Linear Algebra

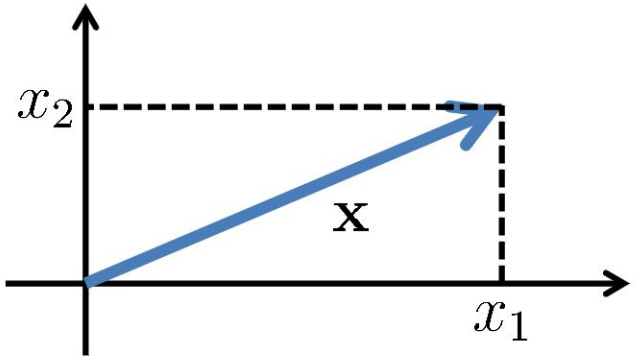
Notation

- Scalar $s \in \mathbb{R}$
- Vector $\mathbf{x} \in \mathbb{R}^n$
- Matrix $\mathbf{M} \in \mathbb{R}^{m \times n}$

Vectors

- Vector and its coordinates

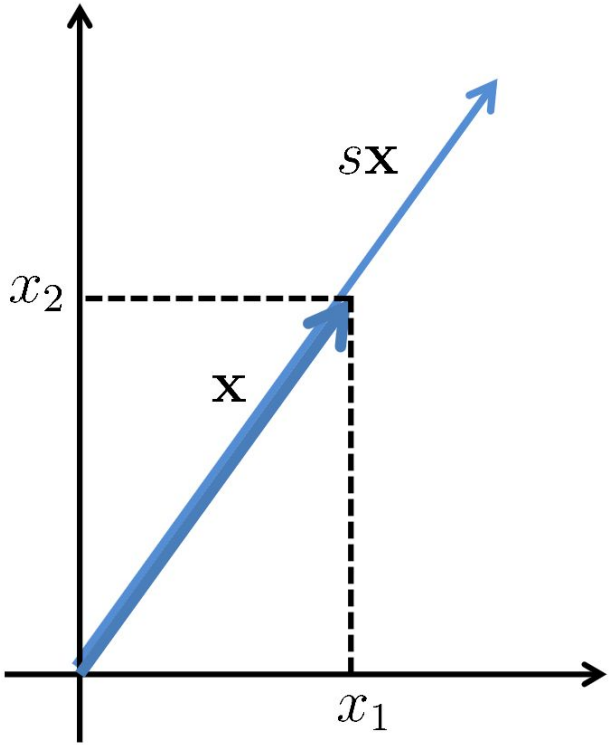
$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \in \mathbb{R}^n$$



- A vector represents a point in n-dimensional space

Operations on Vectors

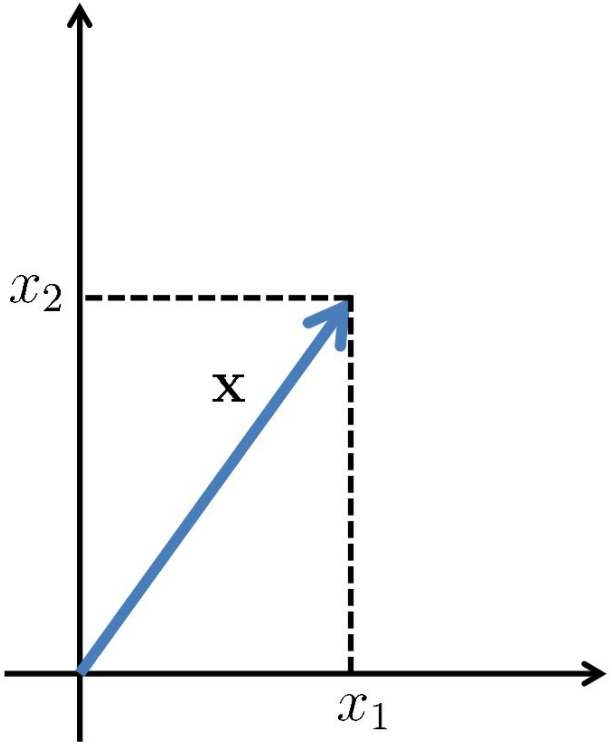
- **Scalar multiplication**
- Addition/subtraction
- Length
- Normalized vector
- Dot product
- Cross product



Operations on Vectors

- Scalar multiplication
- Addition/subtraction
- **Length**
- Normalized vector
- Dot product
- Cross product

