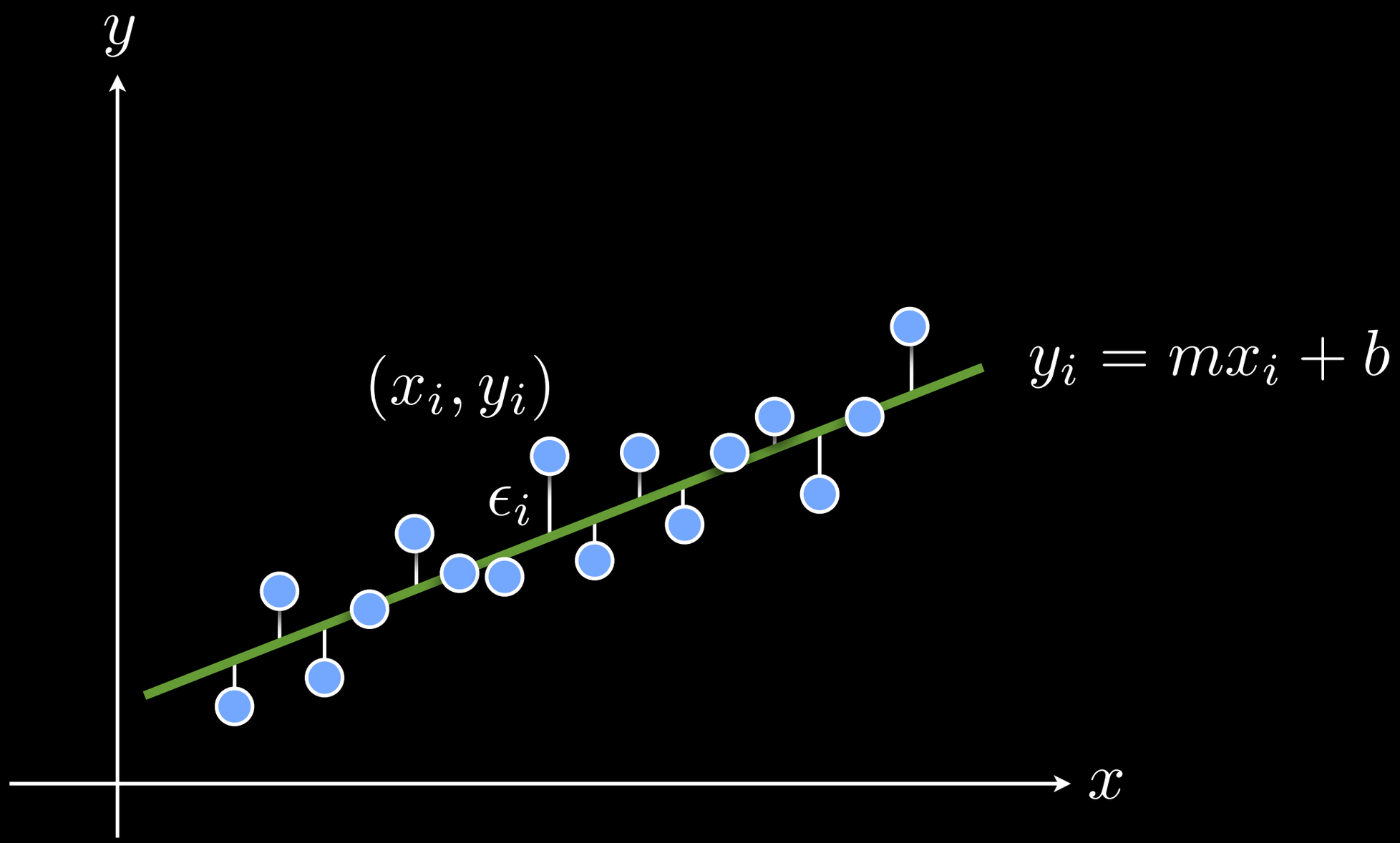
$$\underbrace{\begin{pmatrix} x_0 & 1 \\ x_1 & 1 \end{pmatrix}}_{\mathbf{A}} \underbrace{\begin{pmatrix} m \\ b \end{pmatrix}}_{\mathbf{p}} = \underbrace{\begin{pmatrix} y_0 \\ y_1 \end{pmatrix}}_{\mathbf{b}}$$

What is the solution for the line parameters?

$$\mathbf{p} = \mathbf{A}^{-1}\mathbf{b}$$



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



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

unknown

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

$$= \sum_{i=1}^{N} \left(y_i - \begin{pmatrix} x_i & 1 \end{pmatrix} \begin{pmatrix} m \\ b \end{pmatrix} \right)^2$$

$$= \| \underbrace{\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}}_{\mathbf{b}} - \underbrace{\begin{pmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{pmatrix}}_{\mathbf{A}} \underbrace{\begin{pmatrix} m \\ b \end{pmatrix}}_{\mathbf{x}} \|^2$$

$$= \|\mathbf{b} - \mathbf{A}\mathbf{x}\|^2$$

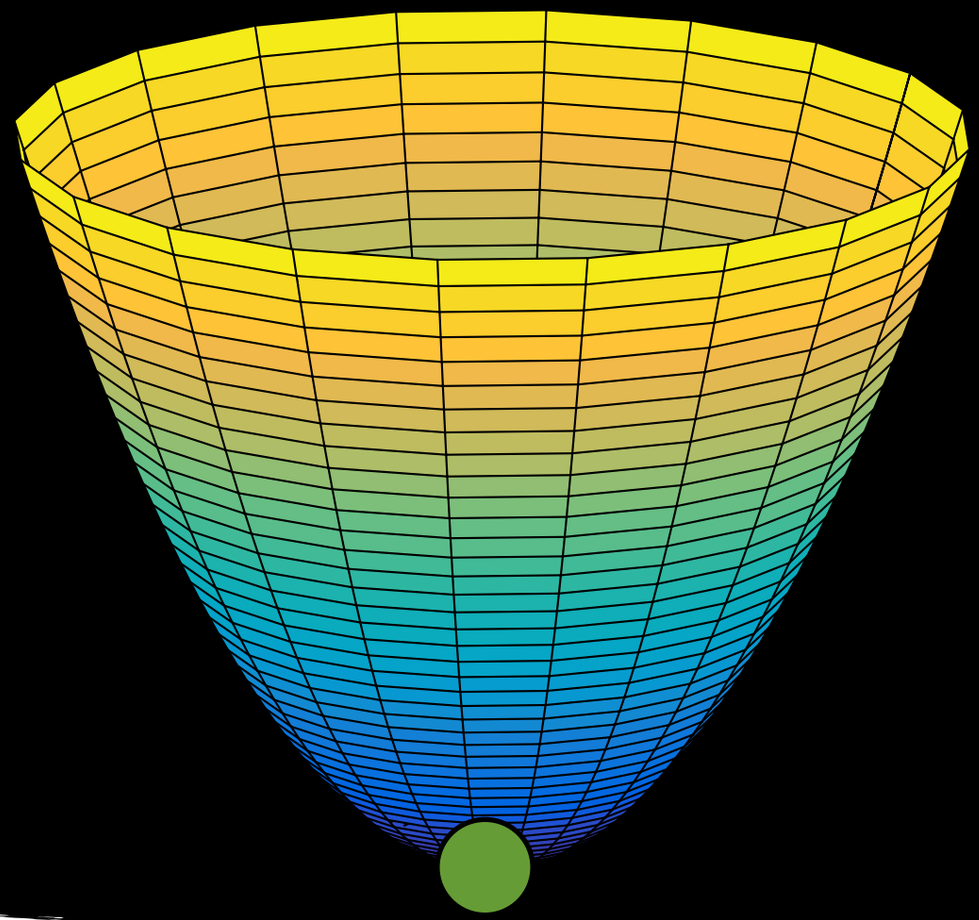
How do we minimize ϵ ?

Calculus 101

Take partial derivatives and set it to zero

$$\frac{\partial \epsilon}{\partial \mathbf{x}} = \mathbf{0}$$

Solve $\dfrac{\partial \epsilon}{\partial \mathbf{x}} = \mathbf{0}$



gradient is equal to zero at extrema

Solve $\dfrac{\partial \epsilon}{\partial \mathbf{x}} = \mathbf{0}$

$\mathbf{A}^\top \mathbf{A}\mathbf{x} = \mathbf{A}^\top \mathbf{b}$ **Normal equation**

Least Squares Solution $\mathbf{x} = (\mathbf{A}^\top \mathbf{A})^{-1} \mathbf{A}^\top \mathbf{b}$

$$\mathbf{A}^{+} = (\mathbf{A}^{\top}\mathbf{A})^{-1}\mathbf{A}^{\top}$$

Generalized Inverse

$$\mathbf{A}^{+} = (\mathbf{A}^{\top}\mathbf{A})^{-1}\mathbf{A}^{\top}$$

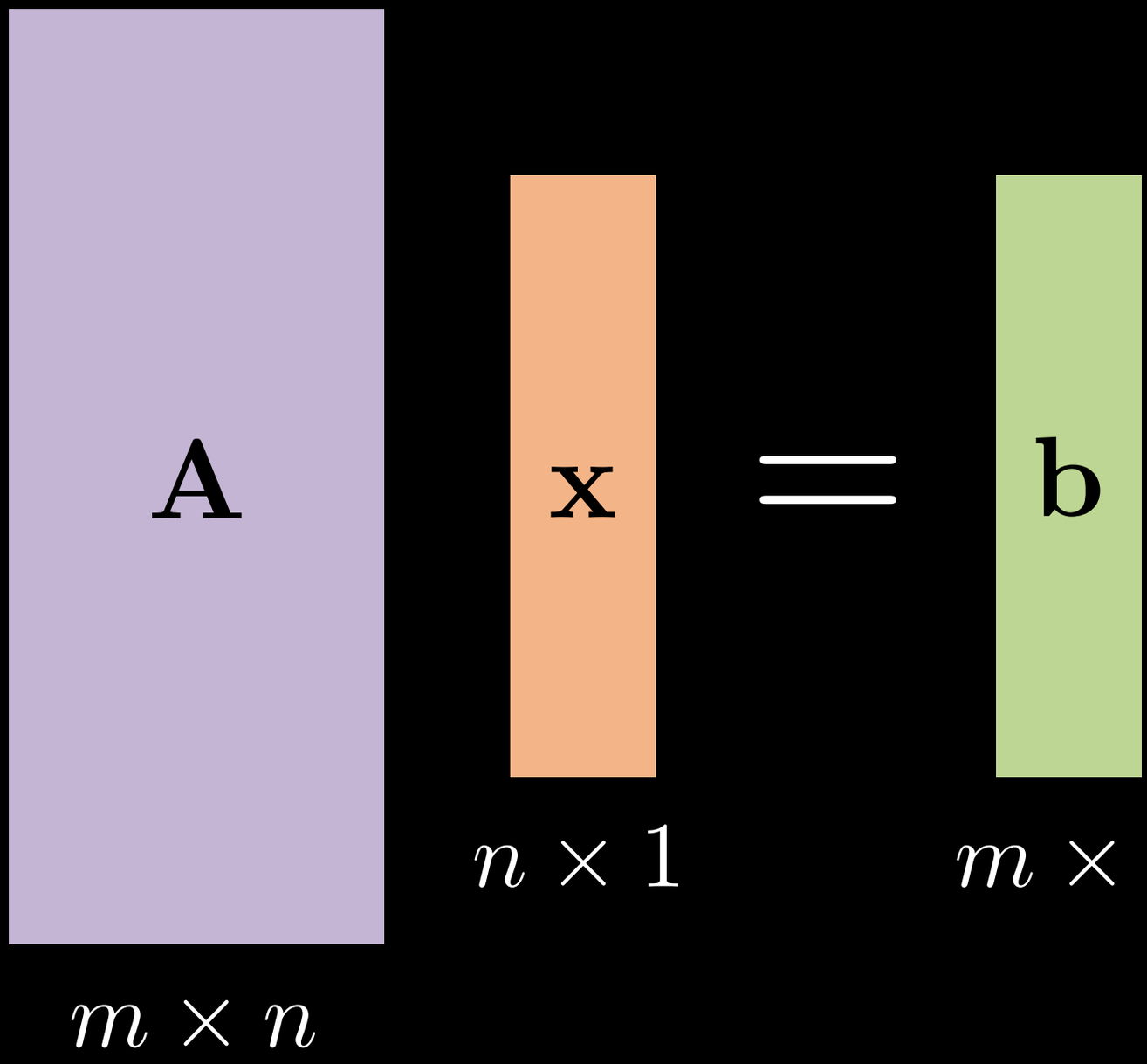
Moore-Penrose Inverse

Solve $\dfrac{\partial \epsilon}{\partial \mathbf{x}} = \mathbf{0}$

$\mathbf{A}^\top \mathbf{A}\mathbf{x} = \mathbf{A}^\top \mathbf{b}$ **Normal equation**

Least Squares Solution $\mathbf{x} = (\mathbf{A}^\top \mathbf{A})^{-1} \mathbf{A}^\top \mathbf{b}$

pseudo-inverse of $\mathbf{A}$

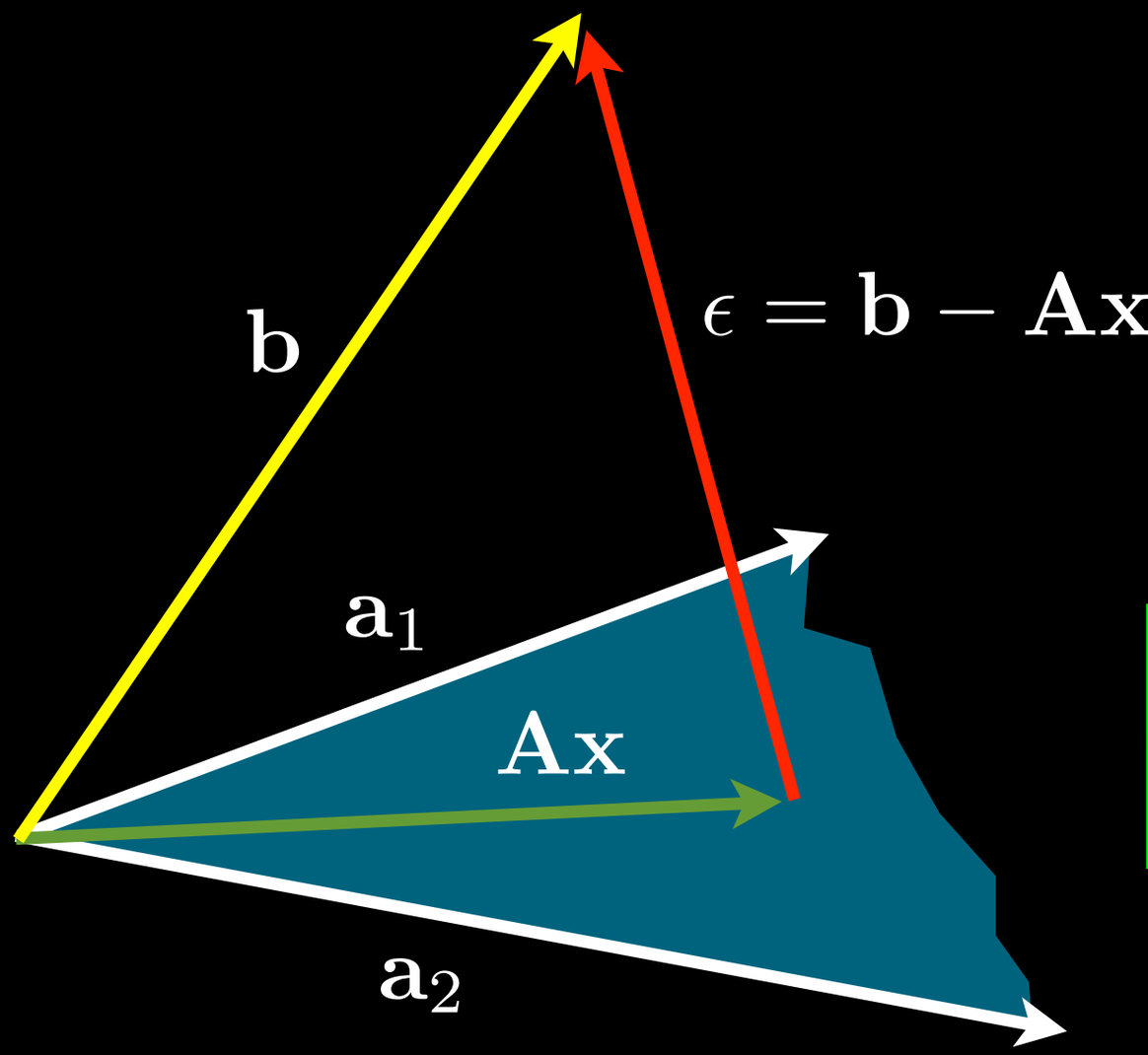


$$\mathbf{x}^* = \arg\min_{\mathbf{x}} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2 = (\mathbf{A}^\top \mathbf{A})^{-1} \mathbf{A}^\top \mathbf{b}$$

Goal: Make $\|\mathrm{b} - \mathrm{A}\mathrm{x}\|$ as small as possible

Goal: Make $\|b - Ax\|$ as small as possible

b is not the interceptAx = bin the exact solution b w

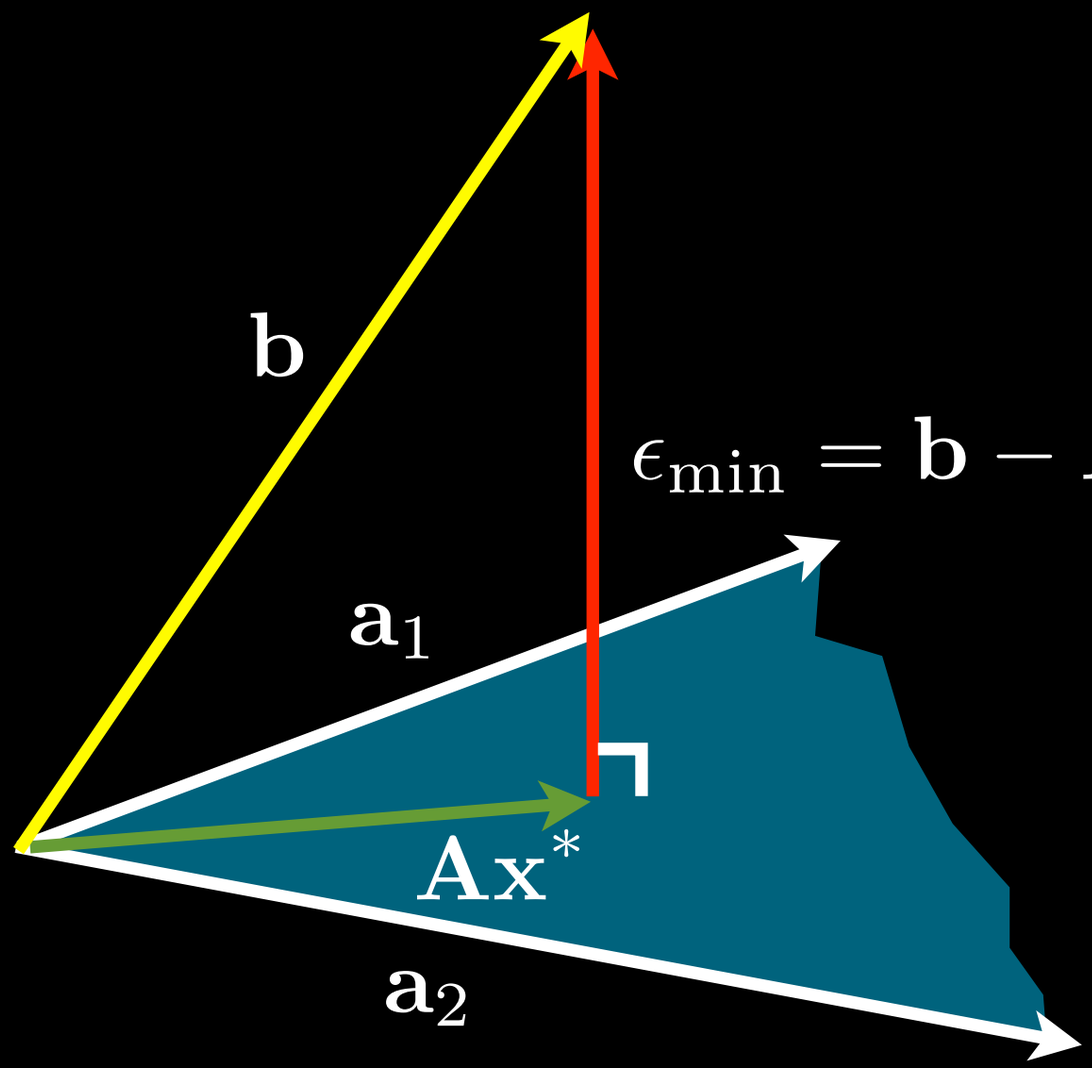


a1 and a2 are the cols of big A,they define a p

How do we minimize the length of ϵ ?

Choose the closest point to b in the plane

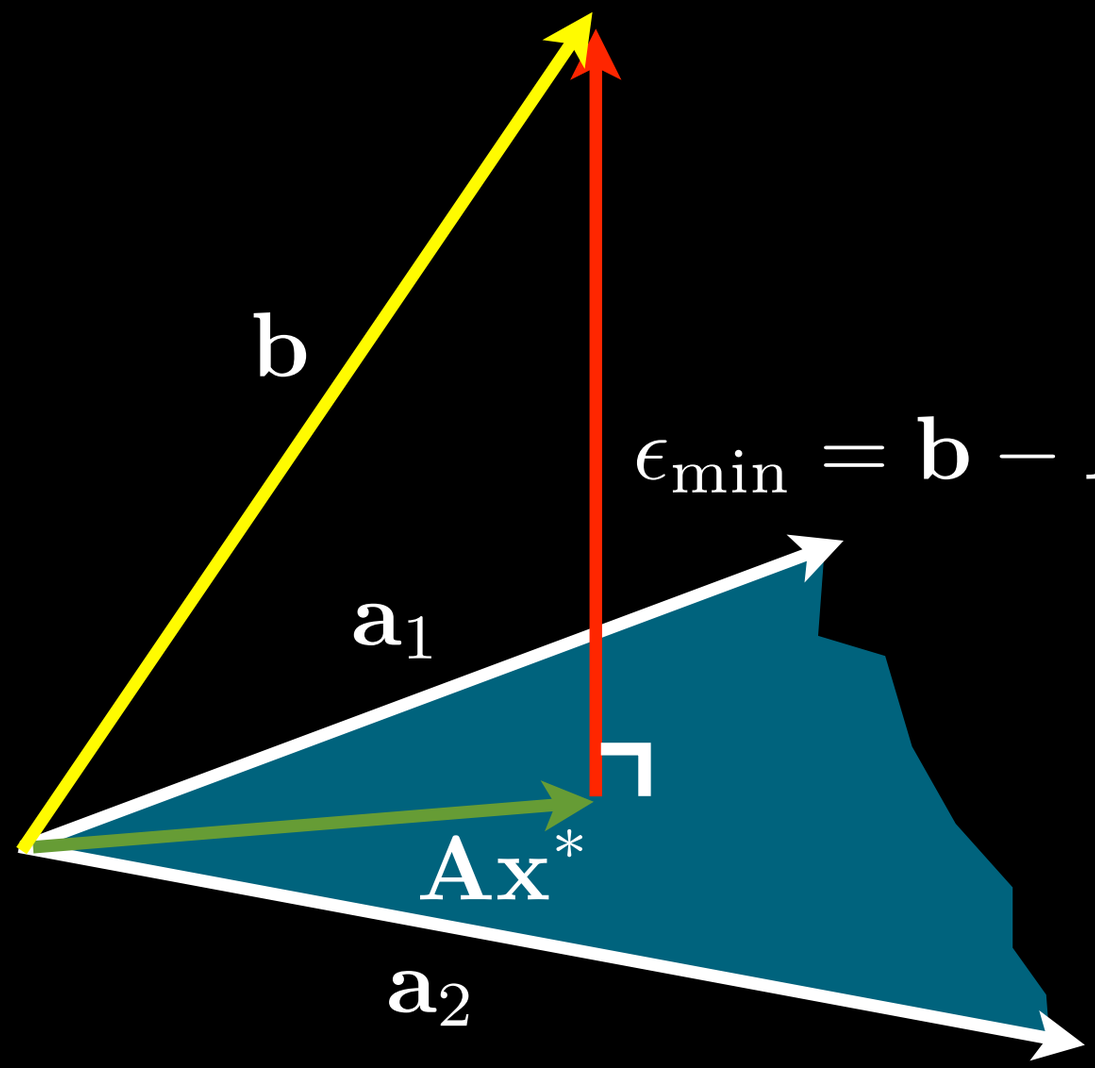
Goal: Make $\|\mathbf{b} - \mathbf{A}\mathbf{x}\|$ as small as possible



$$\mathbf{A}^\top(\mathbf{b} - \mathbf{A}\mathbf{x}) = \mathbf{0}$$

expand this equation out and you get the pseudo inverse. basically solve for x and you

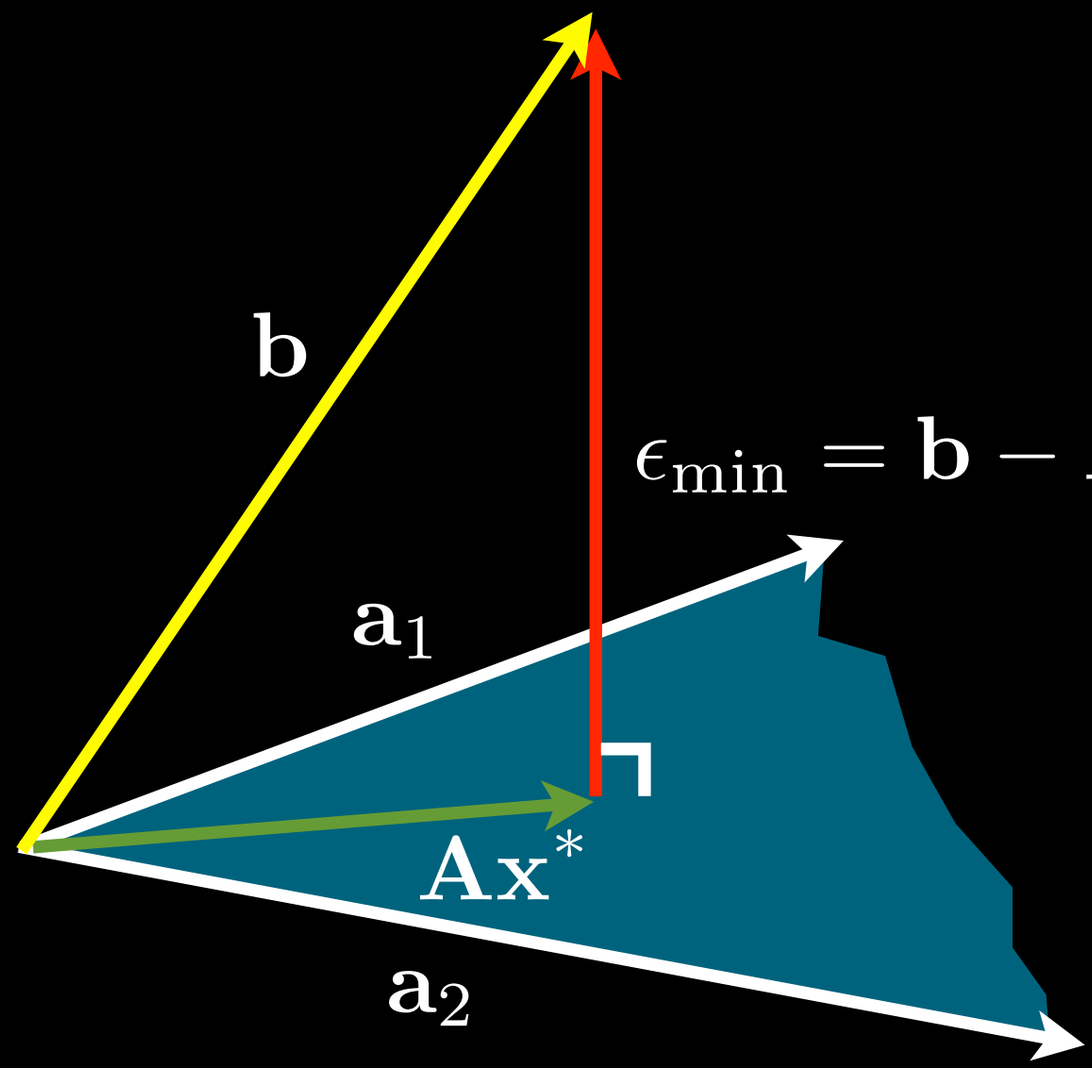
Goal: Make $\|\mathbf{b} - \mathbf{A}\mathbf{x}\|$ as small as possible



$$\mathbf{x} = (\mathbf{A}^\top \mathbf{A})^{-1} \mathbf{A}^\top \mathbf{b}$$

Does this look familiar?

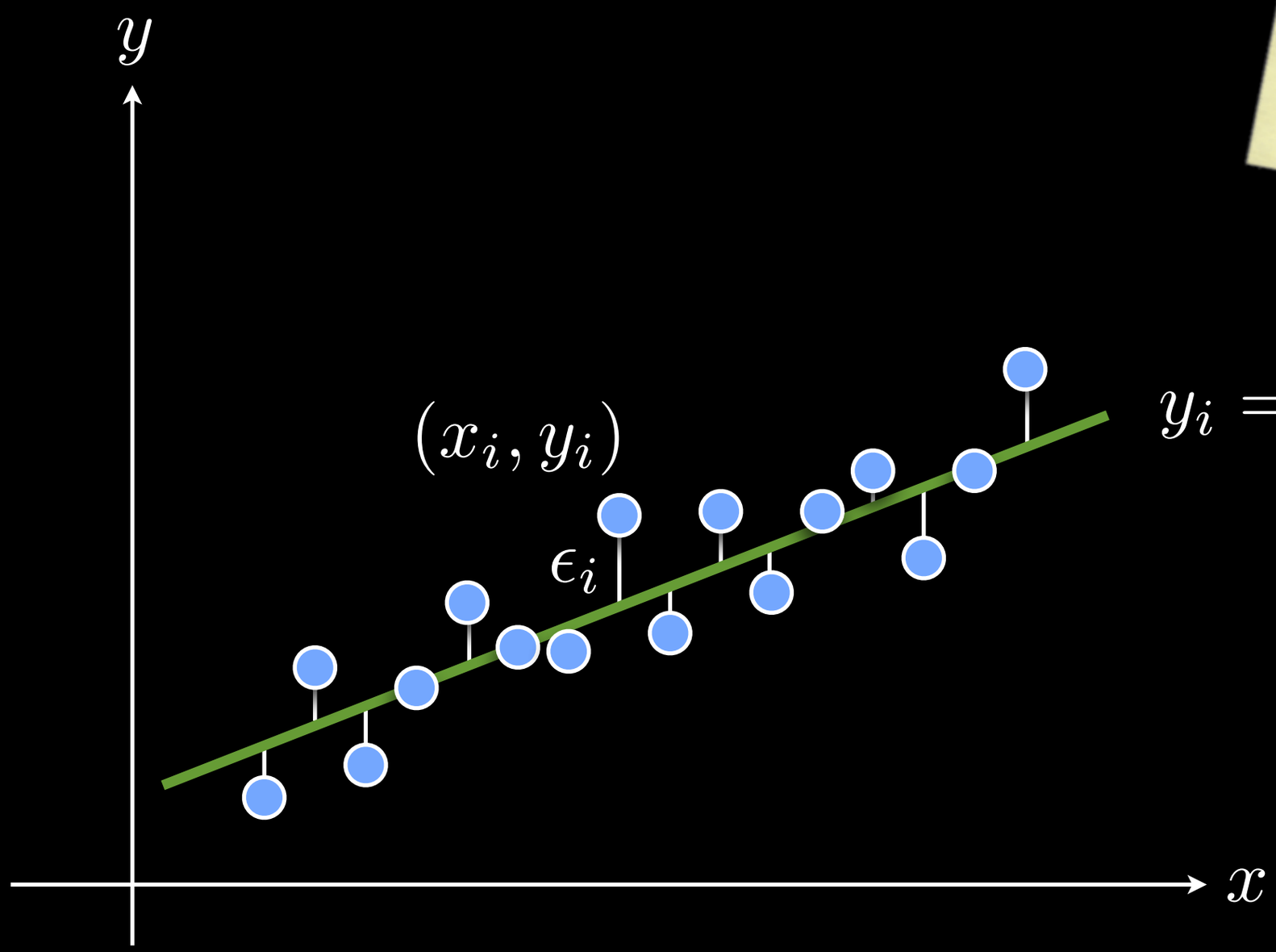
Goal: Make $\|\mathbf{b} - \mathbf{A}\mathbf{x}\|$ as small as possible



Least Squares Solution

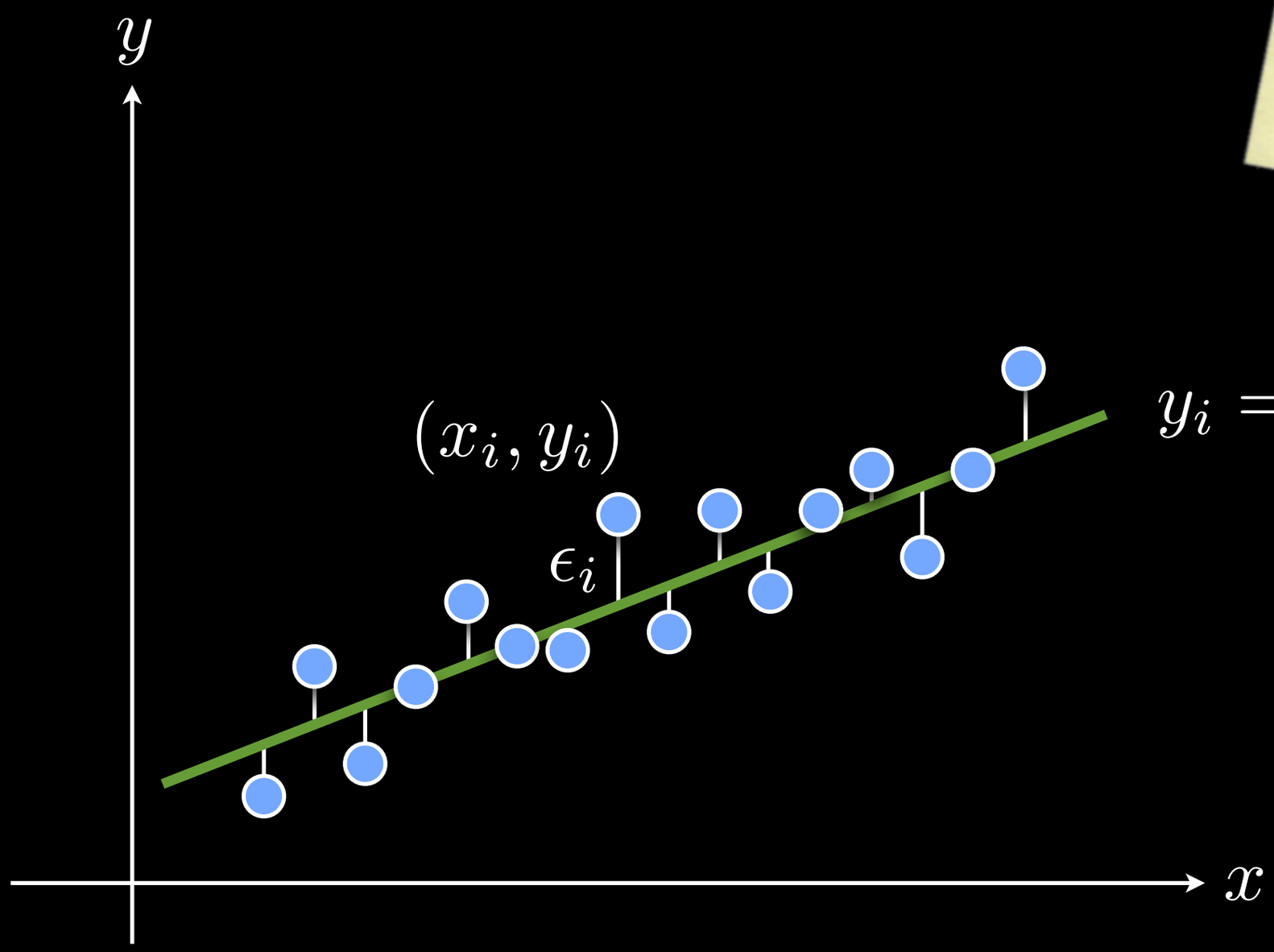
$$\mathbf{x} = (\mathbf{A}^\top \mathbf{A})^{-1} \mathbf{A}^\top \mathbf{b}$$

Issues



Not rotation invariant

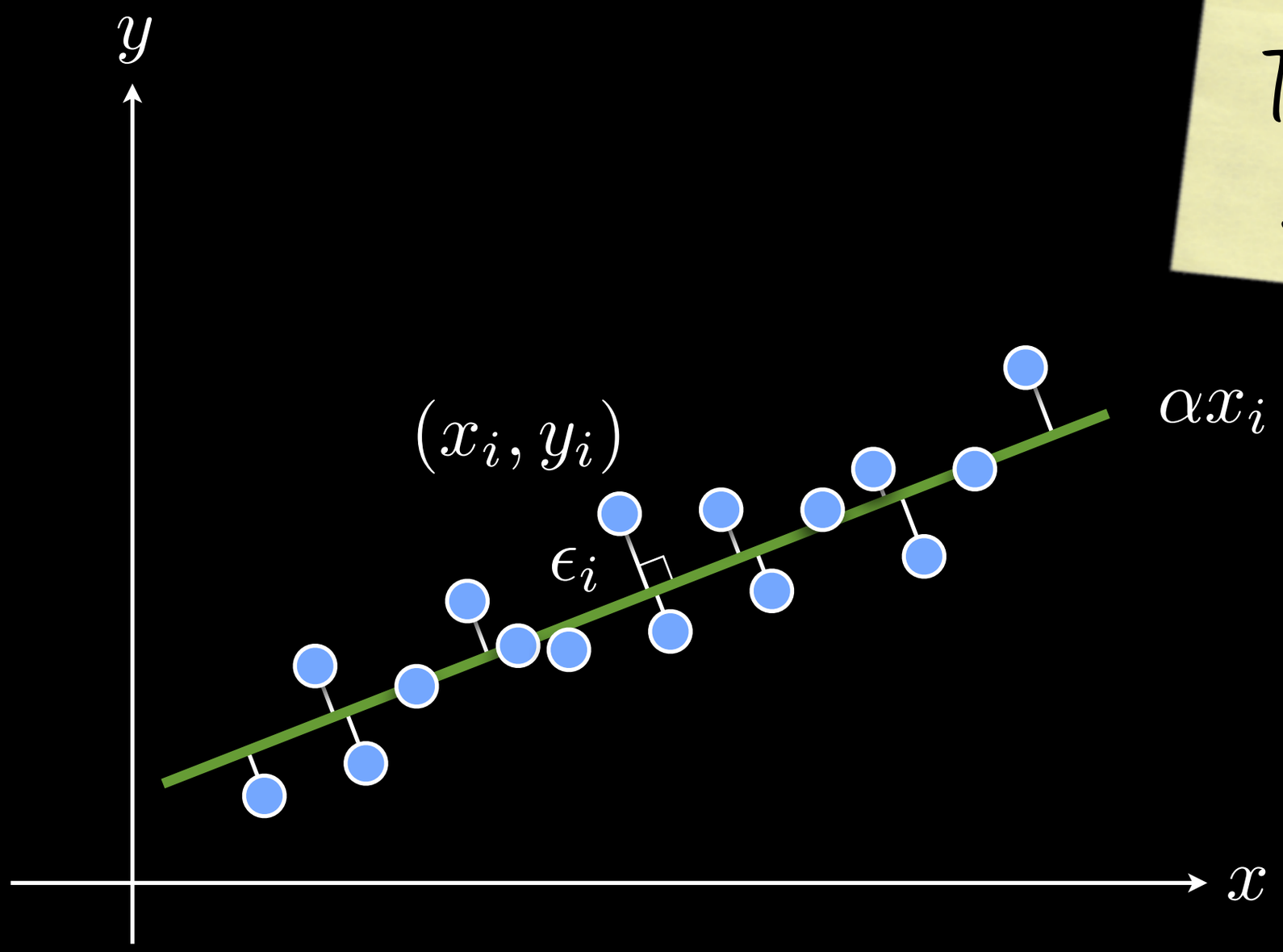
Issues



Not rotation invariant

Fails completely for vertical lines

Total-least Squares



$$\arg\min_{(\alpha,\beta,\gamma)} \sum_{i=1}^{N} (\alpha x_i + \beta y_i - \gamma)^2 \quad \text{subject to} \quad \|(\alpha, \beta)^\top\| = 1$$

