

2D Geometric Transformations

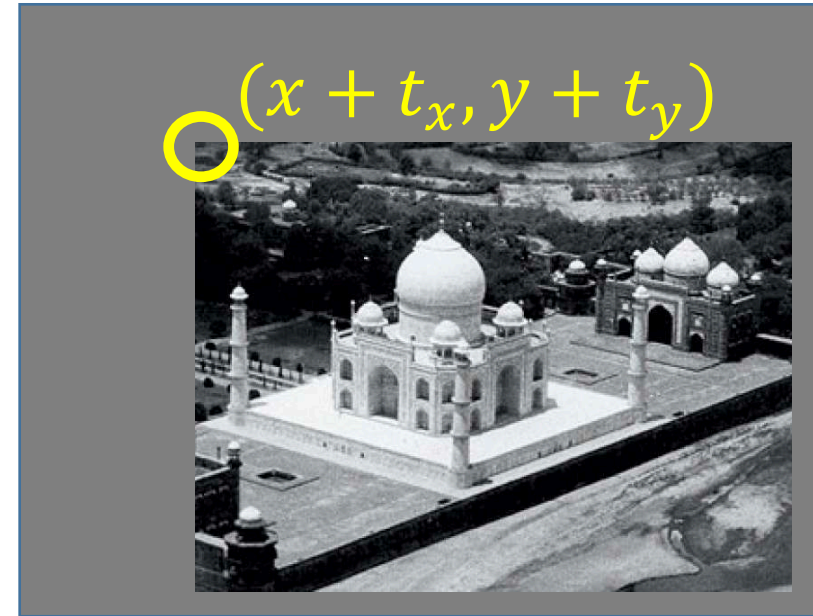
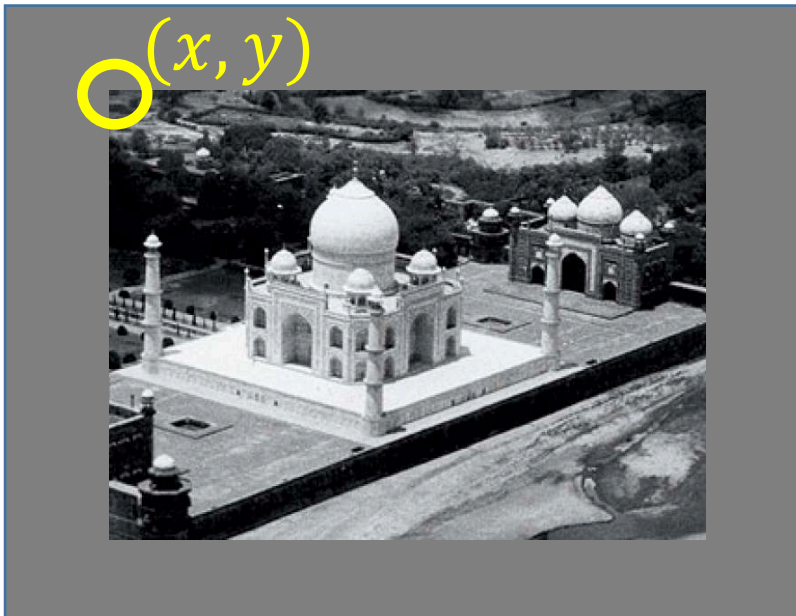
Chetan Arora

Disclaimer: The contents of these slides are taken from various publicly available resources such as research papers, talks and lectures. To be used for the purpose of classroom teaching, and academic dissemination only.



Geometric Operations

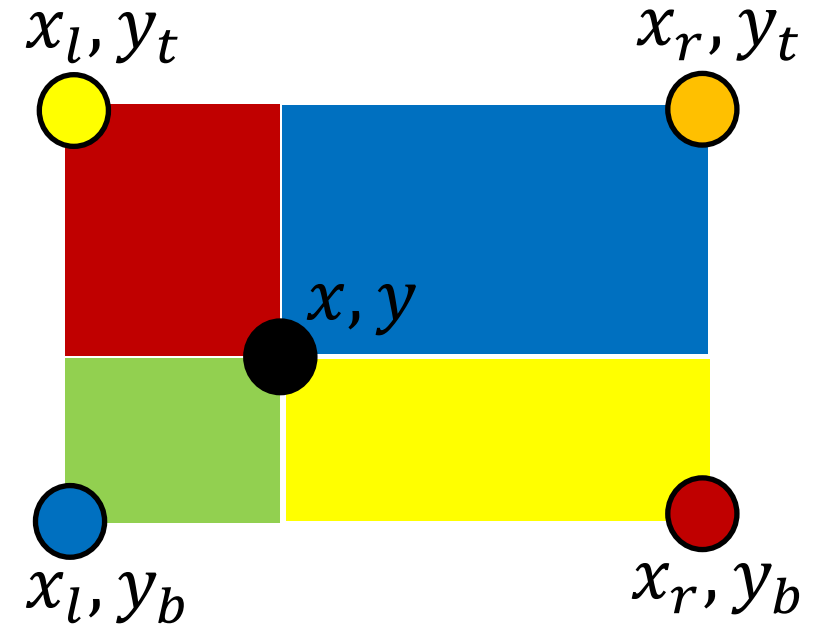
- Geometric operation transforms image I to new image I' by modifying coordinates of image pixels: $I(x, y) \rightarrow I'(x', y')$
- Intensity value originally at (x, y) is moved to a new position (x', y')





Geometric Operations: Bilinear Interpolation

- Since image coordinates can only be discrete values, some transformations may yield (x', y') that's not discrete.
- How to set intensity at non-integer coordinates?





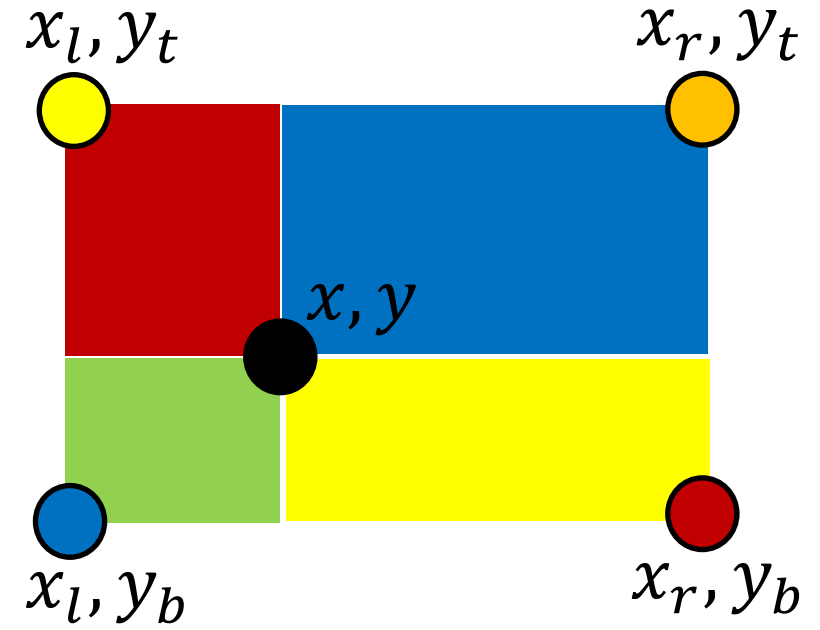
Geometric Operations: Bilinear Interpolation

- Since image coordinates can only be discrete values, some transformations may yield (x', y') that's not discrete.
- How to set intensity at non-integer coordinates?

$$I(x, y_t) = (x - x_l)I(x_r, y_t) + (x_r - x)I(x_l, y_t)$$

$$I(x, y_b) = (x - x_l)I(x_r, y_b) + (x_r - x)I(x_l, y_b)$$

$$I(x, y) = (y - y_t)I(x, y_b) + (y_b - y)I(x, y_t)$$

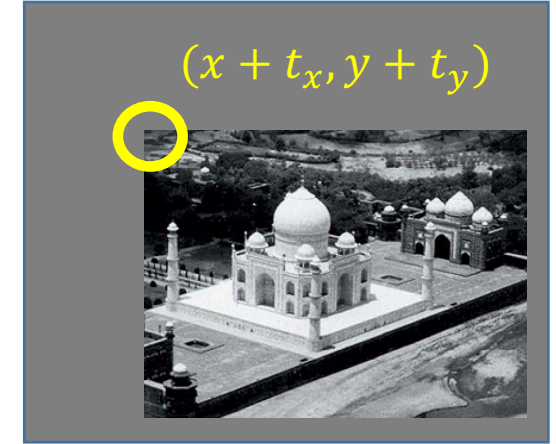
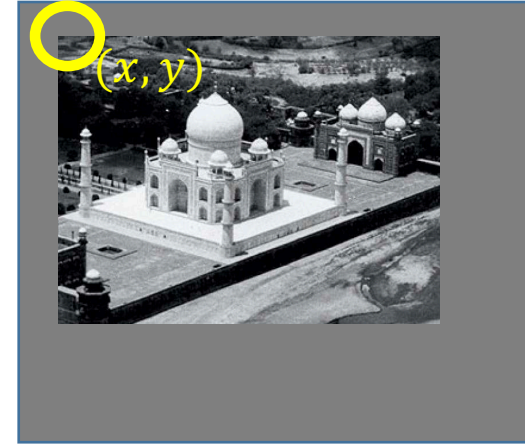




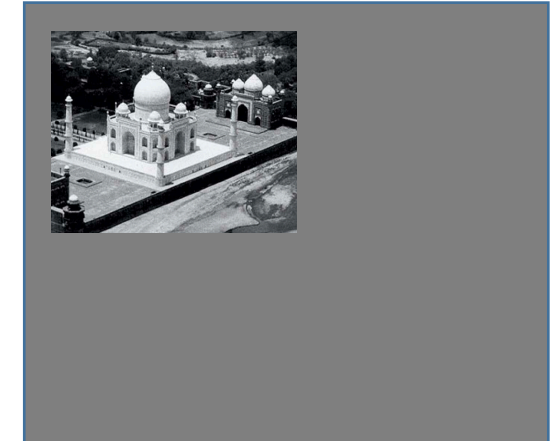
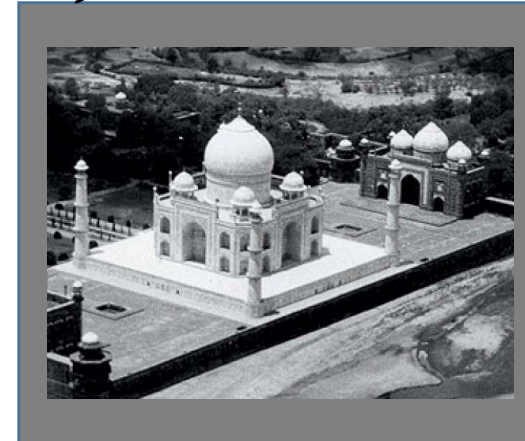
Simple Geometric Transformations

- Translation

$$\begin{aligned} x' &= x + t_x \\ y' &= y + t_y \end{aligned} \quad \text{or} \quad \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$



(0,0)

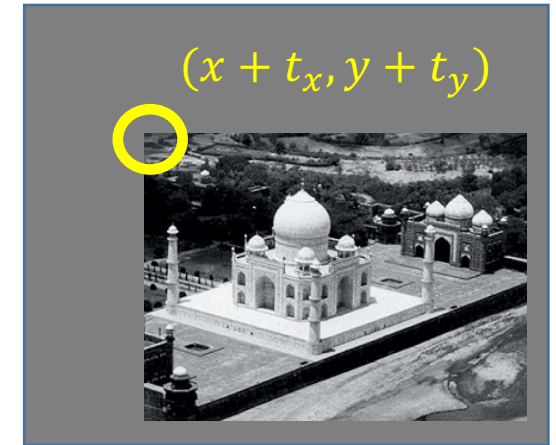
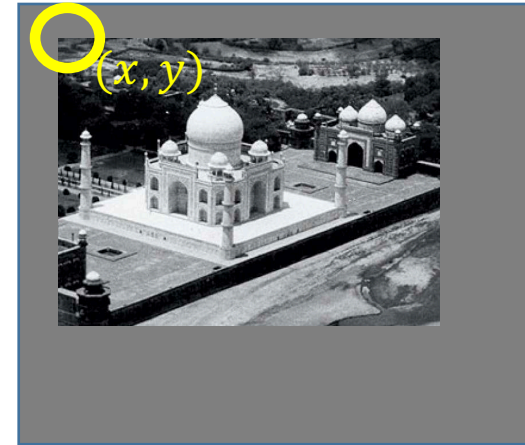