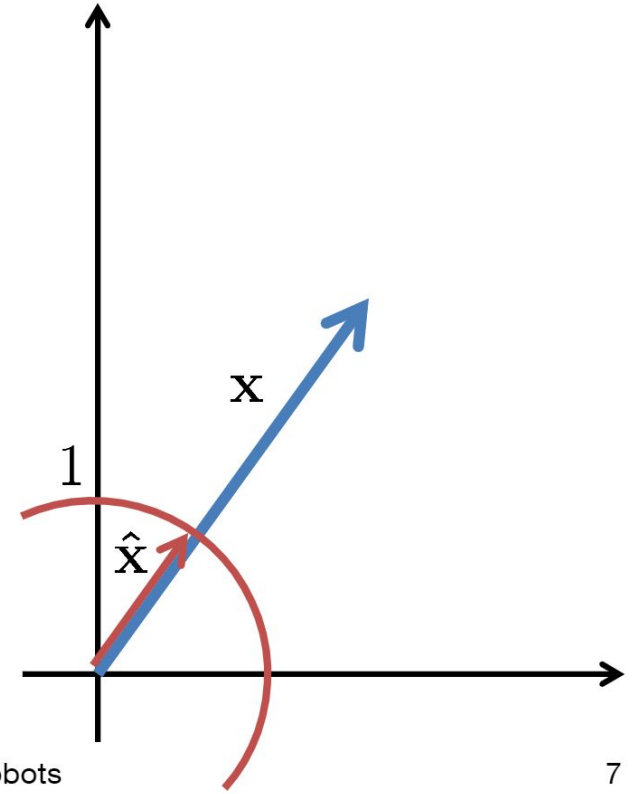
$$\|\mathbf{x}\|_2 = \|\mathbf{x}\| = \sqrt{x_1^2 + x_2^2 + \dots}$$



Operations on Vectors

- Scalar multiplication
- Addition/subtraction
- Length
- **Normalized vector**
- Dot product
- Cross product

$$\hat{\mathbf{x}} = \frac{\mathbf{x}}{\|\mathbf{x}\|}$$

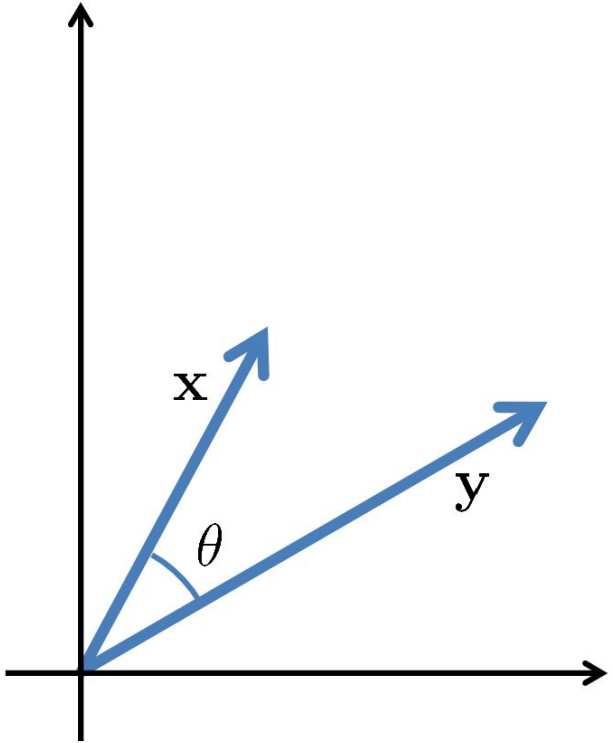


Operations on Vectors

- Scalar multiplication
- Addition/subtraction
- Length
- Normalized vector
- **Dot product**
- Cross product

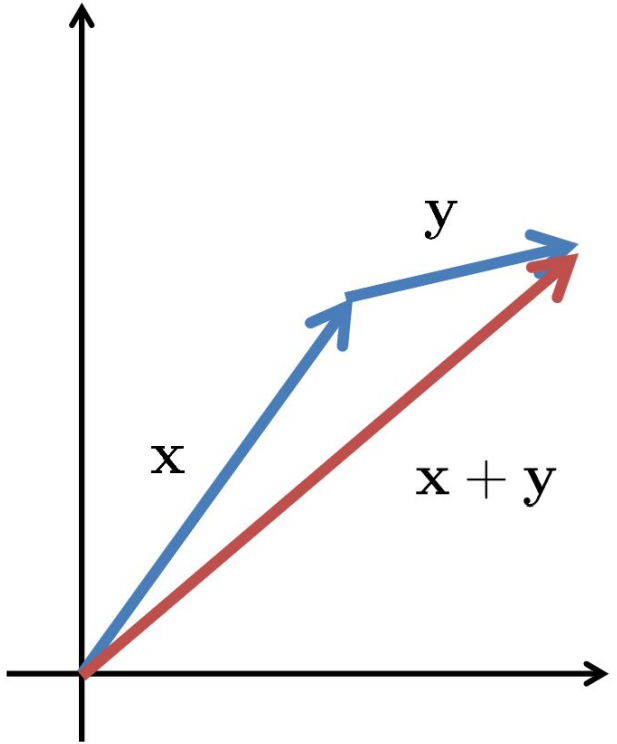
$$\mathbf{x} \cdot \mathbf{y} = \|\mathbf{x}\| \|\mathbf{y}\| \cos\theta$$