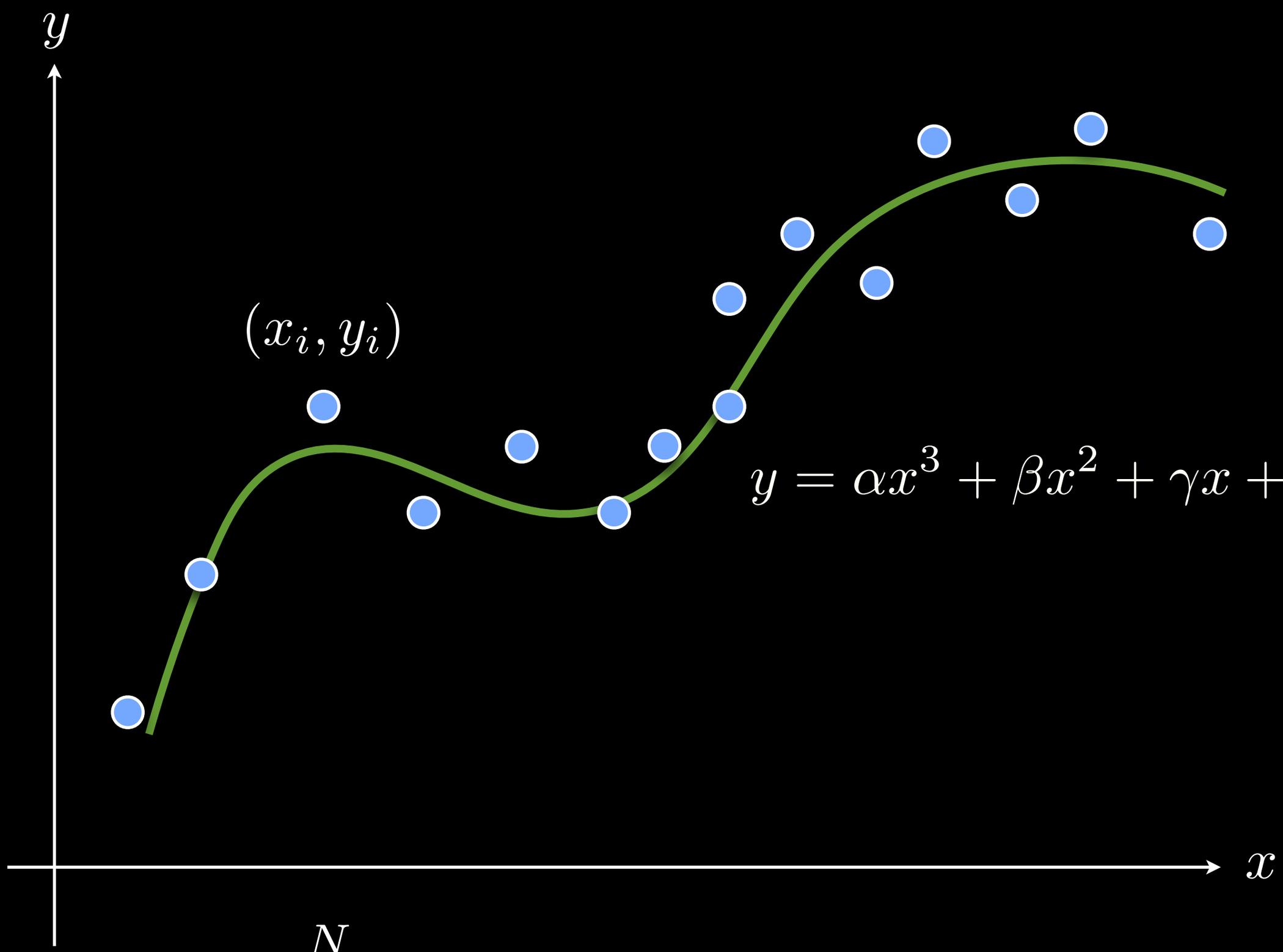
polynomial fitting

y

(x_i, y_i)

$y = \alpha x^3 + \beta x^2 + \gamma x + \delta$

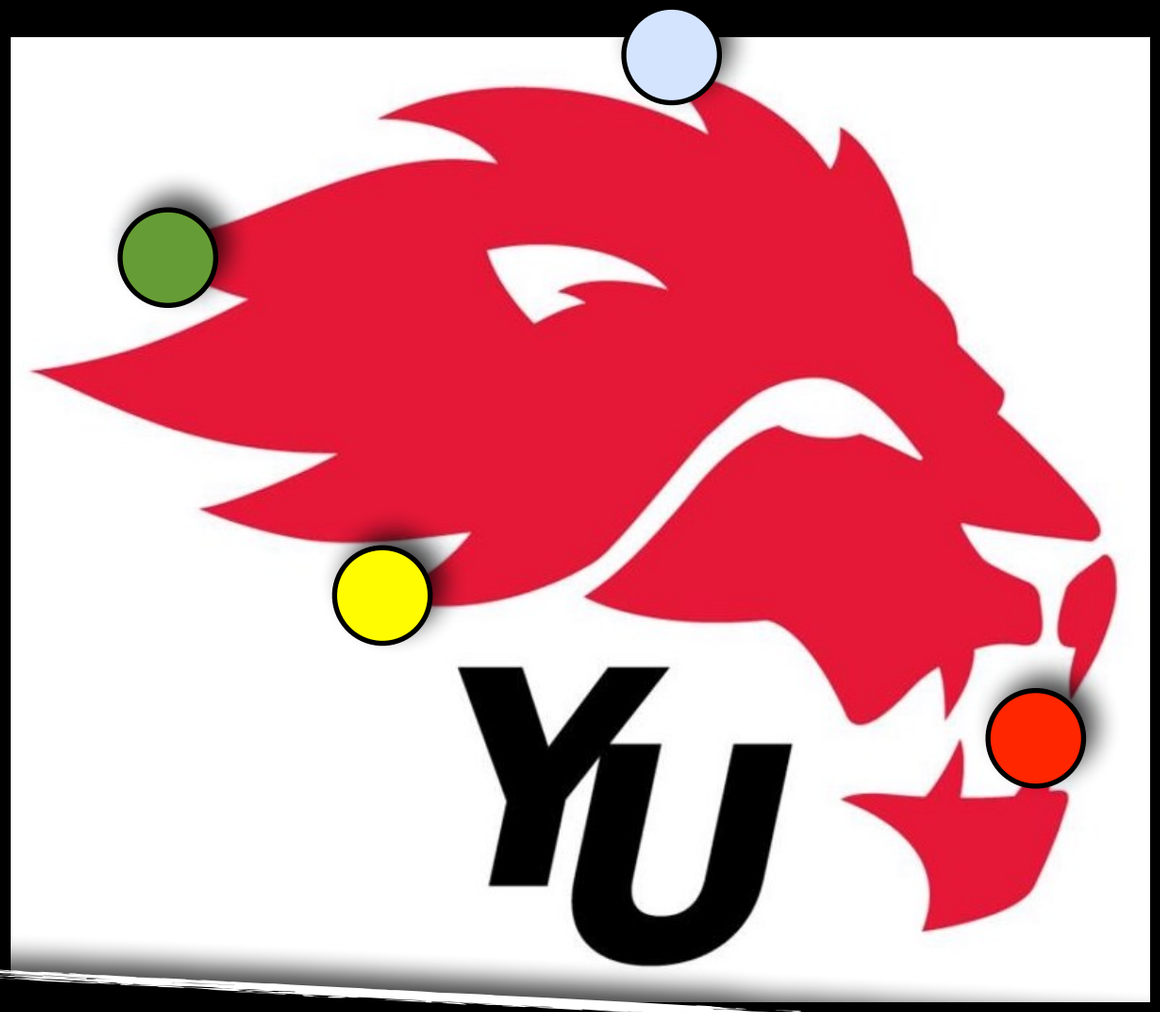
x



$$\epsilon = \sum_{i=1}^{N} (y_i - \alpha x_i^3 - \beta x_i^2 - \gamma x_i - \delta)^2$$

What about estimating geometric transformations?

$$\mathbf{x}_i = (x_i, y_i)$$



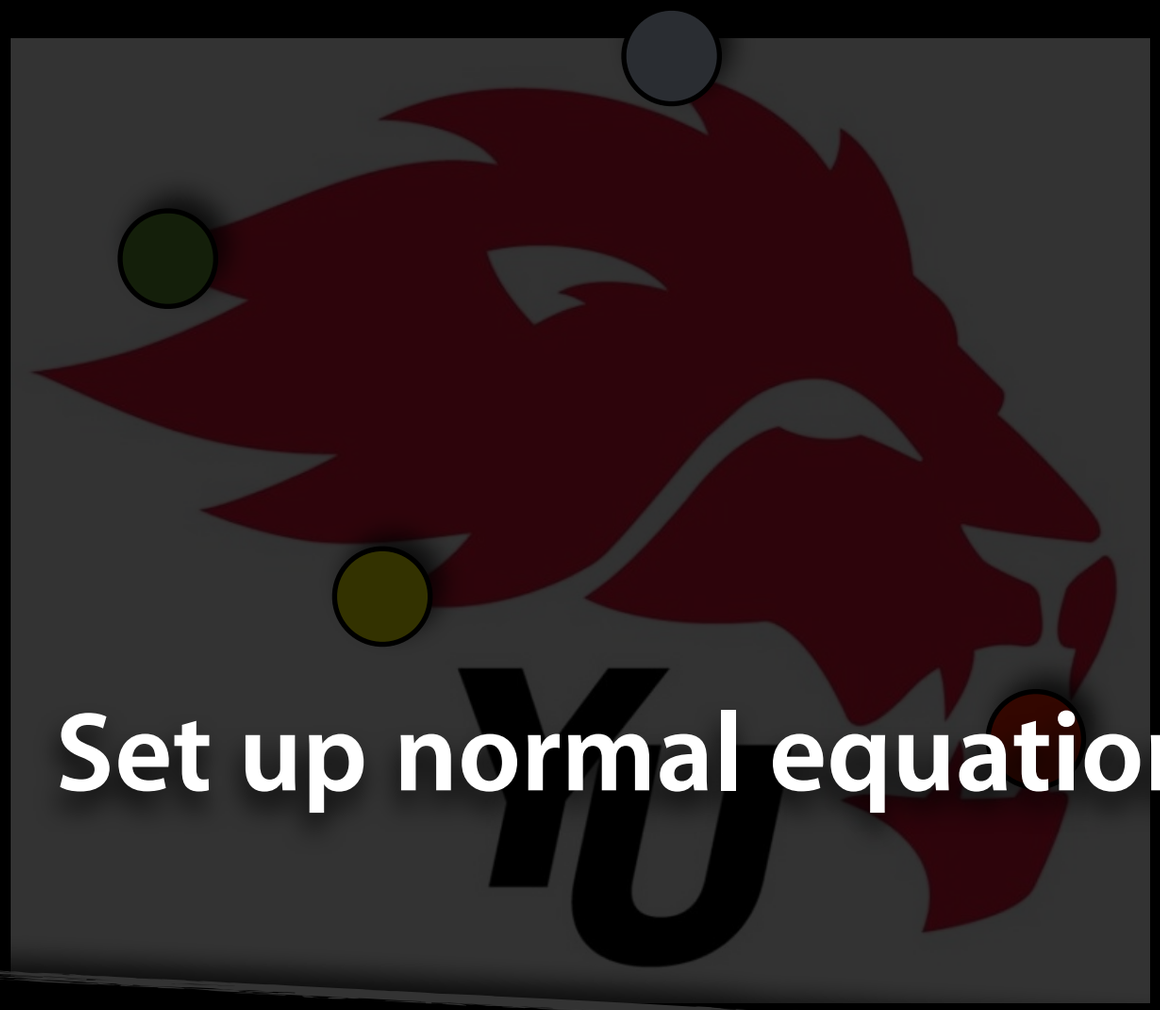
$$\mathbf{x}'_i = \mathbf{T}\mathbf{x}_i + \mathbf{t}$$

Estimate the mapping using least-squares fitting

$$\mathbf{T}^*, \mathbf{t}^* = \arg\min_{\mathbf{T},\mathbf{t}} \sum_{i=1}^{N} \|\mathbf{x}'_i - \mathbf{T}\mathbf{x}_i - \mathbf{t}\|^2$$

$$\mathbf{x}_i = (x_i, y_i)$$

$$\mathbf{x}'_i = \mathbf{T}\mathbf{x}_i + \mathbf{t}$$



Set up normal equation and solve for unknowns

Estimate the mapping using least-squares fitting

$$\mathbf{T}^*, \mathbf{t}^* = \arg\min_{\mathbf{T},\mathbf{t}} \sum_{i=1}^{N} \|\mathbf{x}'_i - \mathbf{T}\mathbf{x}_i - \mathbf{t}\|^2$$

Least-squares with translation

$$\begin{pmatrix} x_i' \\ y_i' \end{pmatrix} = \begin{pmatrix} x_i \\ y_i \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix}$$

rewrite as a linear transformation of the unknowns

Least-squares with translation

$$\begin{pmatrix} x'_i \\ y'_i \end{pmatrix} = \begin{pmatrix} x_i \\ y_i \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix}$$

rewrite as a linear transformation of the unknowns

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \\ & \vdots \\ 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} t_x \\ t_y \end{pmatrix} = \begin{pmatrix} x'_1 - x_1 \\ y'_1 - y_1 \\ \vdots \\ x'_N - x_N \\ y'_N - y_N \end{pmatrix}$$

Least-squares with translation

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \\ & \vdots \\ 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} t_x \\ t_y \end{pmatrix} = \begin{pmatrix} x'_1 - x_1 \\ y'_1 - y_1 \\ \vdots \\ x'_N - x_N \\ y'_N - y_N \end{pmatrix}$$

What is the minimum number of points to solve?

Least-squares with translation

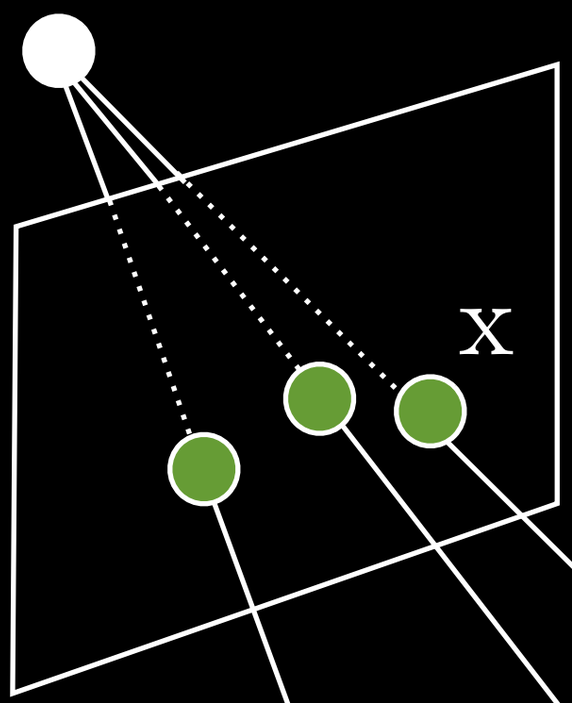
$$\underbrace{\begin{pmatrix} 1 & 0 \\ 0 & 1 \\ \vdots & \\ 1 & 0 \\ 0 & 1 \end{pmatrix}}_{\mathbf{A}} \underbrace{\begin{pmatrix} t_x \\ t_y \end{pmatrix}}_{\mathbf{x}} = \underbrace{\begin{pmatrix} x'_1 - x_1 \\ y'_1 - y_1 \\ \vdots \\ x'_N - x_N \\ y'_N - y_N \end{pmatrix}}_{\mathbf{b}}$$

Set up normal equation and solve for unknowns

$$\mathbf{x} = (\mathbf{A}^\top \mathbf{A})^{-1} \mathbf{A}^\top \mathbf{b}$$

image

Case 1



$$w \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix}$$

$$\mathrm{X} = \begin{pmatrix} X \\ Y \\ 0 \end{pmatrix}$$

world plane

image 1

$$w \begin{pmatrix} x_1 \\ y_1 \\ 1 \end{pmatrix} = \mathbf{H} \begin{pmatrix} x_2 \\ y_2 \\ 1 \end{pmatrix}$$

image 2

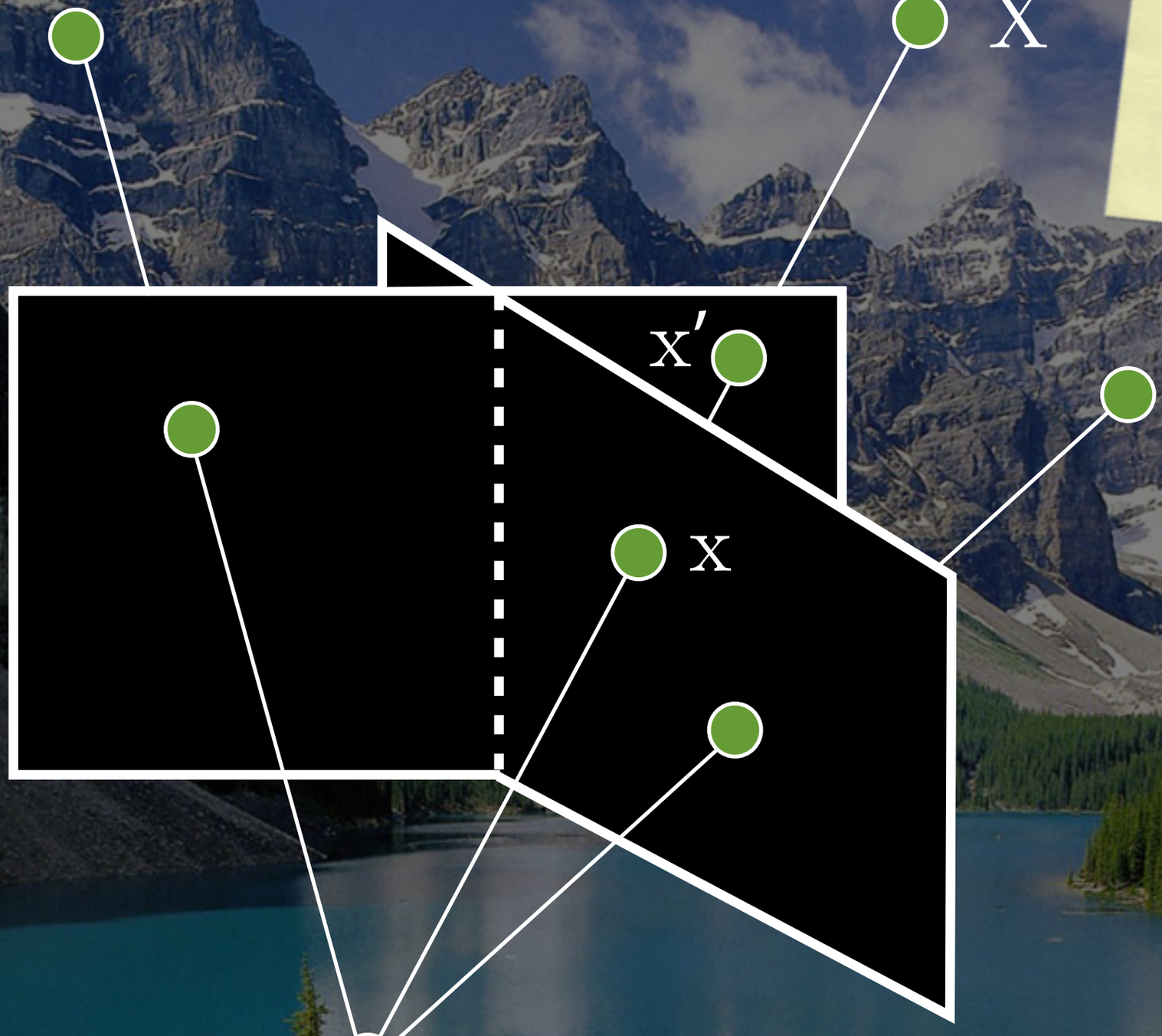
x

x′

X

Case 2

Case 3



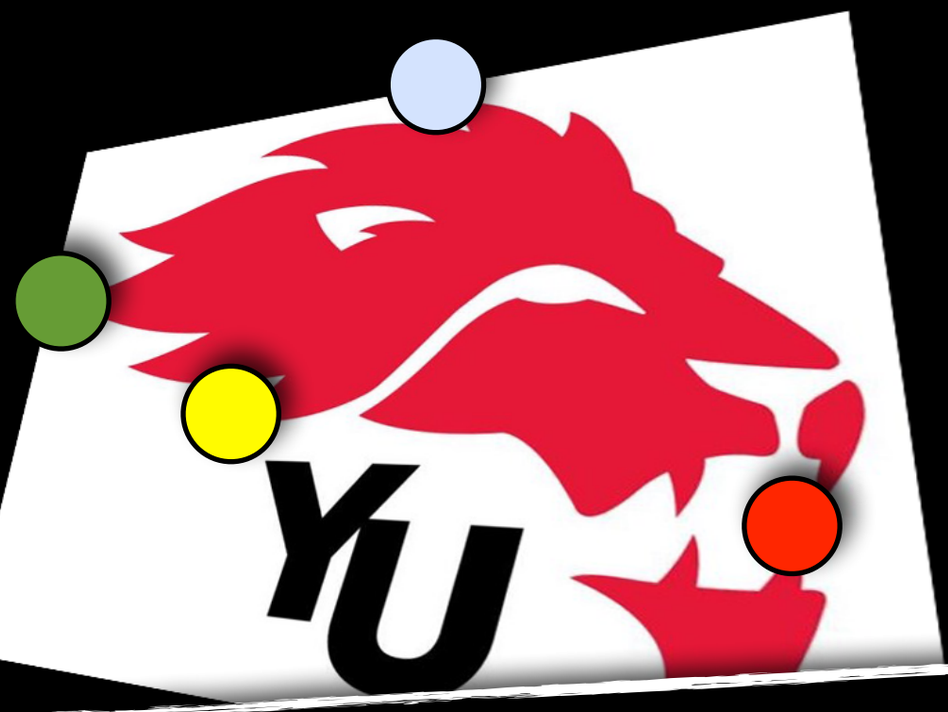
rotate about camera

$$w \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \mathbf{H} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Homography

cannot be estimated with pseudoinverse

$$\mathbf{x}'_i = \begin{pmatrix} \frac{h_{11}x_i + h_{12}y_i + h_{13}}{h_{31}x_i + h_{32}y_i + h_{33}} \\ \frac{h_{21}x_i + h_{22}y_i + h_{23}}{h_{31}x_i + h_{32}y_i + h_{33}} \end{pmatrix}$$



rewrite as homogeneous equations

$$\begin{pmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1x'_1 & -y_1x'_1 & -x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1y'_1 & -y_1y'_1 & -y'_1 \\ & & & & & \vdots & & & \\ x_N & y_N & 1 & 0 & 0 & 0 & -x_Nx'_N & -y_Nx'_N & -x'_N \\ 0 & 0 & 0 & x_N & y_N & 1 & -x_Ny'_N & -y_Ny'_N & -y'_N \end{pmatrix} \mathbf{h} = 0$$

$$\begin{pmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1x'_1 & -y_1x'_1 & -x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1y'_1 & -y_1y'_1 & -y'_1 \\ & & & & & \vdots & & & \\ x_N & y_N & 1 & 0 & 0 & 0 & -x_Nx'_N & -y_Nx'_N & -x'_N \\ 0 & 0 & 0 & x_N & y_N & 1 & -x_Ny'_N & -y_Ny'_N & -y'_N \end{pmatrix} \mathbf{h} = 0$$

$$\begin{pmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1x'_1 & -y_1x'_1 & -x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1y'_1 & -y_1y'_1 & -y'_1 \\ & & & & & \vdots & & & \\ x_N & y_N & 1 & 0 & 0 & 0 & -x_Nx'_N & -y_Nx'_N & -x'_N \\ 0 & 0 & 0 & x_N & y_N & 1 & -x_Ny'_N & -y_Ny'_N & -y'_N \end{pmatrix} \mathbf{h} = 0$$

A minimum of four points is required to estimate $\mathbf{h}$

$$\begin{pmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1x'_1 & -y_1x'_1 & -x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1y'_1 & -y_1y'_1 & -y'_1 \\ & & & & \vdots & & & & \\ x_N & y_N & 1 & 0 & 0 & 0 & -x_Nx'_N & -y_Nx'_N & -x'_N \\ 0 & 0 & 0 & x_N & y_N & 1 & -x_Ny'_N & -y_Ny'_N & -y'_N \end{pmatrix} \mathbf{h} = 0$$

Estimate the mapping using least-squares fitting

$$\begin{pmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1x'_1 & -y_1x'_1 & -x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1y'_1 & -y_1y'_1 & -y'_1 \\ & & & & \vdots & & & & \\ x_N & y_N & 1 & 0 & 0 & 0 & -x_Nx'_N & -y_Nx'_N & -x'_N \\ 0 & 0 & 0 & x_N & y_N & 1 & -x_Ny'_N & -y_Ny'_N & -y'_N \end{pmatrix} \mathbf{h} = 0$$

Estimate the mapping using least-squares fitting

$$\mathbf{h}^* = \arg\min_{\mathbf{h}} \|\mathbf{A}\mathbf{h}\|^2$$

$$\begin{pmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1x'_1 & -y_1x'_1 & -x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1y'_1 & -y_1y'_1 & -y'_1 \\ & & & & \vdots & & & & \\ x_N & y_N & 1 & 0 & 0 & 0 & -x_Nx'_N & -y_Nx'_N & -x'_N \\ 0 & 0 & 0 & x_N & y_N & 1 & -x_Ny'_N & -y_Ny'_N & -y'_N \end{pmatrix} \mathbf{h} = 0$$

Estimate the mapping using least-squares fitting

$$\mathbf{h}^* = \arg\min_{\mathbf{h}} \|\mathbf{A}\mathbf{h}\|^2$$

Cannot be estimated with pseudoinverse

$$\begin{pmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1x'_1 & -y_1x'_1 & -x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1y'_1 & -y_1y'_1 & -y'_1 \\ & & & & \vdots & & & & \\ x_N & y_N & 1 & 0 & 0 & 0 & -x_Nx'_N & -y_Nx'_N & -x'_N \\ 0 & 0 & 0 & x_N & y_N & 1 & -x_Ny'_N & -y_Ny'_N & -y'_N \end{pmatrix} \mathbf{h} = 0$$

Estimate the mapping using least-squares fitting

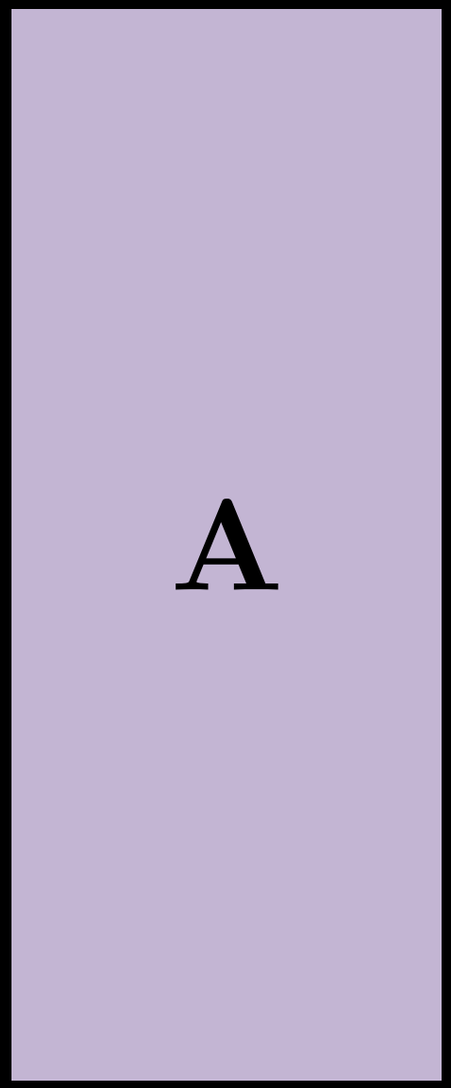
$$\mathbf{h}^* = \arg\min_{\mathbf{h}} \|\mathbf{A}\mathbf{h}\|^2 \text{ subject to } \|\mathbf{h}\| = 1$$

$$\begin{pmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1x'_1 & -y_1x'_1 & -x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1y'_1 & -y_1y'_1 & -y'_1 \\ & & & & \vdots & & & & \\ x_N & y_N & 1 & 0 & 0 & 0 & -x_Nx'_N & -y_Nx'_N & -x'_N \\ 0 & 0 & 0 & x_N & y_N & 1 & -x_Ny'_N & -y_Ny'_N & -y'_N \end{pmatrix} \mathbf{h} = 0$$

Estimate the mapping using least-squares fitting

$$\mathbf{h}^* = \arg\min_{\mathbf{h}} \|\mathbf{A}\mathbf{h}\|^2 \text{ subject to } \|\mathbf{h}\| = 1$$

Compute SVD of A, h^* is the last column of V



$m \times n$

$$\mathbf{A} = \mathbf{U}\,\mathbf{D}\,\mathbf{V}^\top$$

$m \times n$ $m \times n$ $n \times n$ $n \times n$

SVD

$$\mathbf{A} = \mathbf{U} \mathbf{D} \mathbf{V}^\top$$

$m \times n$ $m \times n$ $n \times n$ $n \times n$

Definition: For any given matrix $\mathrm{A} \in \mathbb{R}^{m \times n}$ its Singular Value Decomposition (SVD) is defined as

$$\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{V}^{\top}$$

Definition: For any given matrix $\mathrm{A} \in \mathbb{R}^{m \times n}$ its Singular Value Decomposition (SVD) is defined as

$$\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{V}^{\top}$$

such that

$\mathbf{U}$ is an $m \times n$ matrix with orthogonal columns

Definition: For any given matrix $\mathrm{A} \in \mathbb{R}^{m \times n}$ its Singular Value Decomposition (SVD) is defined as

$$\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{V}^\top$$

such that

$\mathbf{U}$ is an $m \times n$ matrix with orthogonal columns

$\mathbf{V}^\top$ is an $n \times n$ orthogonal matrix

Definition: For any given matrix $\mathrm{A} \in \mathbb{R}^{m \times n}$ its Singular Value Decomposition (SVD) is defined as

$$\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{V}^\top$$

such that

$\mathbf{U}$ is an $m \times n$ matrix with orthogonal columns

$\mathbf{V}^\top$ is an $n \times n$ orthogonal matrix

$\mathbf{D}$ is an $n \times n$ matrix with non-negative entries, termed the singular values

Definition: For any given matrix $\mathrm{A} \in \mathbb{R}^{m \times n}$ its Singular Value Decomposition (SVD) is defined as

$$\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{V}^\top$$

such that

$\mathbf{U}$ is an $m \times n$ matrix with orthogonal columns

$\mathbf{V}^\top$ is an $n \times n$ orthogonal matrix

$\mathbf{D}$ is an $n \times n$ matrix with non-negative entries, termed the singular values

Assume: The diagonal values of D are sorted in descending order, $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$

$$\begin{pmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1x'_1 & -y_1x'_1 & -x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1y'_1 & -y_1y'_1 & -y'_1 \\ & & & & \vdots & & & & \\ x_N & y_N & 1 & 0 & 0 & 0 & -x_Nx'_N & -y_Nx'_N & -x'_N \\ 0 & 0 & 0 & x_N & y_N & 1 & -x_Ny'_N & -y_Ny'_N & -y'_N \end{pmatrix} \mathbf{h} = 0$$

Estimate the mapping using least-squares fitting

$$\mathbf{h}^* = \arg\min_{\mathbf{h}} \|\mathbf{A}\mathbf{h}\|^2 \text{ subject to } \|\mathbf{h}\| = 1$$

Compute SVD of A, h^* is the last column of V

```
% --- least-squares line fitting ---

x = [1:20]';
m = 2;  % --- slope
b = 5;  % --- y-intercept

y = m*x + b;
y = y + randn(size(x));
A = [x ones(size(x))];

mb_estimate = inv(A'*A)*A'*y

% --- or equivalently

mb_estimate = A\y;

% --- or equivalently

mb_estimate = pinv(A)*y;
```