Simple Geometric Transformations

- Translation

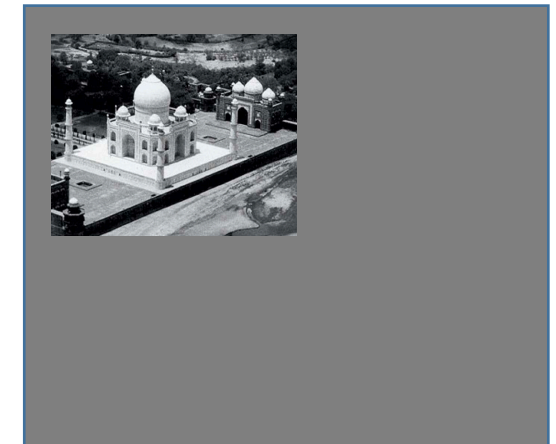
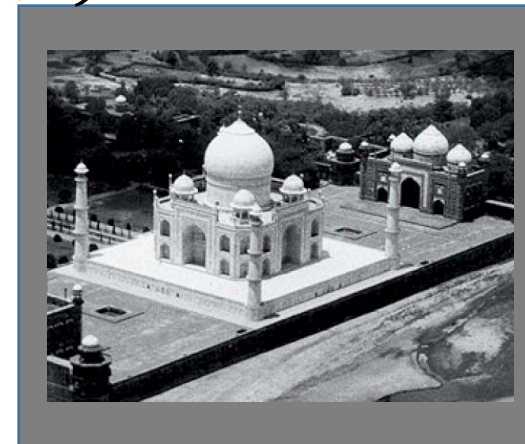
$$\begin{aligned} x' &= x + t_x \\ y' &= y + t_y \end{aligned} \quad \text{or} \quad \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$



- Scaling

$$\begin{aligned} x' &= s_x x \\ y' &= s_y y \end{aligned} \quad \text{or} \quad \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

(0,0)

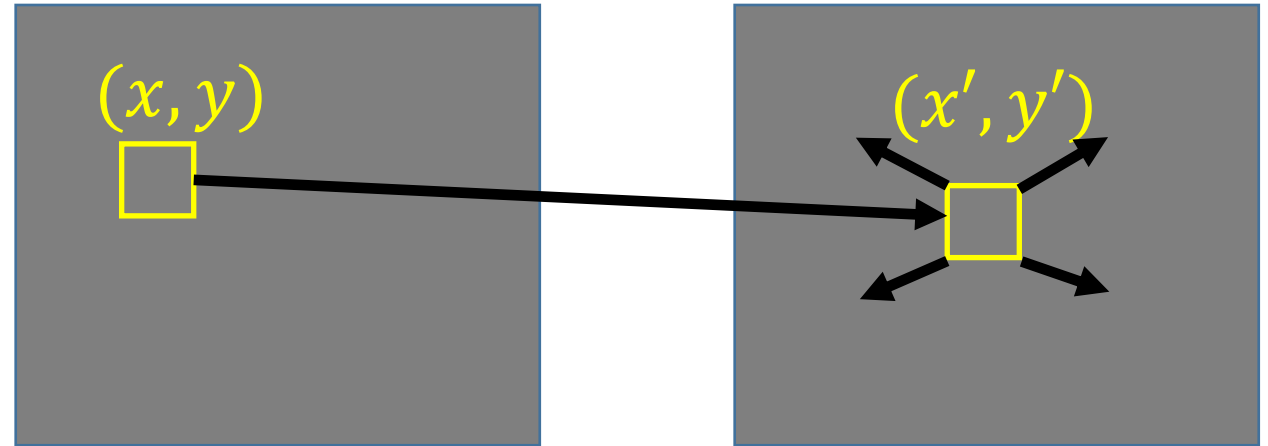




Forward vs Backward Warping

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

What is the problem with forward warping?



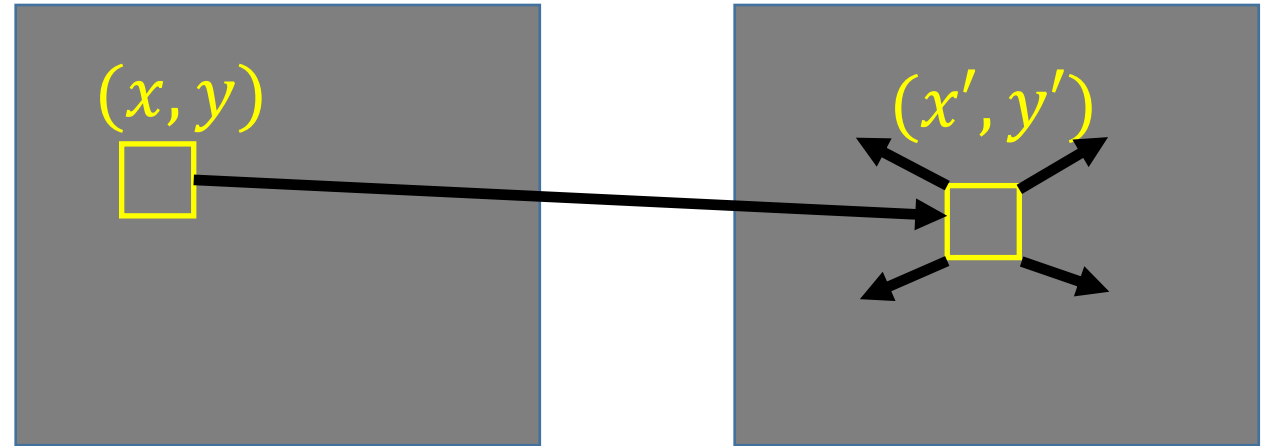


Forward vs Backward Warping

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

What is the problem with forward warping?

Can leave holes in the destination image



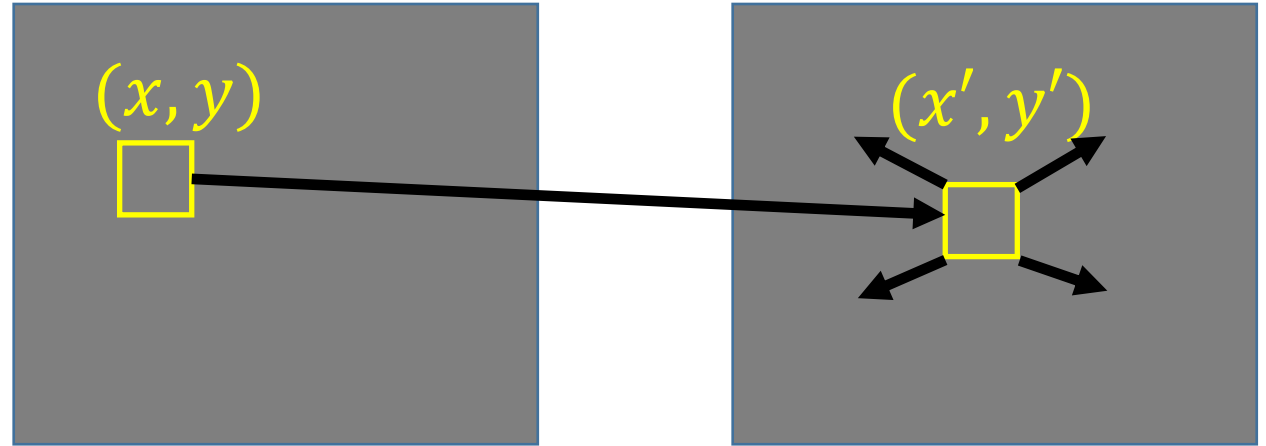


Forward vs Backward Warping

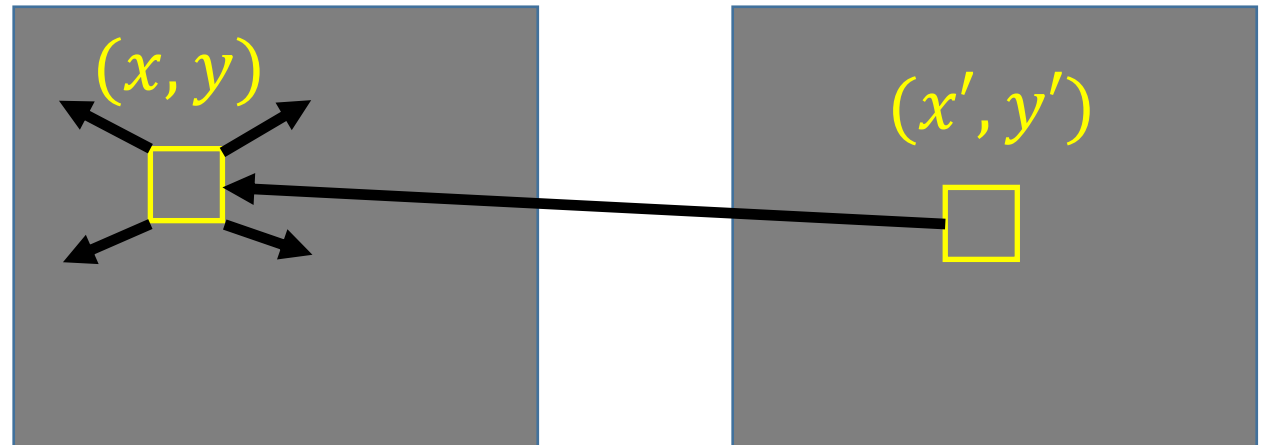
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

What is the problem with forward warping?

Can leave holes in the destination image



$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1/s_x & 0 \\ 0 & 1/s_y \end{bmatrix} \begin{bmatrix} x' \\ y' \end{bmatrix}$$



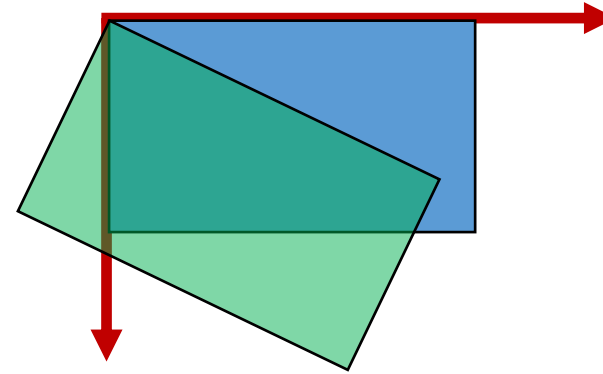


Geometric Transformations

- **Rotation**

$$x' = x \cos(\alpha) - y \sin(\alpha)$$

$$y' = x \sin(\alpha) + y \cos(\alpha)$$





Geometric Transformations

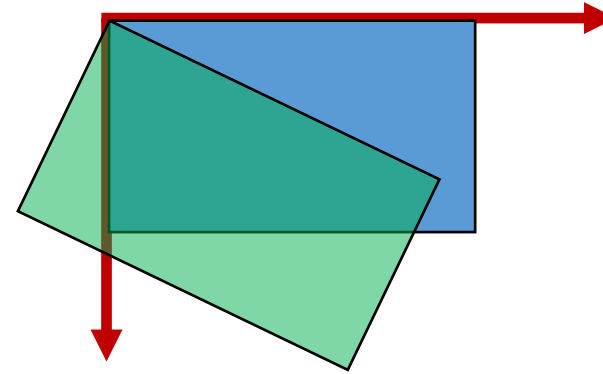
- **Rotation**

$$x' = x \cos(\alpha) - y \sin(\alpha)$$

$$y' = x \sin(\alpha) + y \cos(\alpha)$$

or

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$





Geometric Transformations

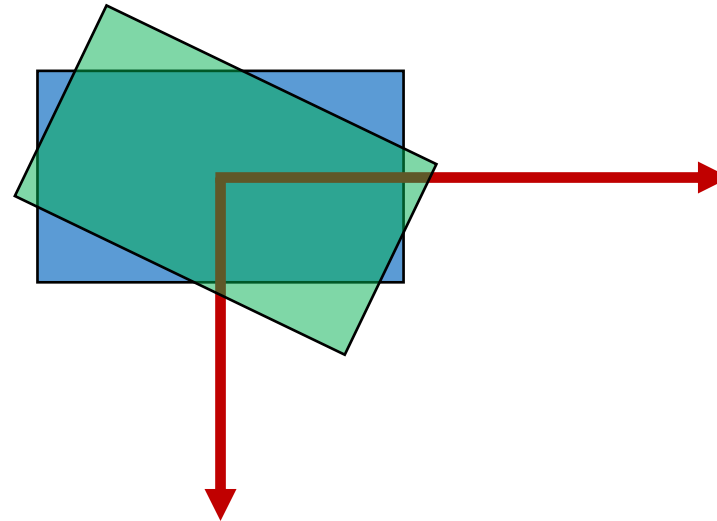
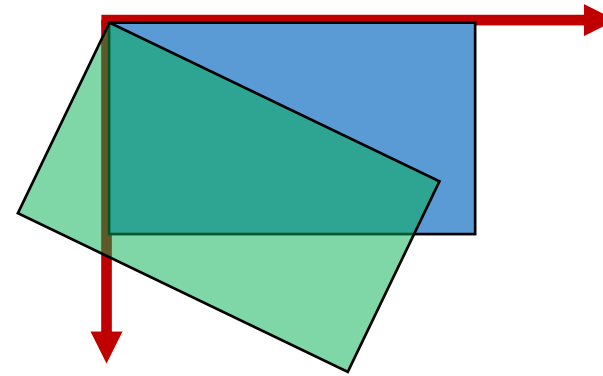
- **Rotation**