$\mathbf{x}, \mathbf{y}$ are orthogonal if $\mathbf{x} \cdot \mathbf{y} = 0$



Operations on Vectors

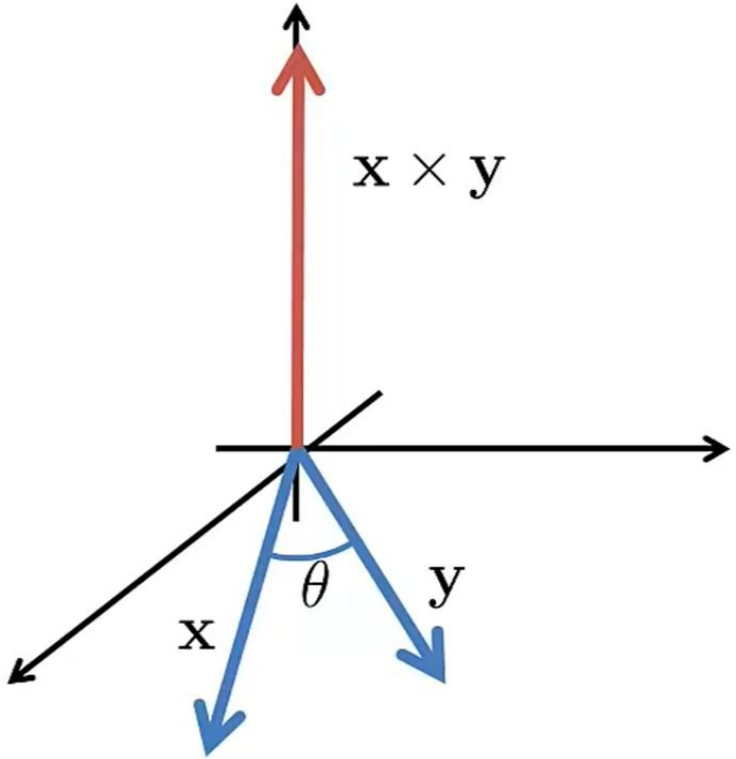
- Scalar multiplication
- **Addition/subtraction**
- Length
- Normalized vector
- Dot product
- Cross product



Operations on Vectors

- Scalar multiplication
- Addition/subtraction
- Length
- Normalized vector
- Dot product
- **Cross product**

$$\mathbf{x} \times \mathbf{y} = \|\mathbf{x}\|\|\mathbf{y}\| \sin(\theta)\mathbf{n}$$



Cross Product

- Definition

$$\mathbf{x} \times \mathbf{y} = \begin{pmatrix} x_2 y_3 - x_3 y_2 \\ x_3 y_1 - x_1 y_3 \\ x_1 y_2 - x_2 y_1 \end{pmatrix}$$

- Matrix notation

$$[\mathbf{x}]_\times = \begin{pmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{pmatrix}$$

- Verify that $\mathbf{x} \times \mathbf{y} = [\mathbf{x}]_\times \mathbf{y}$

Matrices

- Rectangular array of numbers

$$\mathbf{X} = \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1m} \\ x_{21} & x_{22} & \dots & x_{2m} \\ \vdots & & & \\ x_{n1} & x_{n2} & \dots & x_{nm} \end{pmatrix} \in \mathbb{R}^{n \times m}$$

rows columns

- First index refers to row
- Second index refers to column

Types of Matrices

Note that:
Transpose is obtained by mirroring along the diagonal

- Square matrix → No.of rows = no.of columns
- Diagonal matrix → All diagonal elements are not zero
- Upper and lower diagonal matrix → Non-zero elements are below or above the diagonal
- Symmetric matrix $\mathbf{X} = \mathbf{X}^\top$ → The transpose of the matrix is equal to the matrix itself.
- Skew-symmetric matrix $\mathbf{X} = -\mathbf{X}^\top$ → The transpose of the matrix is the negative of the matrix itself.
- (Semi-)positive definite matrix $\mathbf{a}^\top \mathbf{X} \mathbf{a} \geq 0$ → A: vector
- Orthogonal matrix $\mathbf{X}^\top = \mathbf{X}^{-1}$

Operations on Matrices

- Matrix-vector multiplication $\mathbf{Mx}$
- Matrix-matrix multiplication $\mathbf{M_1 M_2}$
- Inverse $\mathbf{M}^{-1}$
- Transpose $\mathbf{M}^{\top}$
- Singular value decomposition $\mathbf{M} = \mathbf{U\Sigma V}^{*}$
- Eigendecomposition (eigenvalues and eigenvectors)

$$\mathbf{M} = \mathbf{Q\Lambda Q}^{-1} \qquad \mathbf{Av} = \lambda\mathbf{v}$$

The singular value decomposition holds for any rectangle matrix.
The matrix (M) can be decomposed into a product of matrices (U) sigma (Σ) (V*), where:

- U & V → denote orthogonal transformations.
- Sigma → A diagonal matrix containing the singular values.

For square matrices, where (M) is decomposed where:

- Q → Orthogonal matrix
- Λ → Diagonal matrix, that contains the eigenvalues little lambda.