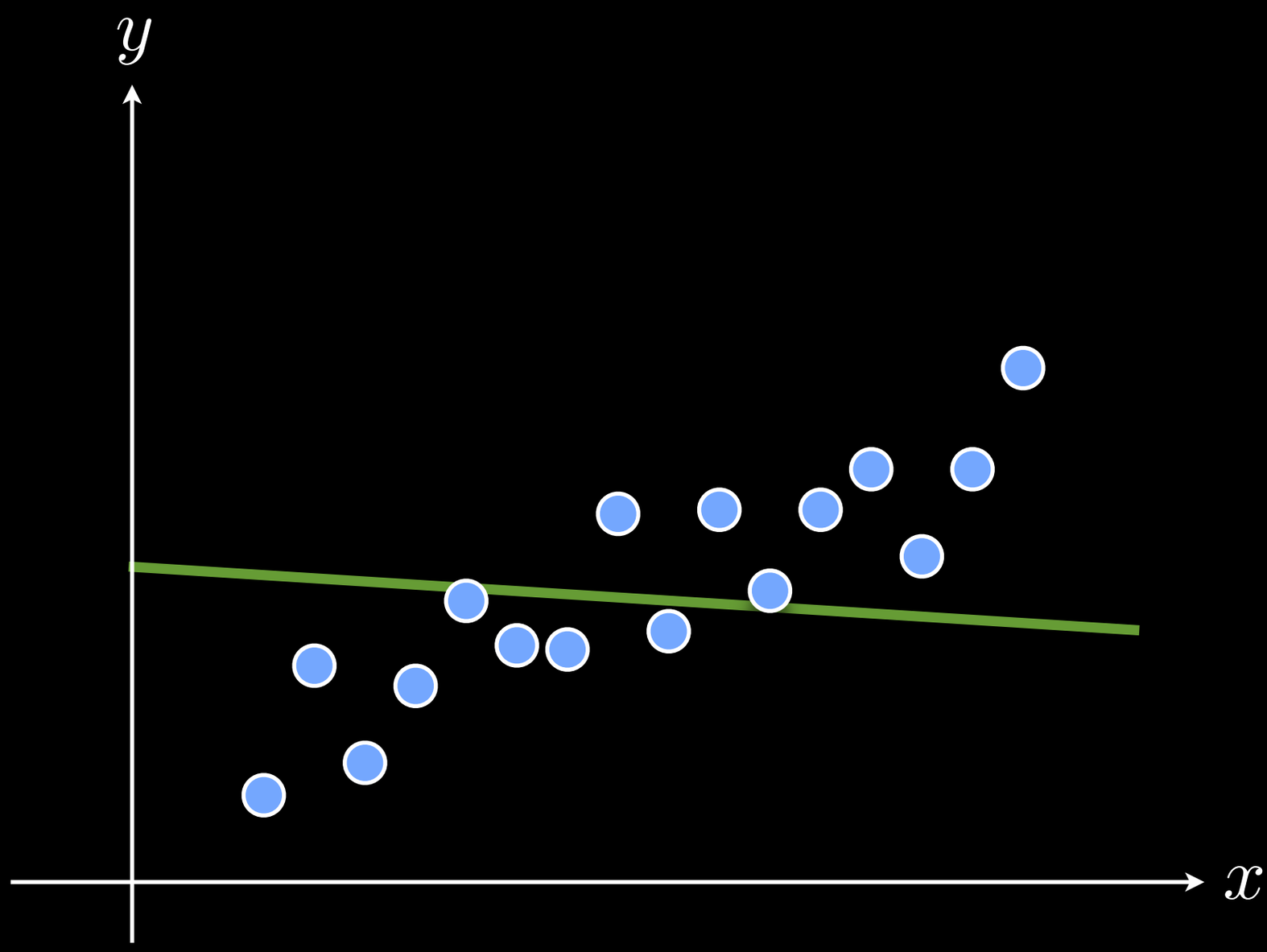
```
% --- least-squares line fitting ---


x = [1:20]';
m = 2;   % --- slope
b = 5;   % --- y-intercept


y = m*x + b;
y = y + randn(size(x));
A = [x ones(size(x))];
mb_estimate = inv(A'*A)*A'*y

% --- or equivalently

mb_estimate = A\y;

% --- or equivalently

mb_estimate = pinv(A)*y;
```

add noise to vertical component of points

```
% --- least-squares line fitting ---

x = [1:20]';
m = 2;  % --- slope
b = 5;  % --- y-intercept

y = m*x + b;
y = y + randn(size(x));
A = [x ones(size(x))];

mb_estimate = inv(A'*A)*A'*y

% --- or equivalently

mb_estimate = A\y;

% --- or equiva

mb_estimate = pinv(A)*y;
```

backslash operator

```
% --- least-squares line fitting ---

x = [1:20]';
m = 2;  % --- slope
b = 5;  % --- y-intercept

y = m*x + b;
y = y + randn(size(x));
A = [x ones(size(x))];

mb_estimate = inv(A'*A)*A'*y

% --- or equivalently

mb_estimate = A\y;

% --- or equivalently

mb_estimate = pinv(A)*y;
```



outlier

Least squares is not robust to outliers

Hough Lines

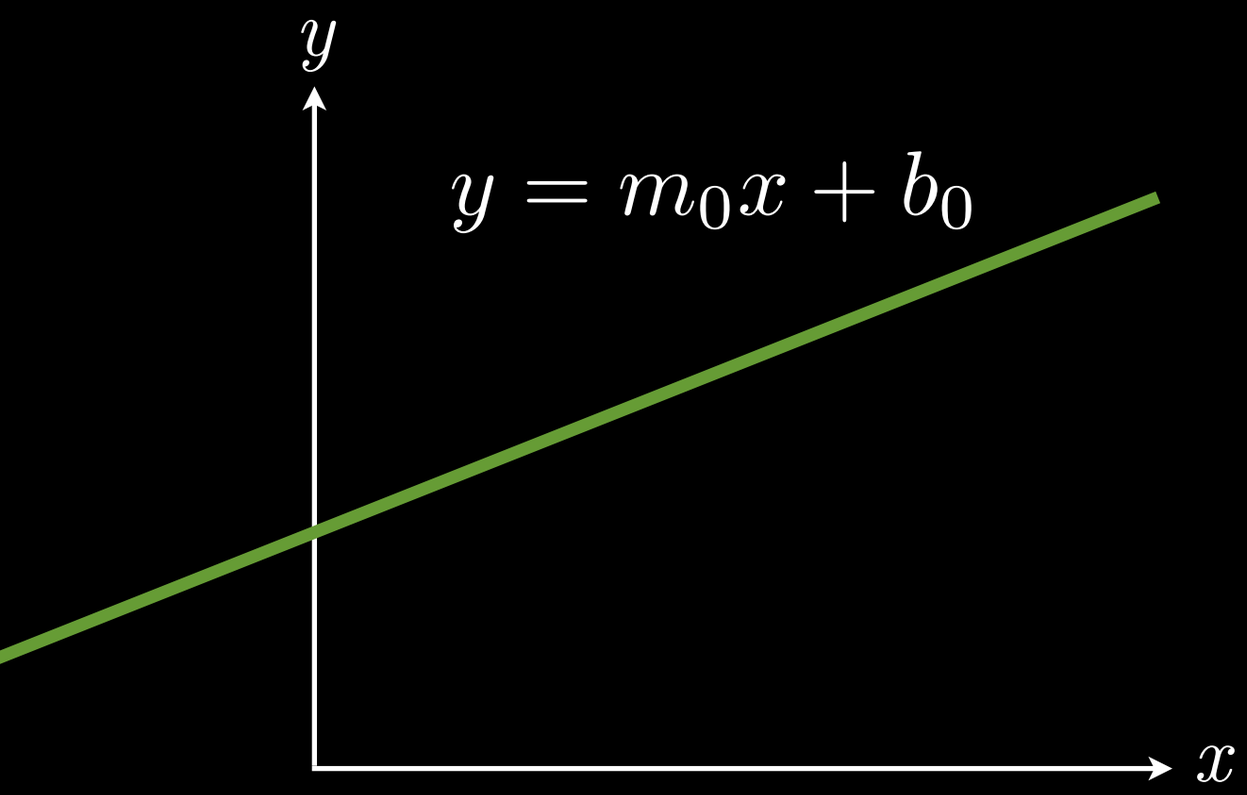
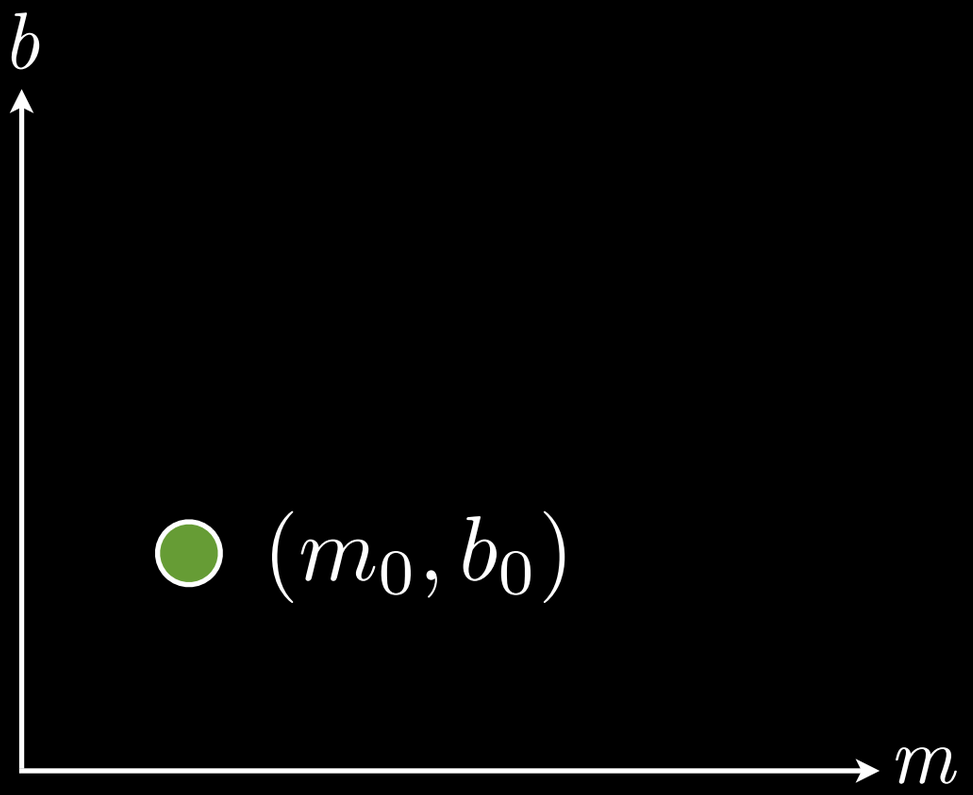


image space



Hough (parameter) space

Point-Line Duality

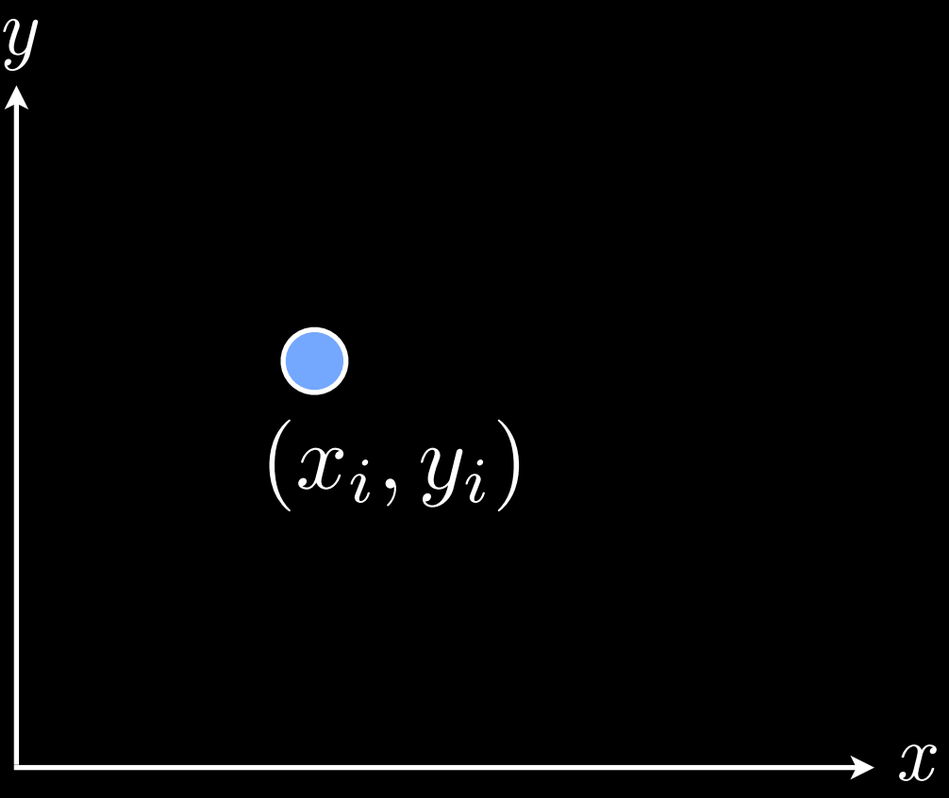
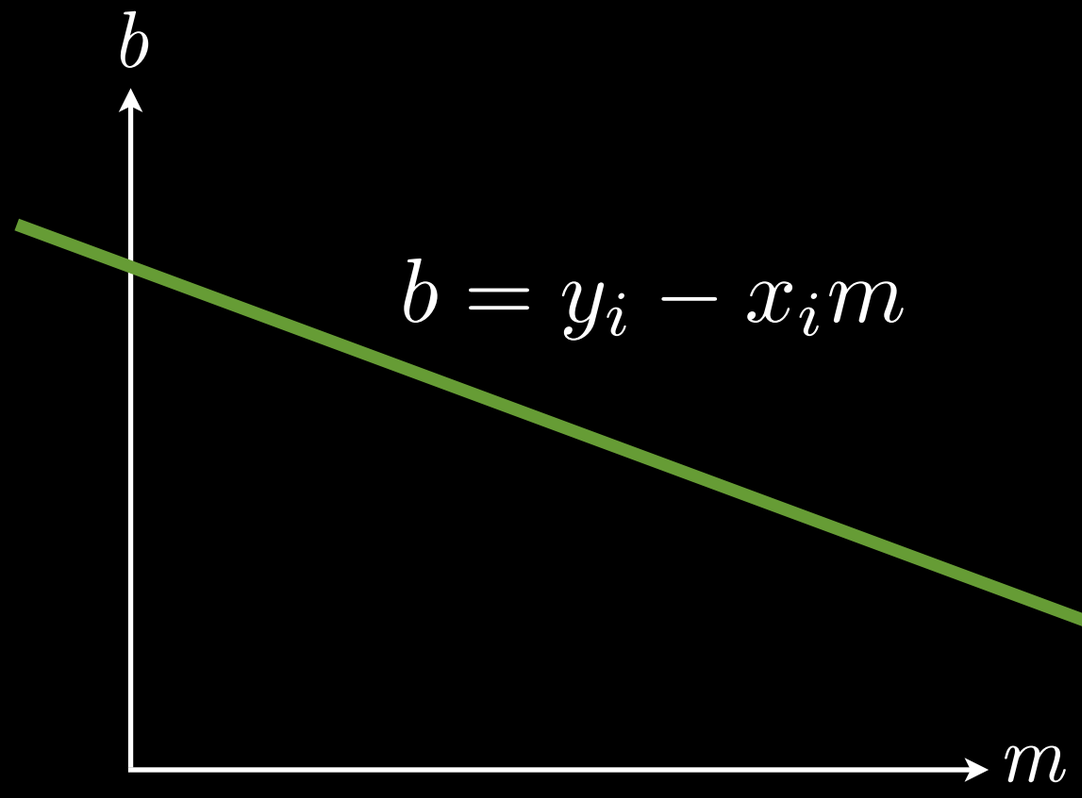


image space



Hough (parameter) space

Hough Lines

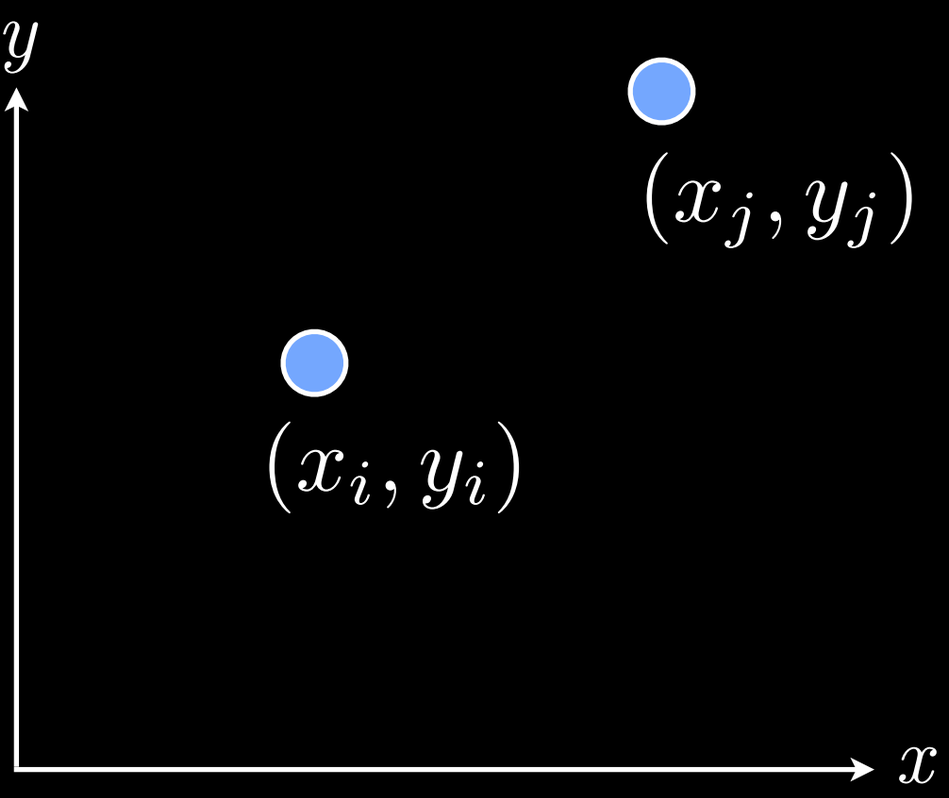
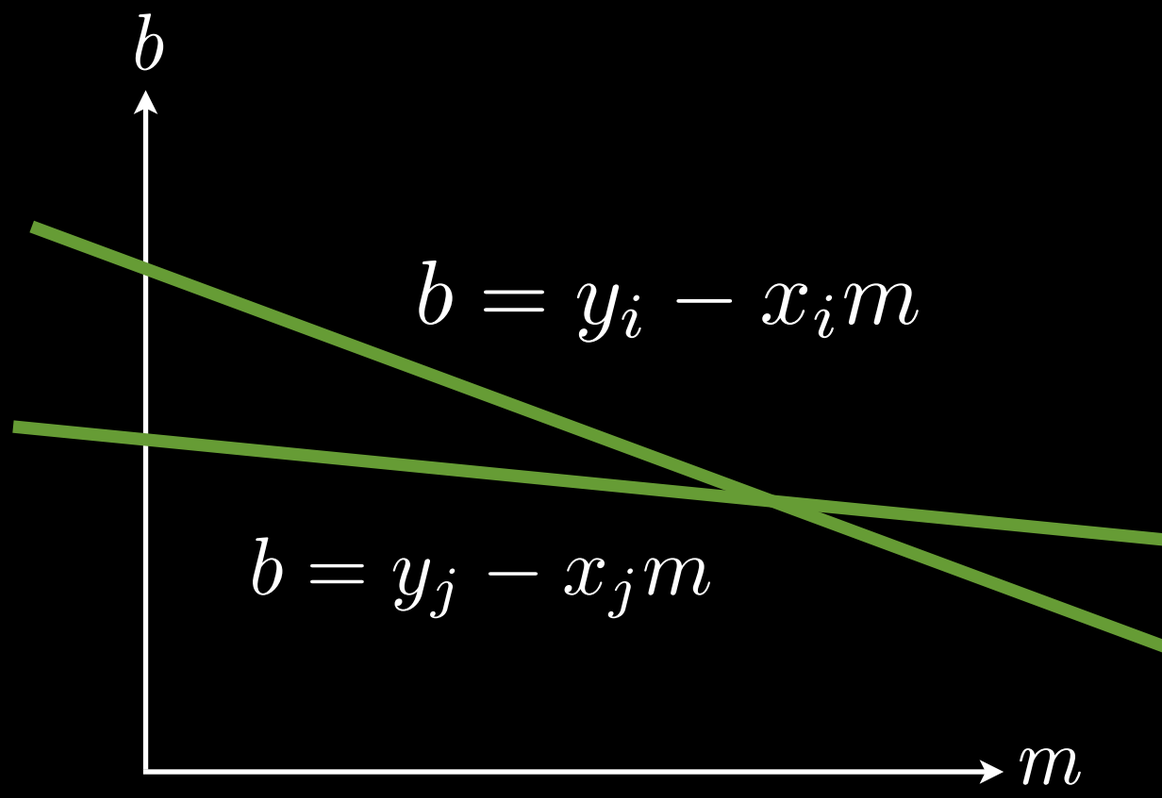
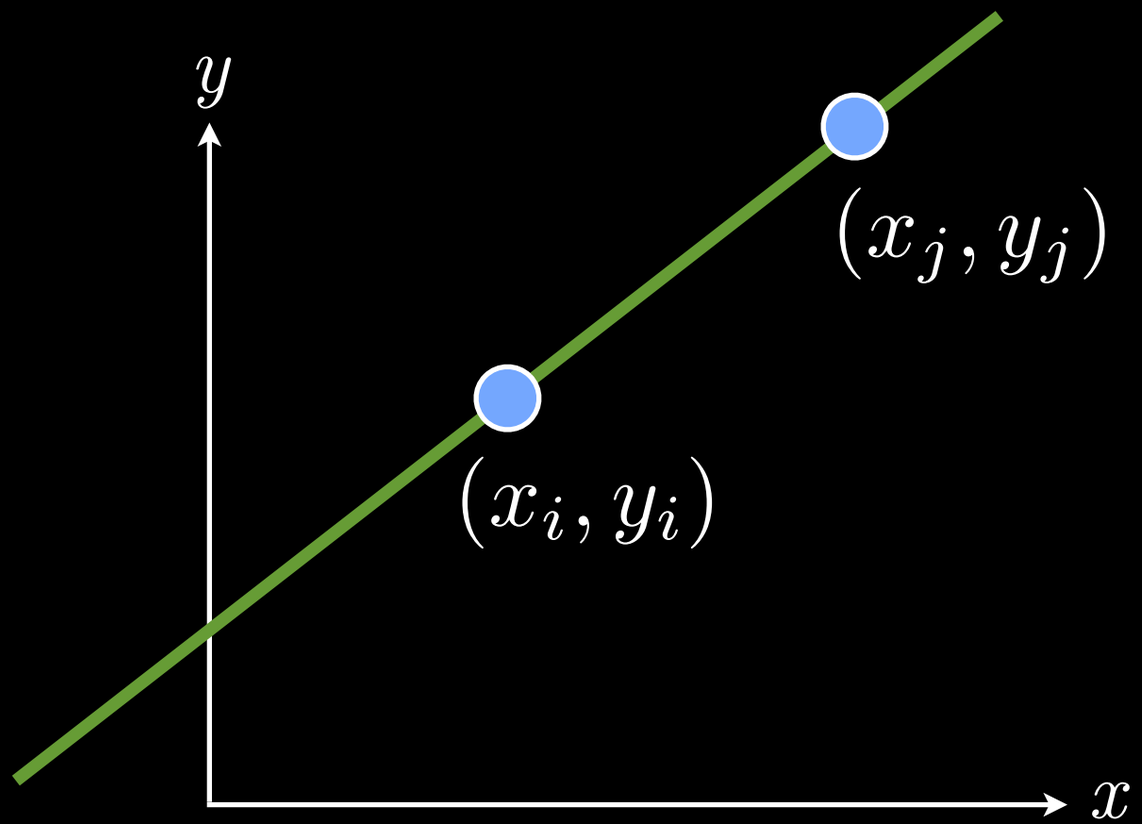


image space



Hough (parameter) space



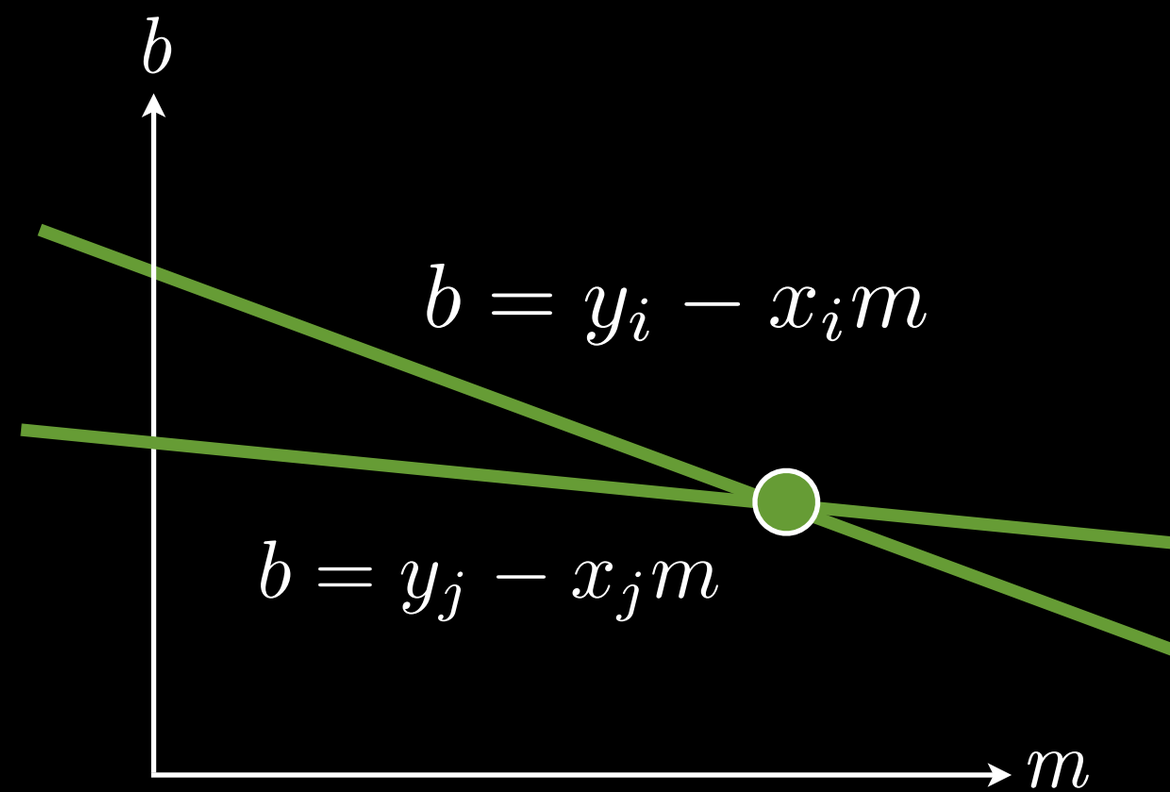
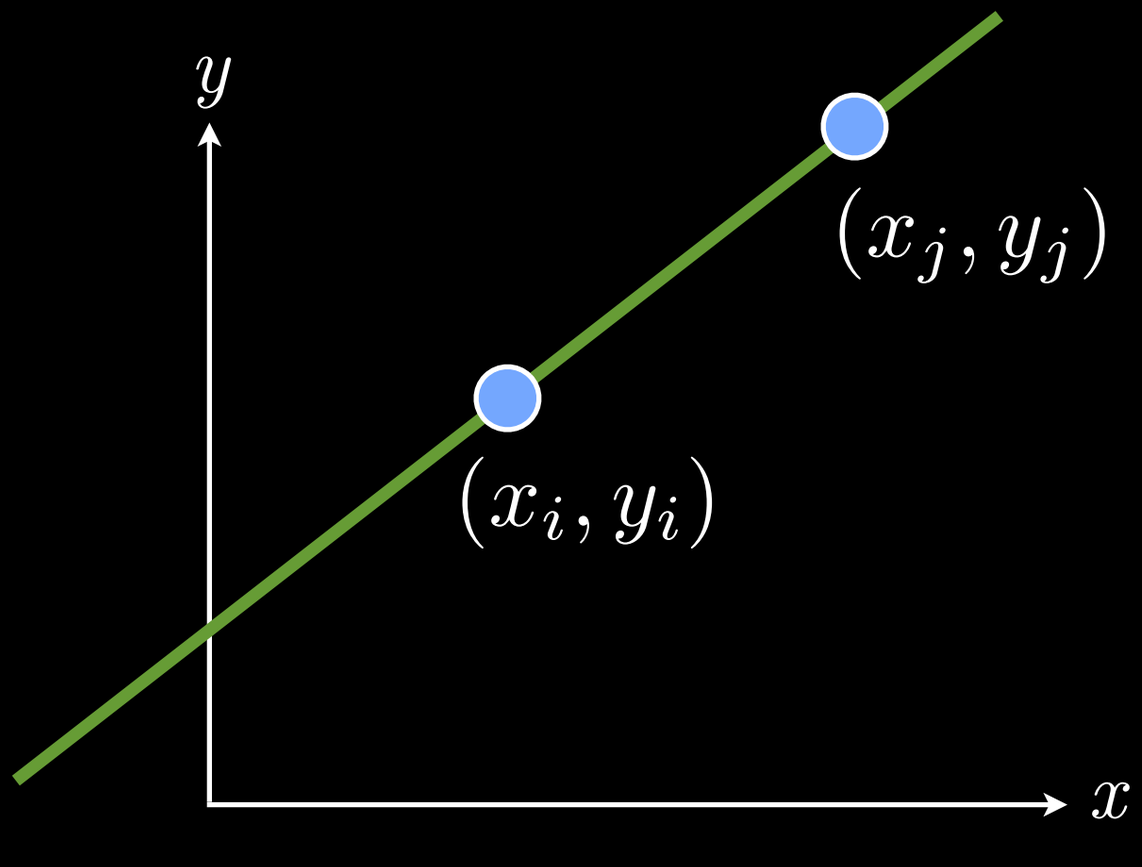


image space

Hough (parameter) space

What are the parameters of the line that contains both points?

Intersection of the two lines in Hough space



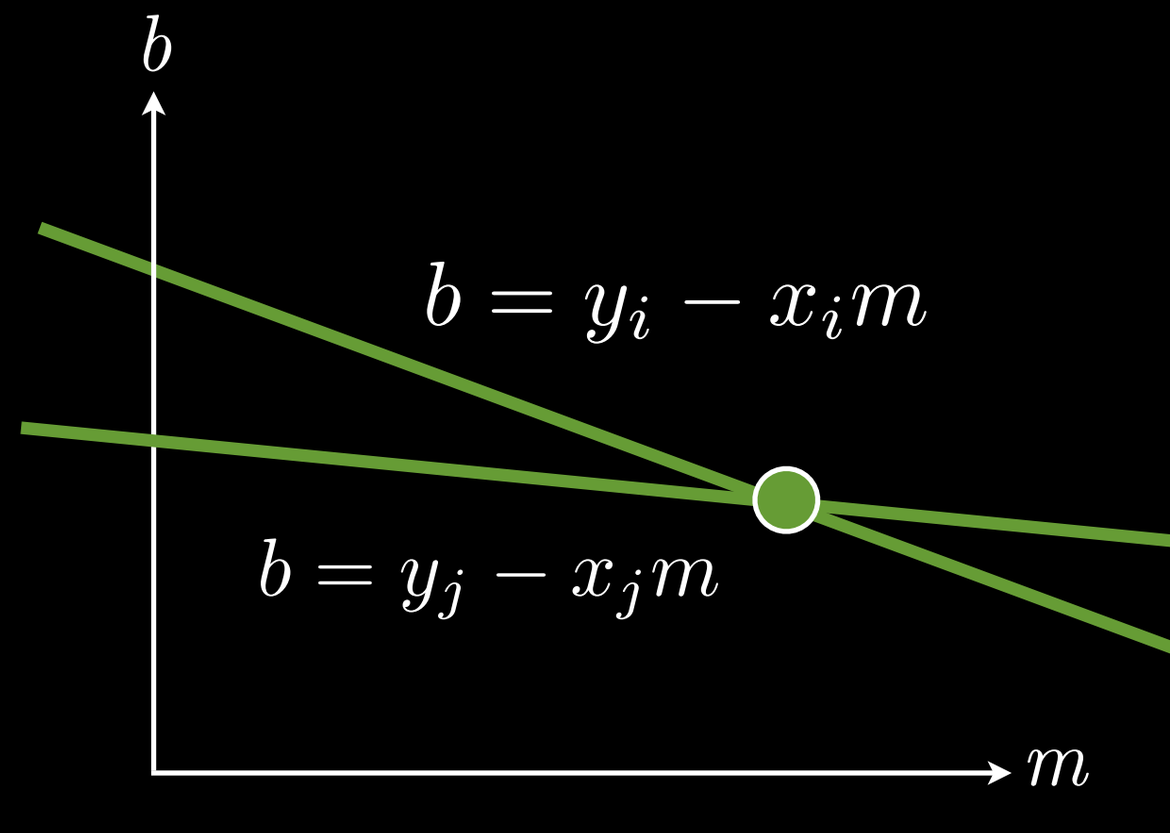
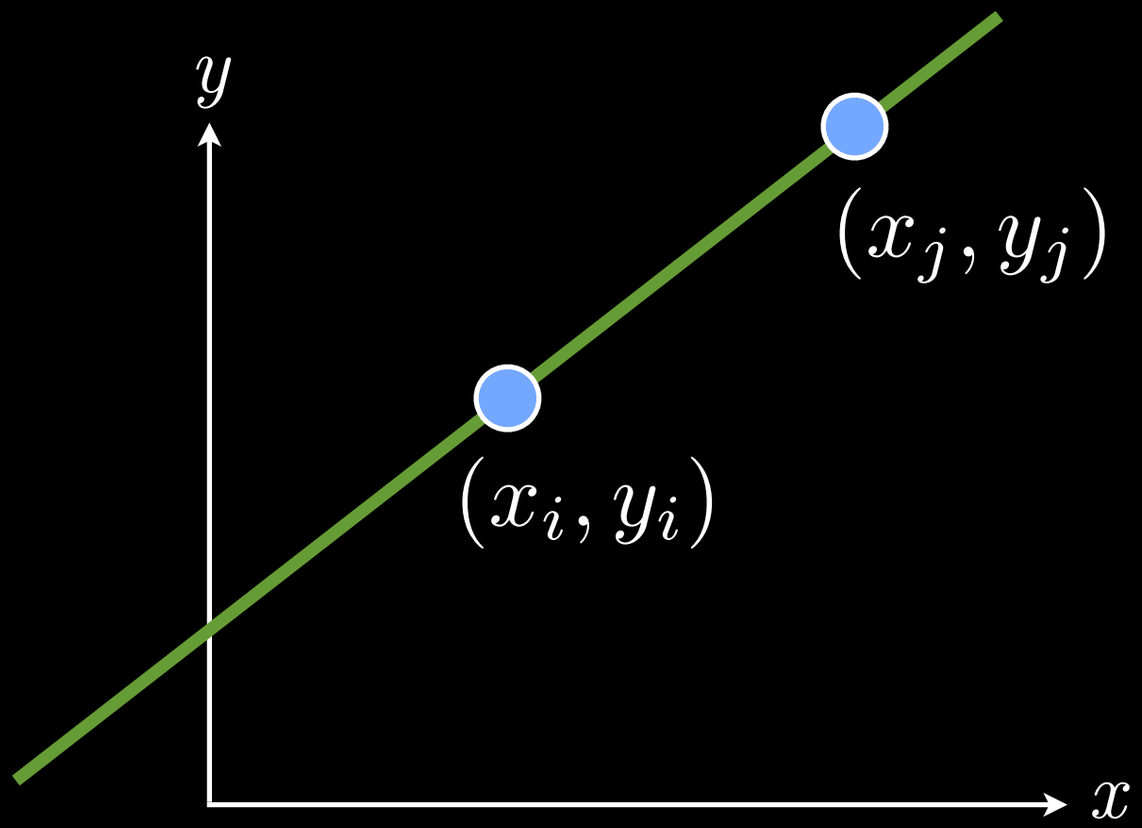


image space

Hough (parameter) space

The parameter space is continuous!
How can we handle this?

Accumulate votes in discrete set of bins



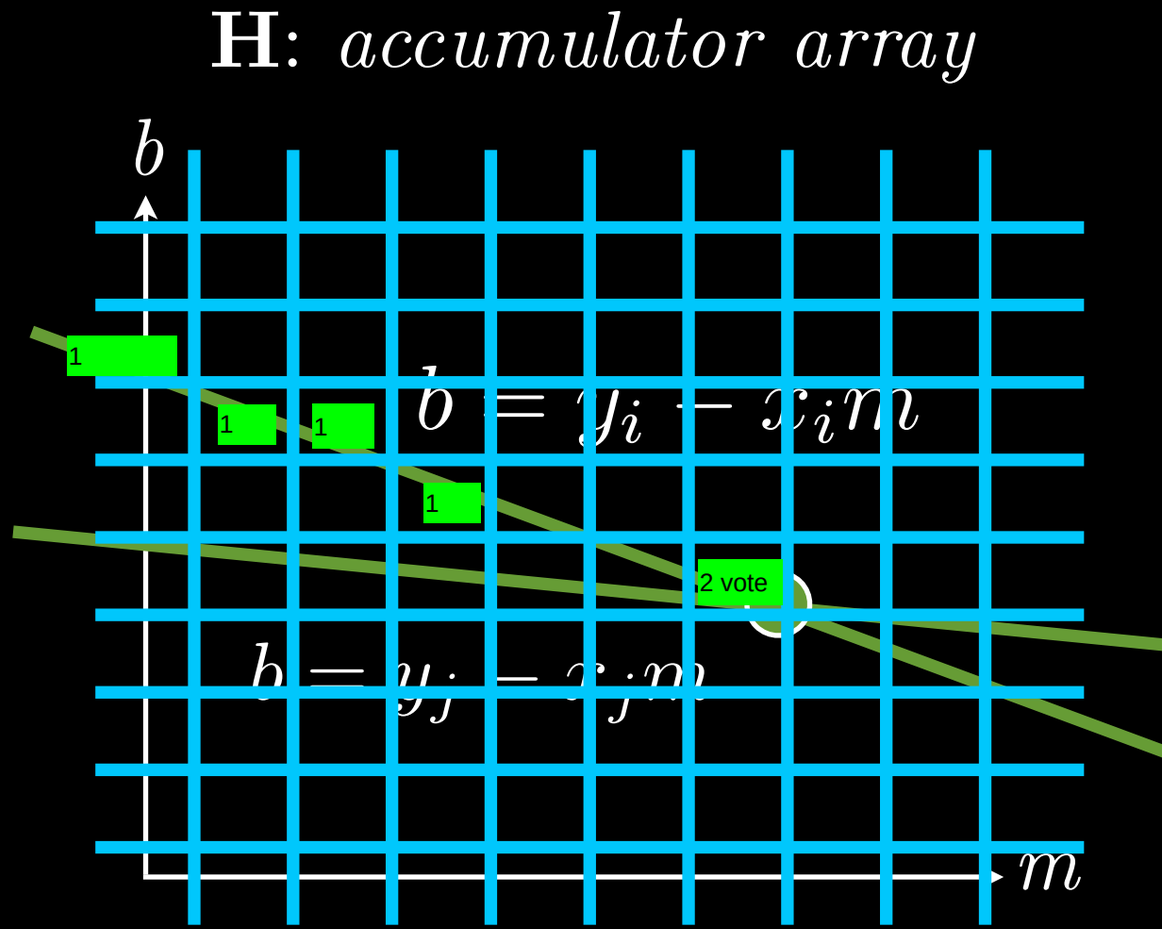


image space

Hough (parameter) space

The parameter space is continuous!
How can we handle this?