$$x' = x \cos(\alpha) - y \sin(\alpha)$$

$$y' = x \sin(\alpha) + y \cos(\alpha)$$

or

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$



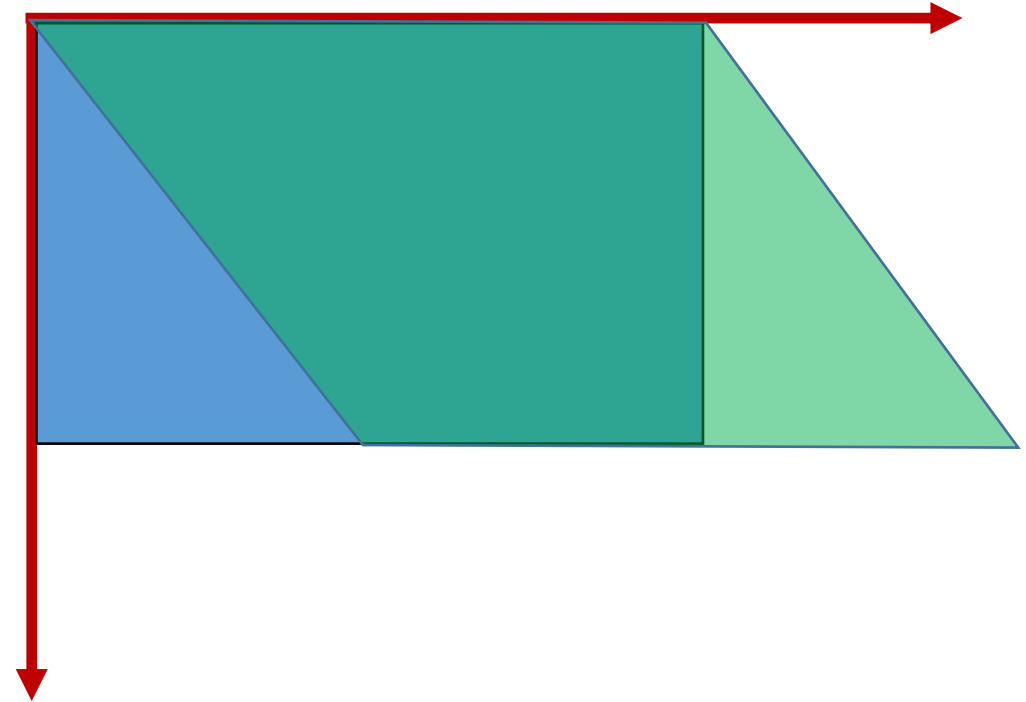


Geometric Transformations

- Affine

$$x' = a_{11}x + a_{12}y + a_{13}$$

$$y' = a_{21}x + a_{22}y + a_{23}$$





Geometric Transformations

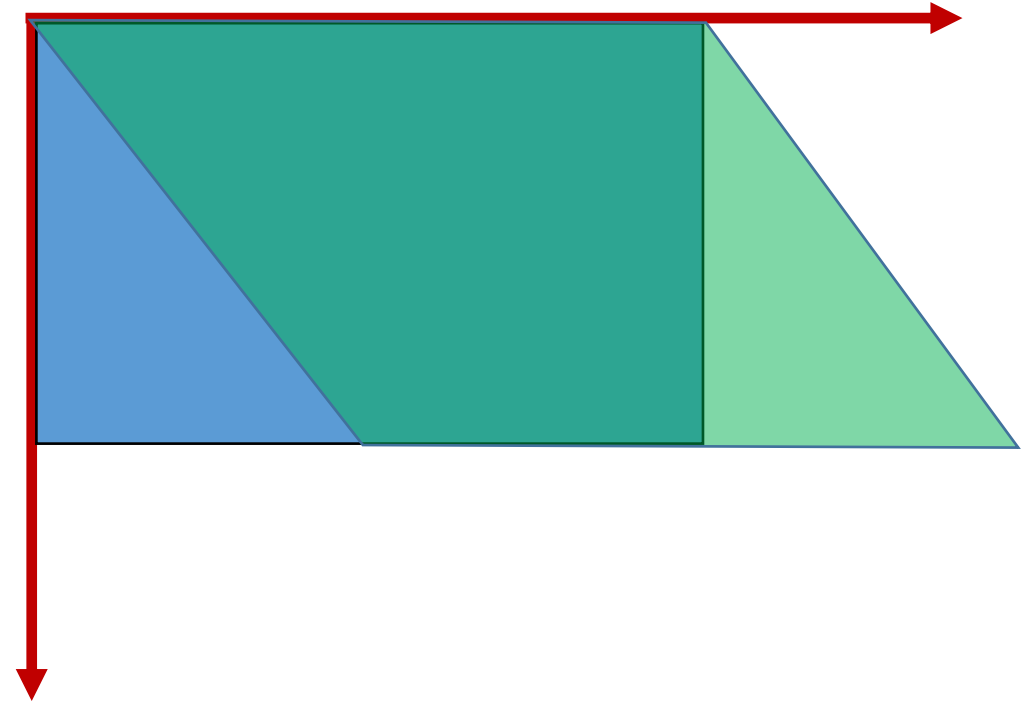
- **Affine**

$$x' = a_{11}x + a_{12}y + a_{13}$$

$$y' = a_{21}x + a_{22}y + a_{23}$$

or

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} a_{13} \\ a_{23} \end{bmatrix}$$





Homogenous Coordinates

- Notation useful for converting scaling, translation, rotating into point-matrix multiplication
- To convert ordinary coordinates into homogeneous coordinates:

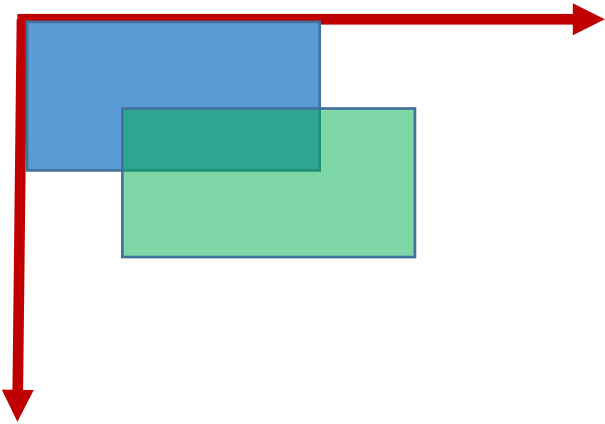
$$\begin{bmatrix} x \\ y \end{bmatrix} \rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \sim \begin{bmatrix} h & x \\ h & y \\ h \end{bmatrix}$$

- In homogenous coordinates, affine transformation becomes:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} a_{13} \\ a_{23} \end{bmatrix} \quad \rightarrow \quad \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