2D Geometry

Geometric Primitives in 2D

- 2D point $\mathbf{x} = \begin{pmatrix} x \\ y \end{pmatrix} \in \mathbb{R}^2$
- Augmented vector $\bar{\mathbf{x}} = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \in \mathbb{R}^3$
- Homogeneous coordinates $\tilde{\mathbf{x}} = \begin{pmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{w} \end{pmatrix} \in \mathbb{P}^2$

Homogeneous Vectors

- Homogeneous vectors that differ only by scale represent the same 2D point
- Convert back to inhomogeneous coordinates by dividing through last element

$$\tilde{\mathbf{x}} = \begin{pmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{w} \end{pmatrix} = \tilde{w} \begin{pmatrix} \tilde{x}/\tilde{w} \\ \tilde{y}/\tilde{w} \\ 1 \end{pmatrix} = \tilde{w} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \tilde{w}\bar{\mathbf{x}}$$

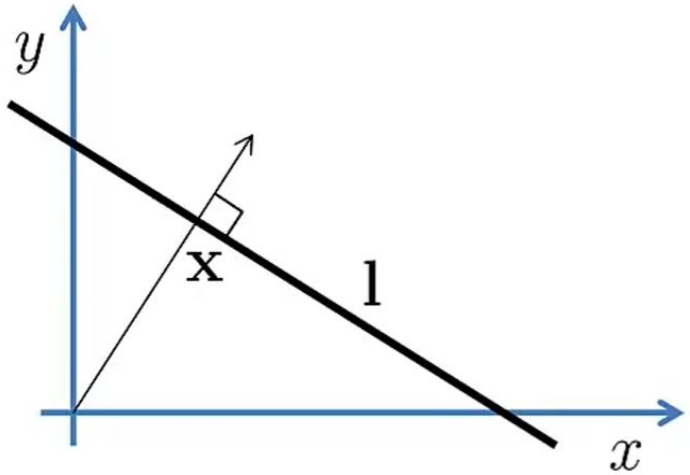
- Points with $\tilde{w} = 0$ are called points at infinity or ideal points

They represent a direction in the space

Geometric Primitives in 2D

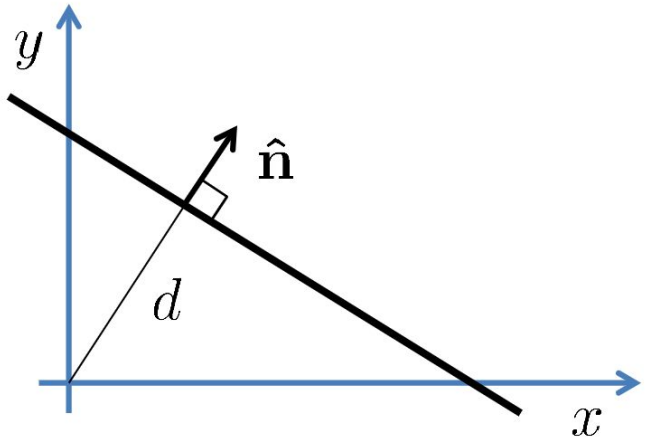
- 2D line $\tilde{\mathbf{l}} = (a, b, c)^\top$

- 2D line equation $\bar{\mathbf{x}} \cdot \tilde{\mathbf{l}} = ax + by + c = 0$



Geometric Primitives in 2D

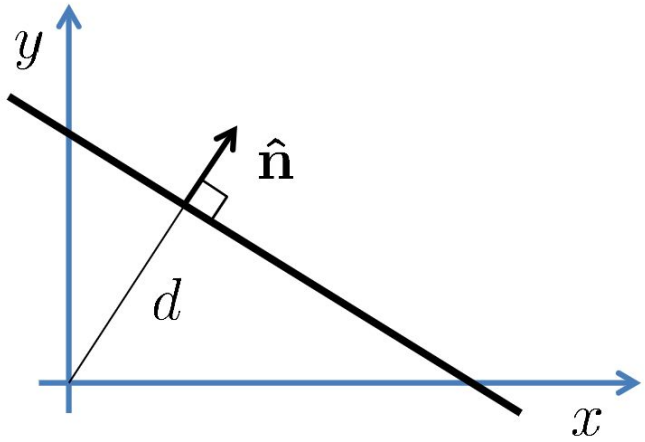
- Normalized line equation $\tilde{\mathbf{l}} = (\hat{n}_x, \hat{n}_y, d)^\top = (\hat{\mathbf{n}}, d)^\top$
  where $\|\hat{\mathbf{n}}\| = 1$
  and d is the distance of the line to the origin



Geometric Primitives in 2D

- Line joining two points $\tilde{\mathbf{l}} = \tilde{\mathbf{x}}_1 \times \tilde{\mathbf{x}}_2$

- Intersection point of two lines $\tilde{\mathbf{x}} = \tilde{\mathbf{l}}_1 \times \tilde{\mathbf{l}}_2$