Accumulate votes in discrete set of bins

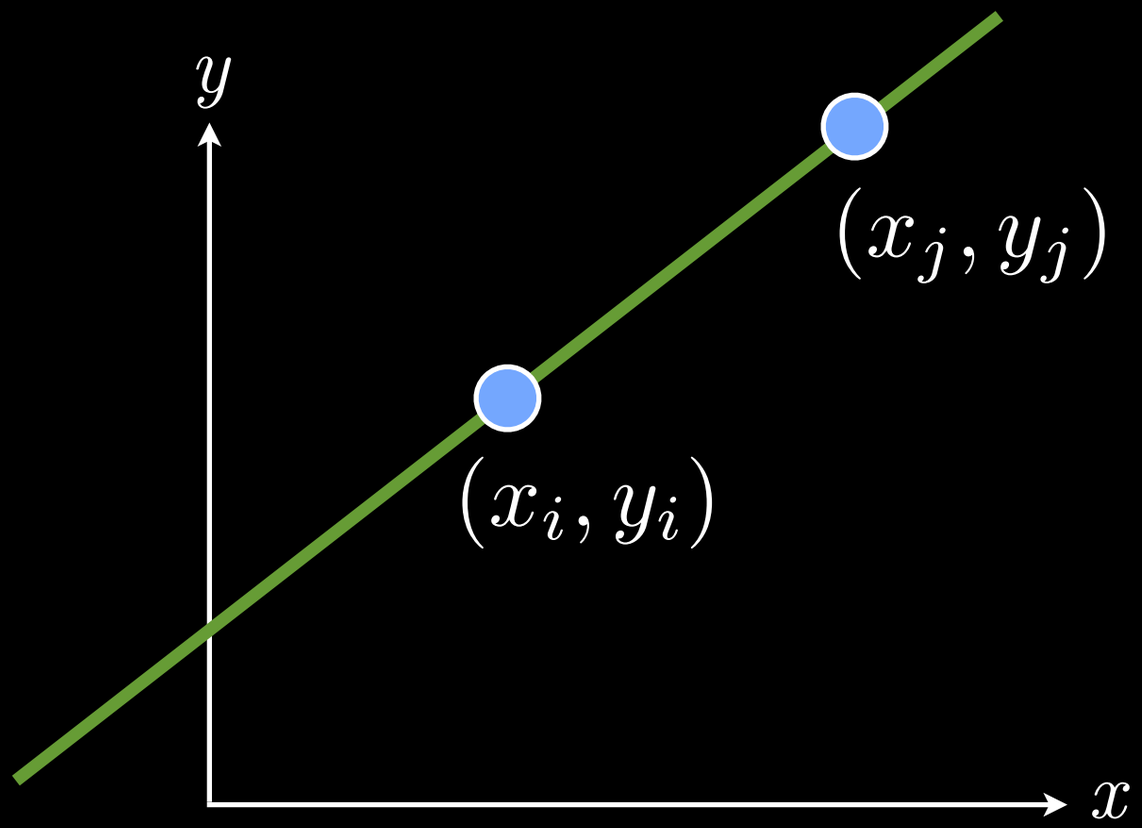
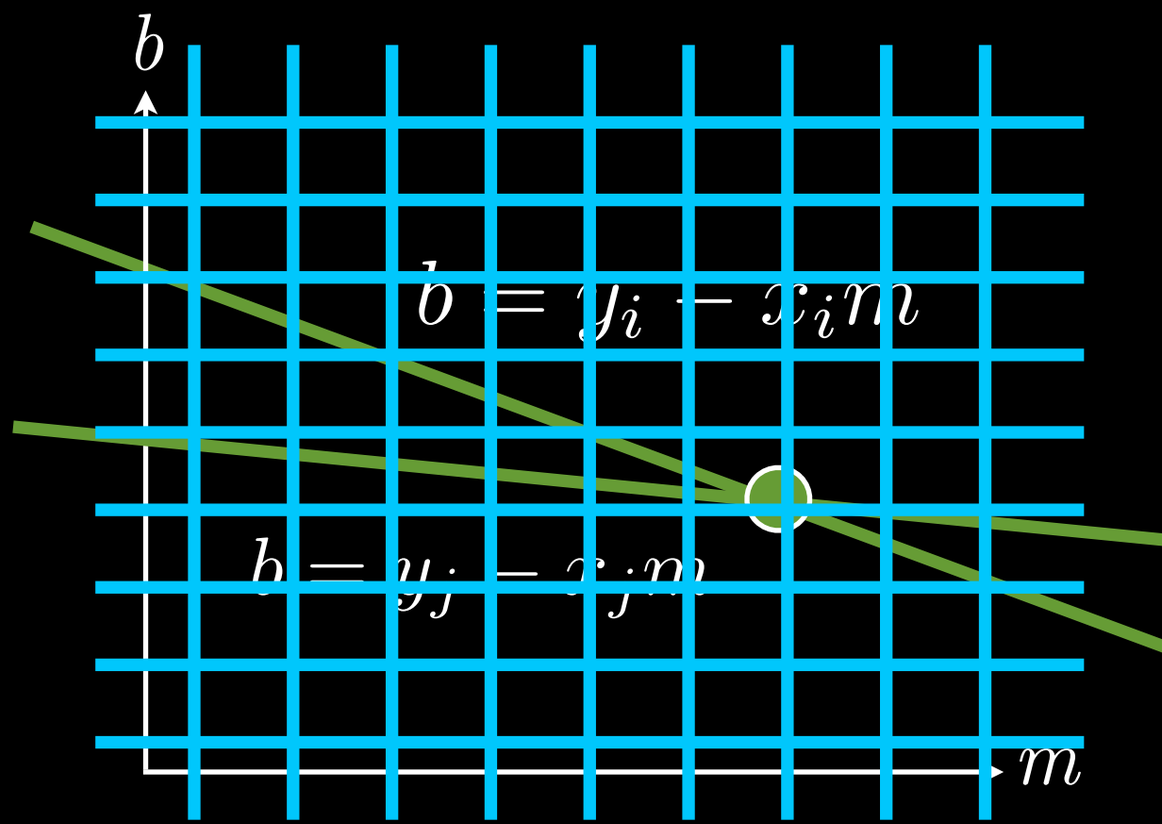


image space

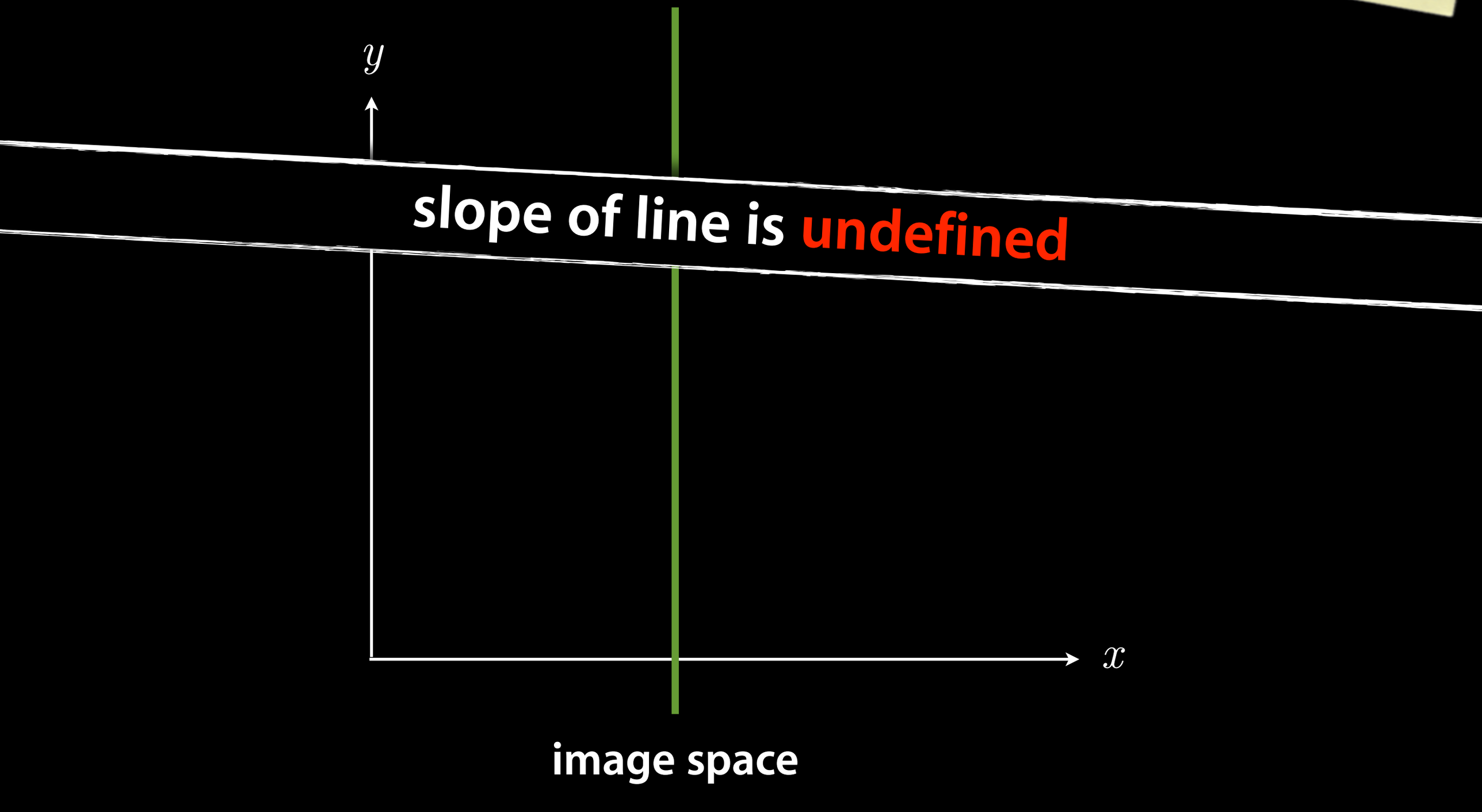
H: *accumulator array*



Hough (parameter) space

Local peaks correspond to detected lines

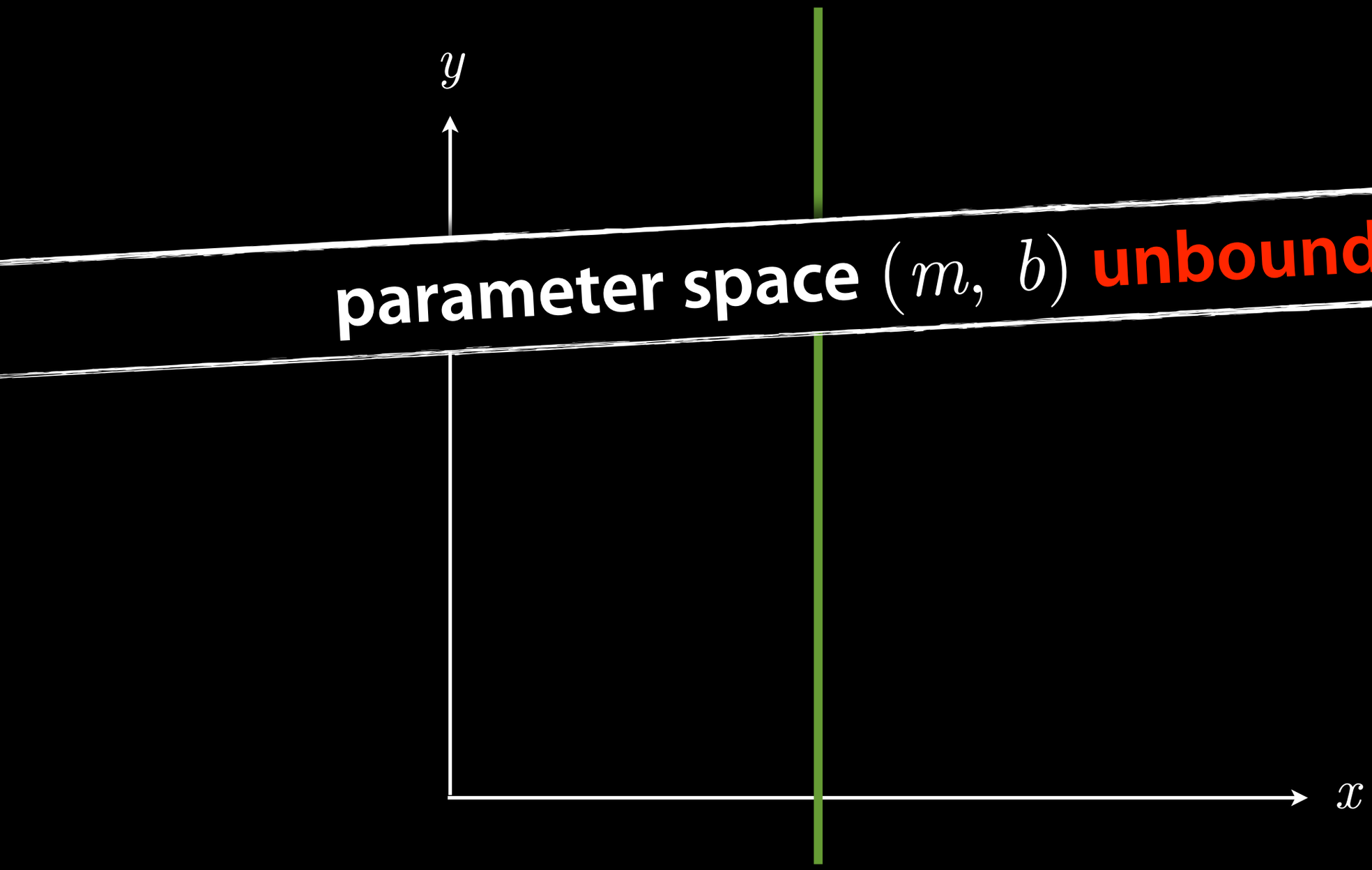
Slope-Intercept



slope of line is **undefined**

image space

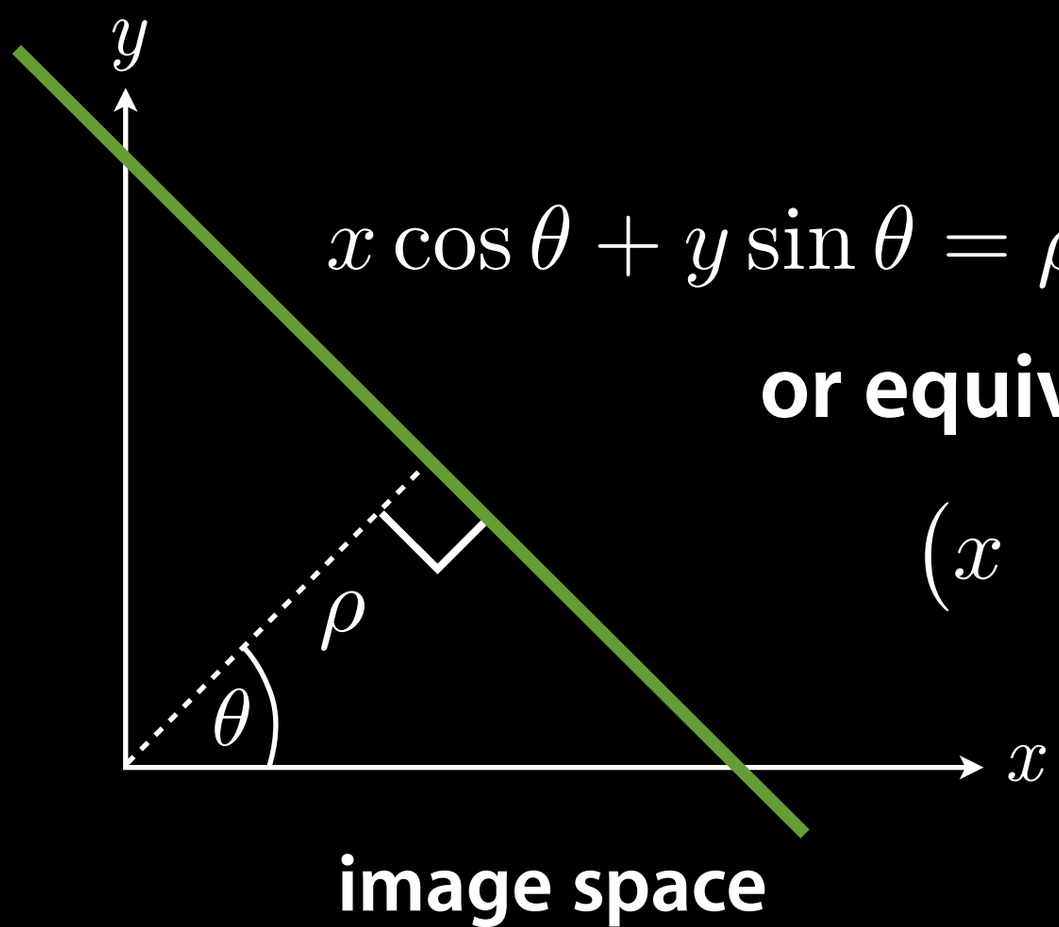
Slope-Intercept



parameter space $(m,\ b)$ **unbounded**

image space

Polar Representation



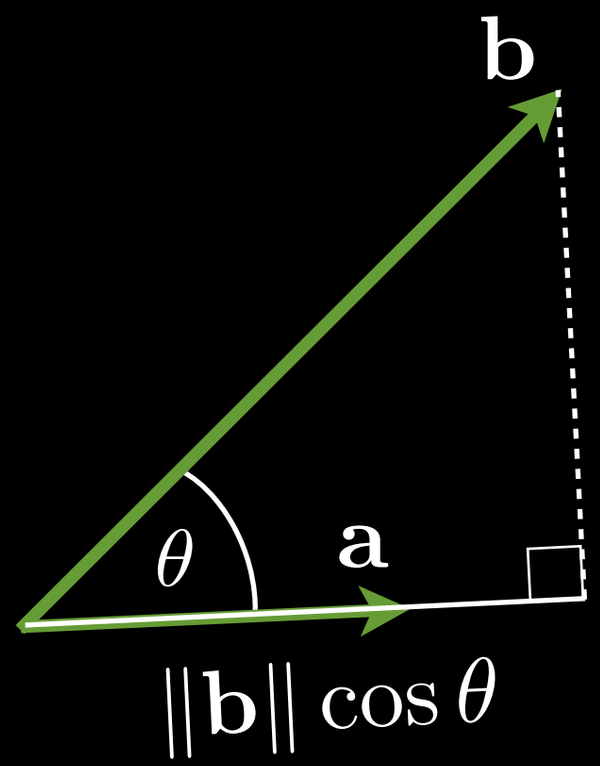
$$x\cos\theta + y\sin\theta = \rho$$

or equivalently

$$\begin{pmatrix} x & y \end{pmatrix} \begin{pmatrix} \cos\theta & \sin\theta \end{pmatrix}^\top = \rho$$

image space

Vector Projection



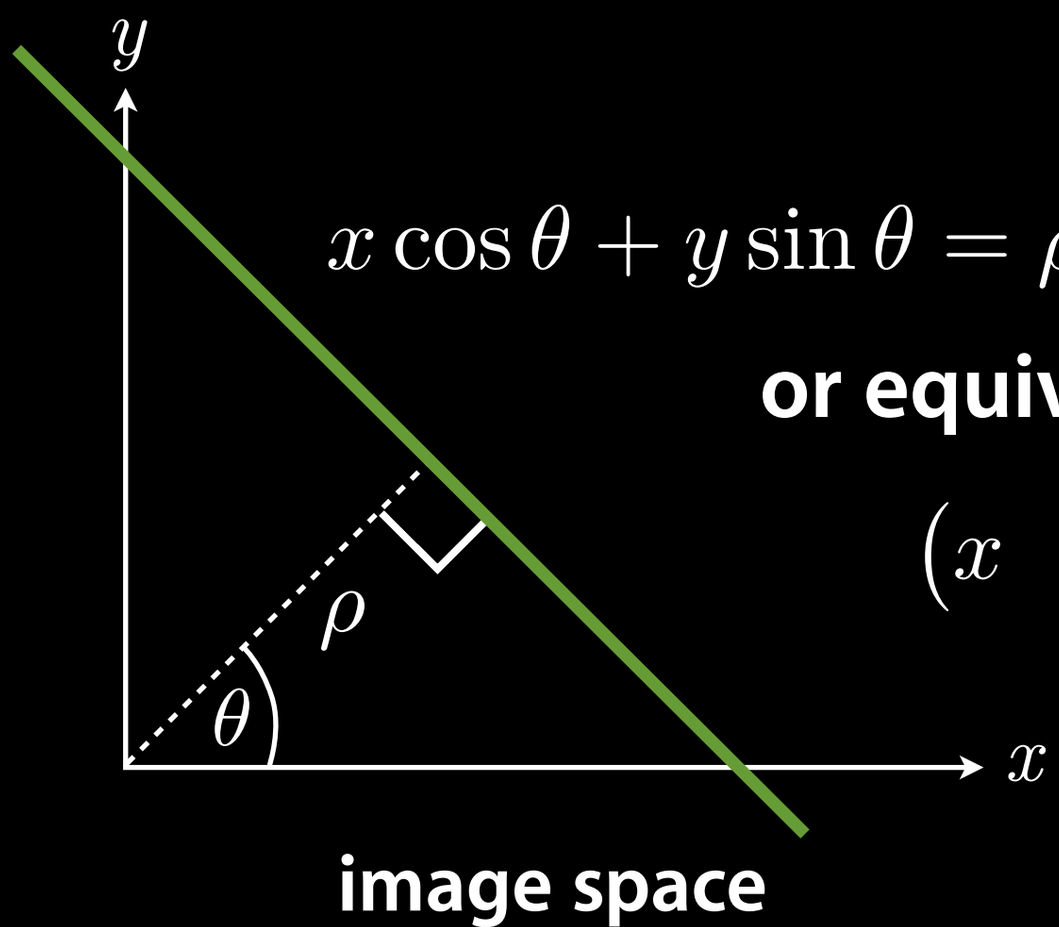
$$\mathbf{a} \cdot \mathbf{b} = \|\mathbf{a}\|\|\mathbf{b}\| \cos\theta$$

assume $\|\mathbf{a}\| = 1$

$$= \|\mathbf{b}\| \cos\theta$$

Length of projection of $\mathbf{b}$ onto $\mathbf{a}$

Polar Representation



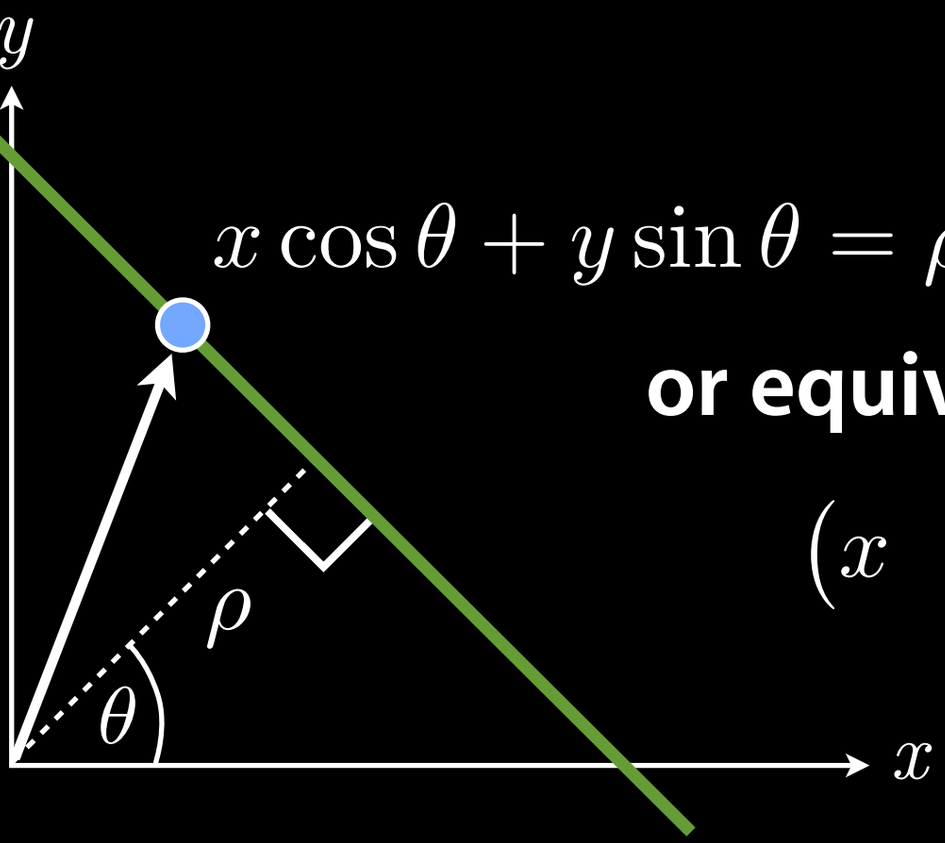
$$x\cos\theta + y\sin\theta = \rho$$

or equivalently

$$\begin{pmatrix} x & y \end{pmatrix} \begin{pmatrix} \cos\theta & \sin\theta \end{pmatrix}^\top = \rho$$

image space

Polar Representation



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

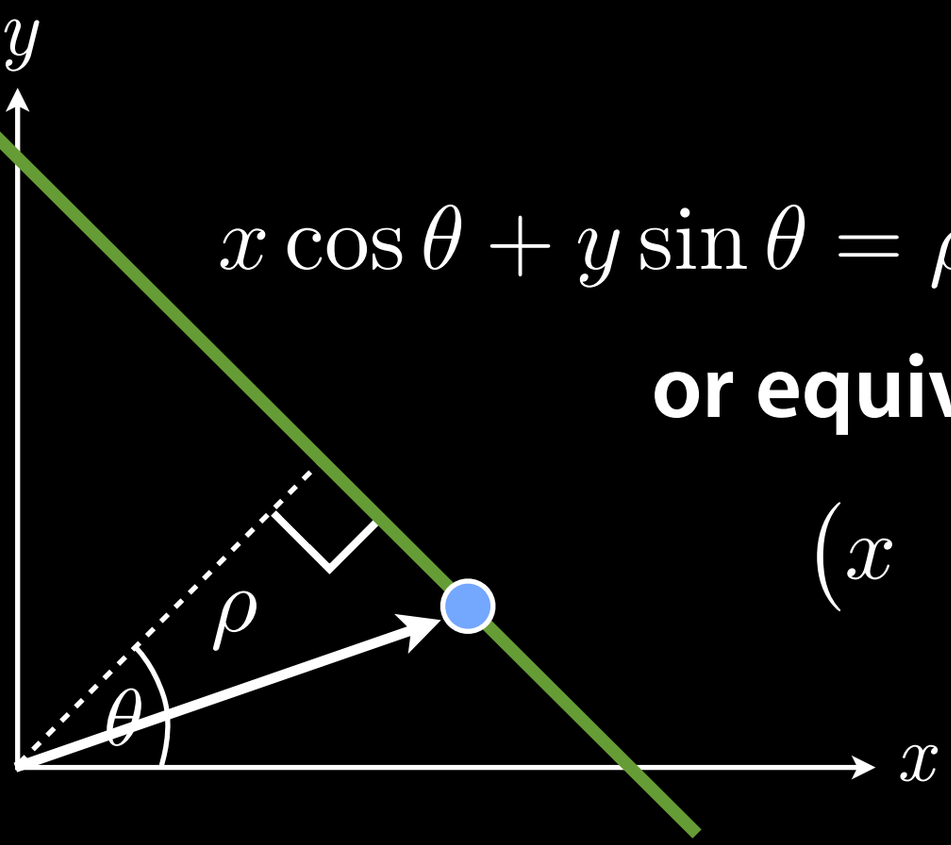
or equivalently

$$\begin{pmatrix} x & y \end{pmatrix} \begin{pmatrix} \cos\theta & \sin\theta \end{pmatrix}^{\top} = \rho$$

image space

any point on the line, the line is defined is defined by unit vector theta, and also the perpendicular distance form that line

Polar Representation



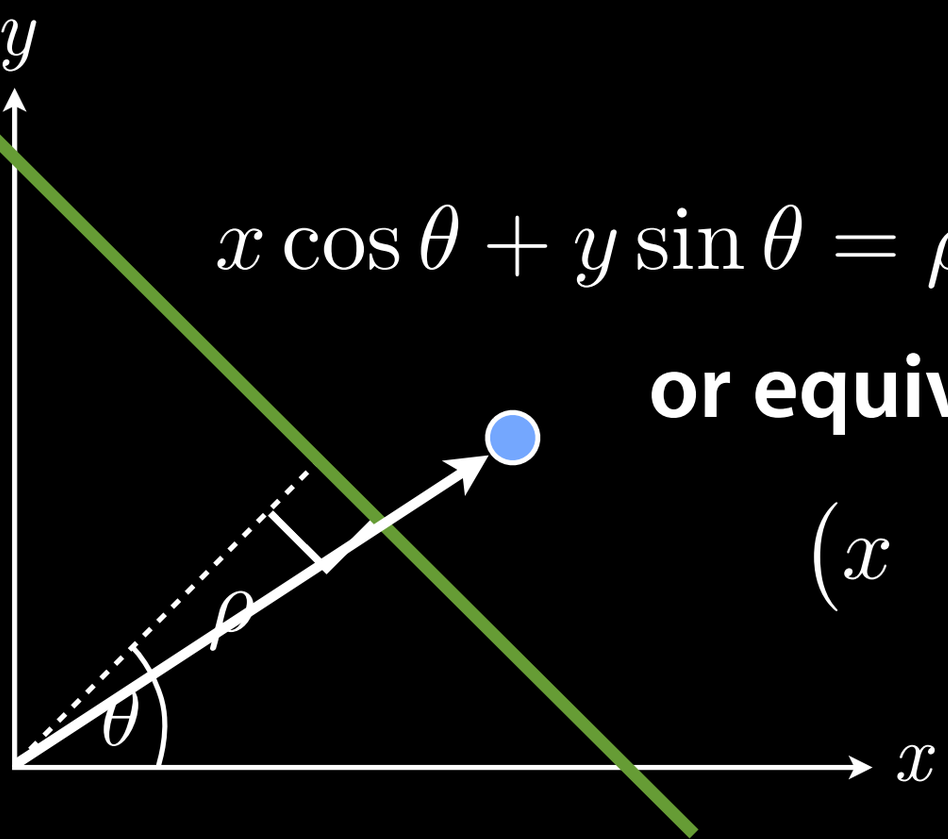
$$x\cos\theta + y\sin\theta = \rho$$

or equivalently

$$\begin{pmatrix} x & y \end{pmatrix} \begin{pmatrix} \cos\theta & \sin\theta \end{pmatrix}^{\top} = \rho$$

image space

Polar Representation



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

or equivalently

$$\begin{pmatrix} x & y \end{pmatrix} \begin{pmatrix} \cos\theta & \sin\theta \end{pmatrix}^\top = \rho$$

image space

Polar Representation

x and y are constant, so obviously its going to be a sinosoid

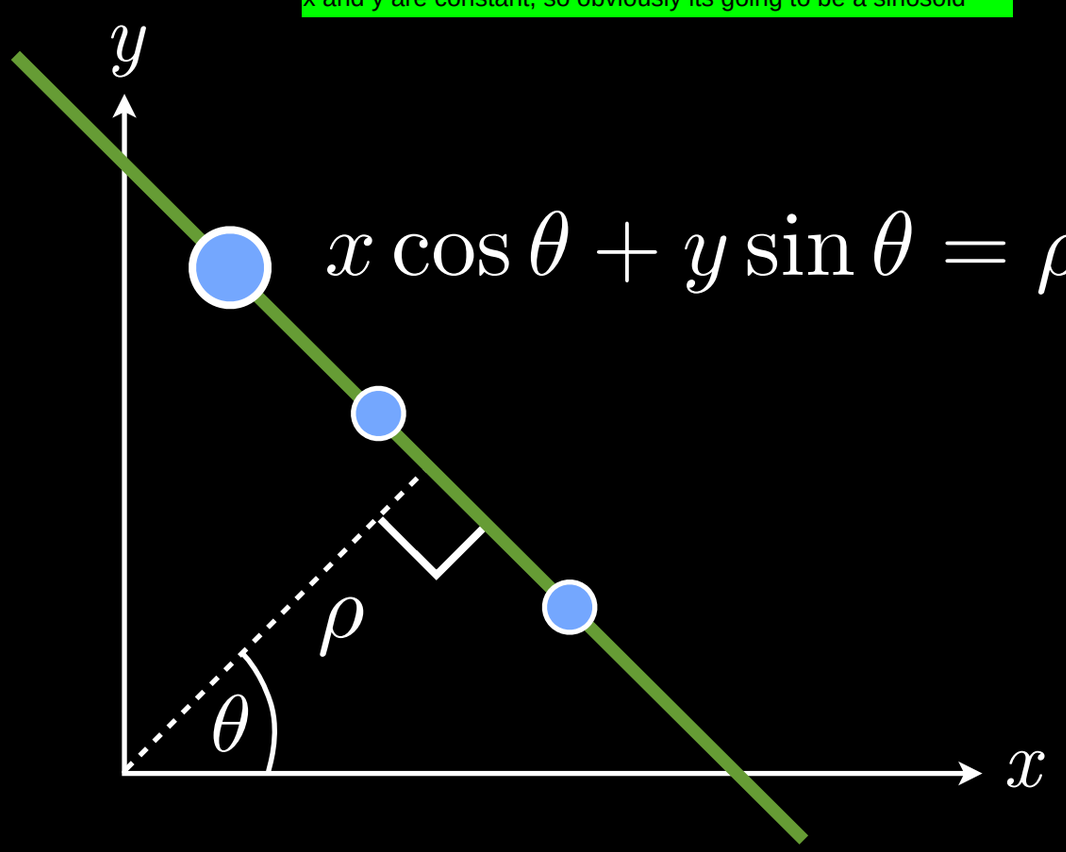
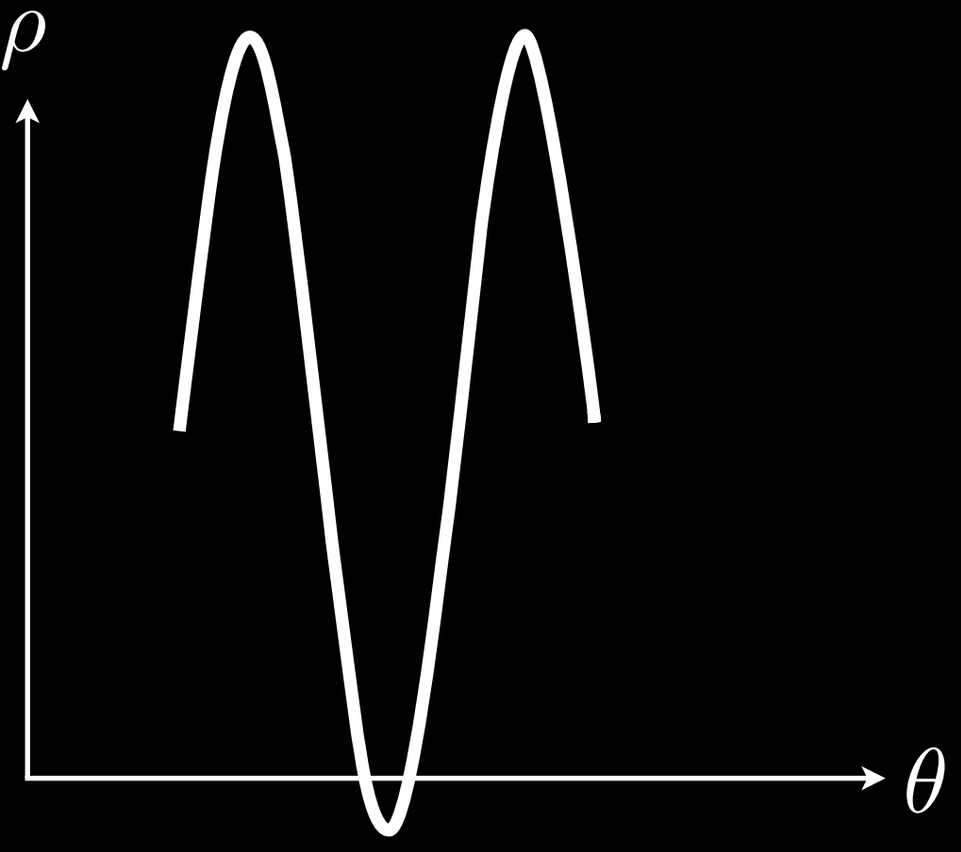


image space



Hough space

Polar Representation

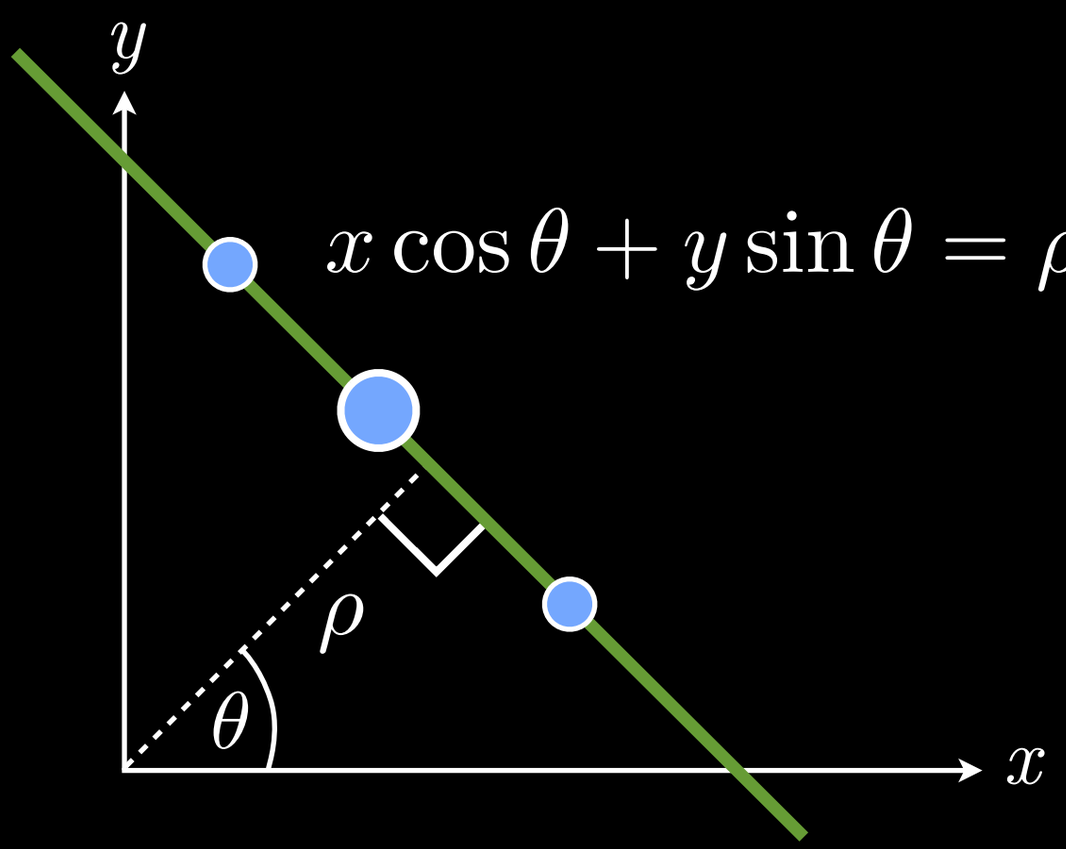
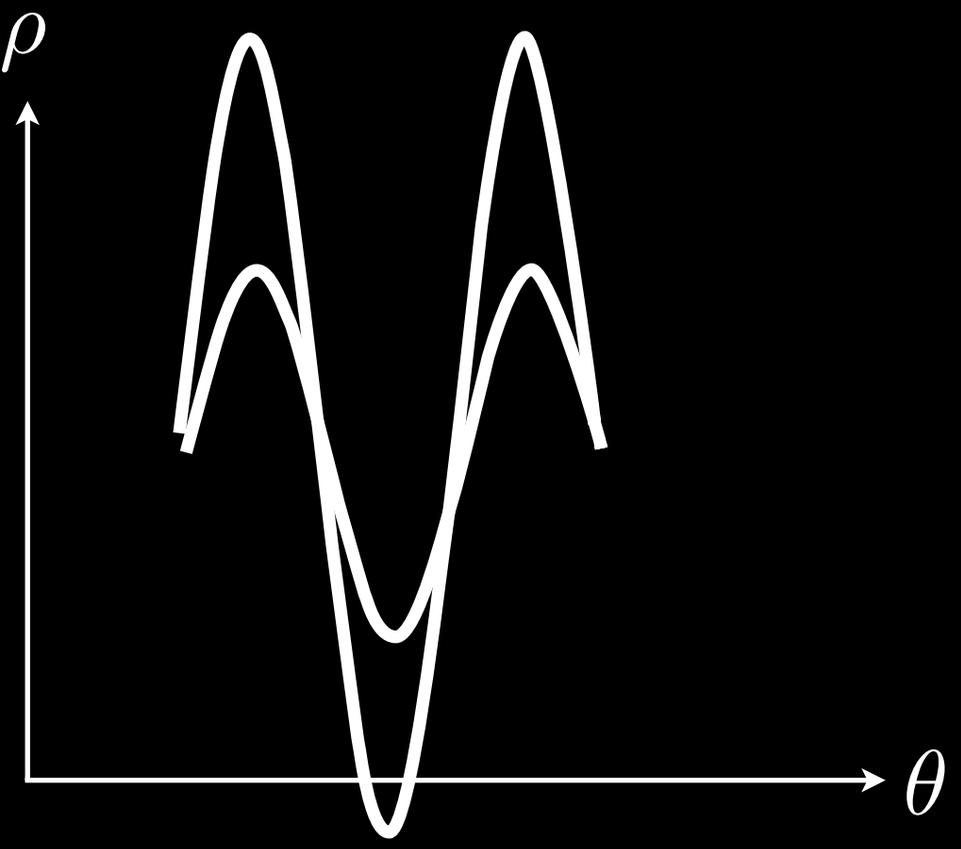