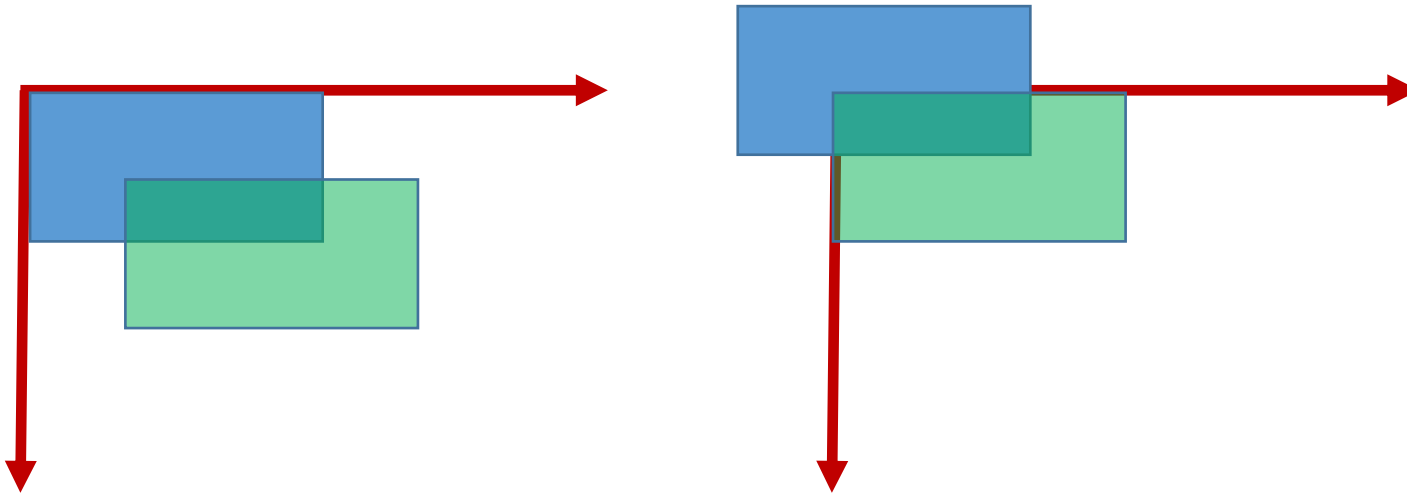


Transformation as Change of Coordinate System



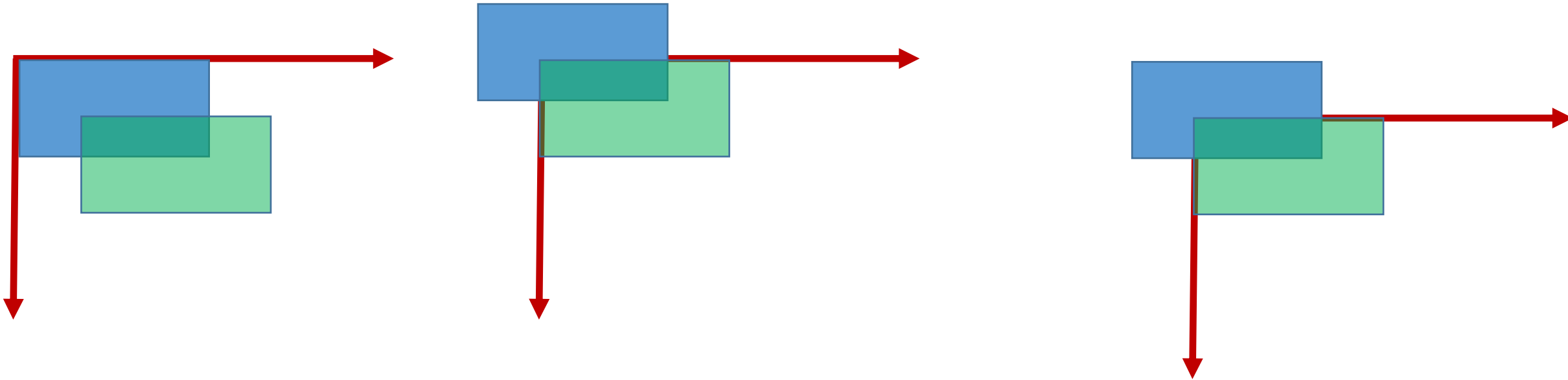


Transformation as Change of Coordinate System



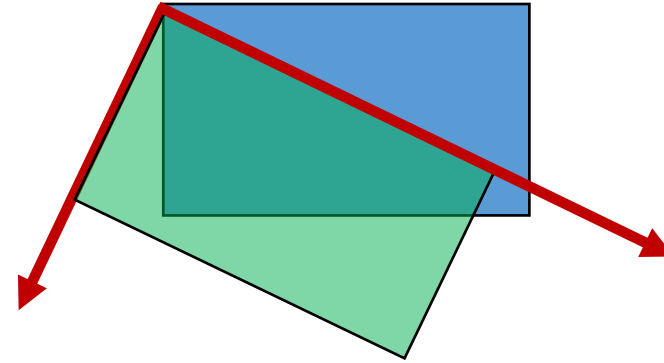
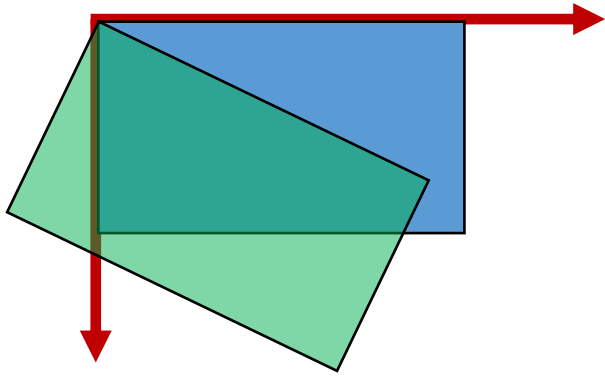


Transformation as Change of Coordinate System





Transformation as Change of Coordinate System





Cascading Transformations

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$



Cascading Transformations

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$


$$\begin{bmatrix} x'' \\ y'' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$$



Cascading Transformations

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x'' \\ y'' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$$



$$\begin{bmatrix} x'' \\ y'' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$



Cascading Transformations

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x'' \\ y'' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$$

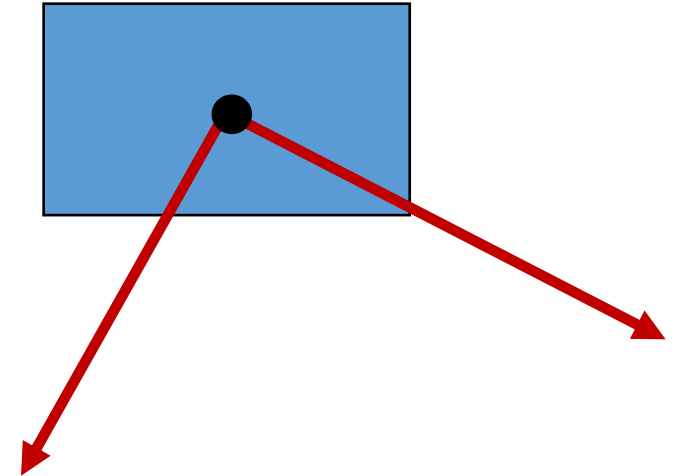
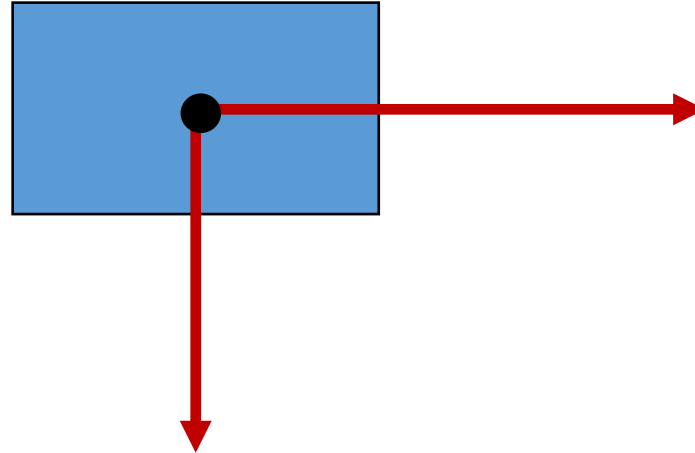
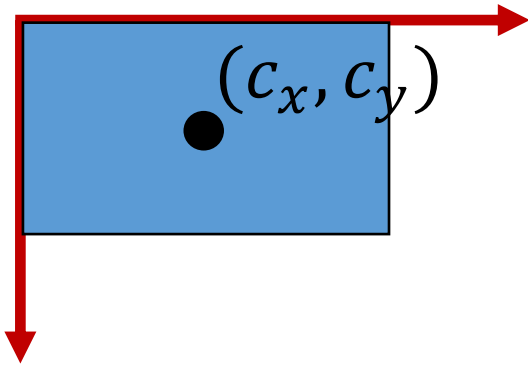

$$\begin{bmatrix} x'' \\ y'' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

=

$$\begin{bmatrix} x'' \\ y'' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) & t_x \cos(\alpha) - t_y \sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) & t_x \sin(\alpha) + t_y \cos(\alpha) \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

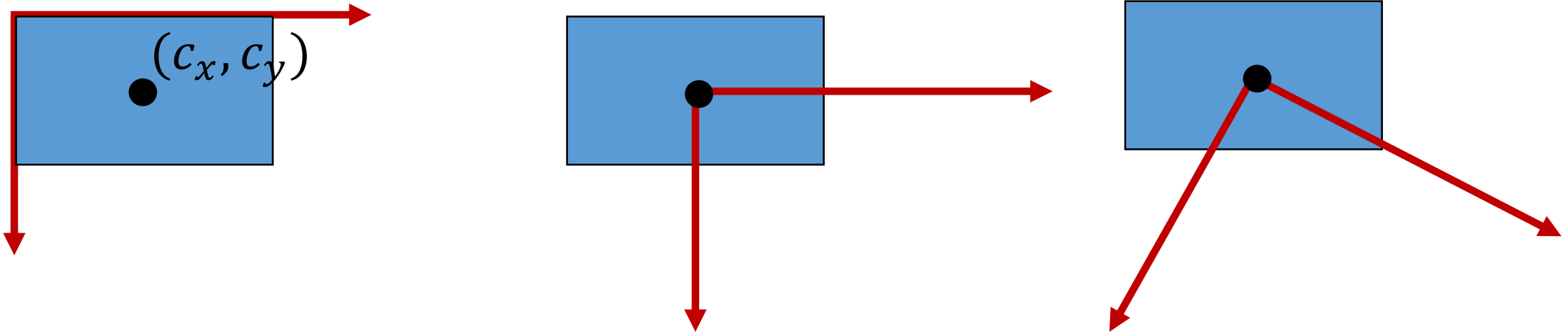


Cascading Transformations





Cascading Transformations



$$\begin{bmatrix} x'' \\ y'' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -c_x \\ 0 & 1 & -c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$



Partitioned Matrix (or Block Matrix)

- A matrix constructed from other smaller matrices.
- The smaller matrices are called blocks or sub-matrices.

$$X = \left[\begin{array}{cc|ccc} 1 & 2 & 5 & 5 & 6 \\ 3 & 4 & 8 & 7 & 6 \\ \hline 2 & 3 & 1 & 0 & 0 \\ 2 & 3 & 0 & 1 & 0 \\ 2 & 3 & 0 & 0 & 1 \end{array} \right]$$

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \quad C = \begin{bmatrix} 5 & 5 & 6 \\ 8 & 7 & 6 \end{bmatrix}$$

$$X = \begin{bmatrix} A & C \\ B & D \end{bmatrix}$$

$$B = \begin{bmatrix} 2 & 3 \\ 2 & 3 \\ 2 & 3 \end{bmatrix} \quad D = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$X = \left[\begin{array}{c|c} A & C \\ \hline B & D \end{array} \right]$$



Partitioned Matrix (or Block Matrix)

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) & t_x \\ \sin(\alpha) & \cos(\alpha) & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) & t_x \\ \sin(\alpha) & \cos(\alpha) & t_y \\ \hline 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$${}^{p'} \begin{bmatrix} x' \\ y' \\ \hline 1 \end{bmatrix} = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) & t_x \\ \sin(\alpha) & \cos(\alpha) & t_y \\ \hline 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ \hline 1 \end{bmatrix} {}^p$$

Matrix Form

$$\begin{matrix} 2 \times 1 \\ \begin{bmatrix} p' \\ 1 \end{bmatrix} \\ 1 \times 1 \end{matrix} = \begin{matrix} 2 \times 2 & 2 \times 1 & 2 \times 1 \\ \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \\ 1 \times 2 & 1 \times 1 & 1 \times 1 \end{matrix} \begin{matrix} 2 \times 1 \\ \begin{bmatrix} p \\ 1 \end{bmatrix} \\ 1 \times 1 \end{matrix}$$