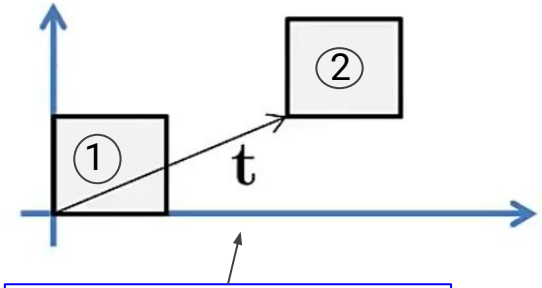


2D Transformations

- Translation $\mathbf{x}' = \mathbf{x} + \mathbf{t}$

$$\mathbf{x}' = \underbrace{\begin{pmatrix} \mathbf{I} & \mathbf{t} \end{pmatrix}}_{2\times 3} \bar{\mathbf{x}}$$

$$\tilde{\mathbf{x}}' = \underbrace{\begin{pmatrix} \mathbf{I} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{pmatrix}}_{3\times 3} \tilde{\mathbf{x}}$$

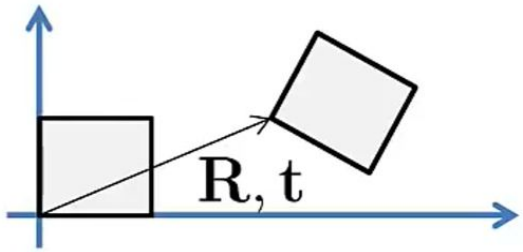


Imagine there is a car/robot moved from position 1 to position 2, and we want to find its location after moving.

where $\mathbf{t} \in \mathbb{R}^2$ is the translation vector, $\mathbf{I}$ is the identity matrix, and $\mathbf{0}$ is the zero vector

2D Transformations

- Rigid body motion or Euclidean transf. (rotation + translation)



$$\mathbf{x}' = \mathbf{R}\mathbf{x} + \mathbf{t} \quad \text{or} \quad \tilde{\mathbf{x}}' = \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{pmatrix} \tilde{\mathbf{x}}$$

where $\mathbf{R} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$

is orthogonal ($\mathbf{R}\mathbf{R}^\top = \mathbf{I}$) with $\det(\mathbf{R}) = 1$

- Distances (and angles) are preserved

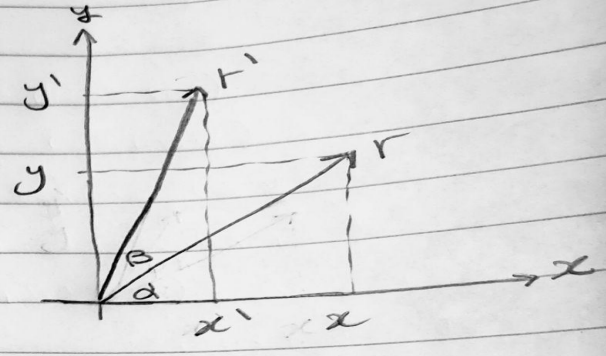
Transformation matrix for a rotation:-

$x = r\cos(\alpha)$

$y = r\sin(\alpha)$

$x' = r'\cos(\alpha+\beta)$

$y' = r'\sin(\alpha+\beta)$



$\cos(\alpha+\beta) = \cos(\alpha)\cos(\beta) - \sin(\alpha)\sin(\beta)$

$\sin(\alpha+\beta) = \cos(\alpha)\sin(\beta) + \sin(\alpha)\cos(\beta)$

$r = r'$

$x' = r\cos(\alpha)\cos(\beta) - r\sin(\alpha)\sin(\beta)$

$y' = r\cos(\alpha)\sin(\beta) + r\sin(\alpha)\cos(\beta)$

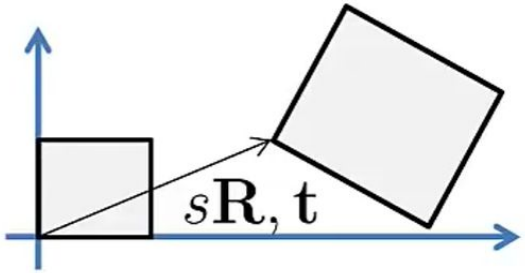
$$\therefore \begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos(\beta) & -\sin(\beta) \\ \sin(\beta) & \cos(\beta) \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

Rotation matrix for angle(β) $\Rightarrow R_\beta$

2D Transformations

- Scaled rotation/similarity transform

$$\mathbf{x}' = s\mathbf{R}\mathbf{x} + \mathbf{t} \quad \text{or} \quad \tilde{\mathbf{x}}' = \begin{pmatrix} s\mathbf{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{pmatrix} \tilde{\mathbf{x}}$$



Here, we can transform a square to a rectangle.

- Preserves angles between lines

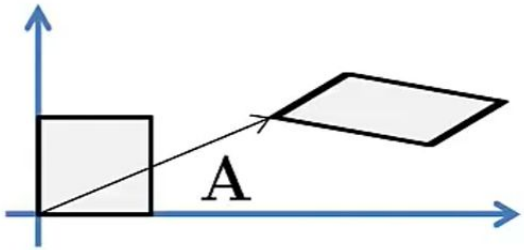
This transform type no longer preserves length due to the scaler (s) which changes the length (size) of the body [non-uniform scaling], but it still preserves angles.