image space



Hough space

Polar Representation

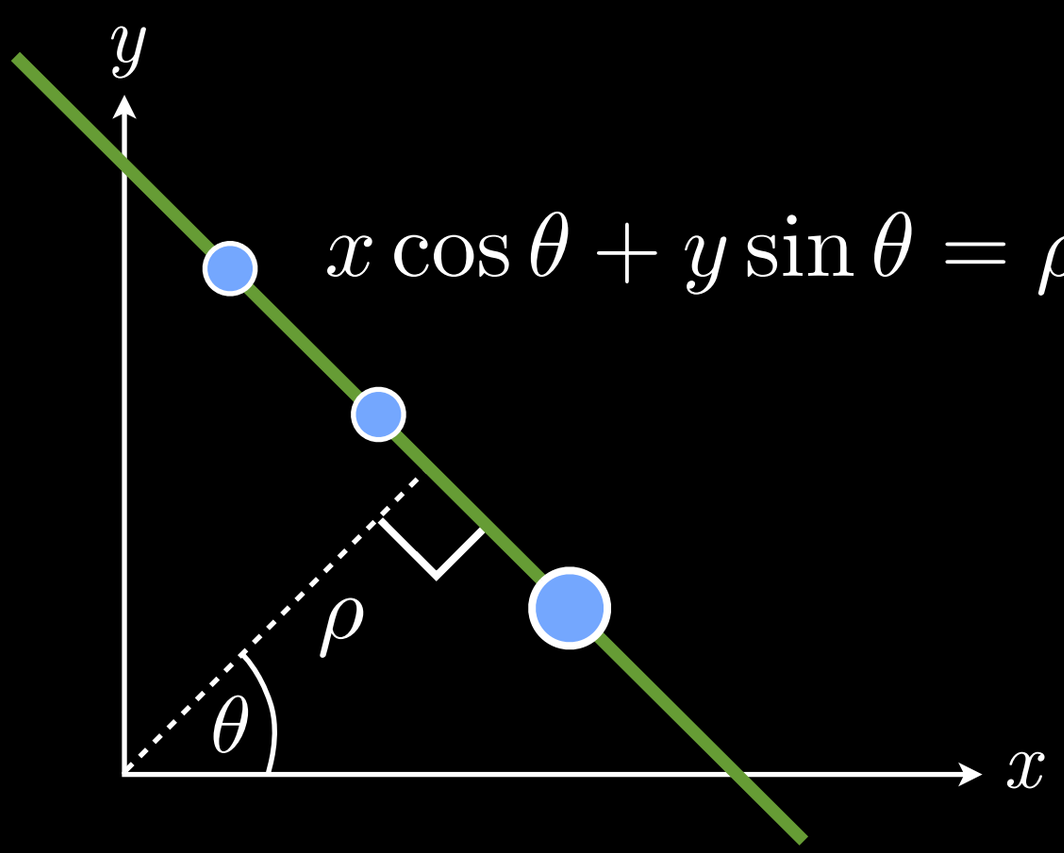
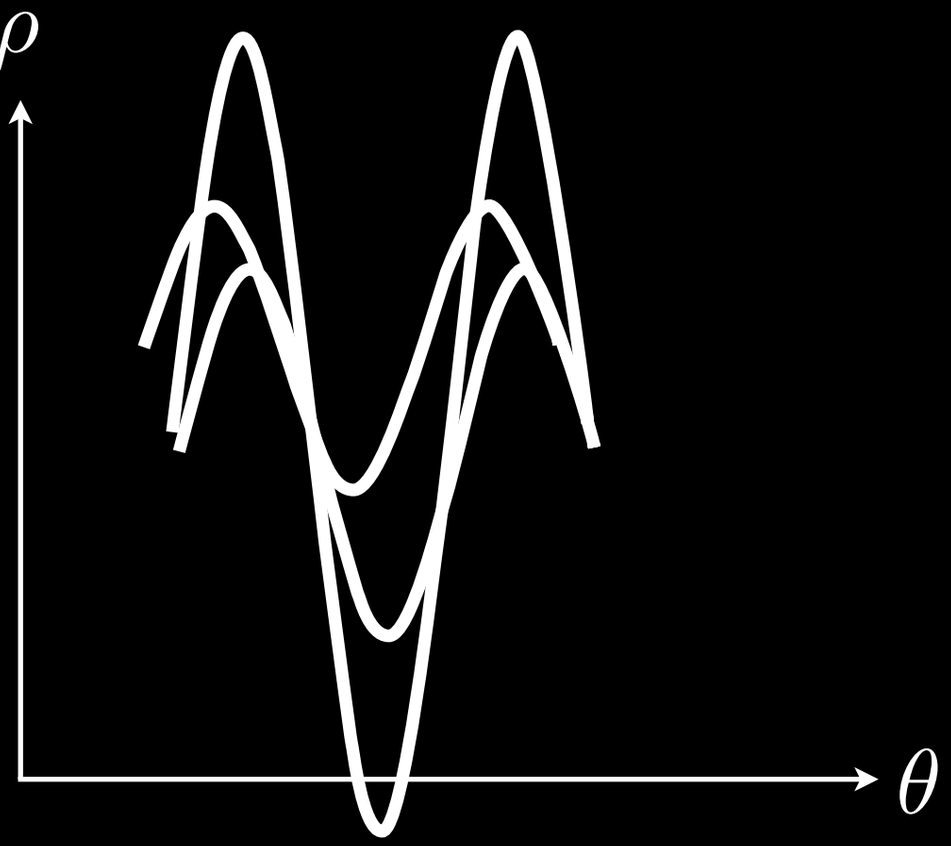


image space



Hough space

Polar Representation

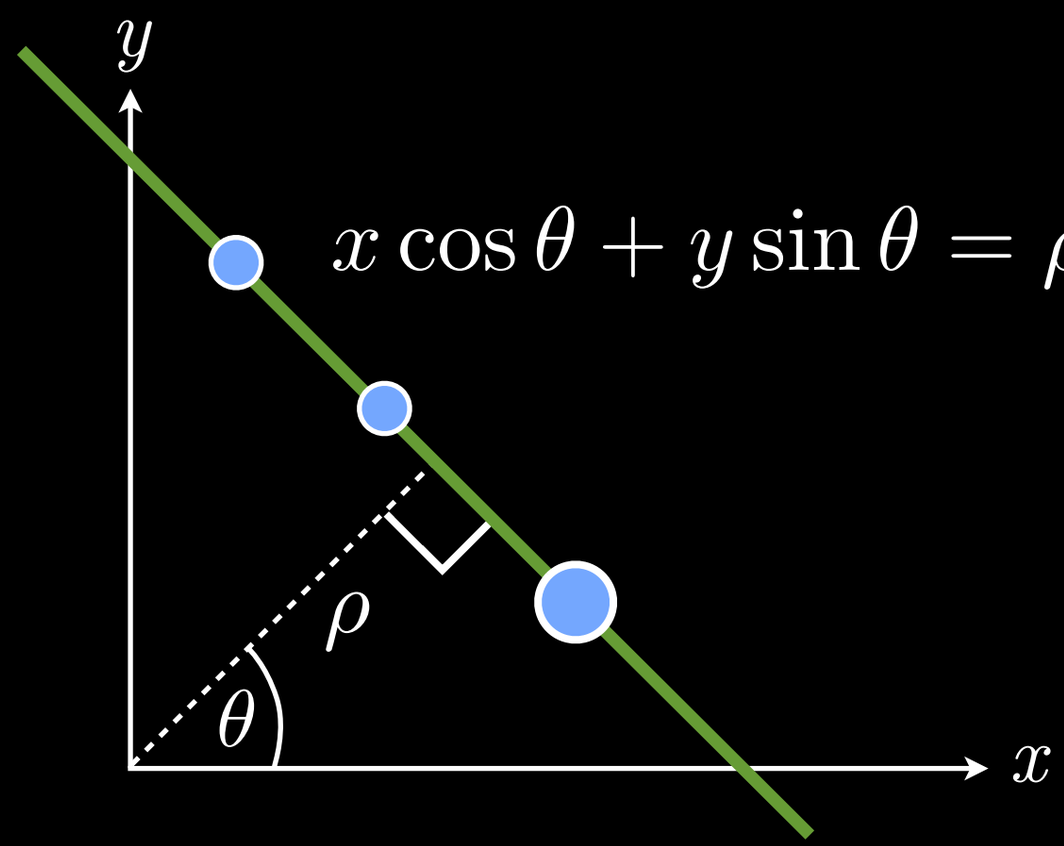
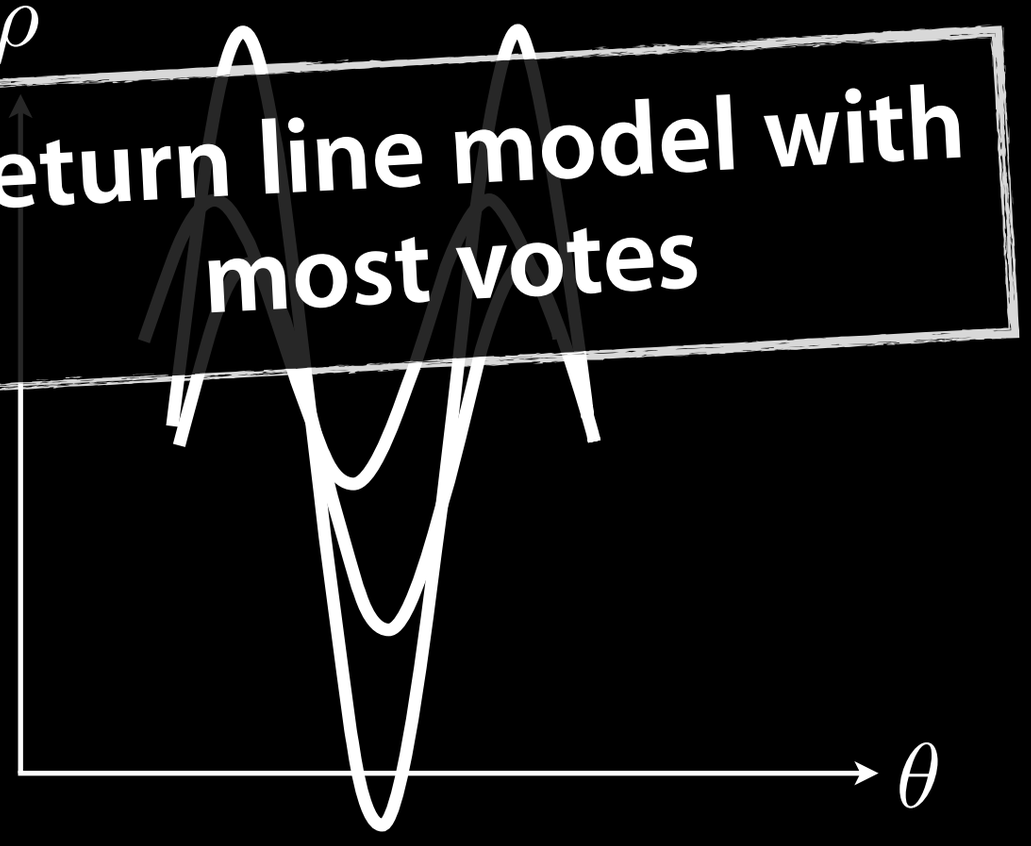


image space

Return line model with most votes



Hough space

Polar Representation

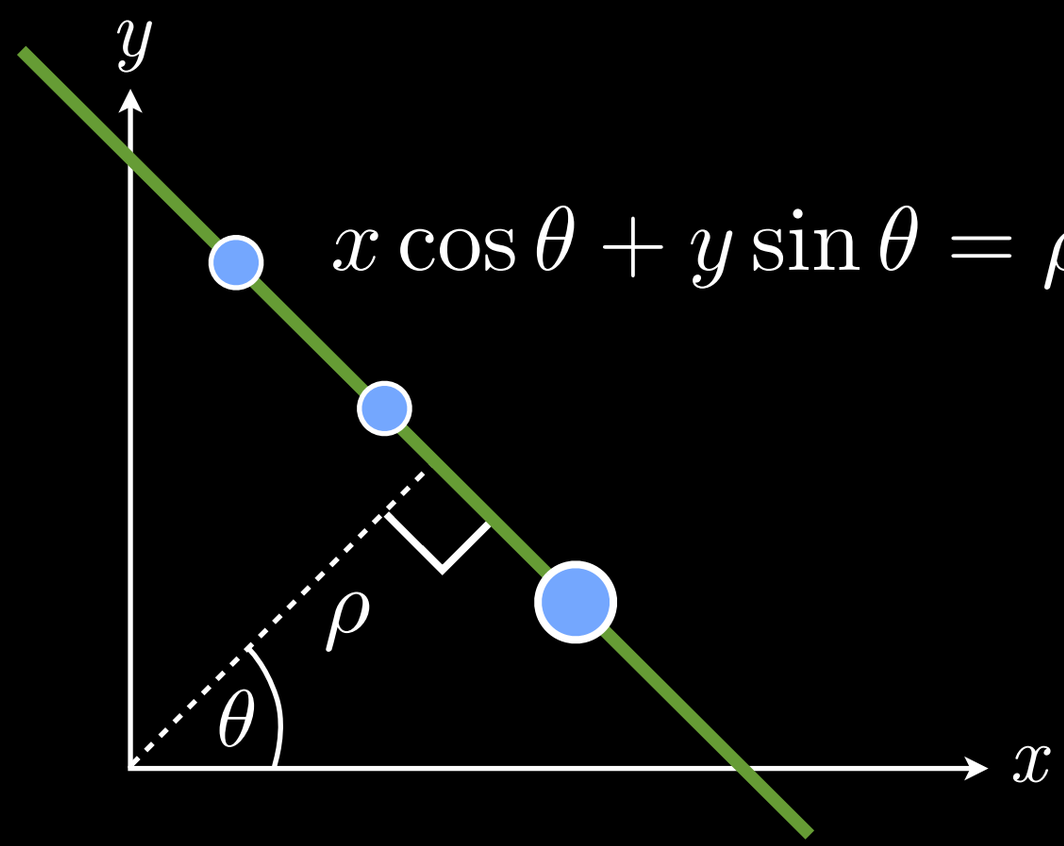
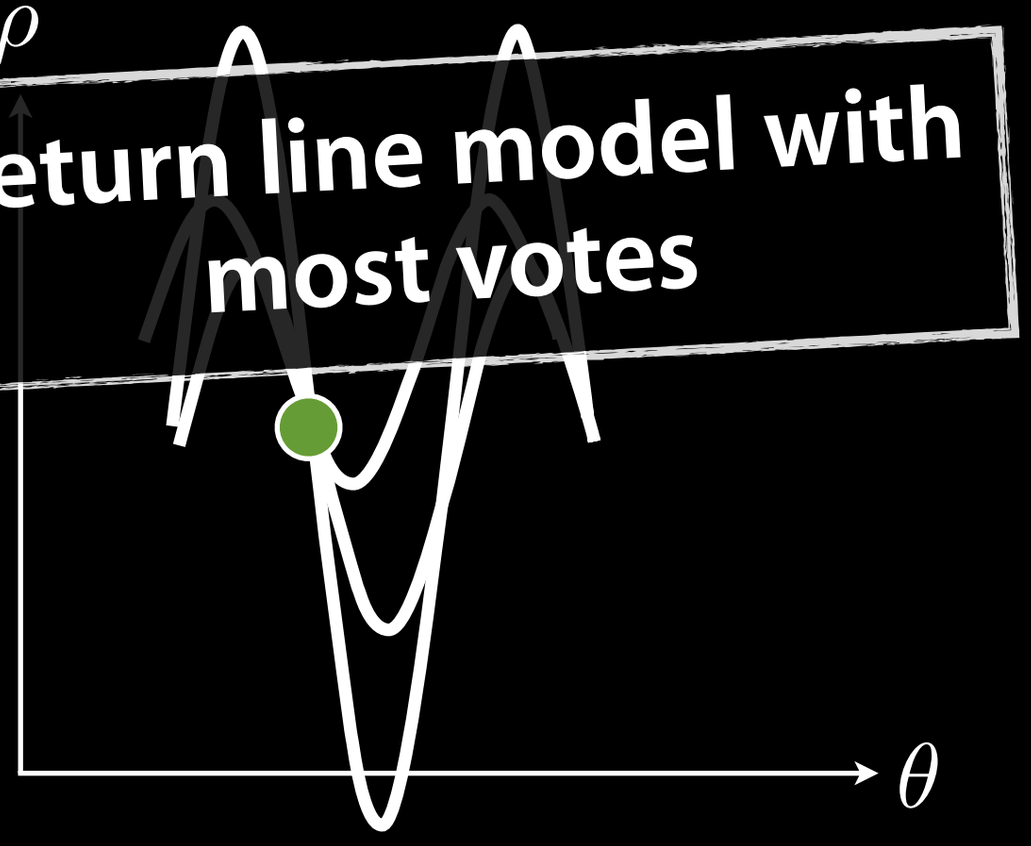


image space

Return line model with most votes



Hough space

1: Initialize $H[\rho, \theta] = 0$
2: **foreach** edgel $e \in I[x, y]$ **do**
3: | **foreach** $\theta = \theta_{\min}$ to $\theta_{\max}$ **do**
4: | | $\rho = x \cos\theta + y \sin\theta$
5: | | $H[\rho, \theta] \mathrel{+}= 1$
6: | **end**
7: **end**
8: $\rho^*, \theta^* = \underset{\rho,\theta}{\operatorname{argmax}} \; H[\rho, \theta]$

Time complexity in terms of number of votes per point?

Hough Lines

extension #1

1: Initialize $H[\rho, \theta] = 0$
2: **foreach** edgel $e \in I[x, y]$ **do**
3: $\quad \theta = \text{angle}(\nabla(I[x, y]))$
4: $\quad \rho = x \cos\theta + y \sin\theta$
5: $\quad H[\rho, \theta] \mathrel{+}= 1$
6:
7: **end**
8: $\rho^*, \theta^* = \underset{\rho,\theta}{\text{argmax}} \; H[\rho, \theta]$

Hough Lines

extension #2

1: Initialize $H[\rho, \theta] = 0$
2: **foreach** edgel $e \in I[x, y]$ **do**
3: $\quad \theta = \text{angle}(\nabla(I[x, y]))$
4: $\quad \rho = x \cos\theta + y \sin\theta$
5: $\quad H[\rho, \theta] \mathrel{+}= \|\nabla(I[x, y])\|$
6:
7: **end**
8: $\rho^*, \theta^* = \underset{\rho,\theta}{\text{argmax}} \; H[\rho, \theta]$

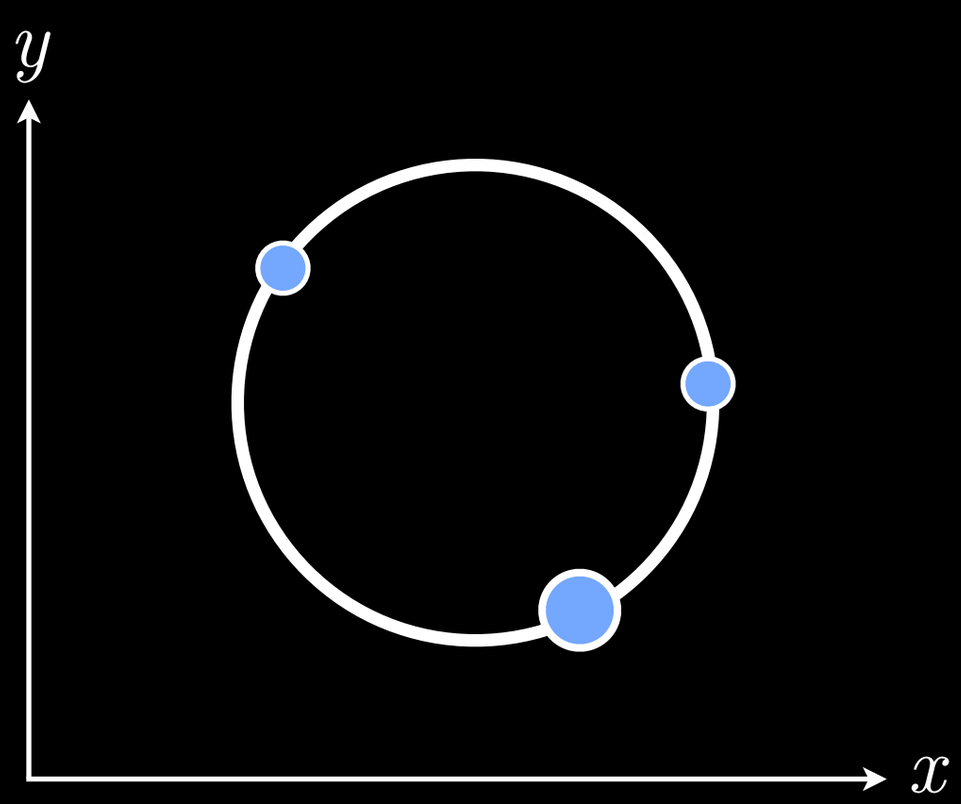
input image

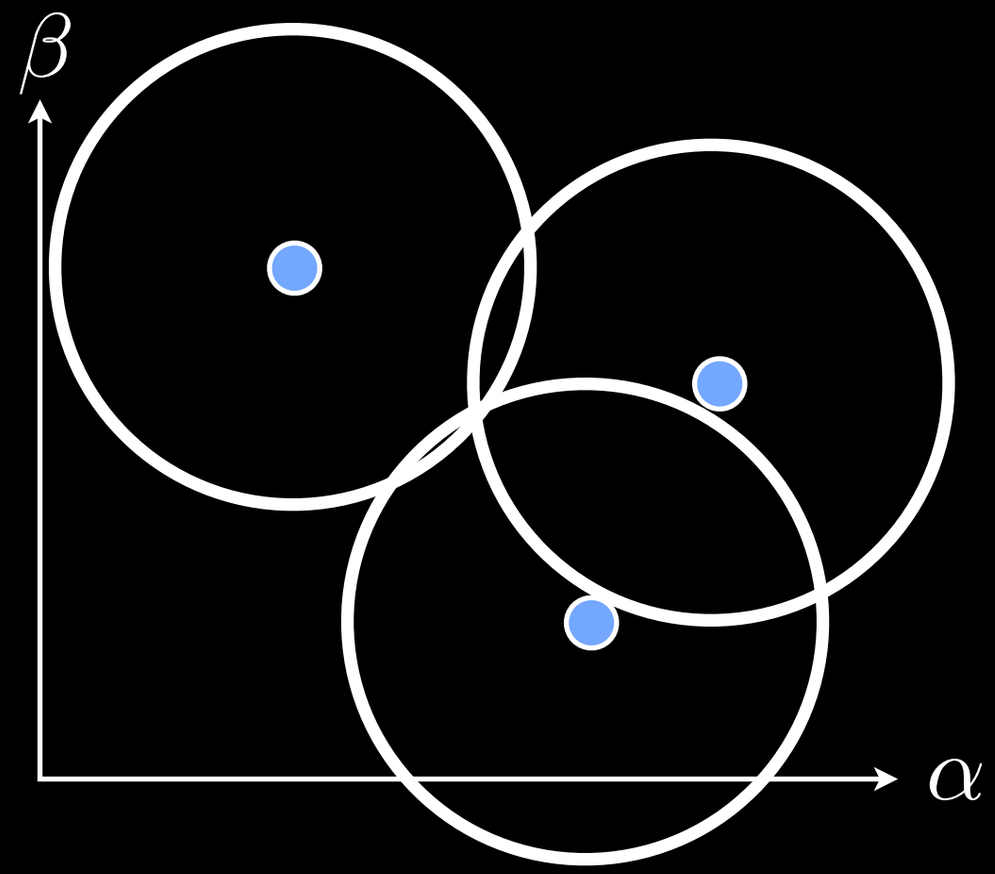
Canny edges

Hough lines

Assume
Fixed Radius

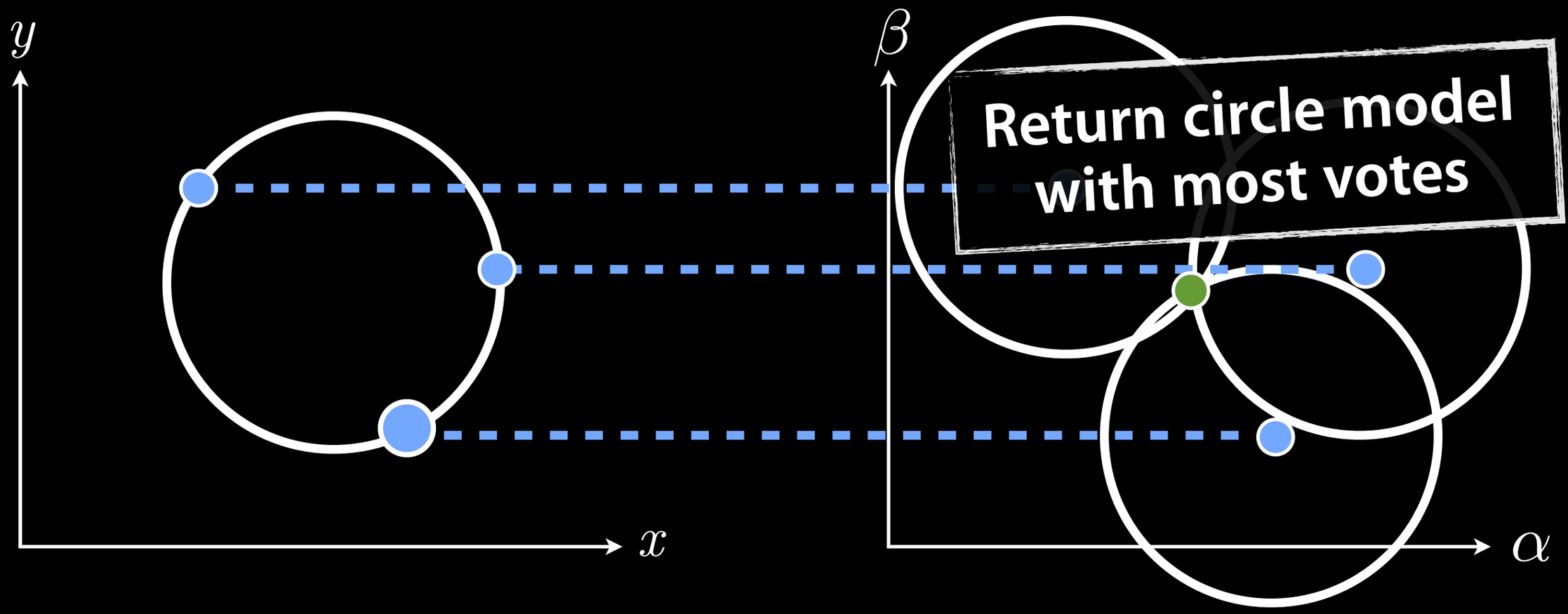
$$(x_i - \alpha)^2 + (y_i - \beta)^2 = r^2$$





Assume
Fixed Radius

$$(x_i - \alpha)^2 + (y_i - \beta)^2 = r^2$$

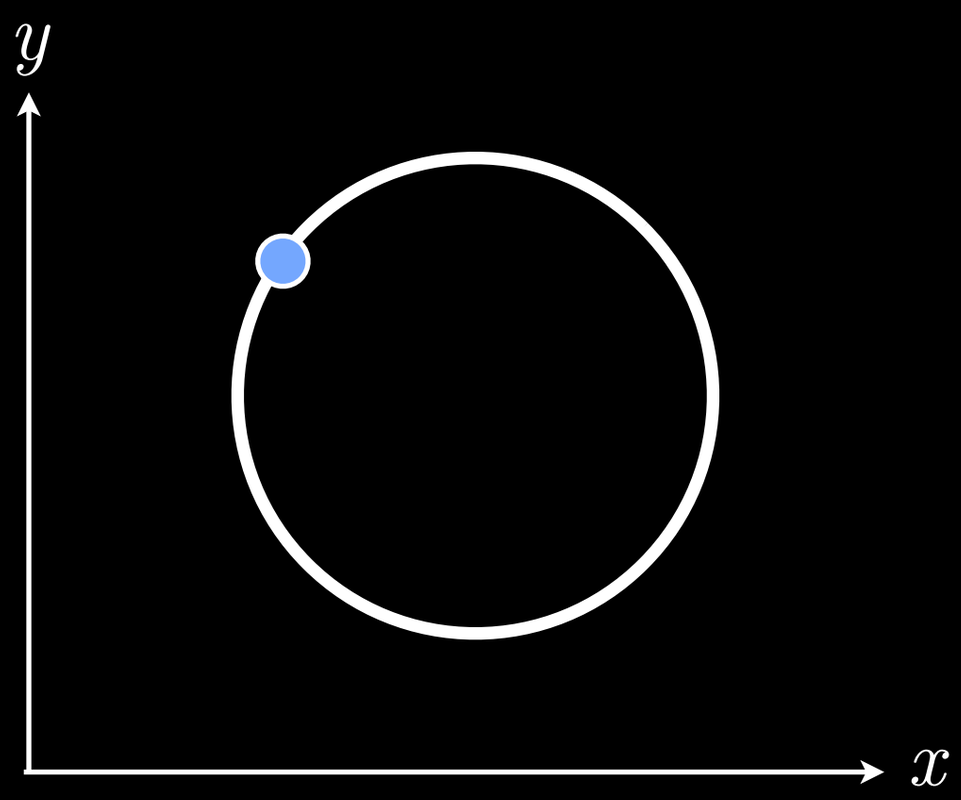


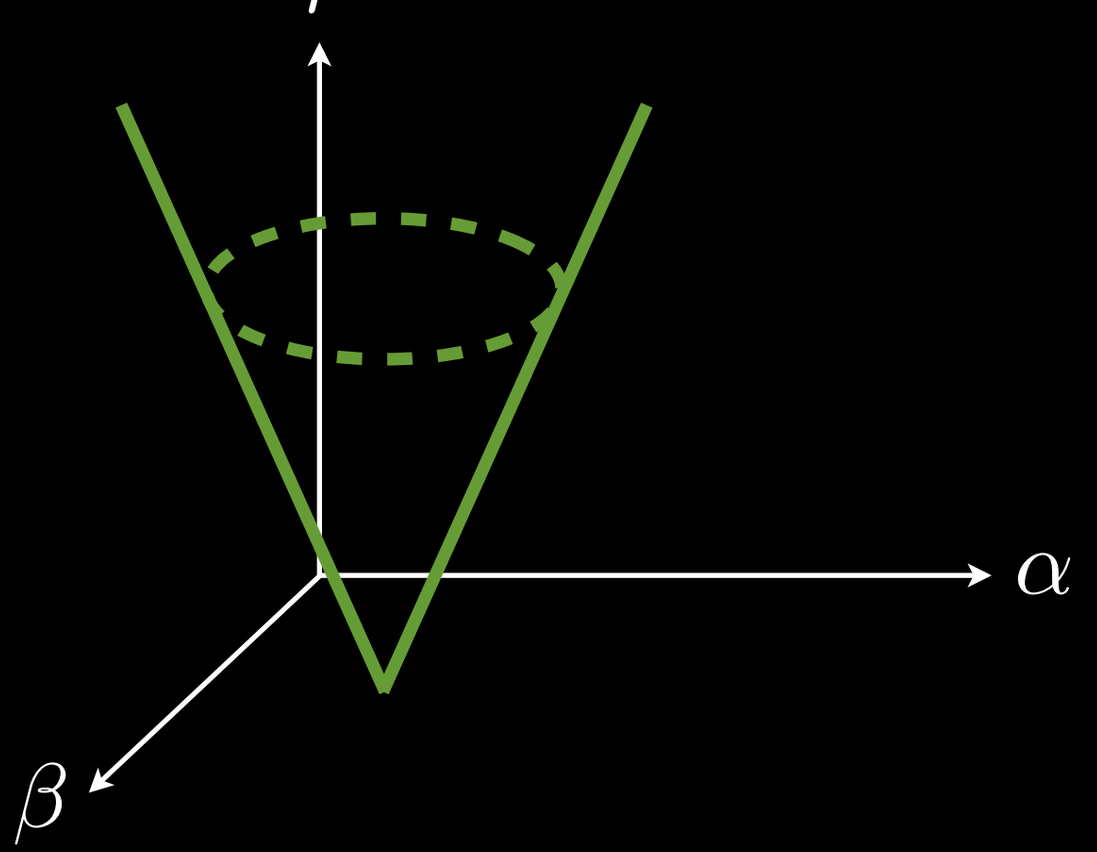
Assume
Unknown Radius

$$(x_i - \alpha)^2 + (y_i - \beta)^2 = r^2$$

Assume Unknown Radius

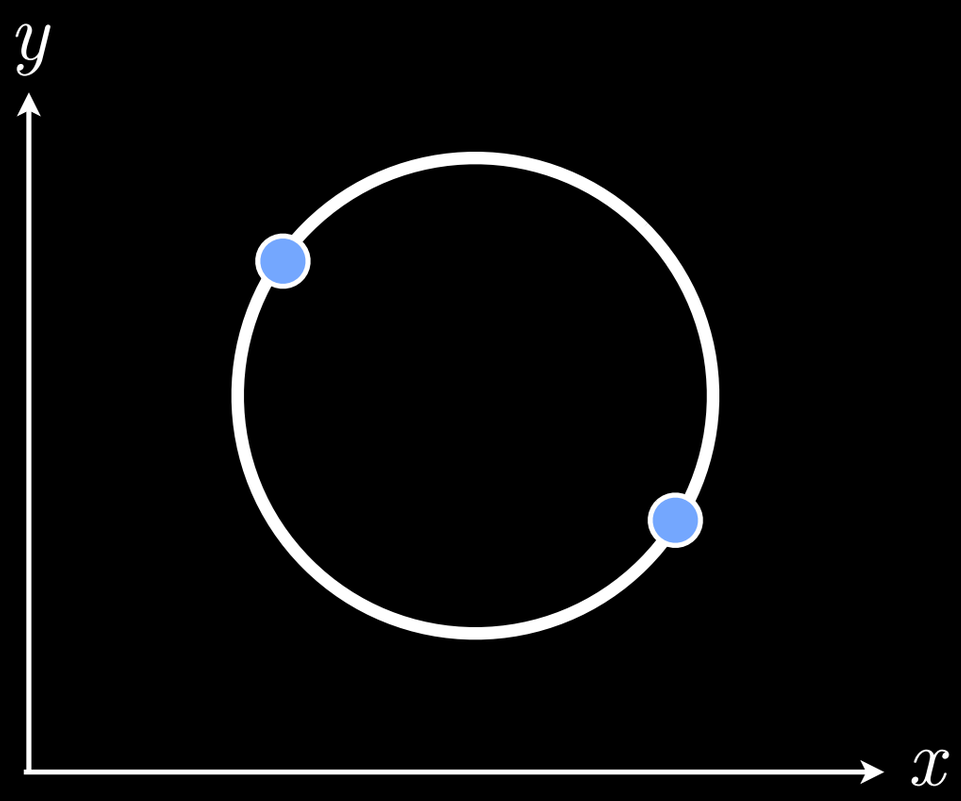
$$(x_i - \alpha)^2 + (y_i - \beta)^2 = r^2$$