Equation Form

$$p' = Rp + t$$

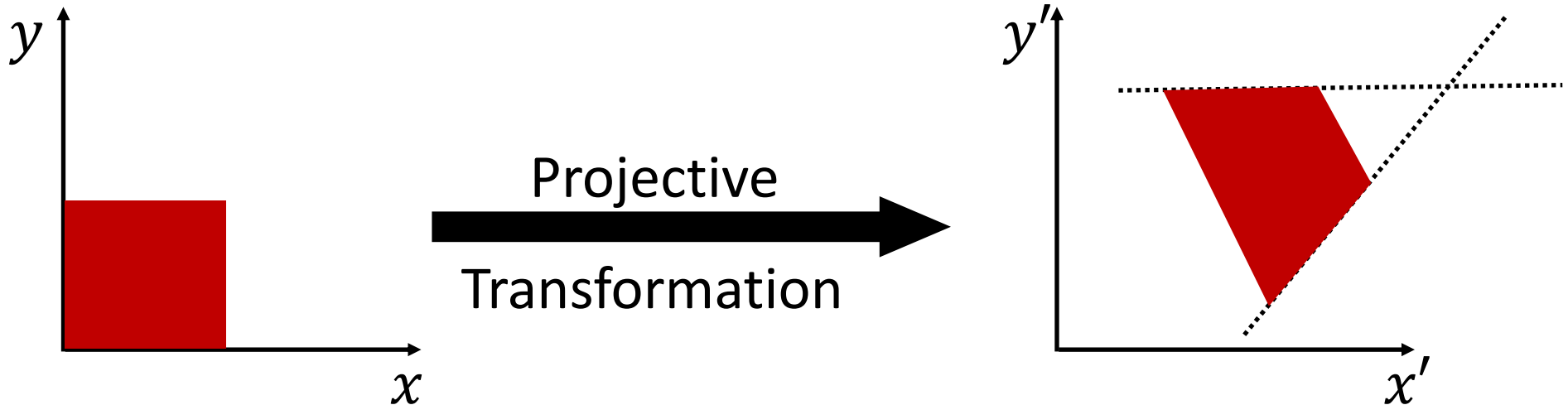


Projective Transformation

$$x' = \frac{a_{11}x + a_{12}y + a_{13}}{a_{31}x + a_{32}y + a_{33}}$$

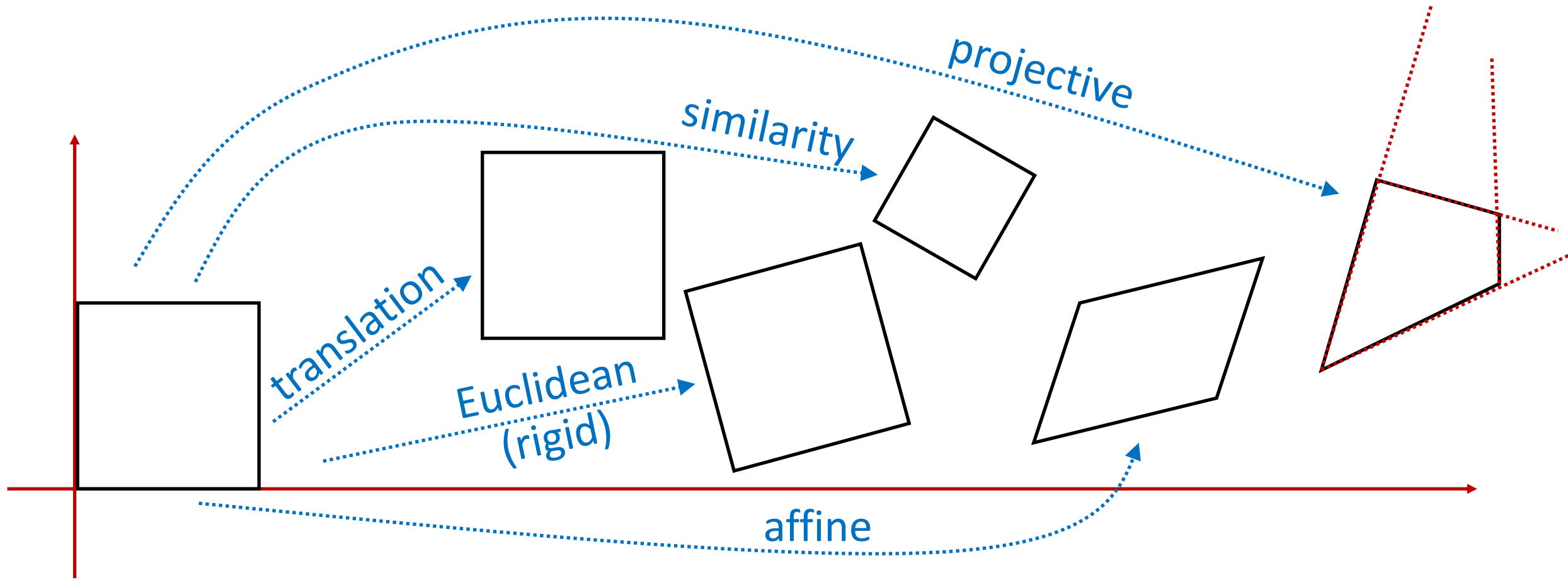
$$y' = \frac{a_{21}x + a_{22}y + a_{23}}{a_{31}x + a_{32}y + a_{33}}$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \sim \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \iff \begin{bmatrix} p' \\ 1 \end{bmatrix} \sim \begin{bmatrix} A & b \\ c^T & 1 \end{bmatrix} \begin{bmatrix} p \\ 1 \end{bmatrix} \iff p' \sim \frac{Ap + b}{c^T p + 1}$$




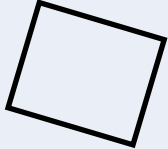
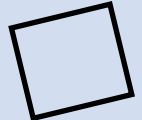

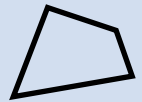


Summary of 2D Geometric Transformations





Summary of 2D Geometric Transformations

Name	Matrix	#D.O.F.	Preserves	Indicative Shape
Translation	$[I \mid t]_{2 \times 3}$	2	Orientation	
Rigid (Euclidean)	$[R \mid t]_{2 \times 3}$	3	Lengths	
Similarity	$[sR \mid t]_{2 \times 3}$	4	Angles	
Affine	$[A]_{2 \times 3}$	6	Parallelism	
Projective	$[H]_{3 \times 3}$	8	Straight Lines	

Parameter Estimation



Parameter Estimation: Fitting Geometric Models

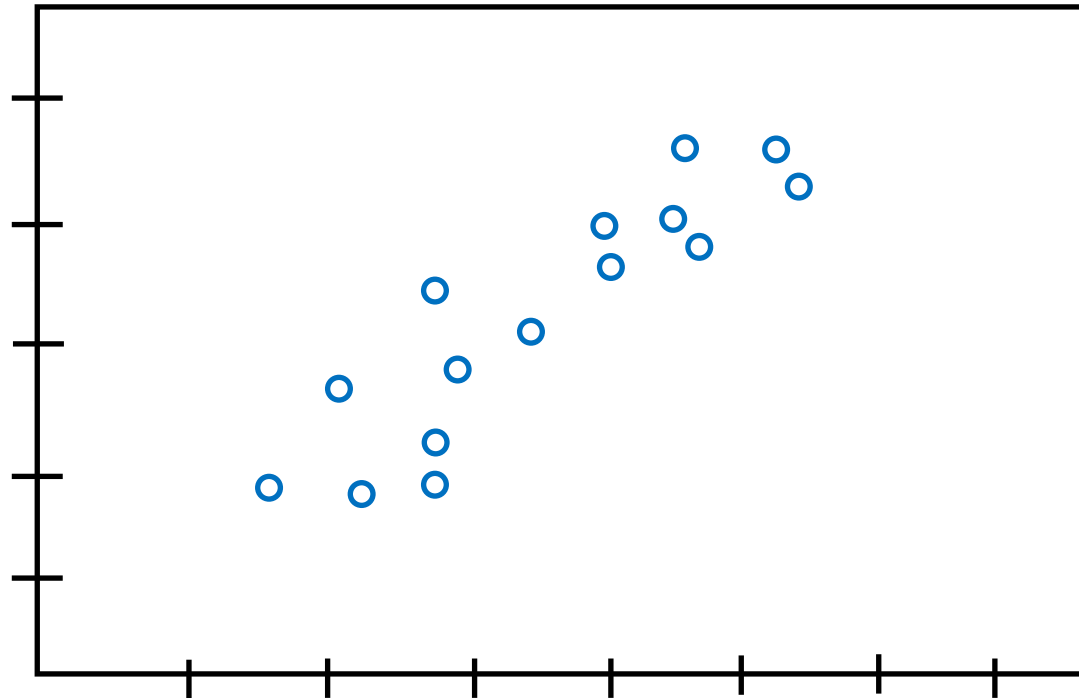
- Want to fit a model to raw image features (data). The features could be points, edges, even regions.
- Parameterize model such that a model instance is an element of \mathbb{R}^n i.e.
model instance = (a_1, a_2, \dots, a_n)
- Define an error function $E(\text{model}_i, \text{data})$ that measures how well a given model instance describes the data
- Solve for the model instance that minimizes E



Line Fitting: Point Feature Data

- Point features = $\{(x_i, y_i) | i = 1, \dots, n\}$

```
pts = [...  
  17 81;  
  23 72;  
  35 58;  
  45 50;  
  57 56;  
  61 36;  
  ⏟  ⏟  
  x  y
```





Parameter Estimation: Fitting Geometric Models

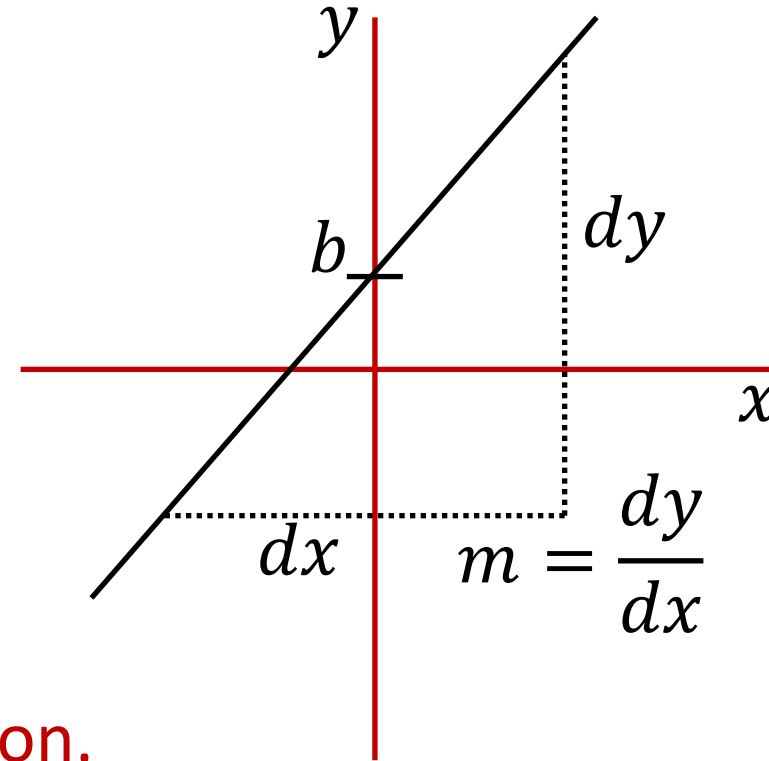
- Want to fit a model to raw image features (data). The features could be points, edges, even regions.
- Parameterize model such that a model instance is an element of \mathbb{R}^n i.e.
model instance = (a_1, a_2, \dots, a_n)
- Define an error function $E(\text{model}_i, \text{data})$ that measures how well a given model instance describes the data
- Solve for the model instance that minimizes E



Line Fitting: Parameterization

Model Instance = (m, b)

Line: $y = m * x + b$



The parameterization is only for illustration.
Ignore the vertical line representation for now



Parameter Estimation: Fitting Geometric Models

- Want to fit a model to raw image features (data). The features could be points, edges, even regions.
- Parameterize model such that a model instance is an element of \mathbb{R}^n i.e.
model instance = (a_1, a_2, \dots, a_n)
- Define an error function $E(\text{model}_i, \text{data})$ that measures how well a given model instance describes the data
- Solve for the model instance that minimizes E



Least Squares

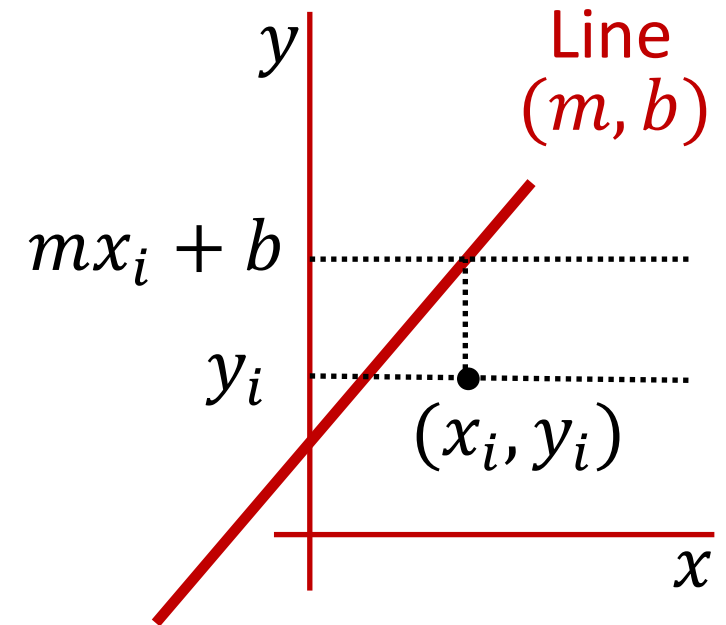
Note: Least squares is only one kind of error function. There are other robust error functions such as LMedS (least median of squares), which are slightly more difficult to optimize.

- Given line (m, b) , find the vertical distance of point (x_i, y_i) to the line:

$$d_i = ((mx_i + b) - y_i)$$

- E is the sum of squared distances over all points:

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





Parameter Estimation: Fitting Geometric Models

- Want to fit a model to raw image features (data). The features could be points, edges, even regions.
- Parameterize model such that a model instance is an element of \mathbb{R}^n i.e.
model instance = (a_1, a_2, \dots, a_n)
- Define an error function $E(\text{model}_i, \text{data})$ that measures how well a given model instance describes the data
- Solve for the model instance that minimizes E



Calculus to Find Extrema

- Take first derivative of E with respect to m , and b and set them to zero.