2D Transformations

- Affine transform

The rotation matrix is replaced by an arbitrary 2x2 matrix.

$$\tilde{\mathbf{x}}' = \mathbf{A}\tilde{\mathbf{x}} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{pmatrix} \tilde{\mathbf{x}}$$



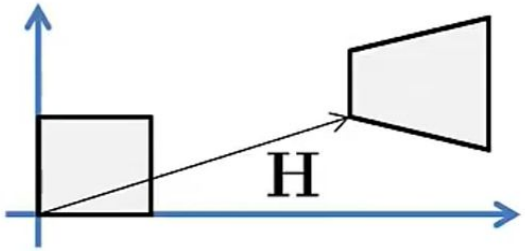
- Parallel lines remain parallel

Angles are no longer preserved and length is not preserved, but parallel lines remain parallel.

2D Transformations

- Homography or projective transf.

$$\tilde{\mathbf{x}}' = \tilde{\mathbf{H}}\tilde{\mathbf{x}} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \tilde{\mathbf{x}}$$



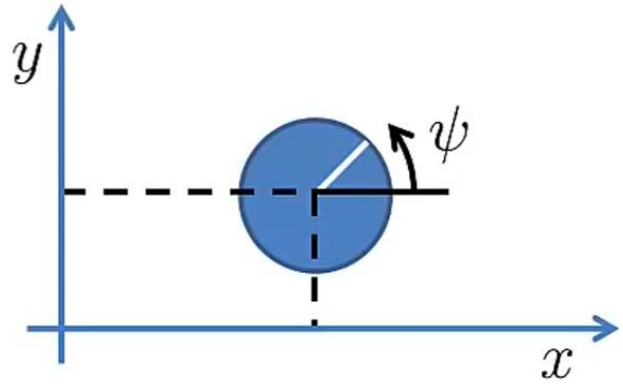
- Note that $\tilde{\mathbf{H}}$ is homogeneous (only defined up to scale)
- Resulting coordinates are homogeneous
- Straight lines remain straight

The two vectors are identical if the only differ is the scaling. Thus, for two homogeneous matrices, they actually correspond to the the same projective transformation if the differ is only the scaling.

2D Robot example

Robot Pose

- Robot is located somewhere in space
- Robot pose:
  - Position x, y — 2D Translation
  - Orientation ψ (yaw angle/heading) — 1D Rotation

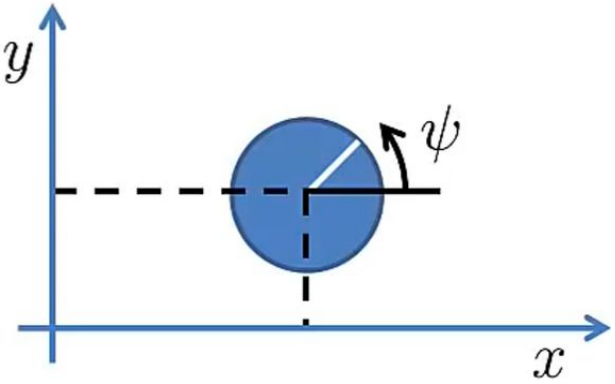


- Robot pose represented as transformation matrix:

$$\mathbf{X} = \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix} = \begin{pmatrix} \cos\psi & -\sin\psi & x \\ \sin\psi & \cos\psi & y \\ 0 & 0 & 1 \end{pmatrix} \in \mathrm{SE}(2) \subset \mathbb{R}^{3x3}$$

Robot Pose

- Robot is located somewhere in space
- Robot pose:
  - Position x, y
  - Orientation ψ (yaw angle/heading)



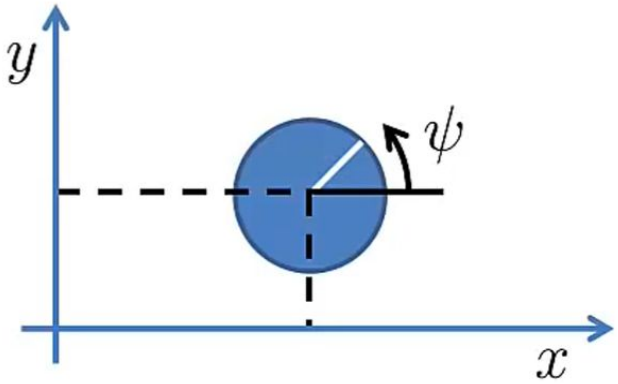
- Robot pose represented as transformation matrix:

$$\mathbf{X} = \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix} = \begin{pmatrix} \cos\psi & -\sin\psi & x \\ \sin\psi & \cos\psi & y \\ 0 & 0 & 1 \end{pmatrix} \in \mathrm{SE}(2) \subset \mathbb{R}^{3x3}$$