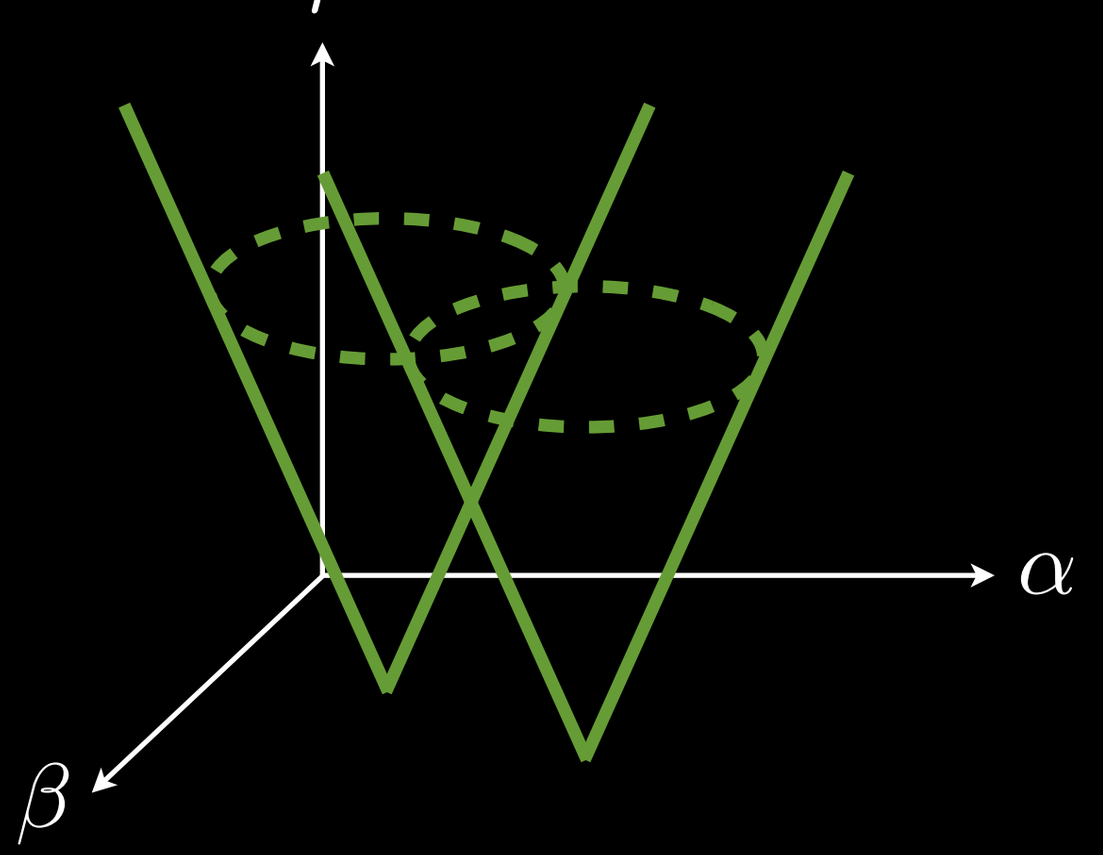




Assume Unknown Radius

$$(x_i - \alpha)^2 + (y_i - \beta)^2 = r^2$$



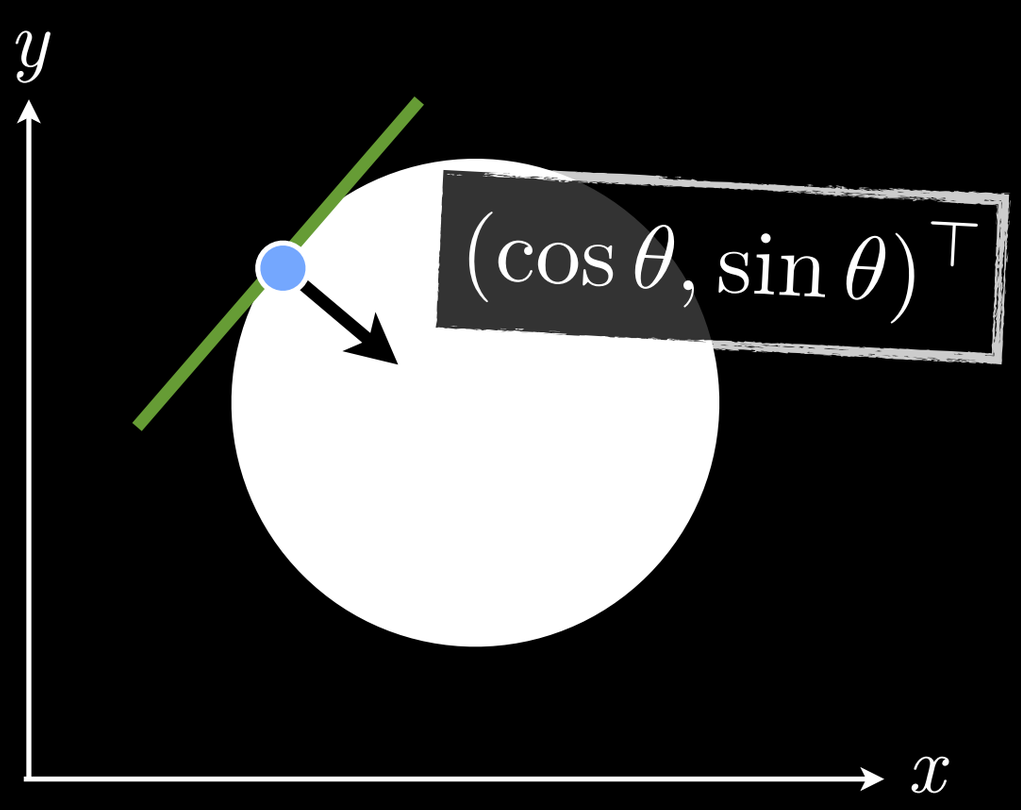


Unknown Radius
Known Gradient
Direction

$$(x_i - \alpha)^2 + (y_i - \beta)^2 = r^2$$

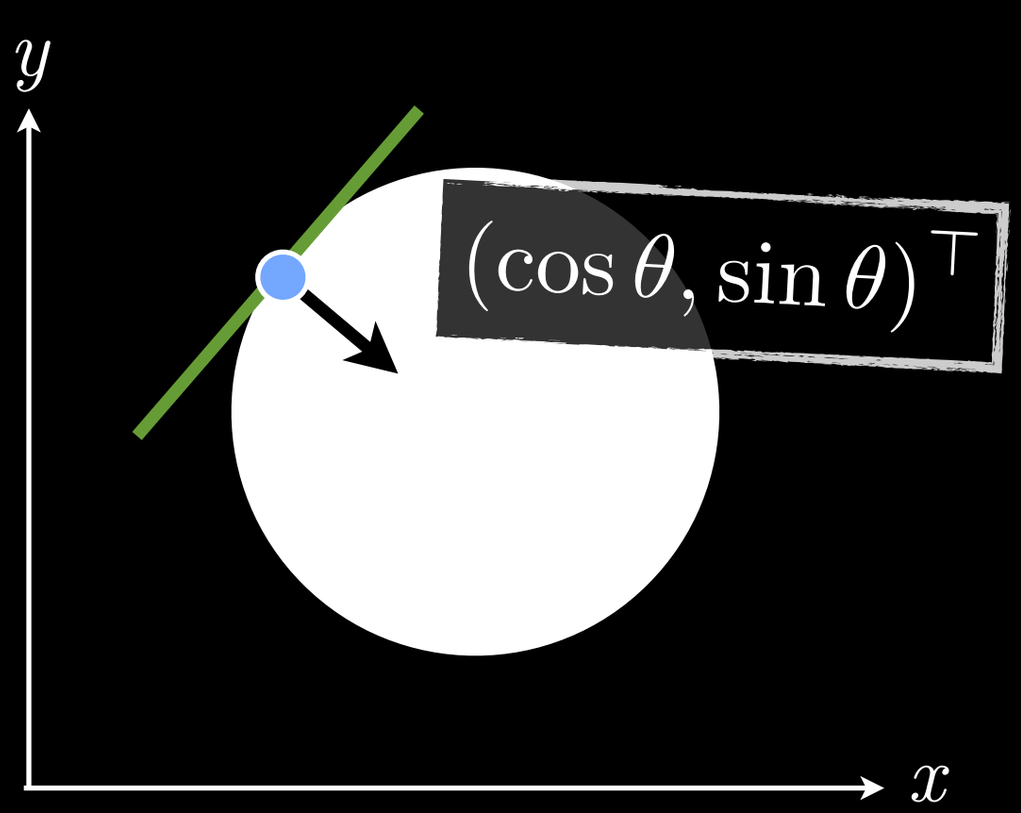
Unknown Radius
Known Gradient
Direction

$$(x_i - \alpha)^2 + (y_i - \beta)^2 = r^2$$

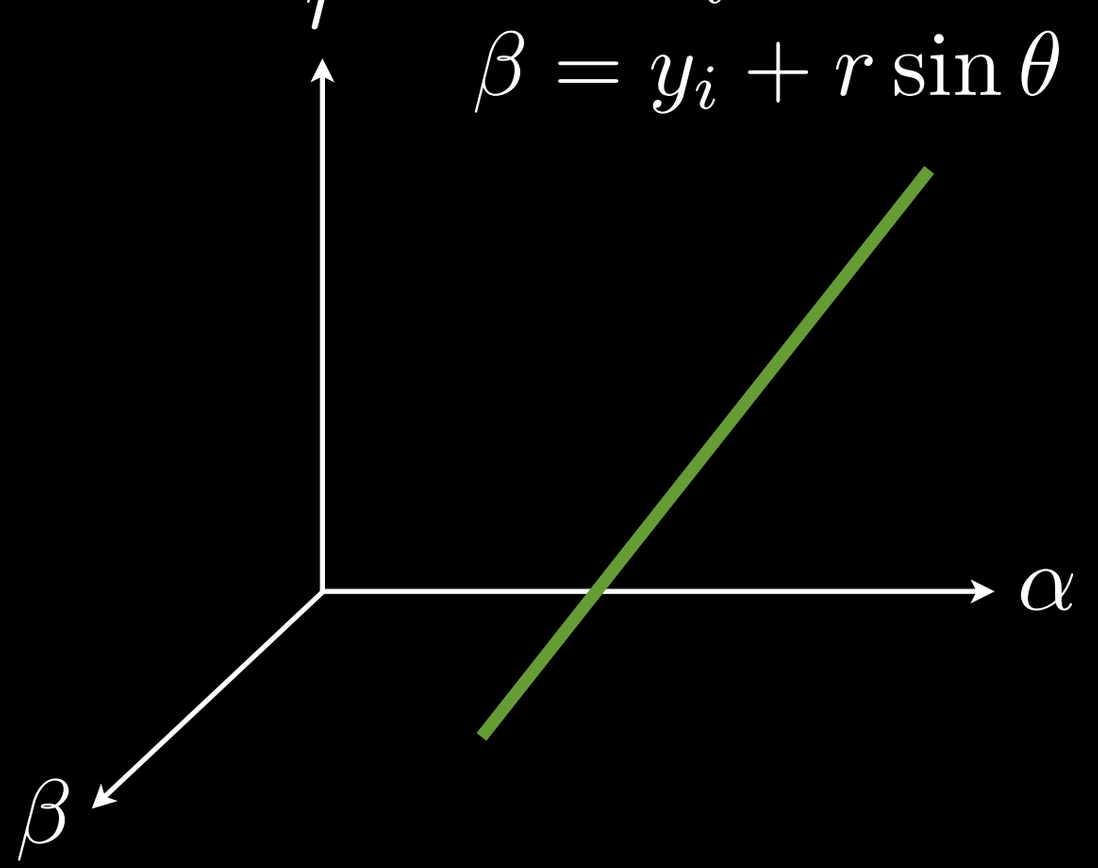


Unknown Radius
Known Gradient
Direction

$$(x_i - \alpha)^2 + (y_i - \beta)^2 = r^2$$



$$\alpha = x_i + r\cos\theta$$
$$\beta = y_i + r\sin\theta$$



Hough Circles

1: Initialize $H[\alpha, \beta, r] = 0$
2: **foreach** edgel $e \in I[x, y]$ **do**
3: | **foreach** possible radius r **do**
3: | | **foreach** possible gradient direction θ **do**
4: | | | $\alpha = x + r \cos\theta$
5: | | | $\beta = y + r \sin\theta$
6: | | | $H(\alpha, \beta, r) += 1$
7: | | **end**
8: | **end**
9: **end**
10: $\alpha^* \beta^*, r^* = \underset{\alpha,\beta,r}{\operatorname{argmax}} \; H[\alpha, \beta, r]$

Hough Circles

1: Initialize $H[\alpha, \beta, r] = 0$
2: **foreach** edgel $e \in I[x, y]$ **do**
3: | **foreach** possible radius r **do**
3: | | **foreach** possible gradient direction θ **do**
4: | | | $\alpha = x + r \cos\theta$
5: | | | $\beta = y + r \sin\theta$
6: | | | $H(\alpha, \beta, r) += 1$
7: | | **end**
8: | **end**
9: **end**
10: $\alpha^* \beta^*, r^* = \underset{\alpha, \beta, r}{\operatorname{argmax}} \; H[\alpha, \beta, r]$

Time complexity in terms of number of votes per point?

PROS

All points processed independently

Robustness to OUTLIERS

Can detect multiple model instances in a single pass

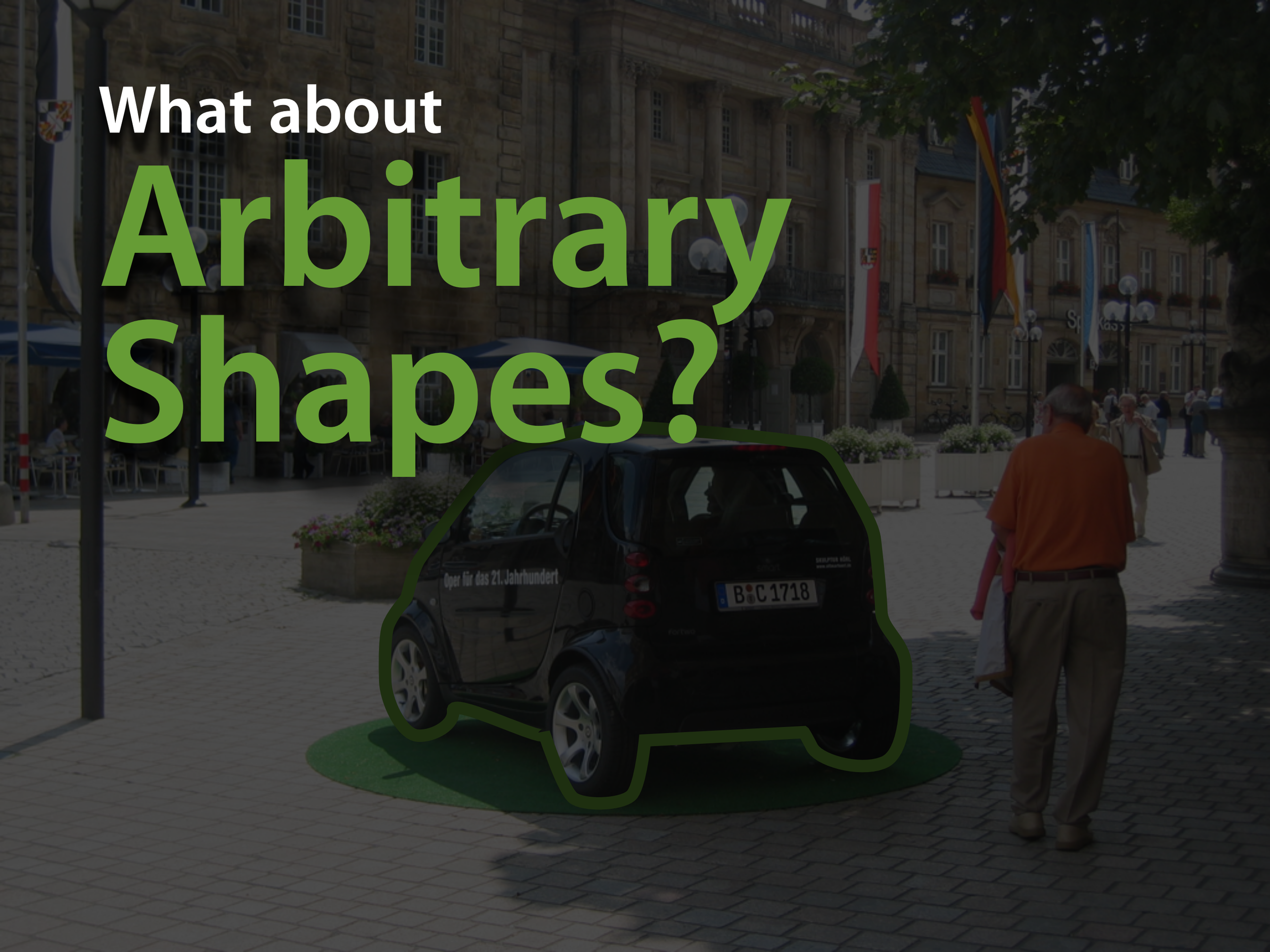
CONS

Search time complexity increases exponentially with number of parameters

Non-target shapes can produce spurious peaks in parameter space

Difficult to select an optimal quantization

What about Arbitrary Shapes?



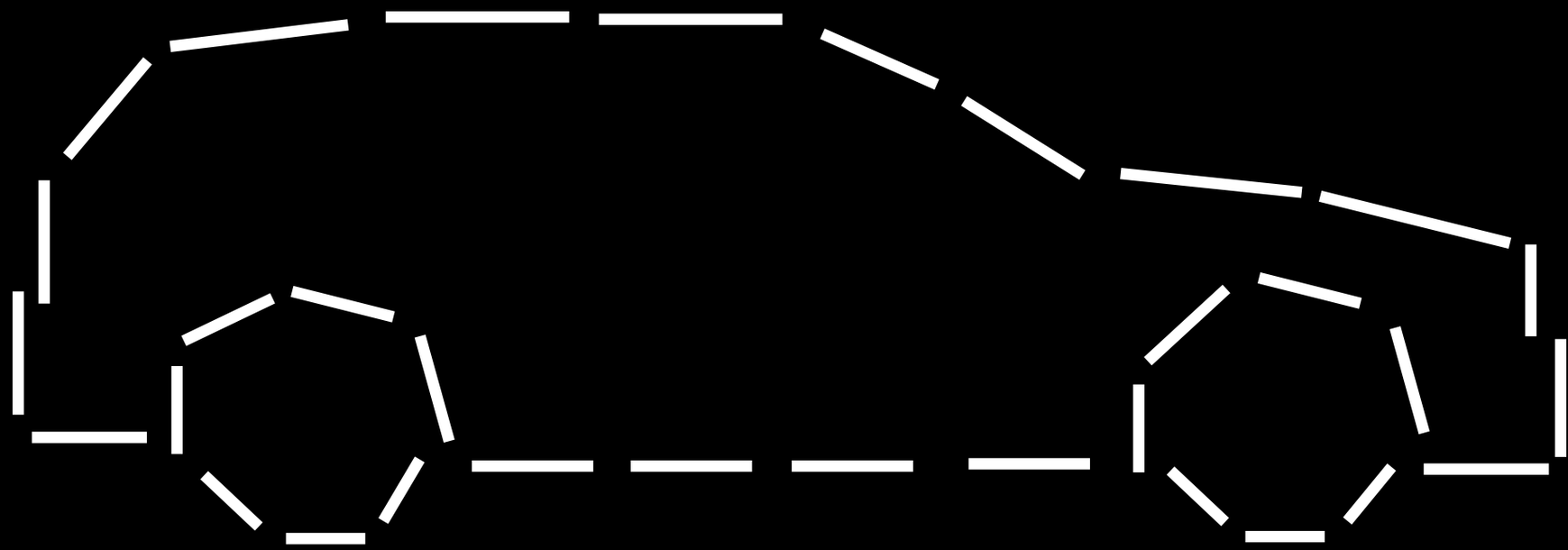
2

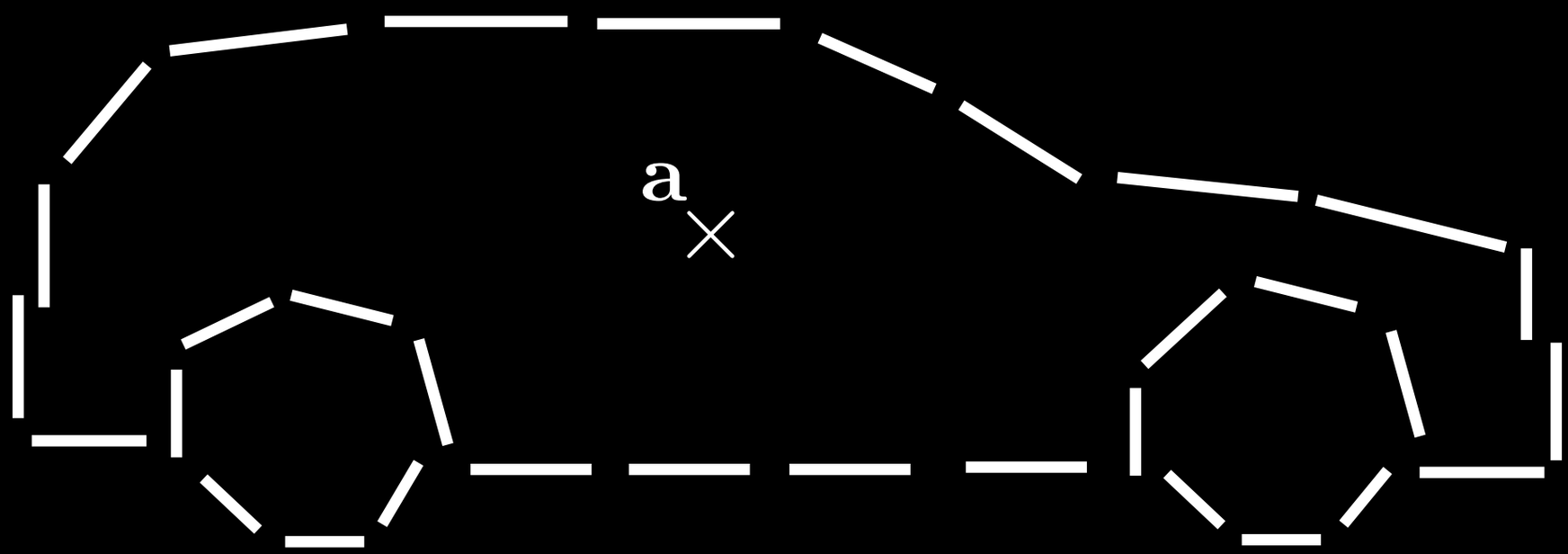
major steps

Step 1

Build shape model

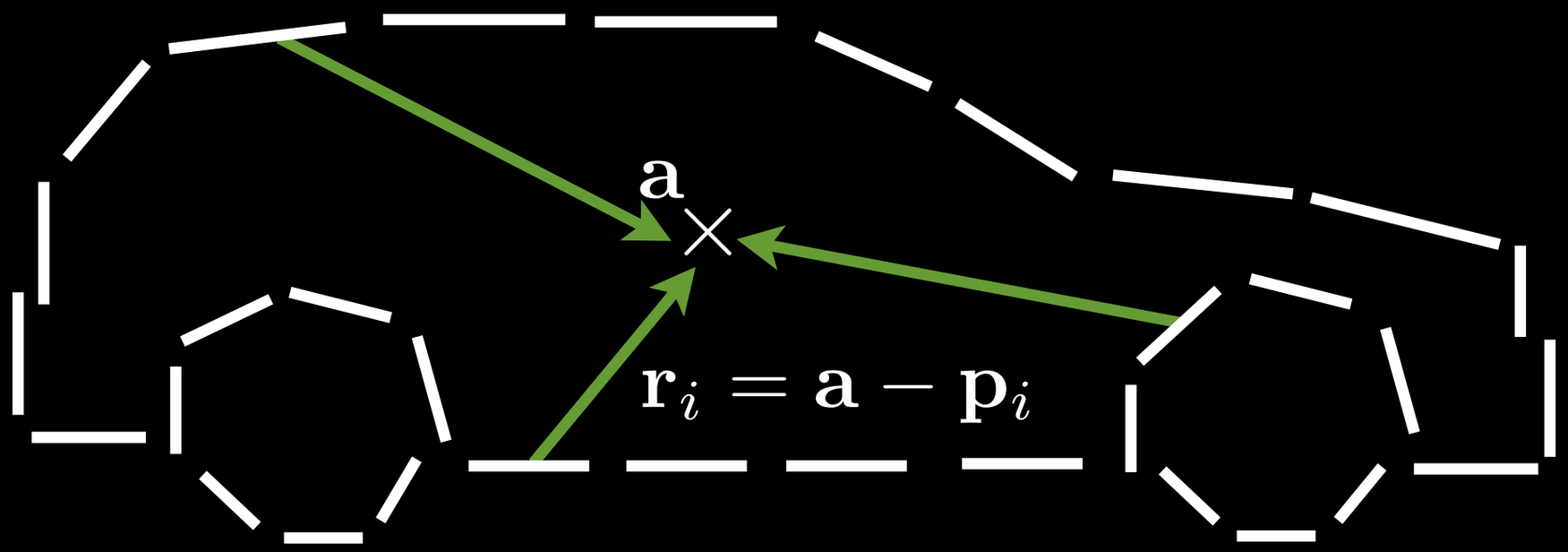
build a model. so compute an edge image



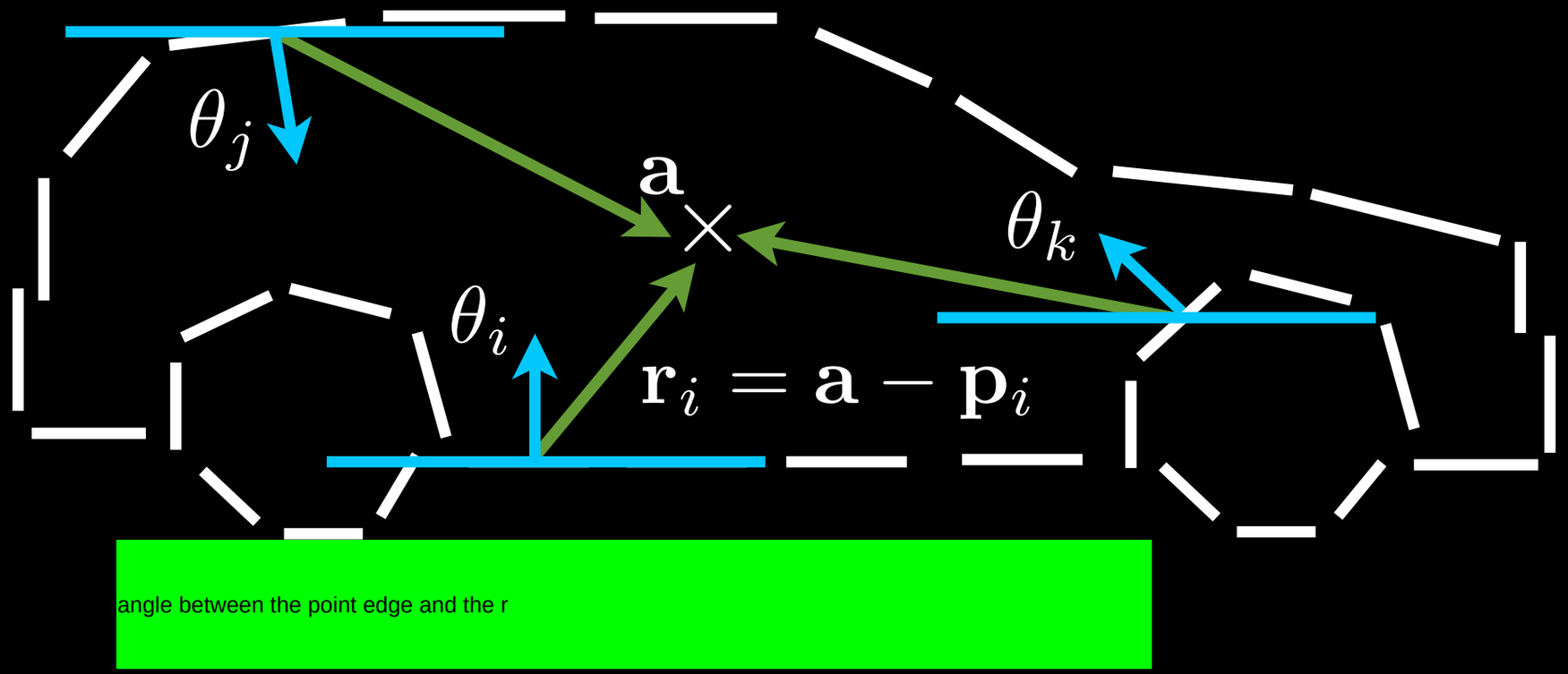


Define a reference point, a

arbbbitray



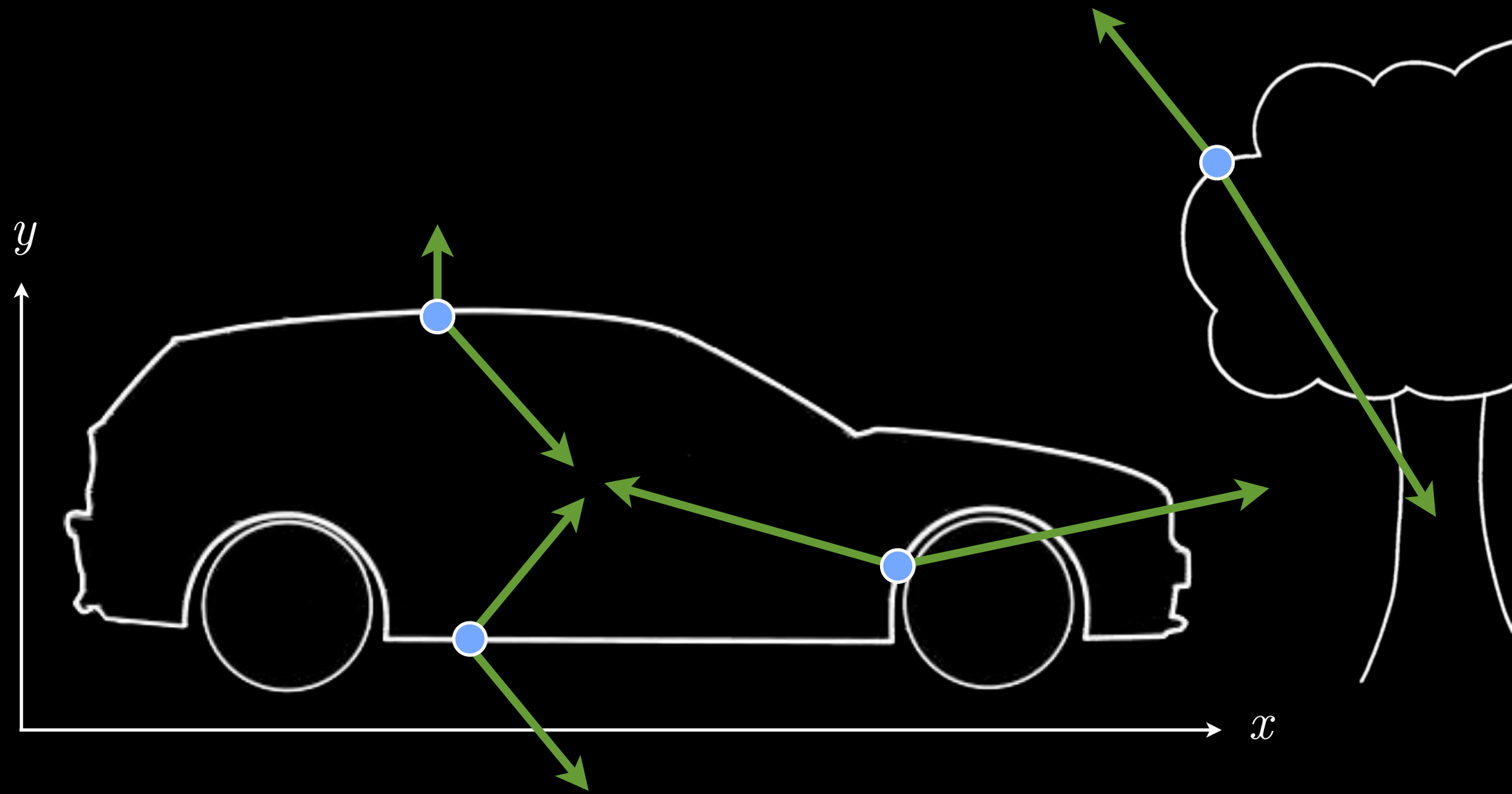
At each boundary point, compute the displacement vector $\mathbf{r}_i = \mathbf{a} - \mathbf{p}_i$

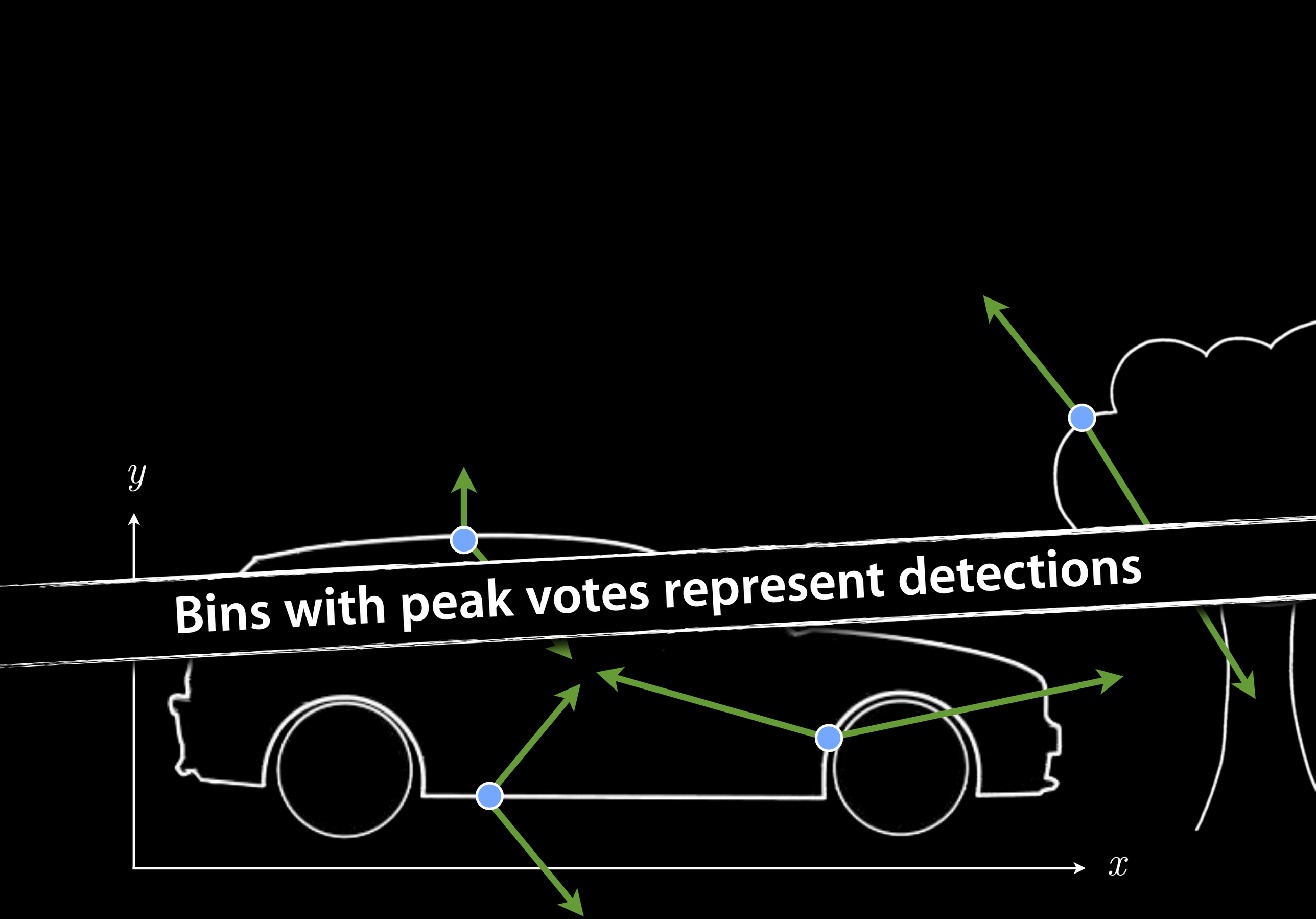


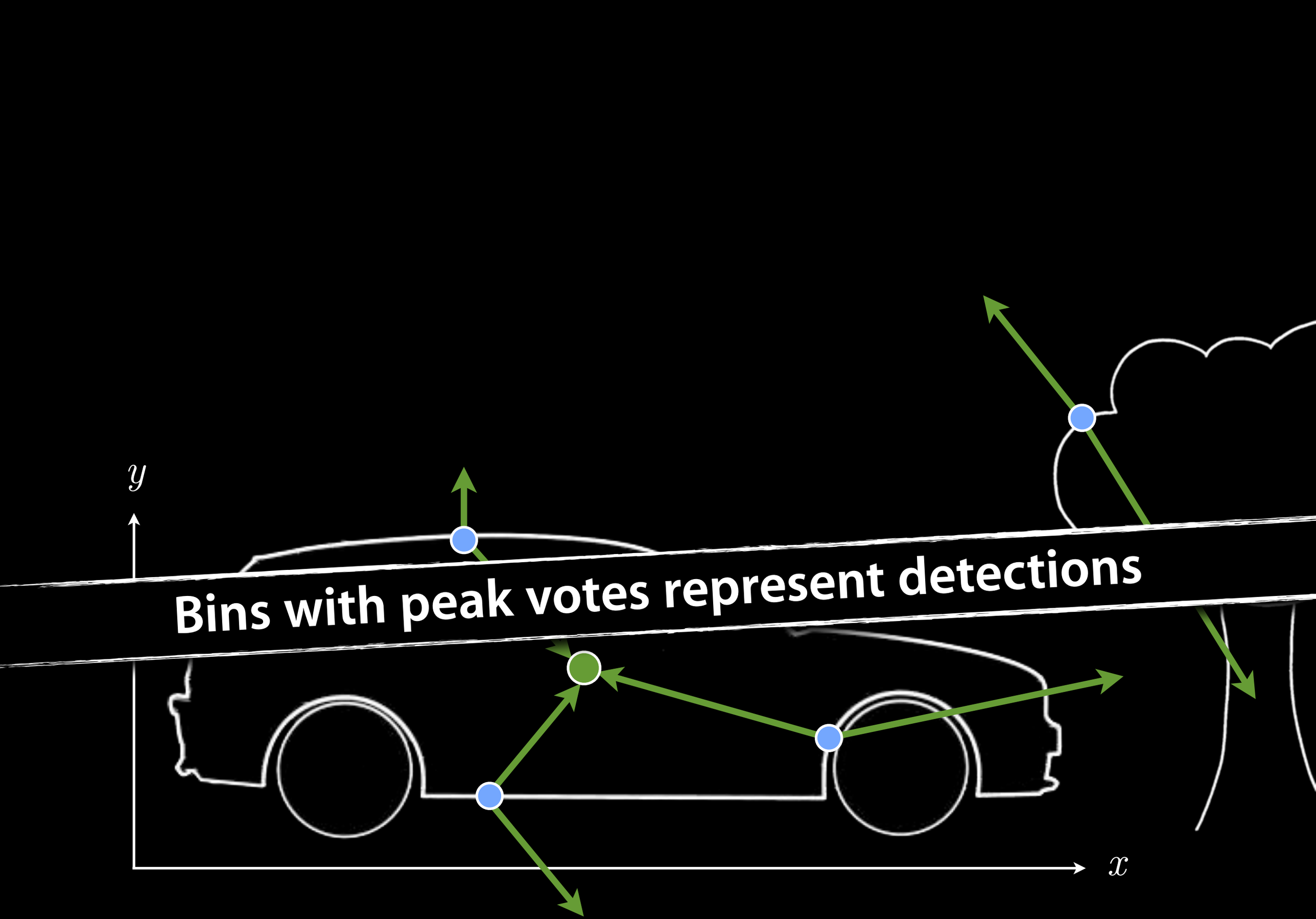
Store displacement vectors in a table indexed by the gradient orientation, θ

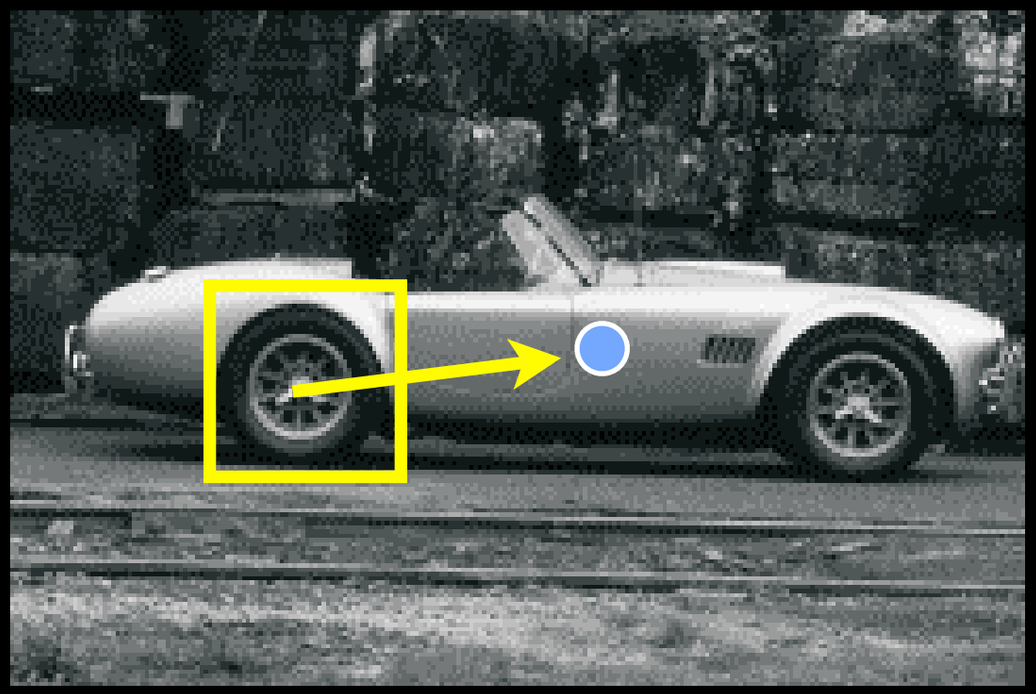
Step 2

Shape detection

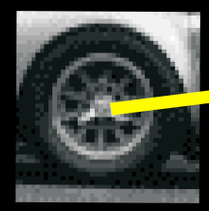

y
x


y
x
Bins with peak votes represent detections


y
x
Bins with peak votes represent detections

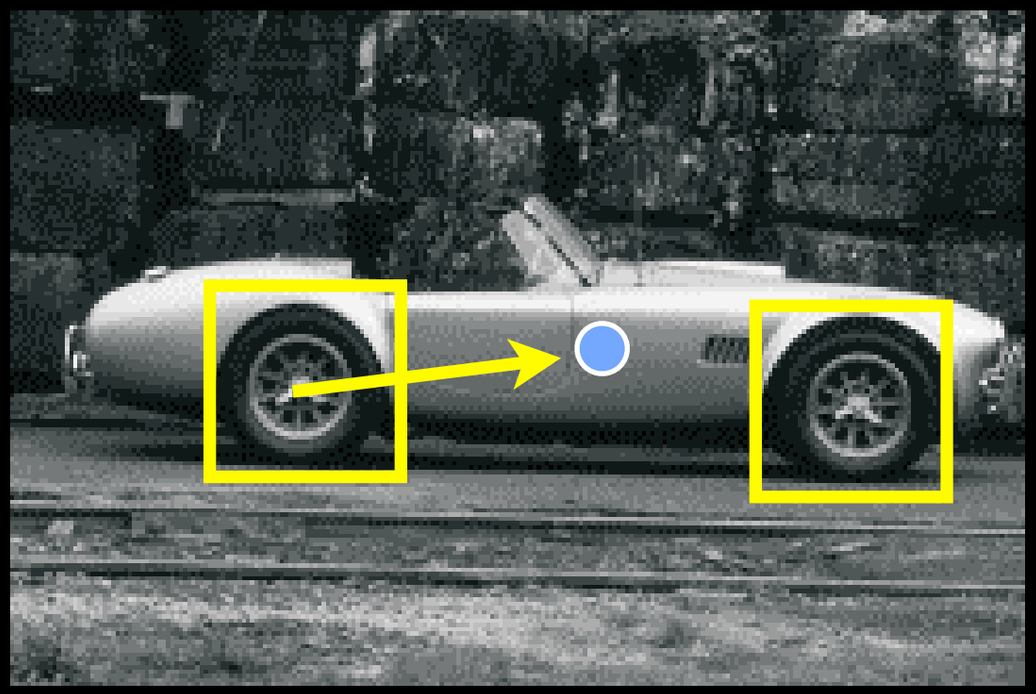


training image

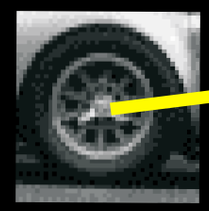


visual codeword

Instead of indexing displacements by gradient orientation index by “visual codeword”