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

$$\frac{\partial E}{\partial m} = 2 \sum_{i=1}^n x_i ((mx_i + b) - y_i) = 0 \quad \Rightarrow \quad \sum_{i=1}^n mx_i^2 + x_i b = \sum_{i=1}^n x_i y_i$$

$$\frac{\partial E}{\partial b} = 2 \sum_{i=1}^n ((mx_i + b) - y_i) = 0 \quad \Rightarrow \quad \sum_{i=1}^n mx_i + b = \sum_{i=1}^n y_i$$



Calculus to Find Extrema

$$\sum_{i=1}^n mx_i^2 + x_i b = \sum_{i=1}^n x_i y_i$$

$$\sum_{i=1}^n mx_i + b = \sum_{i=1}^n y_i$$

Writing the equations in a matrix form

$$\begin{bmatrix} \sum x_i^2 & \sum x_i \\ \sum x_i & \sum 1 \end{bmatrix} \begin{bmatrix} m \\ b \end{bmatrix} = \begin{bmatrix} \sum x_i y_i \\ \sum y_i \end{bmatrix}$$

Estimating a Transformation



Parameter Estimation for Transformations

Problem Statement

- Given point matches between two images, which are related by some parametric transformation (e.g. translation; Euclidean; similarity, or affine), estimate the parameters of that transformation.

General Strategy

- Least-Squares estimation from point correspondences

Two important (related) questions:

- How many degrees of freedom?
- How many point correspondences are needed?



Example: Estimating Translation Parameters

$$\begin{aligned}x' &= x + t_x \\ y' &= y + t_y\end{aligned}$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

How many degrees of freedom?

- How many independent variables are there?



Example: Estimating Translation Parameters

$$\begin{aligned}x' &= x + t_x \\ y' &= y + t_y\end{aligned}$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

How many degrees of freedom?

- How many independent variables are there? **Two**

How many point correspondences are needed?



Example: Estimating Translation Parameters

$$\begin{aligned}x' &= x + t_x \\ y' &= y + t_y\end{aligned}$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

How many degrees of freedom?

- How many independent variables are there? **Two**

How many point correspondences are needed?

- Each correspondence $(x, y) \rightarrow (x', y')$ provides two equations

$$\frac{\text{DoF}}{\text{\#Eqns per Correspondence}} = \frac{2}{2} = 1$$



Example: Estimating Translation Parameters

Least Squares Estimation

Minimize $E = \sum_{i=1}^n \left((x_i + t_x - x_i')^2 + (y_i + t_y - y_i')^2 \right) \quad \text{w.r.t. } t_x, t_y$



Example: Estimating Translation Parameters

Least Squares Estimation

Minimize $E = \sum_{i=1}^n \left((x_i + t_x - x_i')^2 + (y_i + t_y - y_i')^2 \right) \quad \text{w.r.t. } t_x, t_y$

$$\frac{\partial E}{\partial t_x} = 2 \sum_{i=1}^n x_i + t_x - x_i' = 0 \quad \longrightarrow \quad t_x = \sum_{i=1}^n \frac{x_i' - x_i}{n}$$



Example: Estimating Translation Parameters

Least Squares Estimation

Minimize $E = \sum_{i=1}^n \left((x_i + t_x - x_i')^2 + (y_i + t_y - y_i')^2 \right) \quad \text{w.r.t. } t_x, t_y$

$$\frac{\partial E}{\partial t_x} = 2 \sum_{i=1}^n x_i + t_x - x_i' = 0 \quad \longrightarrow \quad t_x = \sum_{i=1}^n \frac{x_i' - x_i}{n}$$

$$\frac{\partial E}{\partial t_y} = 2 \sum_{i=1}^n y_i + t_y - y_i' = 0 \quad \longrightarrow \quad t_y = \sum_{i=1}^n \frac{y_i' - y_i}{n}$$



Example: Estimating Similarity Parameters

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & -b & c \\ b & a & d \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad a = s * \cos(\theta) \quad b = s * \sin(\theta)$$

Minimize w.r.t. a, b, c, d

$$E = \sum_{i=1}^n ((ax_i - by_i + c - x_i')^2 + (bx_i + ay_i + d - y_i')^2)$$

Take partial derivative with respect to a, b, c, d and set to 0



Example: Estimating Similarity Parameters

Minimize $E = \sum_{i=1}^n ((ax_i - by_i + c - x_i')^2 + (bx_i + ay_i + d - y_i')^2)$

$$\frac{\partial E}{\partial a} = 2 \sum_{i=1}^n x_i (ax_i - by_i + c - x_i') + y_i (bx_i + ay_i + d - y_i') = 0$$

$$\sum_{i=1}^n (ax_i^2 - bx_iy_i + cx_i - x_ix_i') + (bx_iy_i + ay_i^2 + dy_i - y_iy_i') = 0$$

$$\sum_{i=1}^n a(x_i^2 + y_i^2) + b(0) + c(x_i) + d(y_i) = x_ix_i' + y_iy_i'$$



Example: Estimating Similarity Parameters

Minimize $E = \sum_{i=1}^n ((ax_i - by_i + c - x_i')^2 + (bx_i + ay_i + d - y_i')^2)$

$$\frac{\partial E}{\partial b} = 2 \sum_{i=1}^n -y_i(ax_i - by_i + c - x_i') + x_i(bx_i + ay_i + d - y_i') = 0$$

$$\sum_{i=1}^n (-ax_iy_i + by_i^2 - cy_i + x_i'y_i) + (bx_i^2 + ax_iy_i + dx_i - x_iy_i') = 0$$

$$\sum_{i=1}^n a(0) + b(x_i^2 + y_i^2) + c(-y_i) + d(x_i) = x_iy_i' - x_i'y_i$$



Example: Estimating Similarity Parameters

Similarly for c and d . Putting it all together

$$\sum_{i=1}^n a(x_i^2 + y_i^2) + b(0) + c(x_i) + d(y_i) = x_i x'_i + y_i y'_i$$

$$\sum_{i=1}^n a(0) + b(x_i^2 + y_i^2) + c(-y_i) + d(x_i) = x_i y'_i - x'_i y_i$$

$$\begin{bmatrix} \sum x_i^2 + y_i^2 & 0 & \sum x_i & \sum y_i \\ 0 & \sum x_i^2 + y_i^2 & -\sum y_i & \sum x_i \\ - & - & - & - \\ - & - & - & - \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} \sum x_i x'_i + y_i y'_i \\ \sum x_i y'_i - x'_i y_i \\ - \\ - \end{bmatrix}$$

RANSAC



Fitting as Search in a Parametric Space

- Choose a parametric model to represent a set of features
- Membership criterion is not local
- Can't tell whether a point belongs to a given model just by looking at that point.



Fitting as Search in a Parametric Space

- Three main questions:
 - What model represents this set of features best?
 - How many model instances are there?
 - Which of several model instances gets which feature?
- Computational complexity is important
 - It is infeasible to examine every possible set of parameters and every possible combination of features



Why is Hough Voting not Sufficient

- Voting: Cycle through features, cast votes for model parameters. Look for model parameters that receive a lot of votes.
- It's not feasible to check all combinations of features by fitting a model to each possible subset. Outliers can impact estimation, if all allowed to vote.
- Noise & clutter features will cast votes too, *but* typically their votes should be inconsistent with the majority of “good” features.



RANSAC

- **RAN**dom **SA**mple **C**onsensus
- Approach: we want to avoid the impact of outliers, so let's look for “inliers”, and use only those.
- Intuition: if an outlier is chosen to compute the current fit, then the resulting line won't have much support from rest of the points.



RANSAC

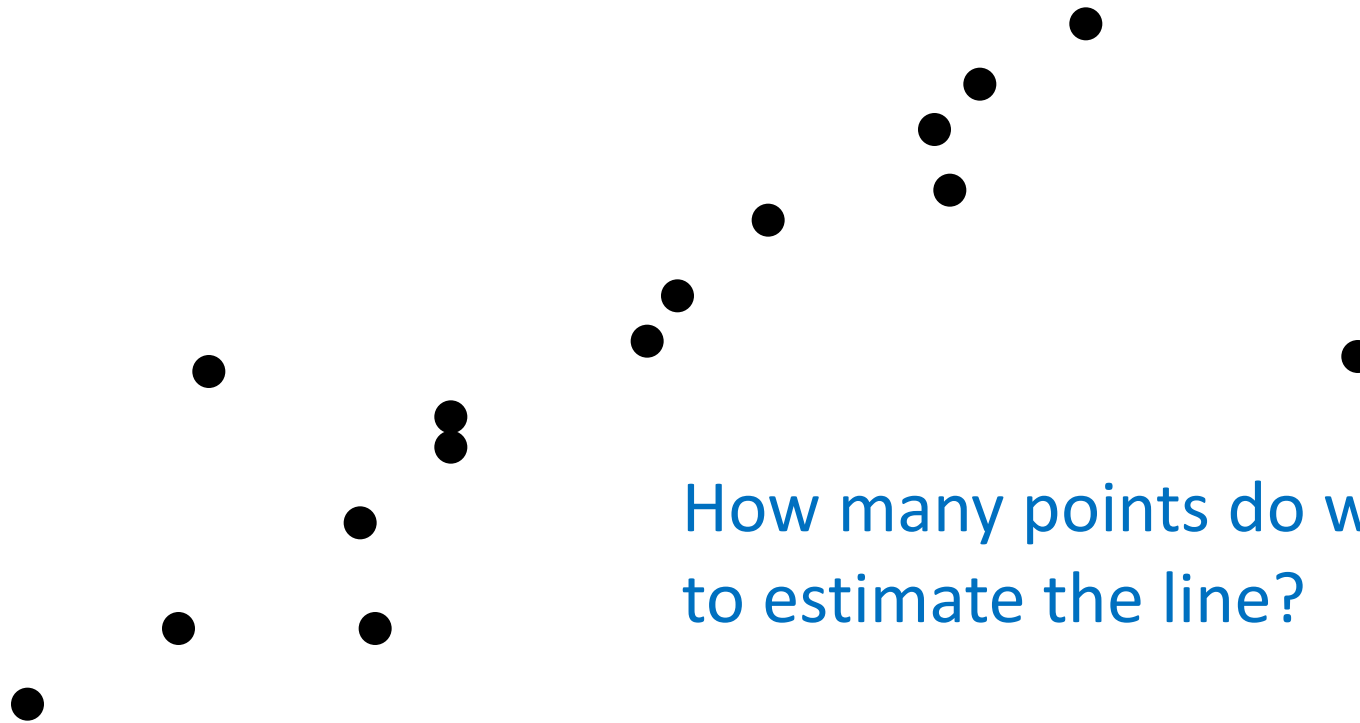
RANSAC loop:

1. Randomly select a seed group of points on which to base transformation estimate (e.g., a group of matches)
 2. Compute transformation from seed group
 3. Find inliers (**consensus** set) to this transformation. Size of consensus set is called model's **support**
 4. If the number of inliers is sufficiently large, re-compute least-squares estimate of transformation on all of the inliers
- Keep the transformation with the largest number of inliers