Robot Pose

- Robot is located at

$$x = 0.7$$
$$y = 0.5$$
$$\psi = 45°$$

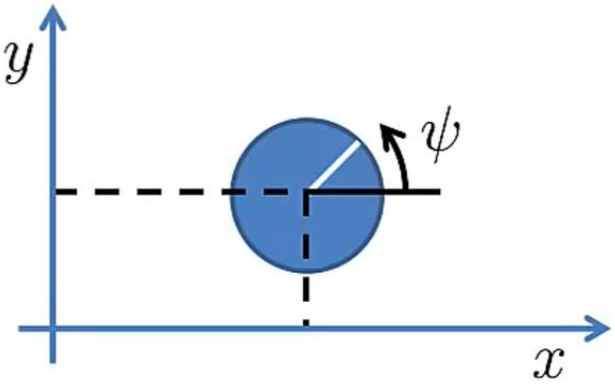


- Robot pose

$$\mathbf{X} = \begin{pmatrix} \cos 45 & -\sin 45 & 0.7 \\ \sin 45 & \cos 45 & 0.5 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0.71 & -0.71 & 0.7 \\ 0.71 & 0.71 & 0.5 \\ 0 & 0 & 1 \end{pmatrix}$$

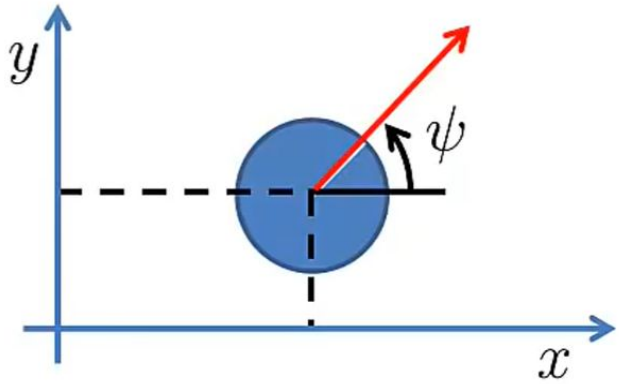
Coordinate Transformations

- Robot is located somewhere in space
- Robot pose:
  - Position x, y
  - Orientation ψ (yaw angle/heading)
- What is the pose after moving 1m forward?
- How do we need to move to reach a certain position?



Coordinate Transformations

- Robot moves forward by 1m
- What is its position afterwards?

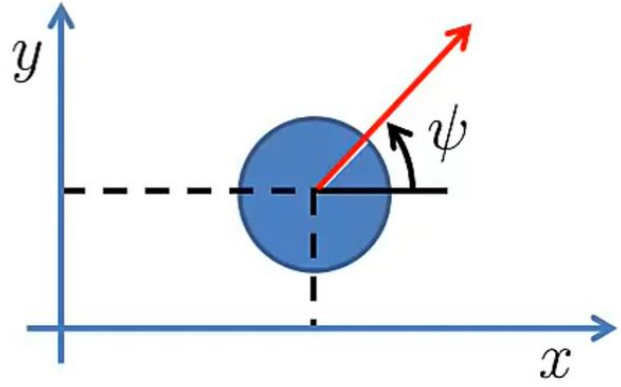


- Point located 1m in front of the robot in local coordinates:

$$\tilde{\mathbf{p}}_{\text{local}} = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}$$

Coordinate Transformations

- Robot moves forward by 1m
- What is its position afterwards?



- Point located 1m in front of the robot in global coordinates:

$$\tilde{\mathbf{p}}_{\text{global}} = \mathbf{X}\tilde{\mathbf{p}}_{\text{local}} = \begin{pmatrix} 0.71 & -0.71 & 0.7 \\ 0.71 & 0.71 & 0.5 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1.41 \\ 1.21 \\ 1 \end{pmatrix}$$

Coordinate Transformations

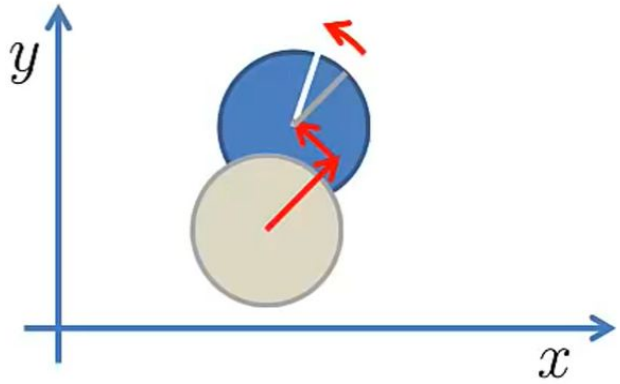
- We transformed local to global coordinates
- Sometimes we need to do the inverse
- How can we transform global coordinates into local coordinates?

$$\tilde{\mathbf{p}}_{\text{global}} = \mathbf{X}\tilde{\mathbf{p}}_{\text{local}} = \begin{pmatrix} R & \mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix} \tilde{\mathbf{p}}_{\text{local}}$$

$$\tilde{\mathbf{p}}_{\text{local}} = \mathbf{X}^{-1}\tilde{\mathbf{p}}_{\text{global}} = \begin{pmatrix} R^\top & -R^\top\mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix} \tilde{\mathbf{p}}_{\text{global}}$$