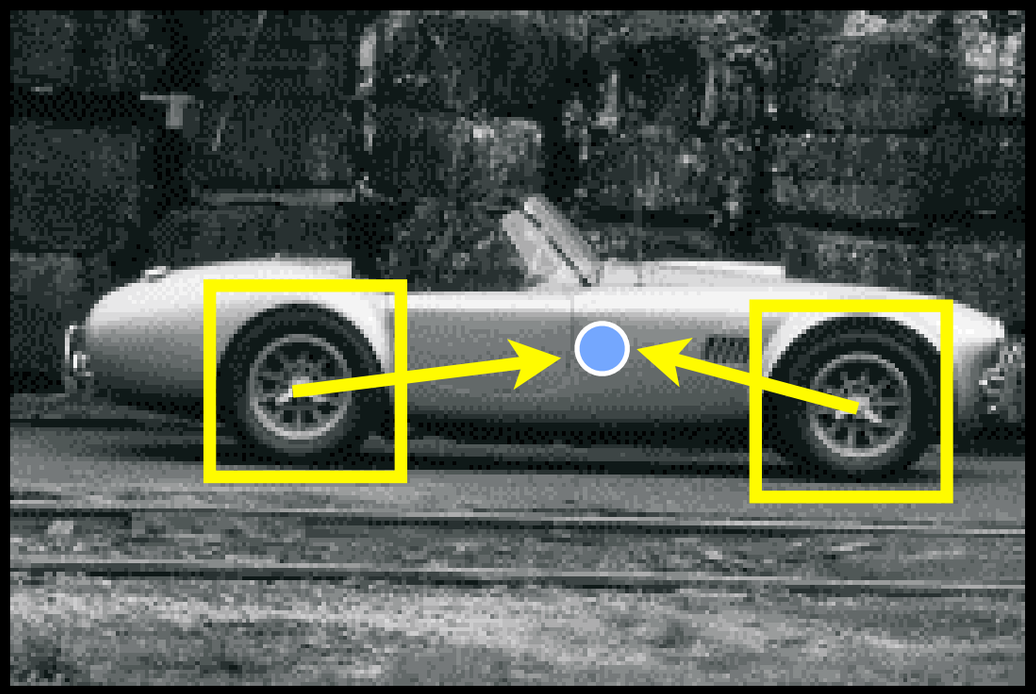


training image

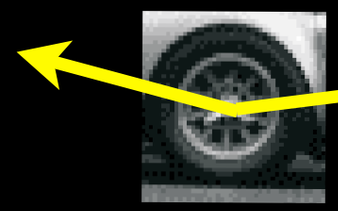


visual codeword

Instead of indexing displacements by gradient orientation index by “visual codeword”

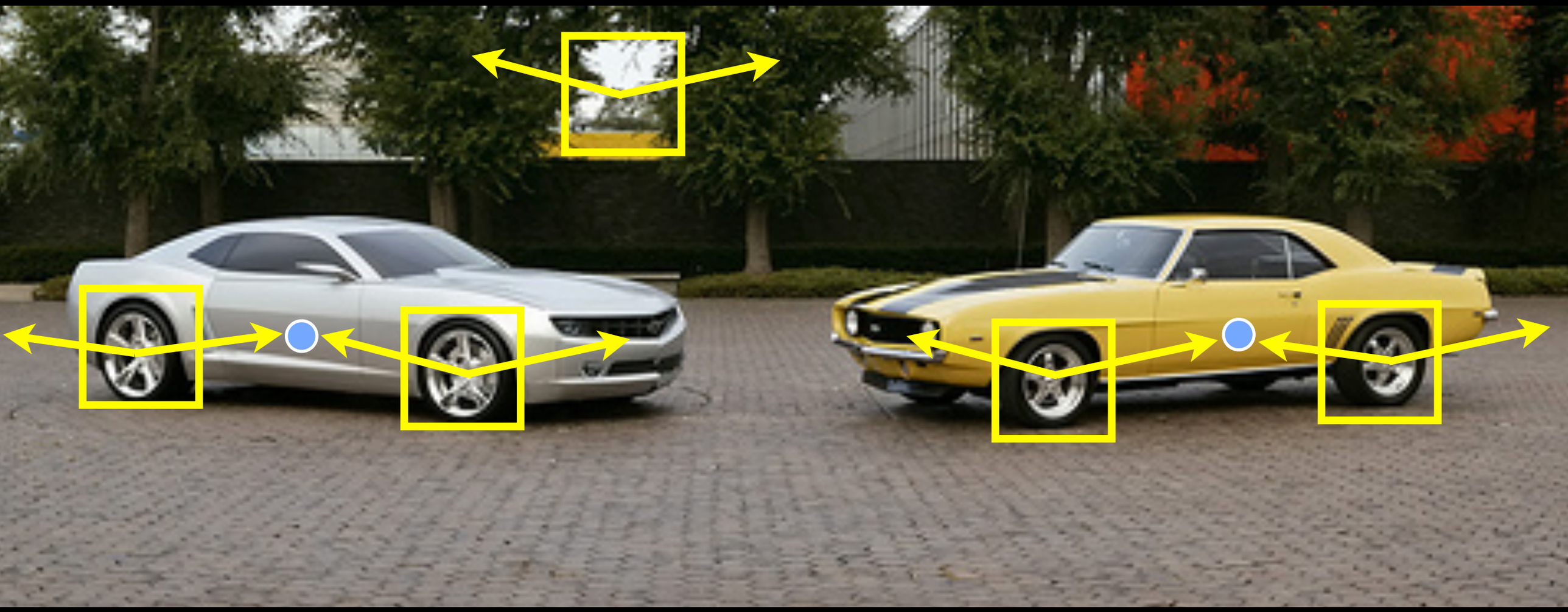


training image



visual codeword

Instead of indexing displacements by gradient orientation index by “visual codeword”

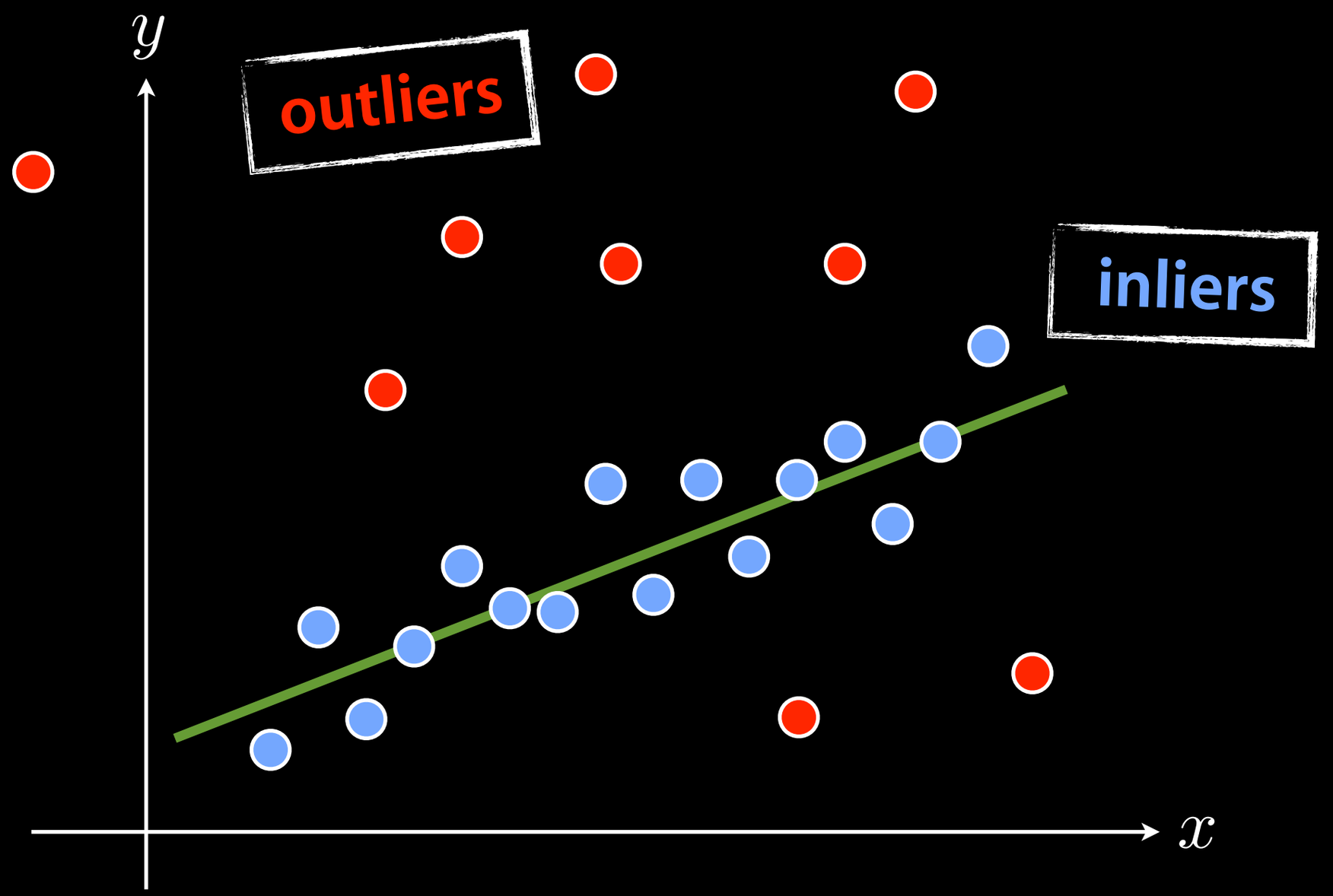


test image

RANSAC

RANdom **SA**mple **C**onsensus

General procedure for fitting a parametric model to data containing outliers



Goal: Fit the model to the data with outliers

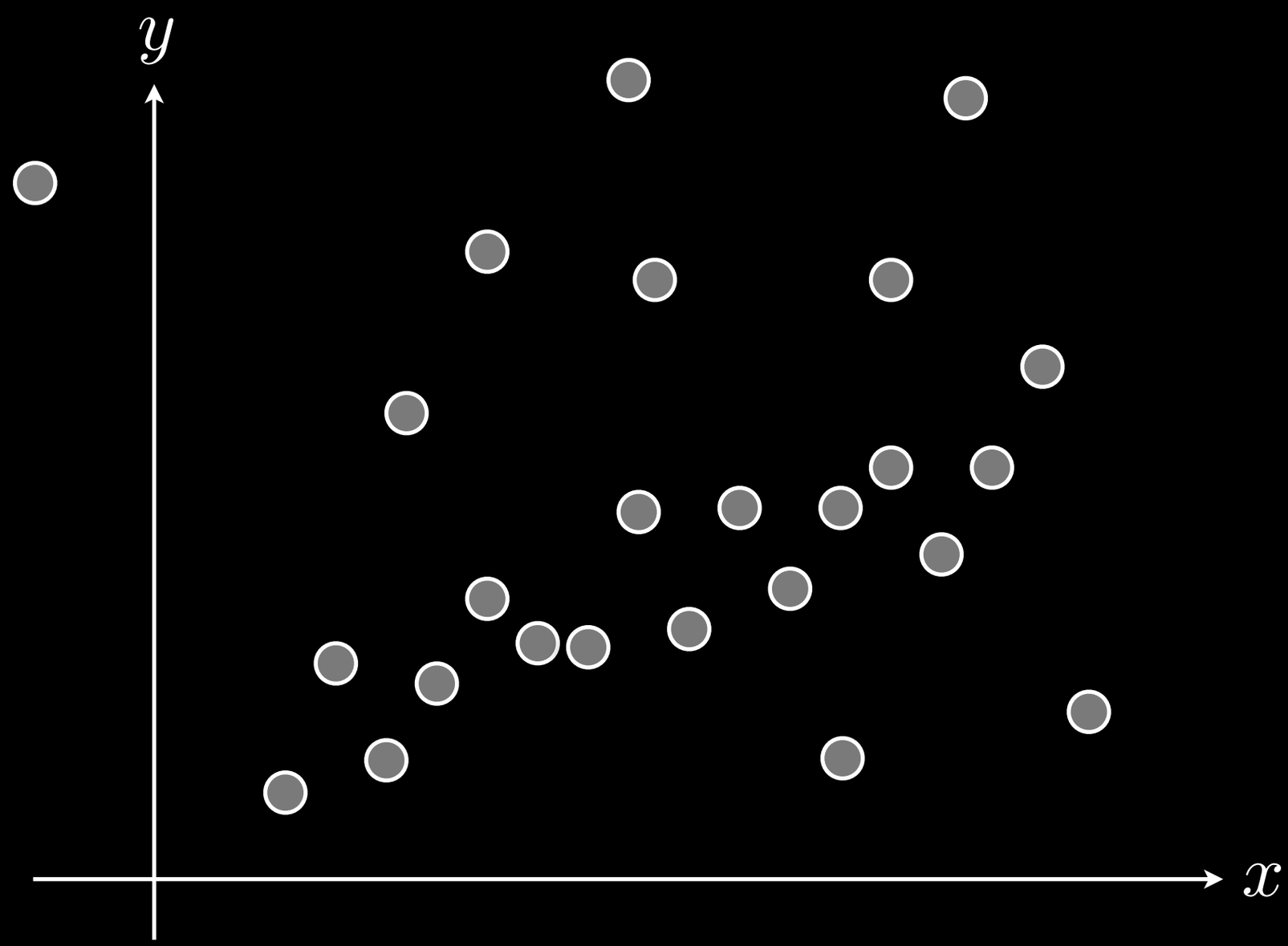
unknown

4

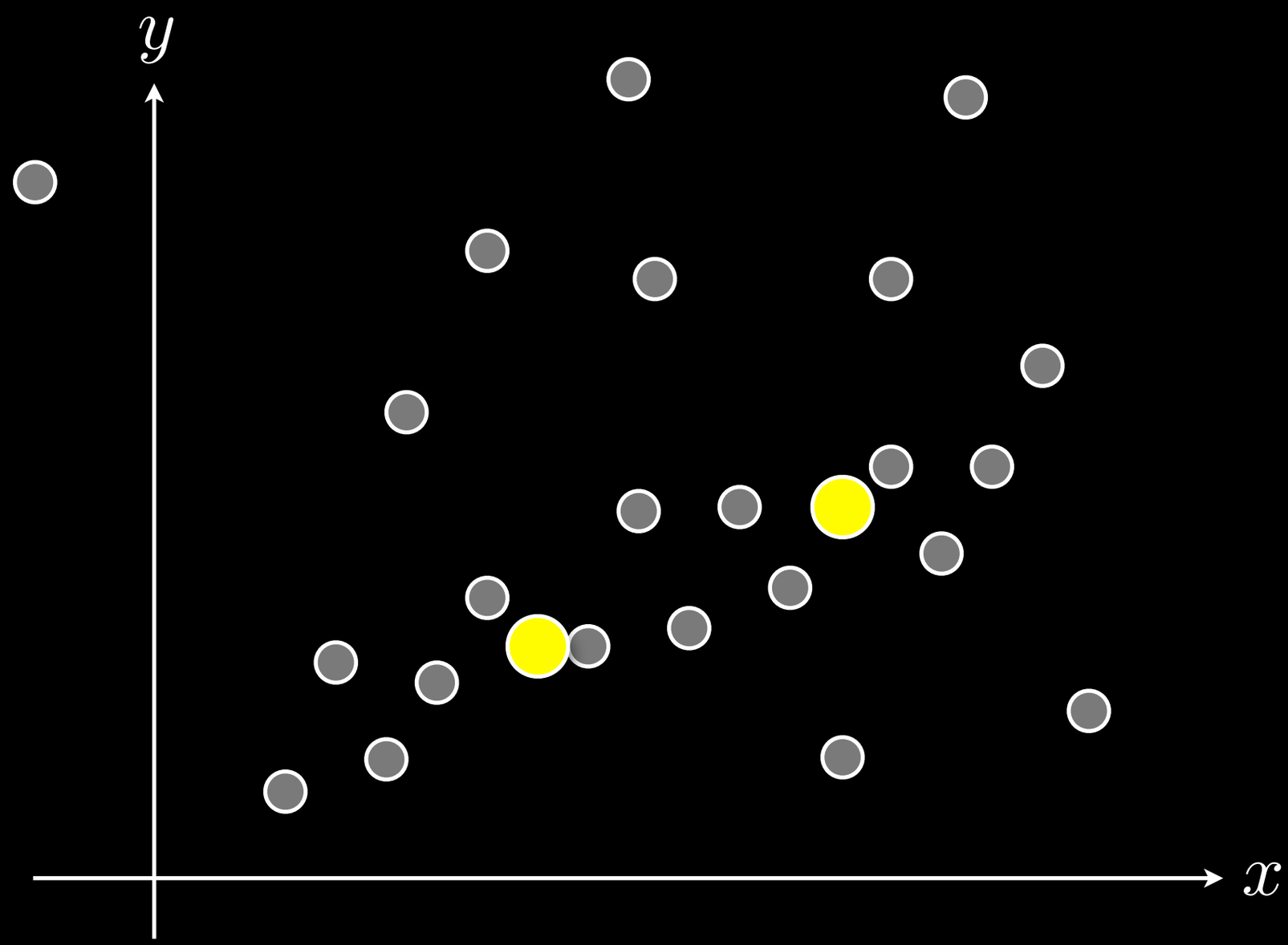
major steps

Step 1

Select minimum point set



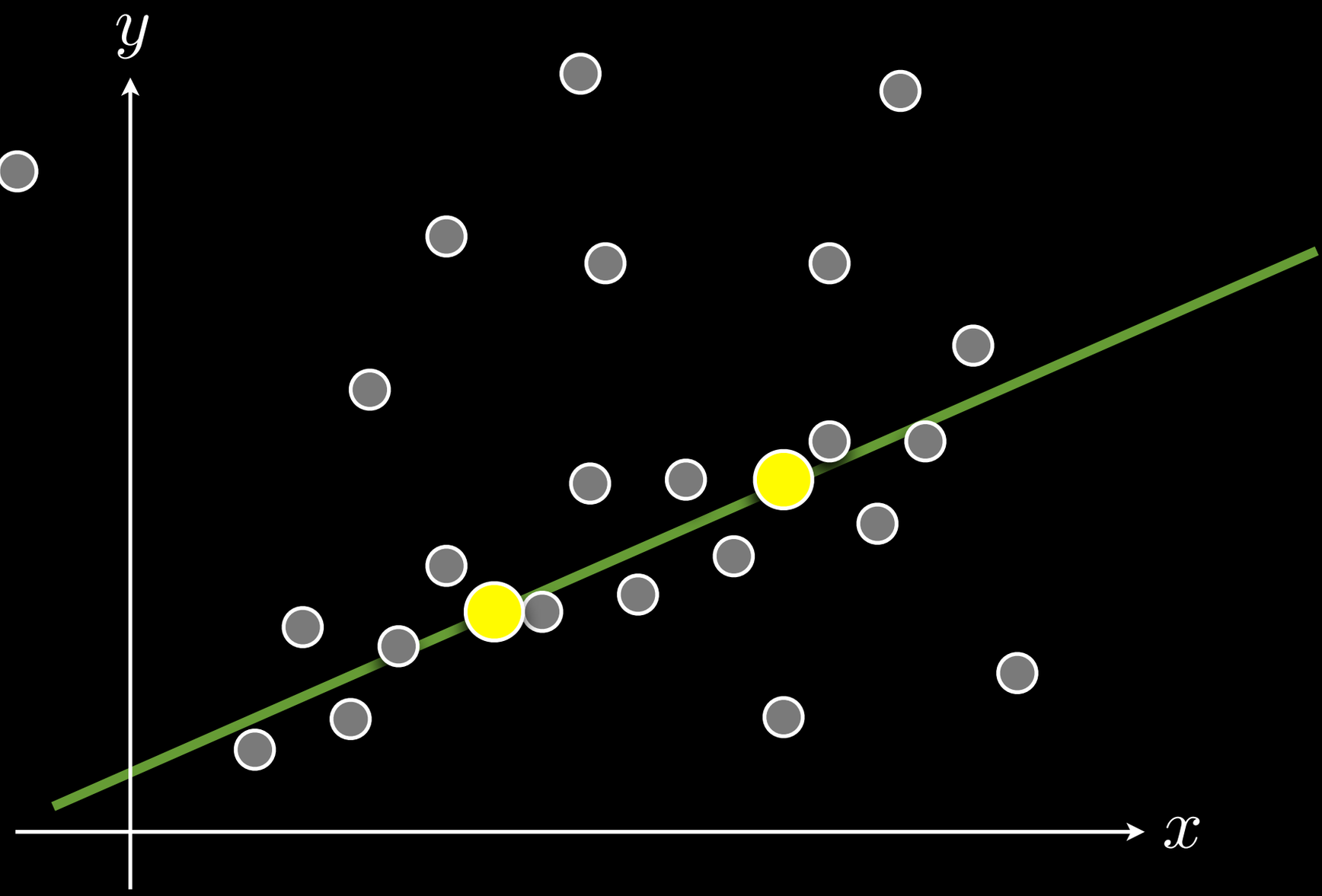
Select random **minimum** set of points to fit the model



What is the size of the minimum set?

Step 2

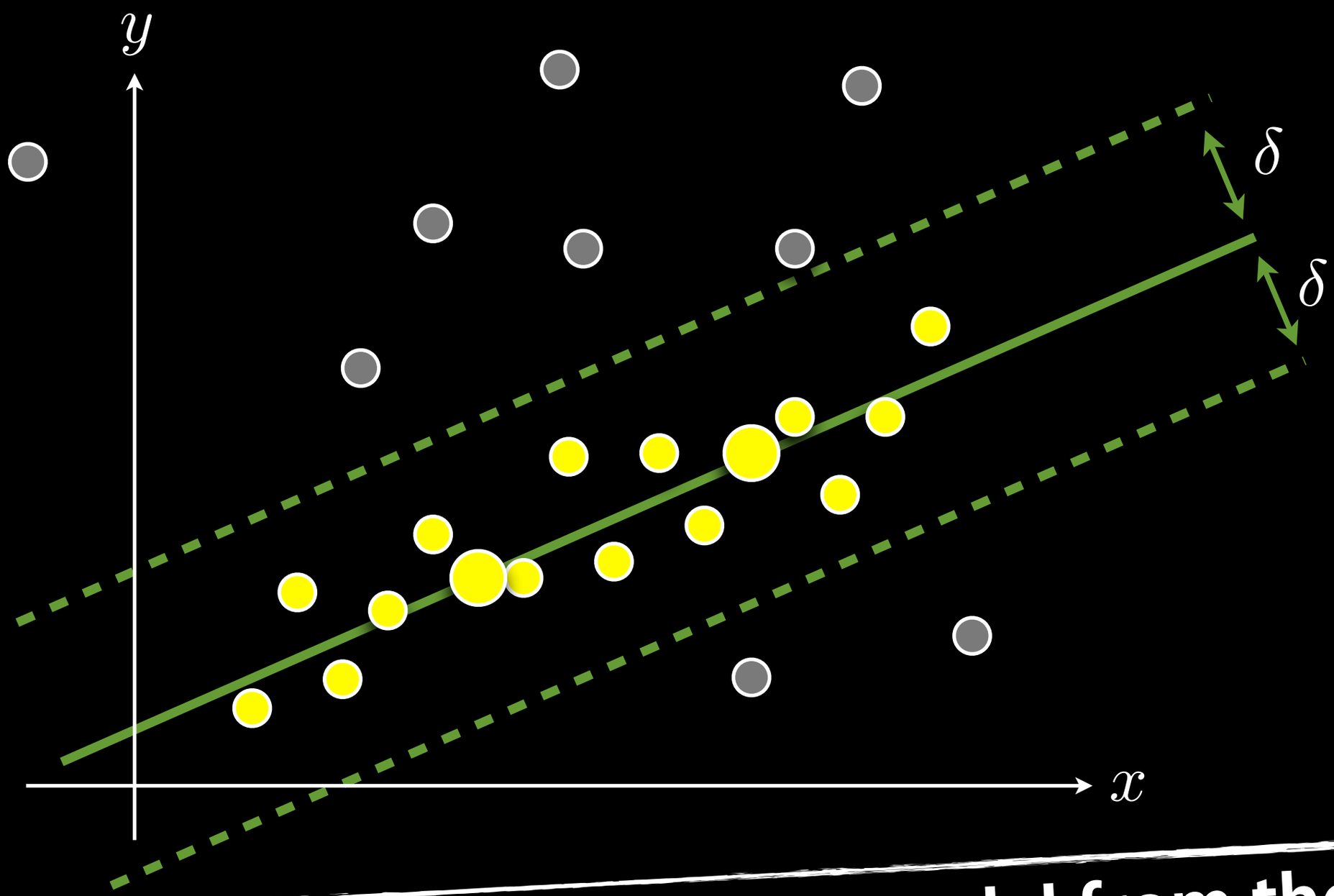
Find best fitting line



Compute the best fitting line

Step 3

Determine inliers



Compute the set of inliers to model from the dataset based on threshold

Step 4

Repeat steps 1-3 for N steps and estimate the final model using the sample with the most inliers

How many samples?

$N =$ **number of samples**

$e =$ **probability that a point is an outlier**

$s =$ **number of points in a sample**

$p =$ **desired probability that we get a good sample**

$N =$ **number of samples**

$e =$ **probability that a point is an outlier**

$s =$ **number of points in a sample**

$p =$ **desired probability that we get a good sample**

$$N = \frac{\log(1-p)}{\log(1-(1-e)^s)}$$

$N =$ **number of samples**

$e =$ **probability that a point is an outlier**

$s =$ **number of points in a sample**

$p =$ **desired probability that we get a good sample**

$$1 - e$$

Probability that choosing one point yields an inlier

$N =$ **number of samples**

$e =$ **probability that a point is an outlier**

$s =$ **number of points in a sample**

$p =$ **desired probability that we get a good sample**

$$(1 - e)^s$$

Probability of choosing s inliers in a row

$N =$ **number of samples**

$e =$ **probability that a point is an outlier**

$s =$ **number of points in a sample**

$p =$ **desired probability that we get a good sample**

$$(1 - e)^s$$

Sample only contains inliers

$N =$ **number of samples**

$e =$ **probability that a point is an outlier**

$s =$ **number of points in a sample**

$p =$ **desired probability that we get a good sample**

$$1 - (1 - e)^s$$

Probability that one or more sample points are outliers

$N =$ **number of samples**

$e =$ **probability that a point is an outlier**

$s =$ **number of points in a sample**

$p =$ **desired probability that we get a good sample**

$$1 - (1 - e)^s$$

Sample is contaminated

$N =$ **number of samples**

$e =$ **probability that a point is an outlier**

$s =$ **number of points in a sample**

$p =$ **desired probability that we get a good sample**

$$(1-(1-e)^s)^N$$

Probability that N samples were contaminated

$N =$ **number of samples**

$e =$ **probability that a point is an outlier**

$s =$ **number of points in a sample**

$p =$ **desired probability that we get a good sample**

$$1-(1-(1-e)^s)^N$$

Probability at least one sample of s points contains only inliers

$N =$ **number of samples**

$e =$ **probability that a point is an outlier**

$s =$ **number of points in a sample**

$p =$ **desired probability that we get a good sample**

$$1-(1-(1-e)^s)^N$$

At least one sample was not contaminated

$N =$ **number of samples**

$e =$ **probability that a point is an outlier**

$s =$ **number of points in a sample**

$p =$ **desired probability that we get a good sample**

$$p = 1 - (1 - (1 - e)^s)^N$$

$N =$ **number of samples**

$e =$ **probability that a point is an outlier**

$s =$ **number of points in a sample**

$p =$ **desired probability that we get a good sample**

$$p = 1 - (1 - (1 - e)^s)^N$$

How to solve for N ?

$N =$ **number of samples**

$e =$ **probability that a point is an outlier**

$s =$ **number of points in a sample**

$p =$ **desired probability that we get a good sample**

$$p = 1 - (1 - (1 - e)^s)^N$$

$$N = \frac{\log(1 - p)}{\log(1 - (1 - e)^s)}$$

$$N = \frac{\log(1-p)}{\log(1-(1-e)^s)}$$

Choose N such that with probability $p = 0.99$ at least one random sample is free from outliers

	proportion of outliers, e (%)						
s	5	10	20	25	30	40	50
2	2	3	5	6	7	11	17
3	3	4	7	9	11	19	35
4	3	5	9	13	17	34	72
5	4	6	12	17	26	57	146
6	4	7	16	24	37	97	293
7	4	8	20	33	54	163	588
8	5	9	26	44	78	272	1177

PROS

Simple to implement

Applicable in many diverse contexts

CONS

Many parameters to tune

Cannot be used if ratio of inliers/outliers is too small

Planar
Panorama

map the images onto a common image plane

each image is warped with a homography

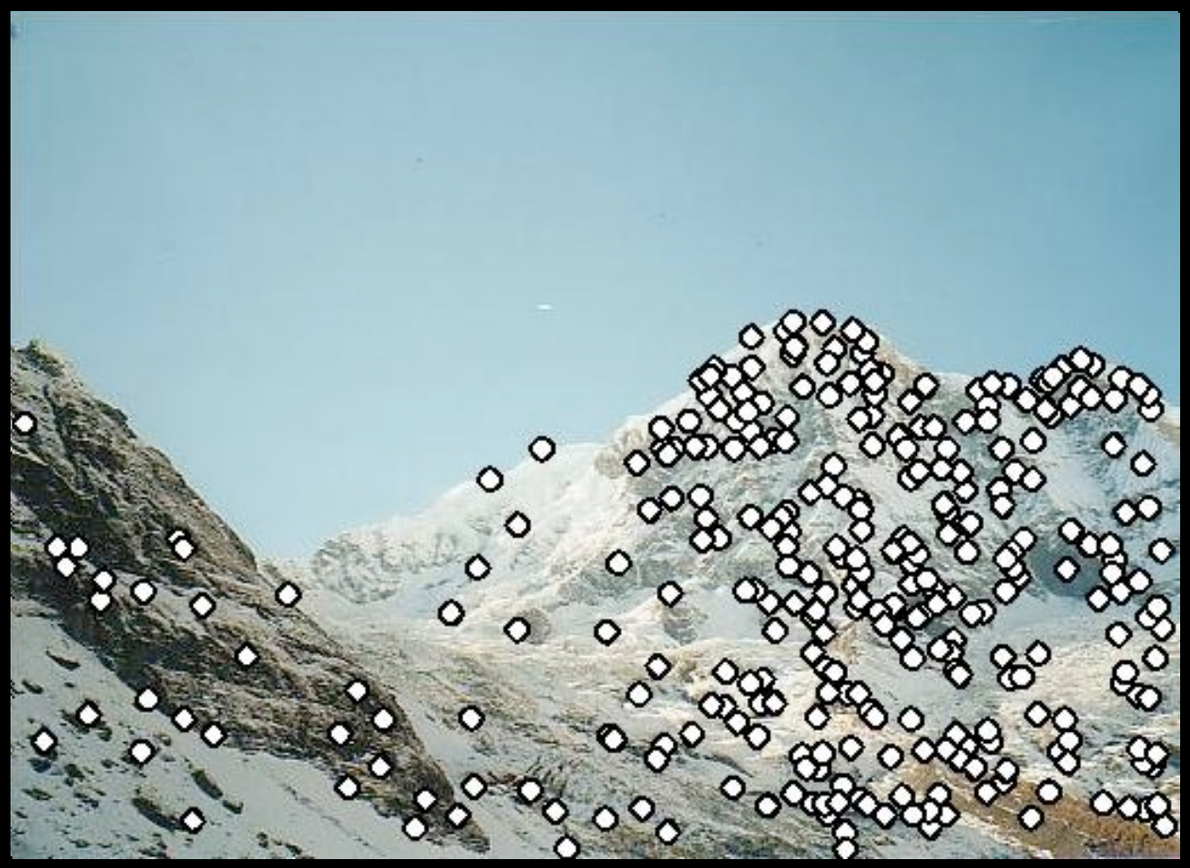
5

major steps

Step 1

Extract features

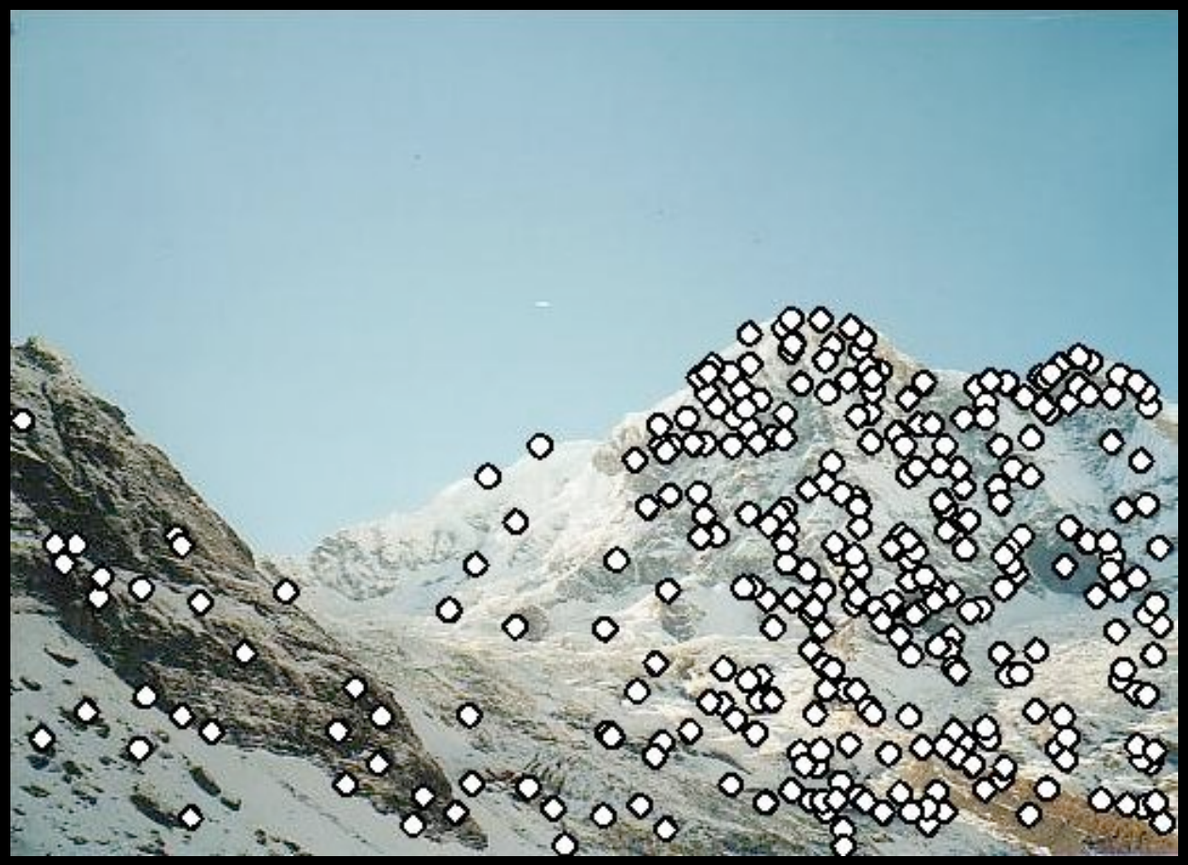
Independently extract features in each image