RANSAC Line Fitting Example

- Task: Estimate the best line

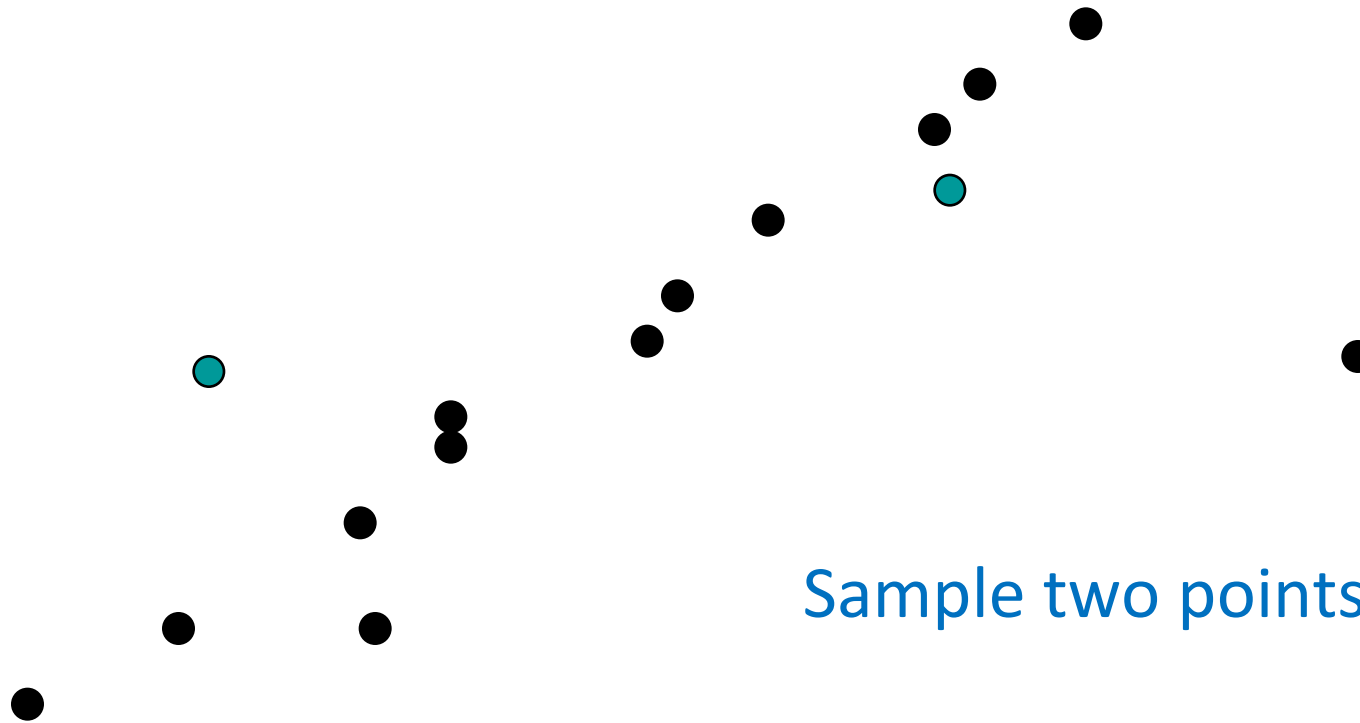


How many points do we need
to estimate the line?



RANSAC Line Fitting Example

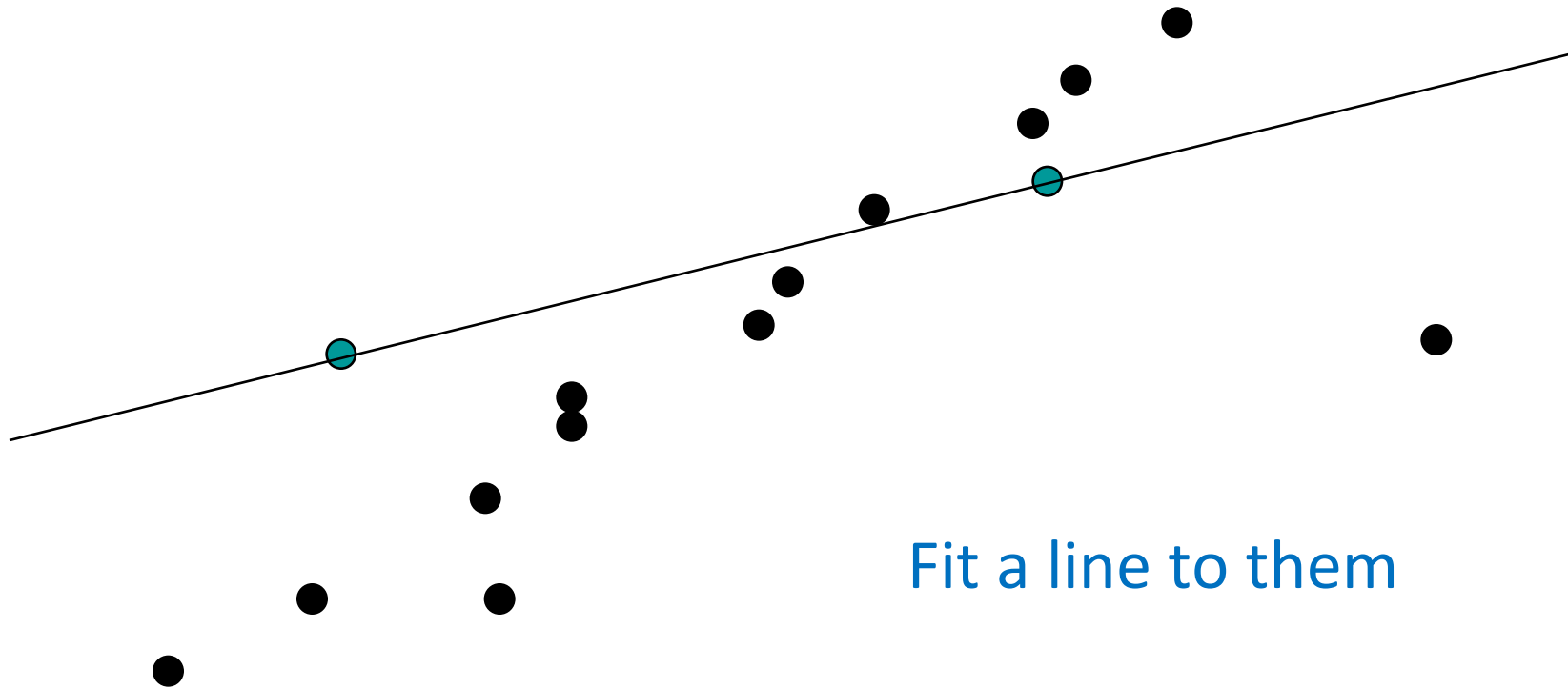
- Task: Estimate the best line





RANSAC Line Fitting Example

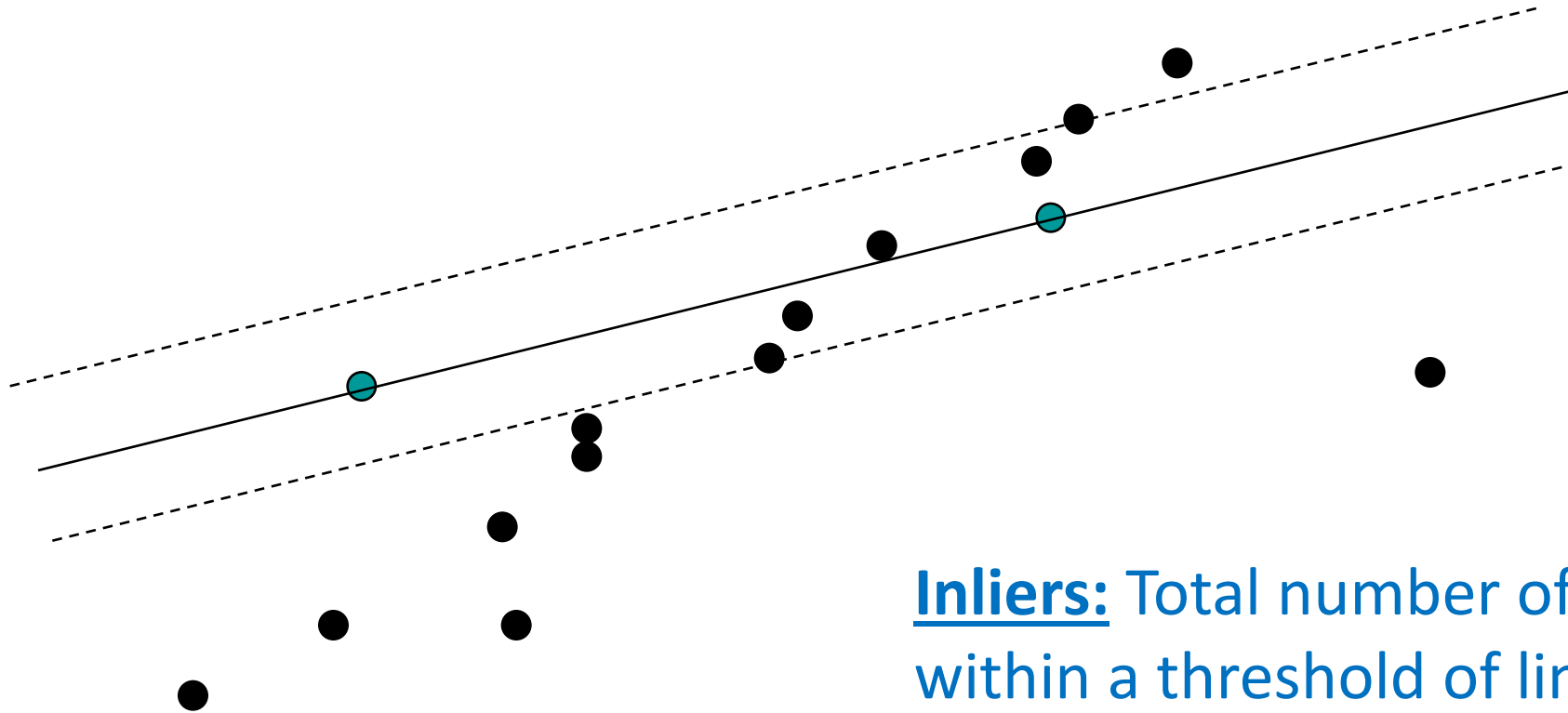
- Task: Estimate the best line





RANSAC Line Fitting Example

- Task: Estimate the best line

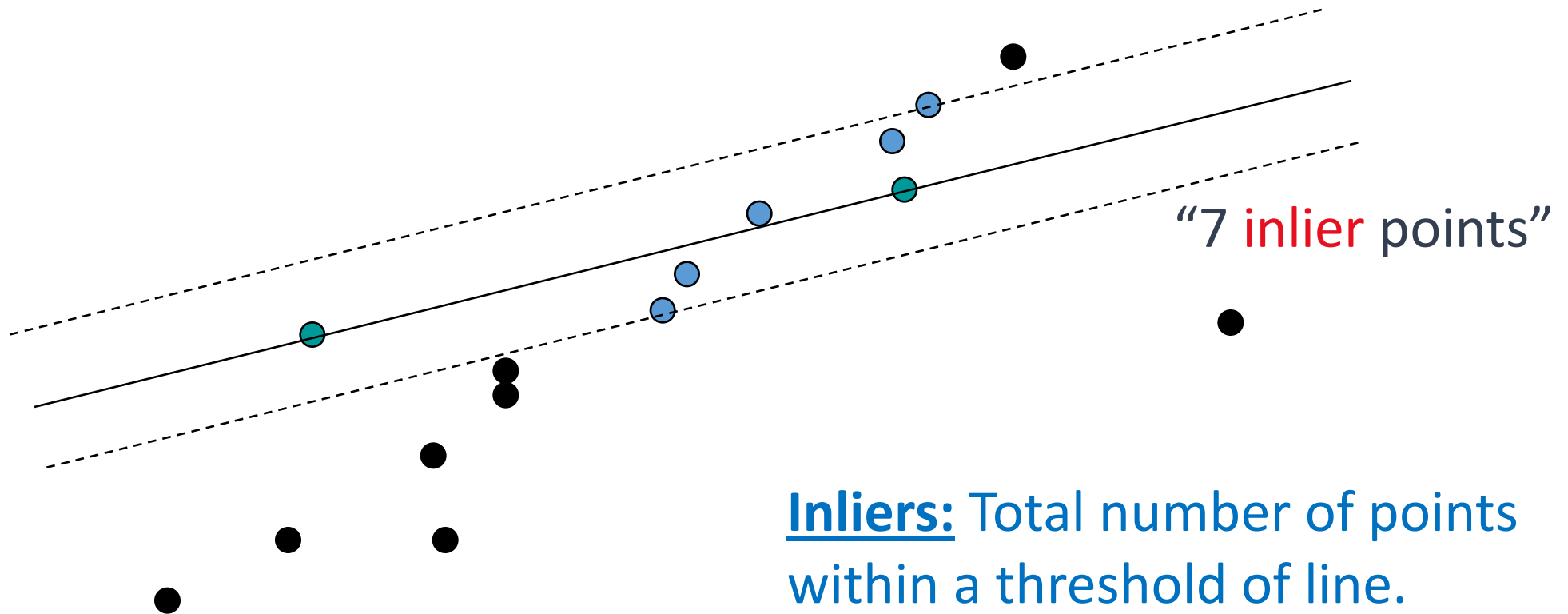


Inliers: Total number of points within a threshold of line.