Coordinate Transformations

- Now consider a different motion
- Robot moves 0.2m forward, 0.1m sideward and turns by 10deg

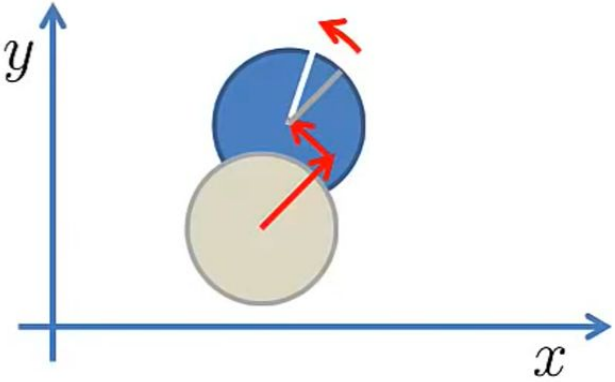


- Euclidean transformation:

$$\mathbf{U} = \begin{pmatrix} \cos 10 & -\sin 10 & 0.2 \\ \sin 10 & \cos 10 & 0.1 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0.98 & -0.17 & 0.2 \\ 0.17 & 0.98 & 0.1 \\ 0 & 0 & 1 \end{pmatrix}$$

Coordinate System Transformations

- Now consider a different motion
- Robot moves 0.2m forward, 0.1m sideward and turns by 10deg



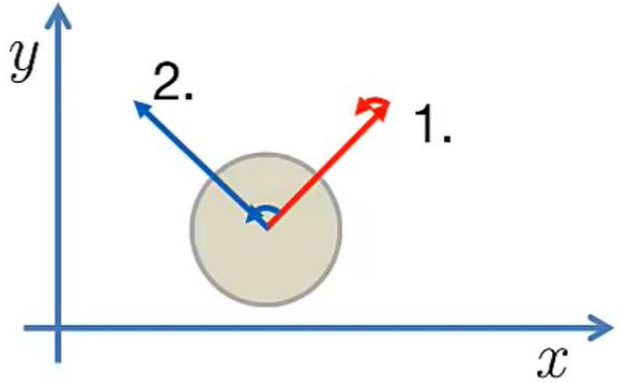
- After this motion, the robot pose becomes

$$\mathbf{X}' = \mathbf{X}\mathbf{U} = \begin{pmatrix} 0.71 & -0.71 & 0.7 \\ 0.71 & 0.71 & 0.5 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0.98 & -0.17 & 0.2 \\ 0.17 & 0.98 & 0.1 \\ 0 & 0 & 1 \end{pmatrix} = \cdots$$

Coordinate System Transformations

Note: The order matters!

$$\mathbf{AB} \neq \mathbf{BA}$$



Compare:

1. Move 1m forward, then turn 90deg left
2. Turn 90deg left, then move 1m forward

Robot Odometry

How can we estimate the robot motion?

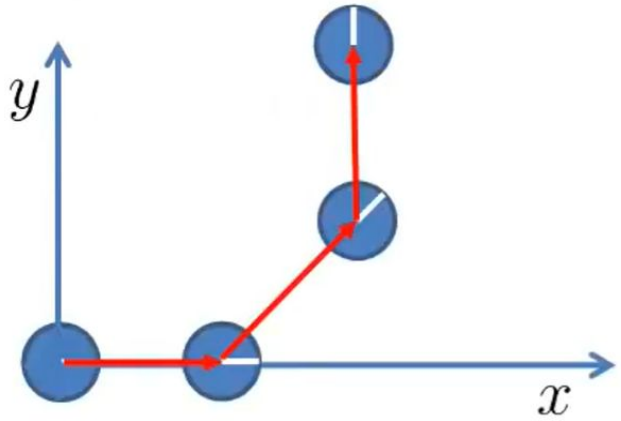
- **Control-based** models predict the estimated motion from the issued control commands
- **Odometry-based** models are used when systems are equipped with distance sensors (e.g., wheel encoders)
- **Velocity-based** models have to be applied when no wheel encoders are given

Dead Reckoning

- Integration of odometry is also called dead reckoning
- Mathematical procedure to determine the present location of a vehicle
- Achieved by calculating the current pose of the vehicle based on the estimated/measured velocities and the elapsed time

Motion Models

- Estimating the robot pose $\mathbf{X}_t$ based on the issued controls (or IMU readings) $\mathbf{u}_t$ and the previous location $\mathbf{X}_{t-1}$
- Motion model $\mathbf{X}_t = f(\mathbf{X}_{t-1}, \mathbf{u}_t)$



Exercise

- **Given:**
  - IMU readings from real flight of Ardrone quadrotor
  - Horizontal speed in the local frame
  - Yaw angular speed
- **Wanted:**
  - Position and orientation in global frame
  - Integrate these values to get robot pose