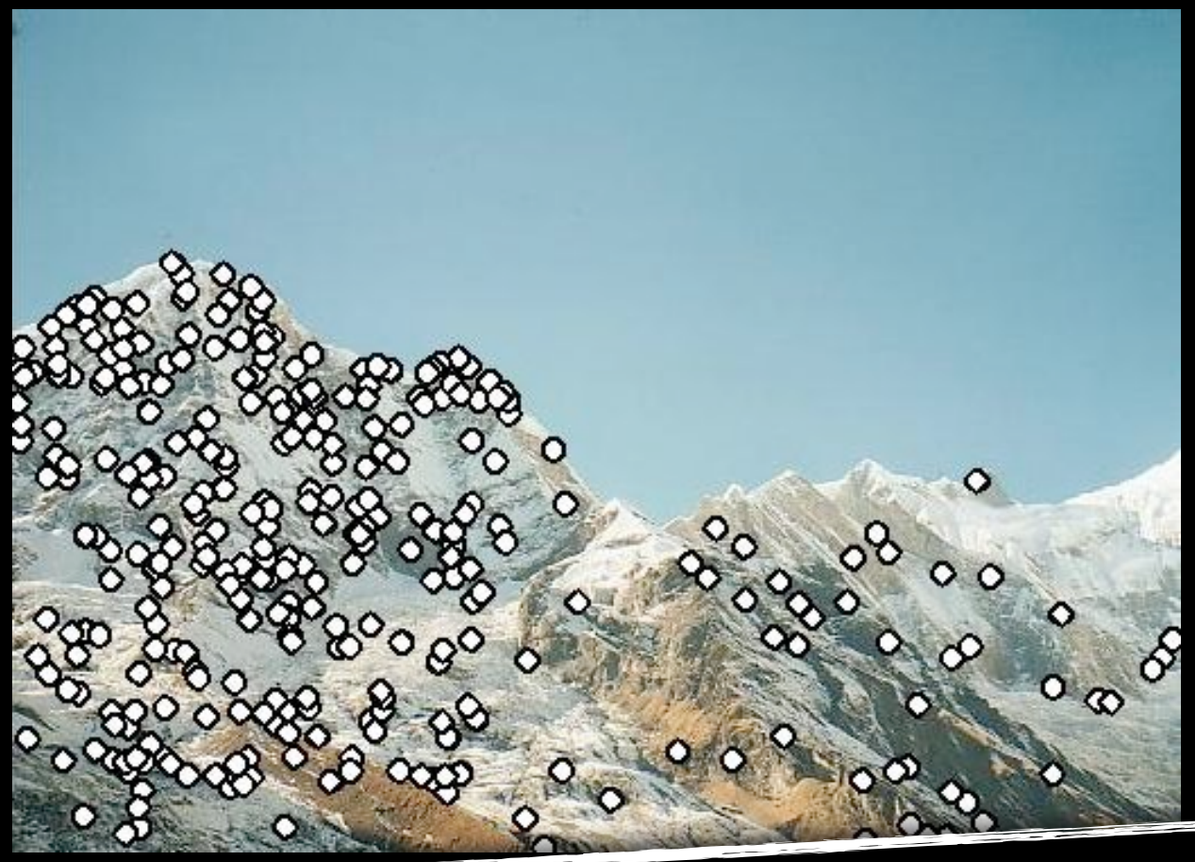


Independently extract features in each image

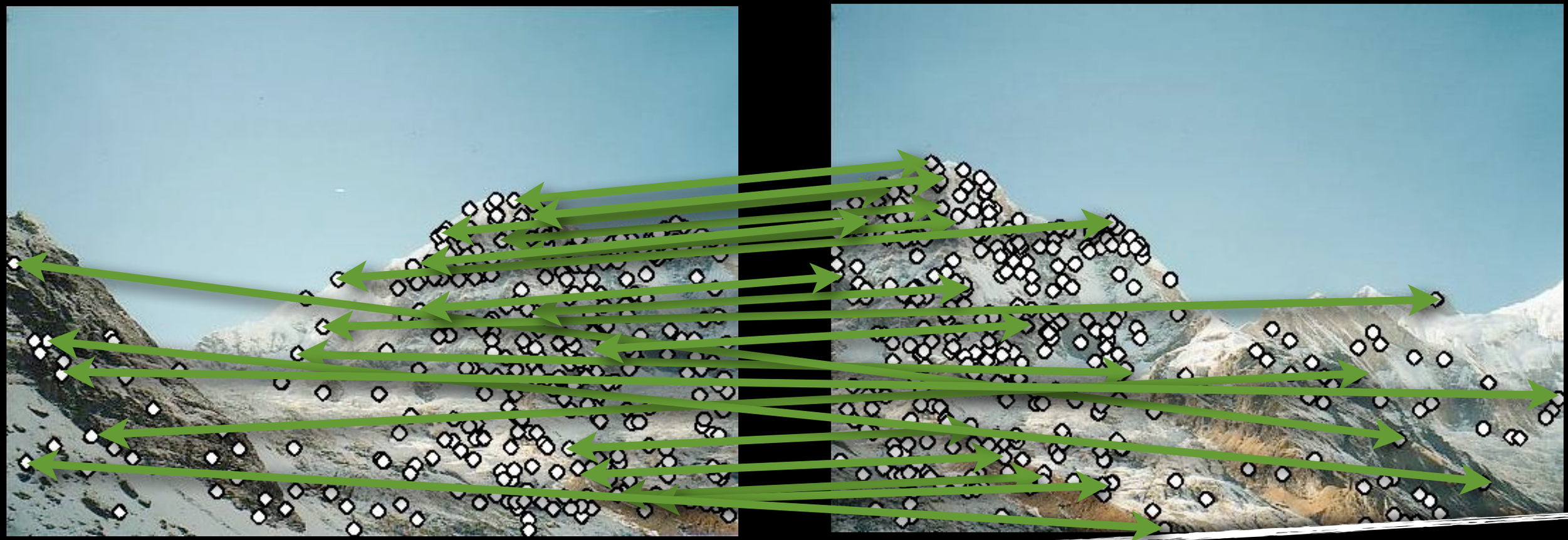
Step 2

Compute putative matches





Compute putative matches between images

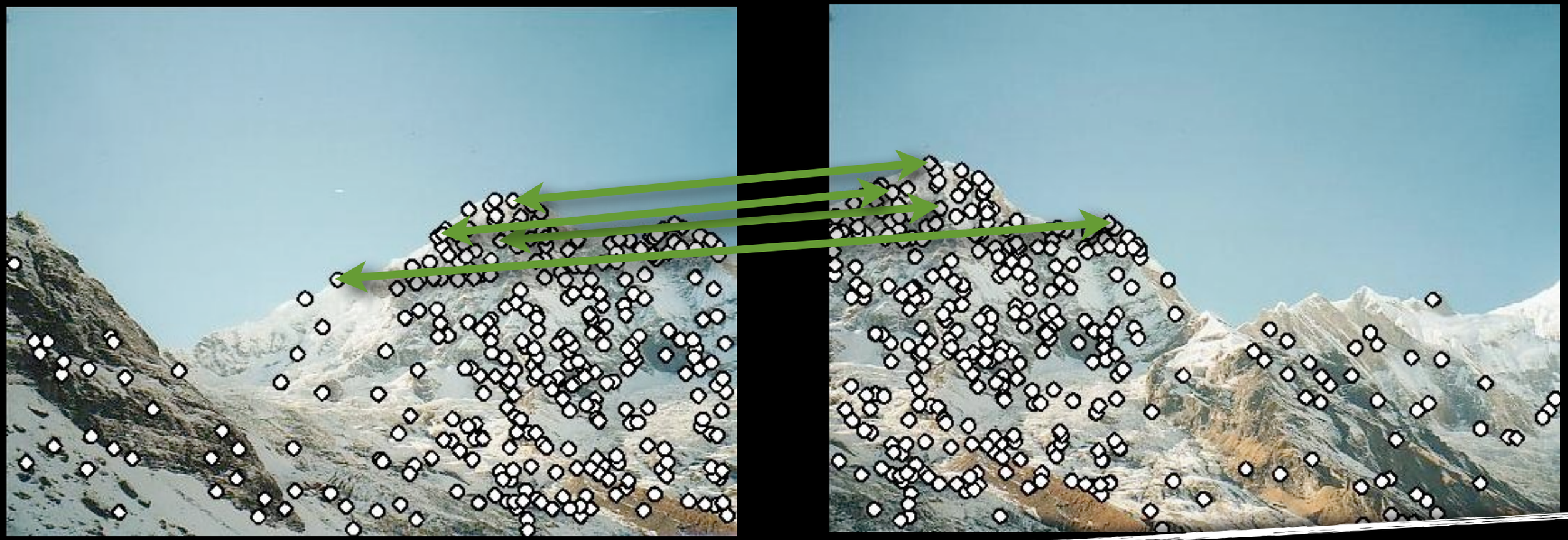


Compute putative matches between images

for every point on the left, find closest match in right imatch, compare feature descriptor to the left, take the one that is closest in terms

Step 3

Hypothesize transformation

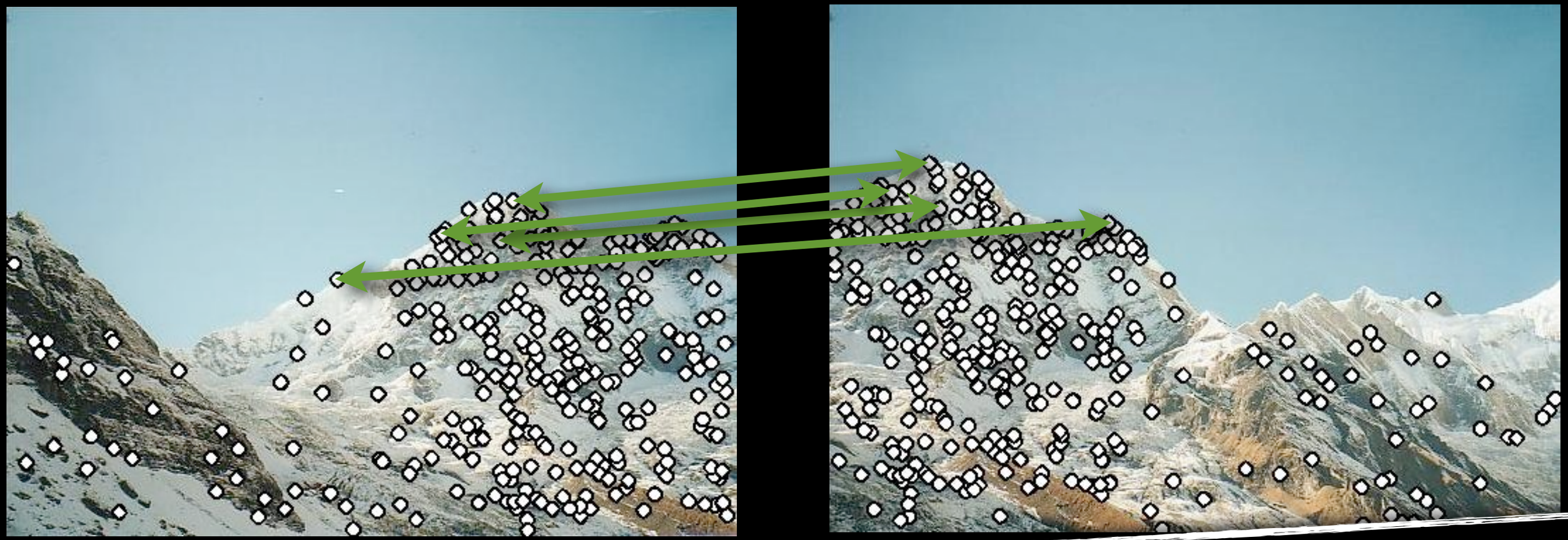


Select minimal point set and compute transformation

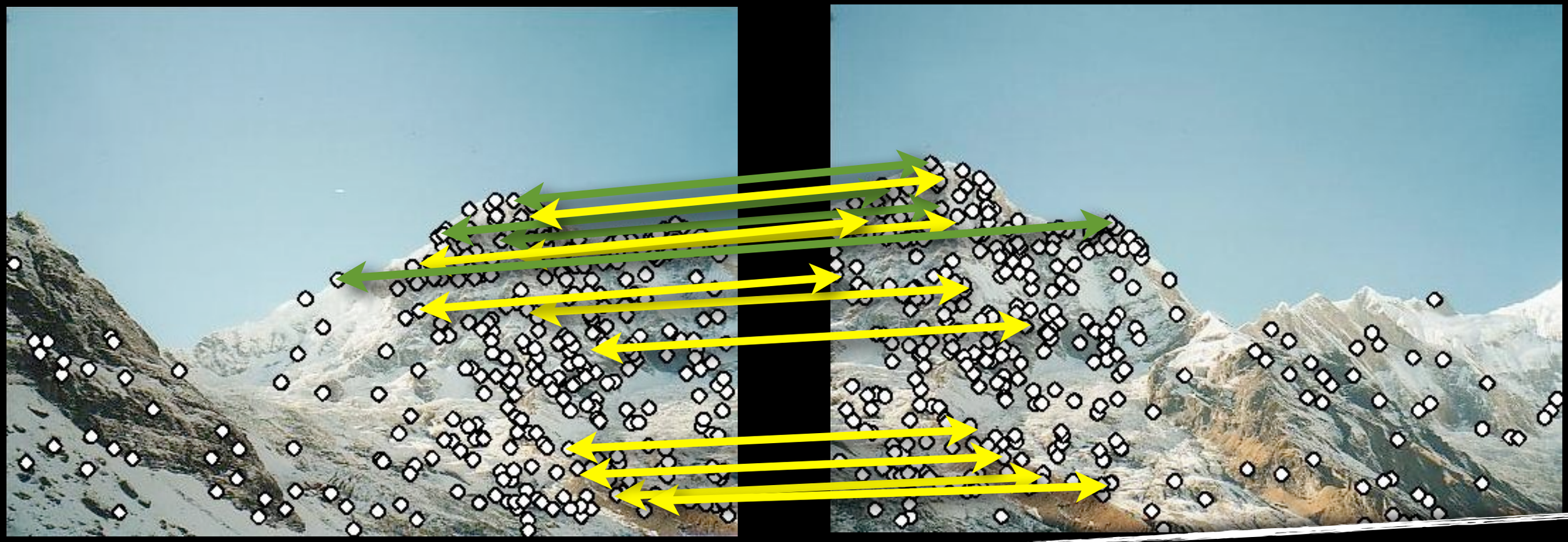
take the minimal set in which the matches are most similar

Step 4

Verify transformation



Search for matches consistent with transformation



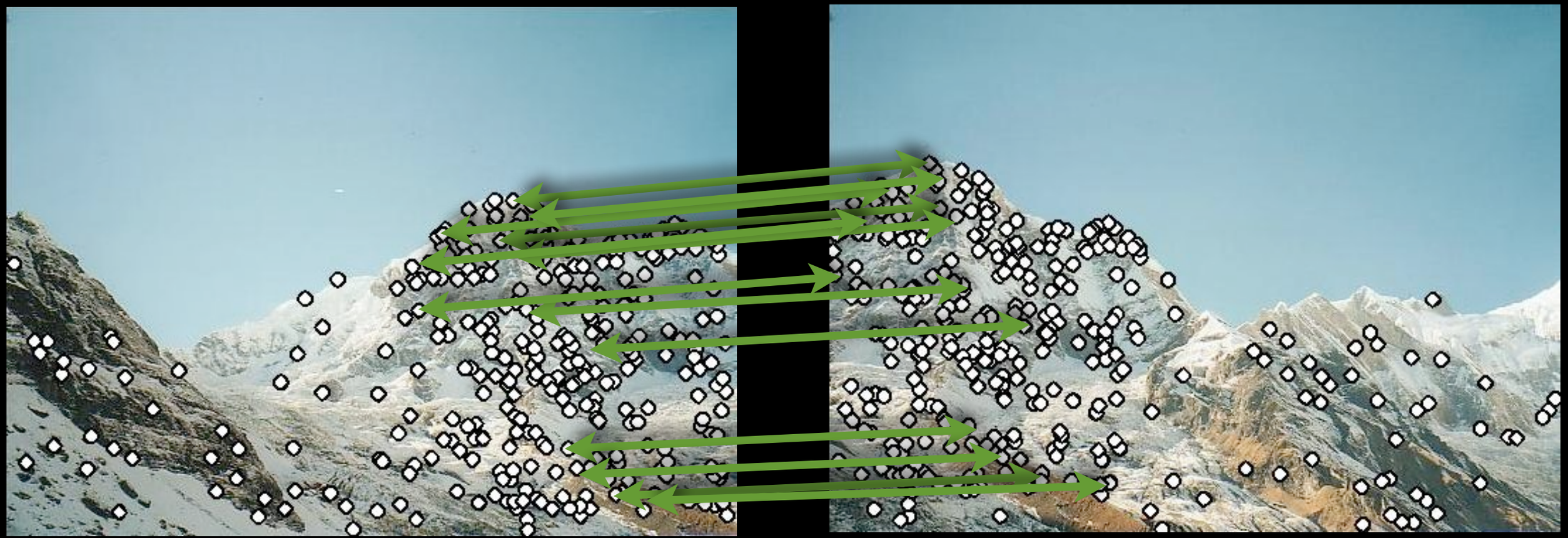
Search for matches consistent with transformation

Steps 3-4

Repeat hypothesize and verify

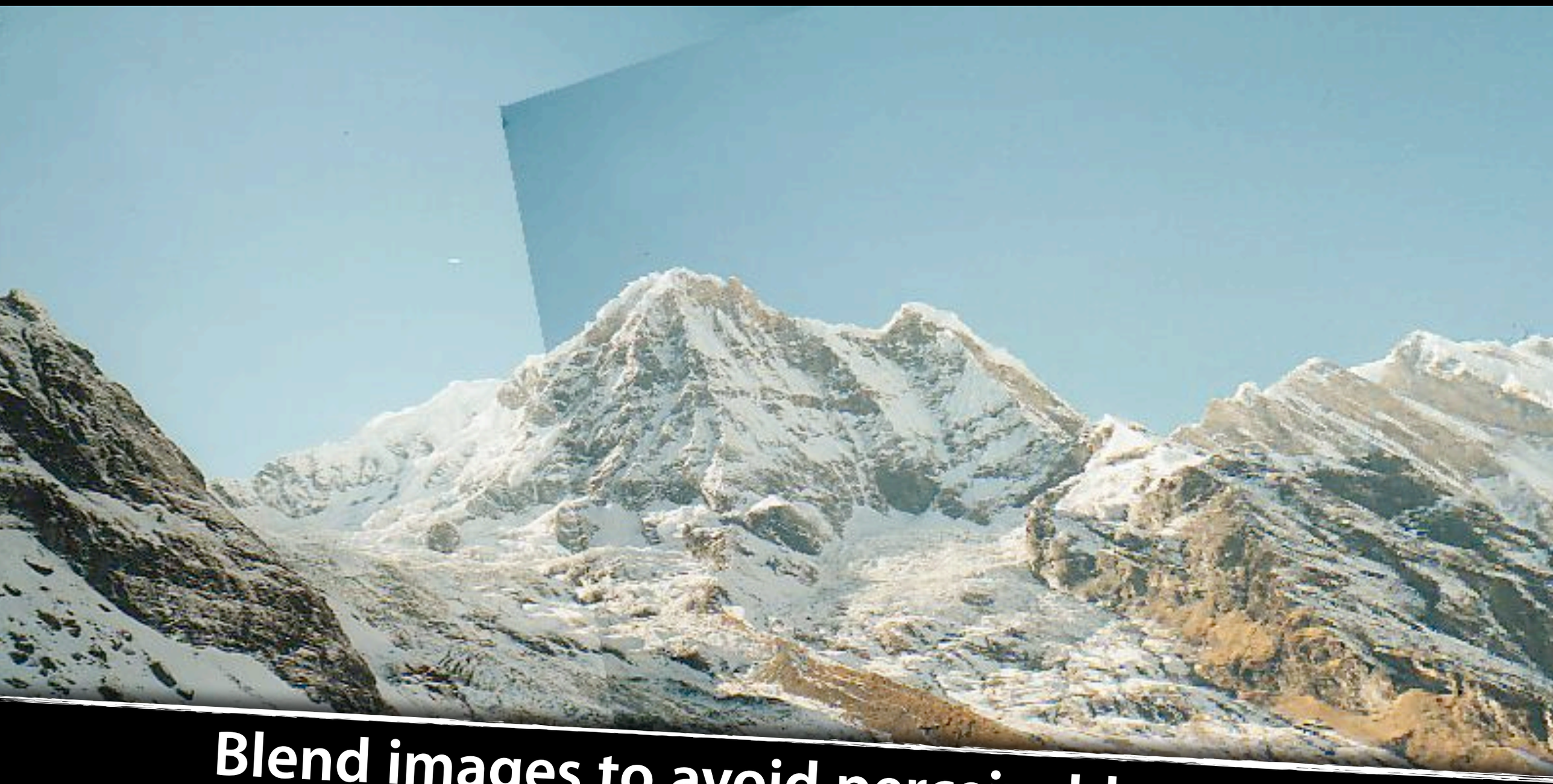
Step 5

Estimate final mapping with best hypothesis



Estimate final transformation with best hypothesis

Estimate final transformation with best hypothesis



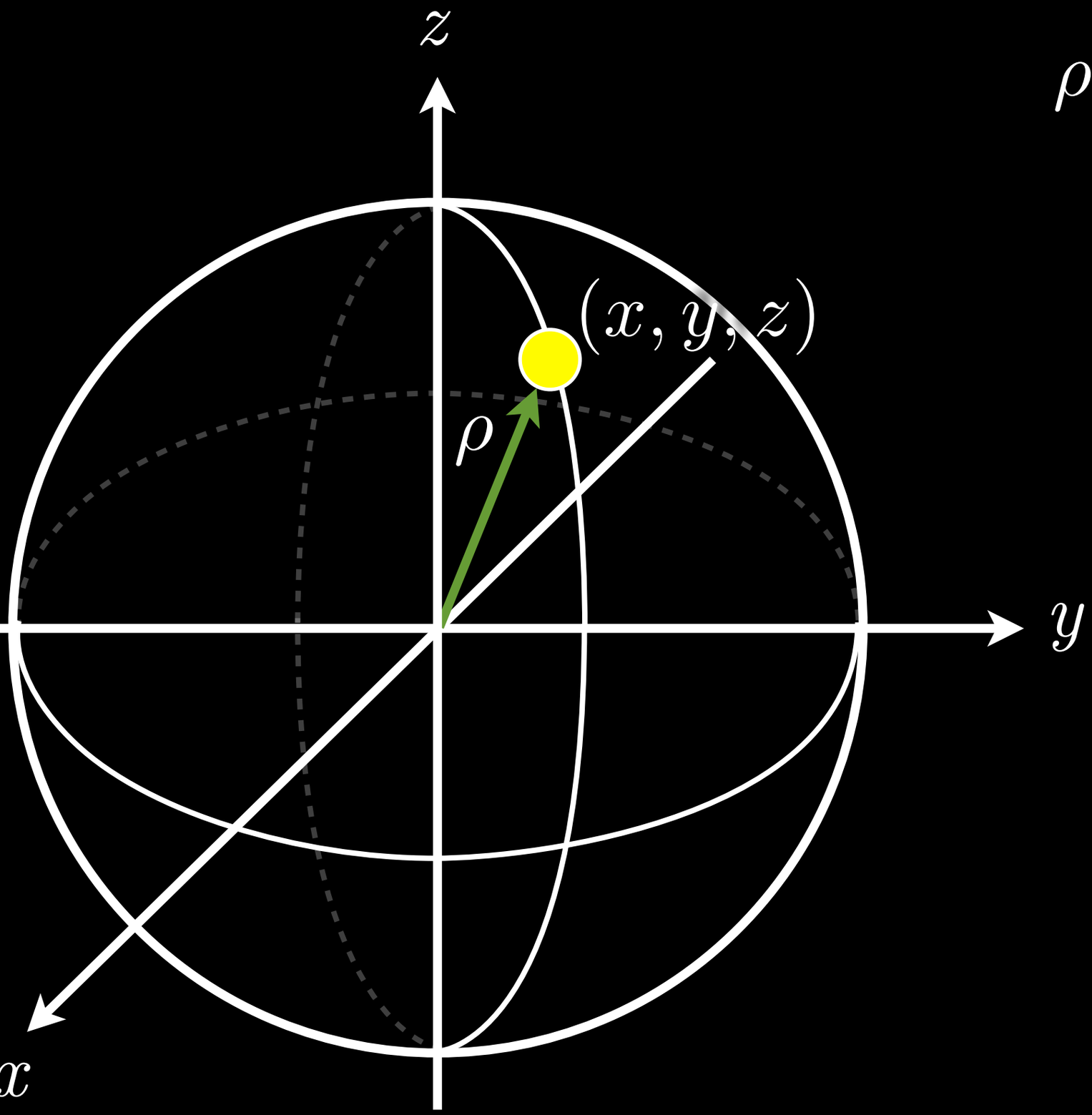
Blend images to avoid perceivable seams

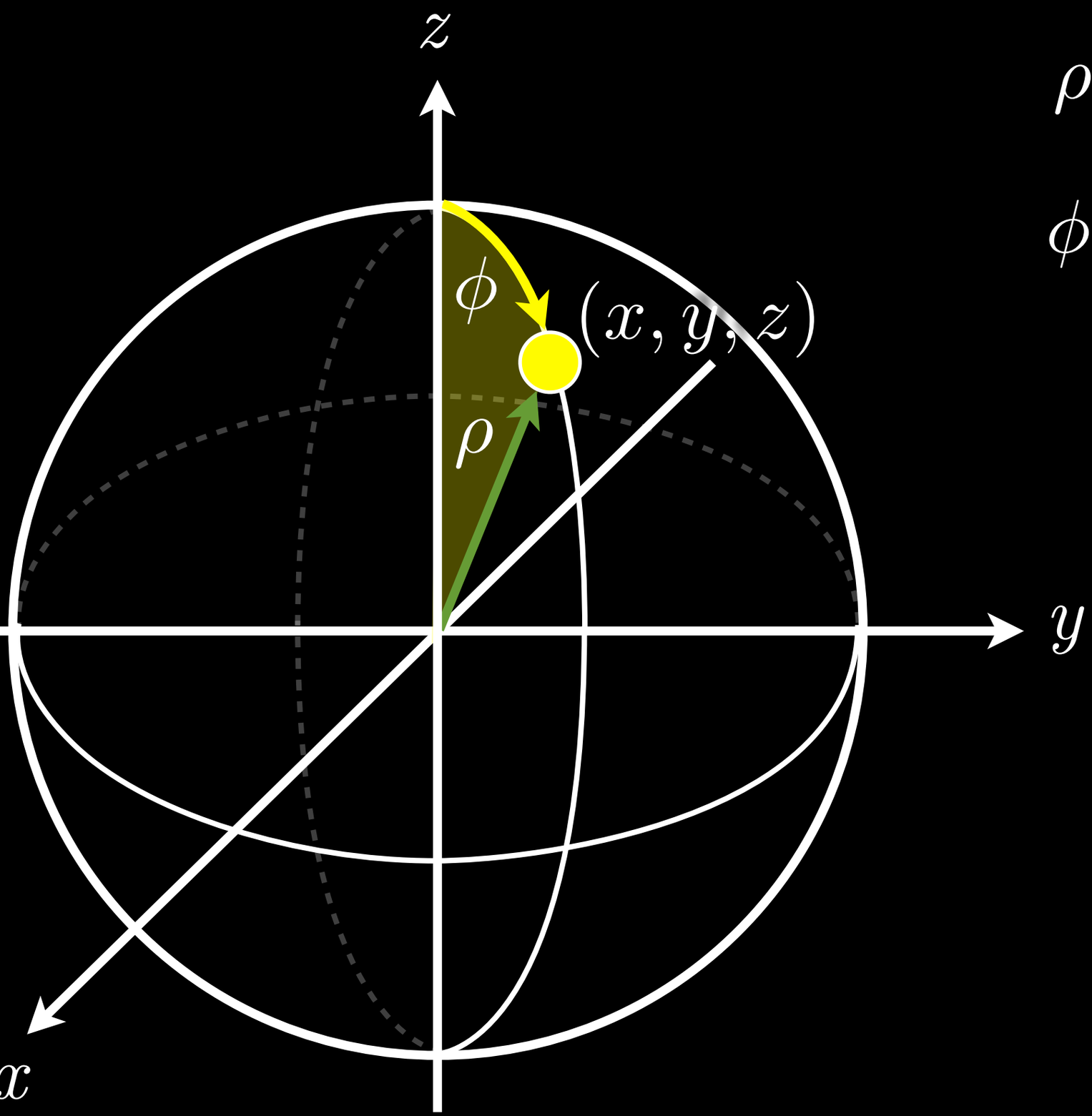

CANON ZOOM LENS EF 24-105mm 1:4 L IS USM
24-105mm
24 35 50 70 105
IMAGE STABILIZER
WB AF-DRIVE ISO
8000 5.6 93
6400

project images onto a sphere using spherical coordinates

Spherical Coordinates

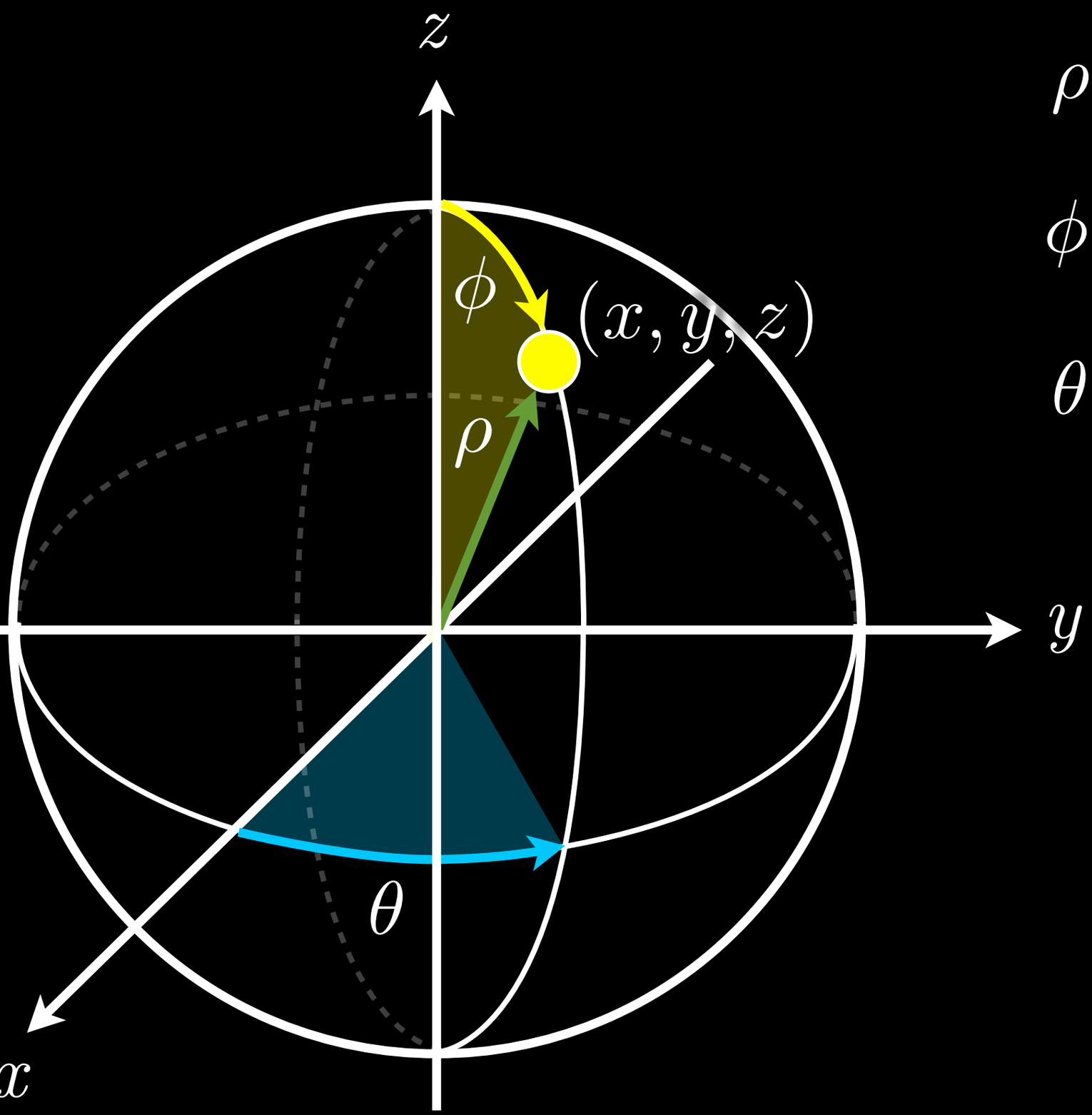
$$\rho = \sqrt{x^2 + y^2 + z^2}$$





$$\rho = \sqrt{x^2 + y^2 + z^2}$$

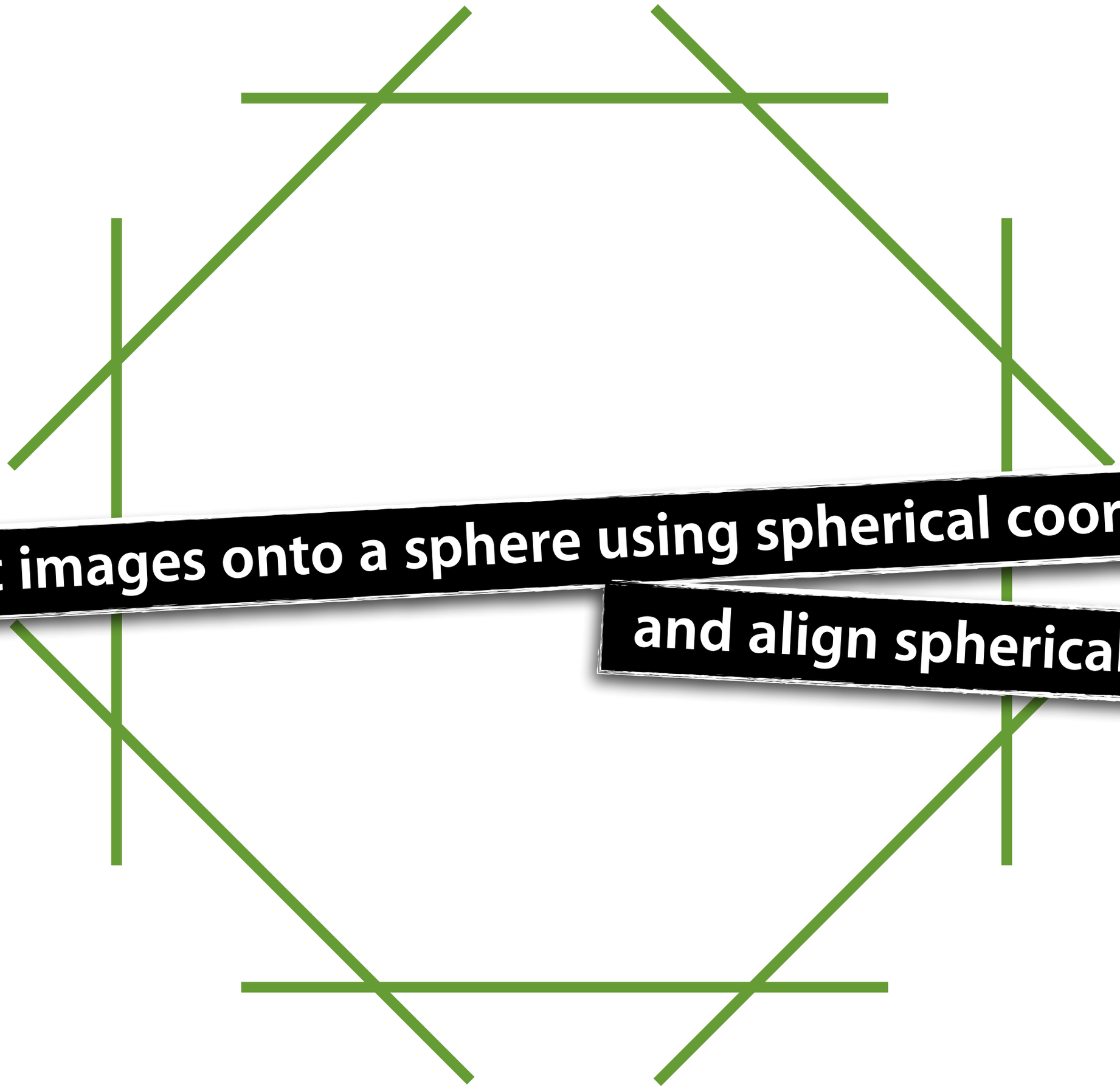
$$\phi = \cos^{-1}(z/\rho)$$



$$\rho = \sqrt{x^2 + y^2 + z^2}$$

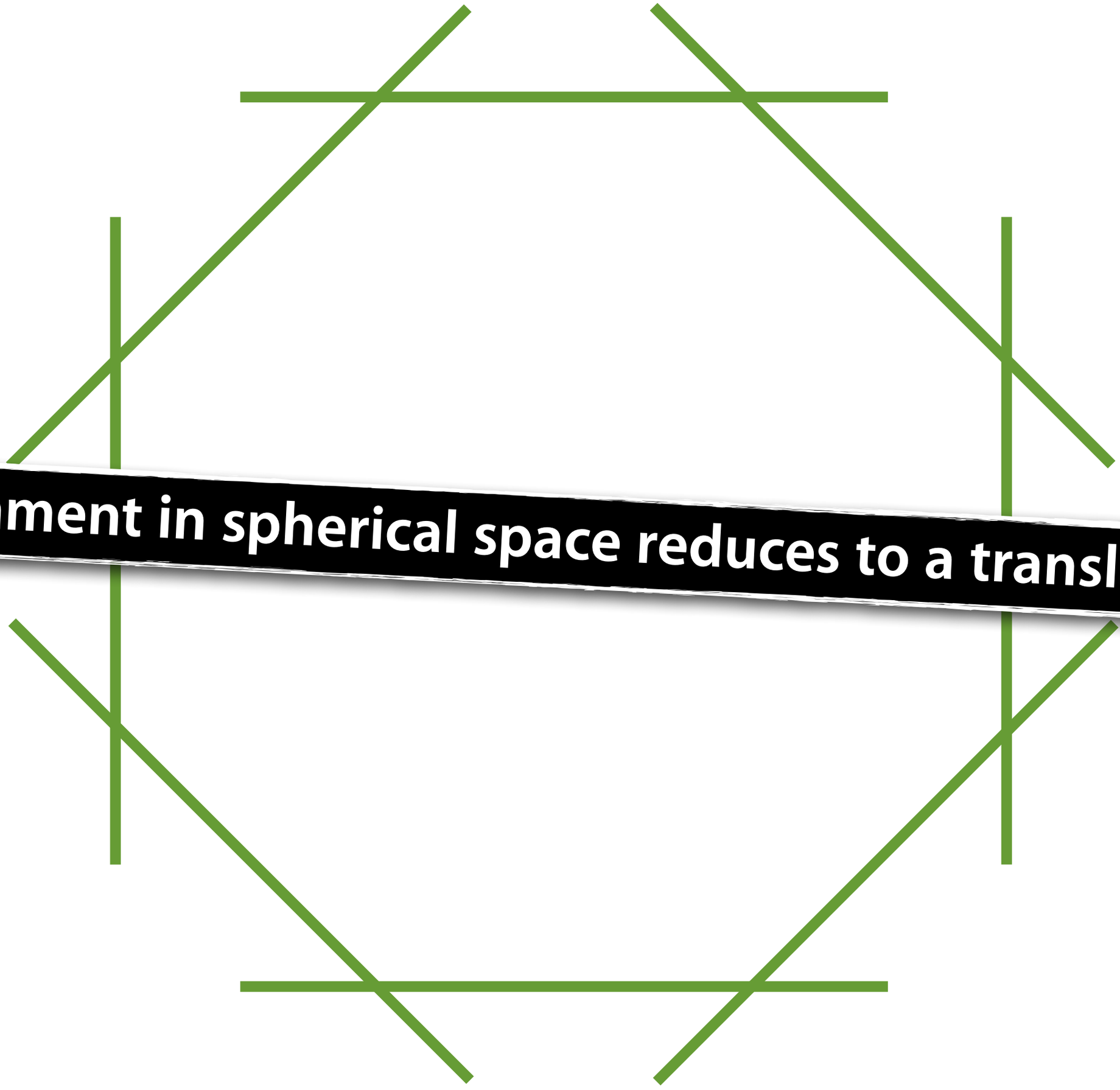
$$\phi = \cos^{-1}(z/\rho)$$

$$\theta = \tan^{-1}(y/x)$$



project images onto a sphere using spherical coordinates

and align spherical images



alignment in spherical space reduces to a translation

Cylindrical Panorama