RANSAC Line Fitting Example

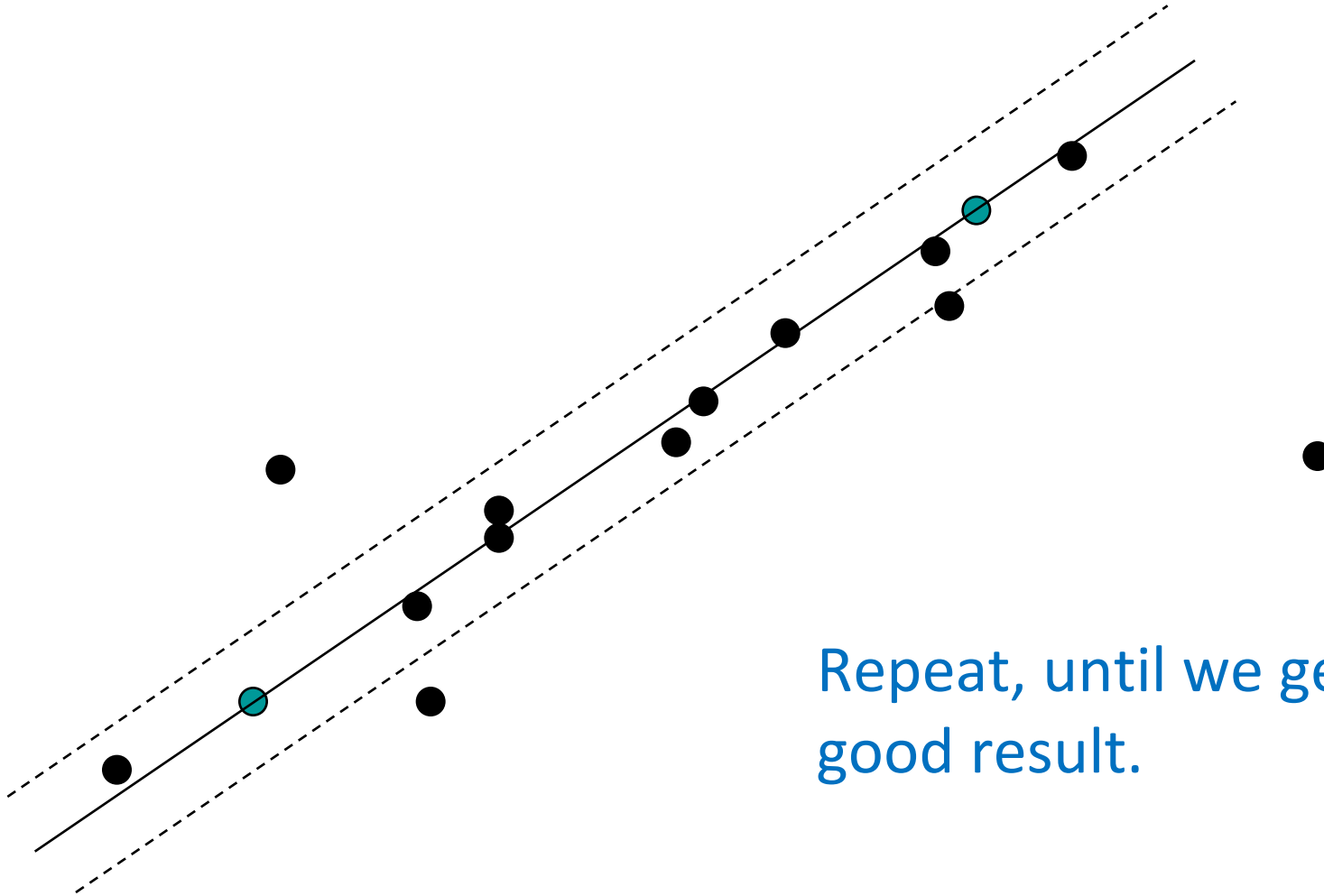
- Task: Estimate the best line





RANSAC Line Fitting Example

- Task: Estimate the best line

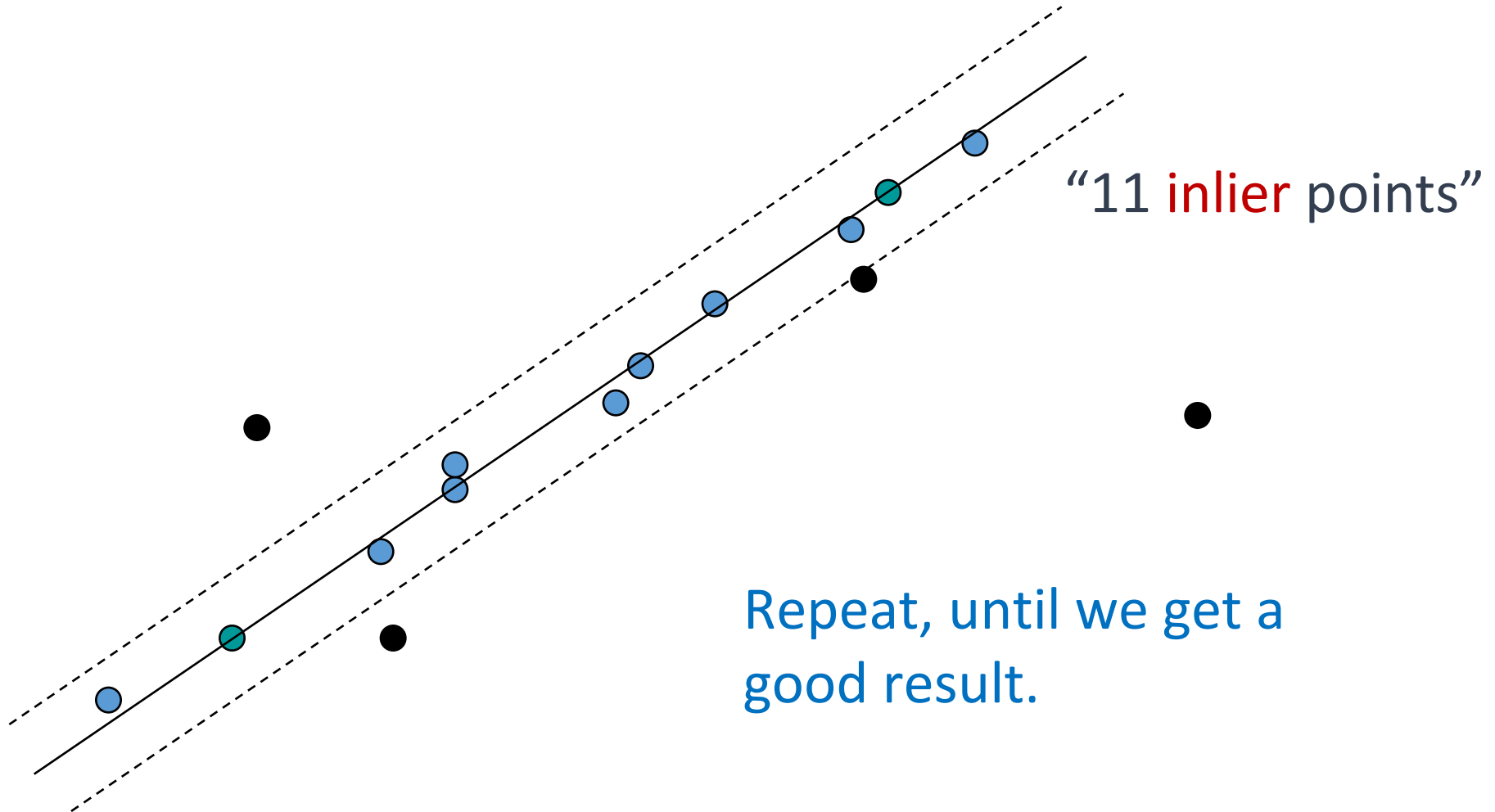


Repeat, until we get a good result.



RANSAC Line Fitting Example

- Task: Estimate the best line





Complete Algorithm

- Forsyth & Ponce (1st ed.)

Algorithm 15.4: RANSAC: fitting lines using random sample consensus

Determine:

n — the smallest number of points required

k — the number of iterations required

t — the threshold used to identify a point that fits well

d — the number of nearby points required
to assert a model fits well

Until k iterations have occurred

Draw a sample of n points from the data
uniformly and at random

Fit to that set of n points

For each data point outside the sample

Test the distance from the point to the line
against t ; if the distance from the point to the line
is less than t , the point is close

end

If there are d or more points close to the line
then there is a good fit. Refit the line using all
these points.

end

Use the best fit from this collection, using the
fitting error as a criterion



RANSAC: How many samples?

- How many samples are needed?
 - w : fraction of inliers (points from line).
 - n : points needed to define hypothesis (2 for lines).
 - k : samples chosen.
- Probability that a single sample of n points is correct: w^n
- Probability that all k samples fail: $(1 - w^n)^k$

Choose k high enough to keep $(1 - w^n)^k$ below desired failure rate.



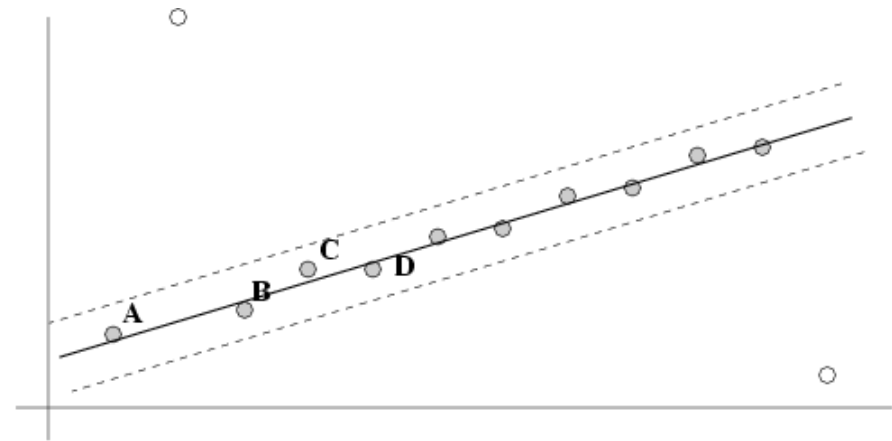
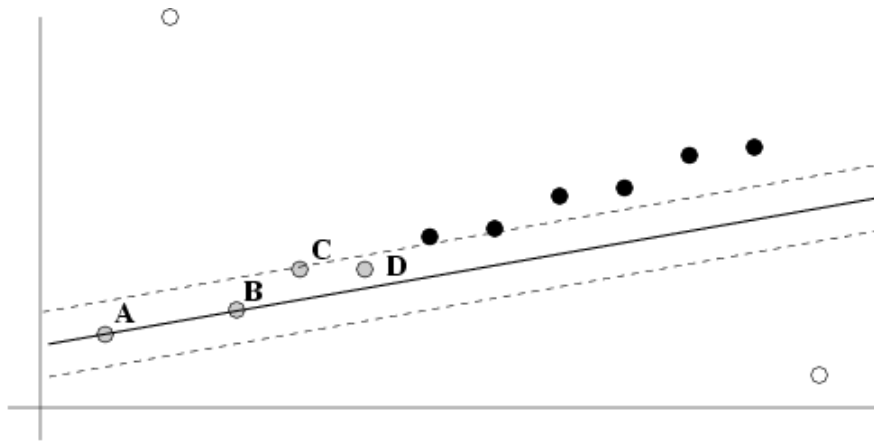
RANSAC: Computed k ($p = 0.99$)

Sample size (n)	Proportion of outliers						
	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



After RANSAC

- RANSAC divides data into inliers and outliers and yields estimate computed from minimal set of inliers.
- Improve this initial estimate with estimation over all inliers (e.g. with standard least-squares minimization).
- But this may change inliers, so alternate fitting with re-classification as inlier/outlier.





RANSAC: Pros and Cons

- **Pros:**

- General method suited for a wide range of model fitting problems
- Easy to implement and easy to calculate its failure rate

- **Cons:**

- Only handles a moderate percentage of outliers without cost blowing up
- Many real problems have high rate of outliers (but sometimes selective choice of random subsets can help)