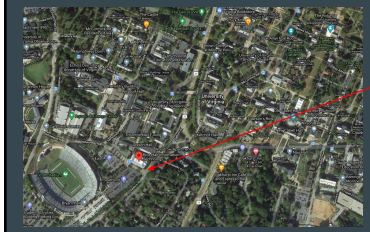
CS4501
Robotics for Soft Eng

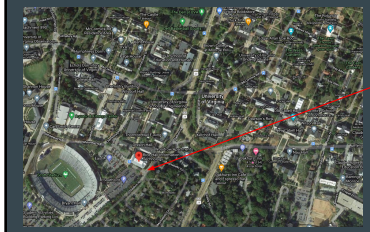
•••

Coordinates and Transformations



Lat: 38.03178779534993
Long: -78.5108566305418

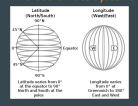
Lat: 38 01' 52.6" N
Long: 78 30' 38.7" W



Lat: 38.03178779534993
Long: -78.5108566305418

Lat: 38 01' 52.6" N
Long: 78 30' 38.7" W

Two Coordinate Systems

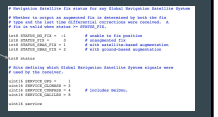


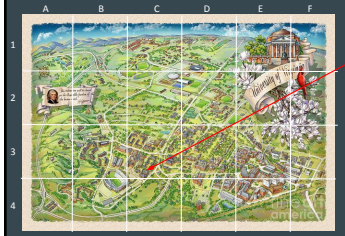
https://en.wikipedia.org/wiki/Geographic_coordinate_system

ROS Support

- **Specialized Message types**
  sensor_msgs/NavSatFix.msg

std_msgs/Header header
sensor_msgs/NavSatStatus status
float64 latitude
float64 longitude
float64 altitude
float64[9] position_covariance
uint8 position_covariance_type



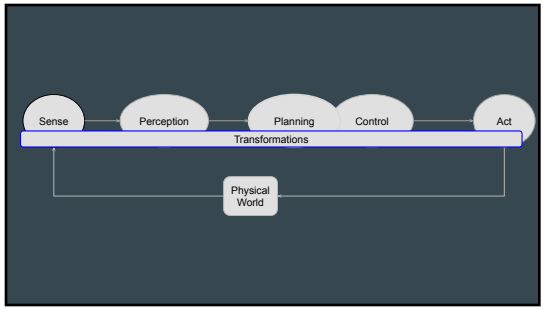


Another Coordinate System

B3

Yet Another Coordinate System

10 yards, 45 degrees



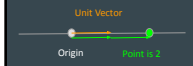
System of coordinates

- Method to associate unique numbers to a point
- Requires
  - Origin
  - Basis unit vector (positive)

System of coordinates

- Method to associate unique numbers to a point
- Requires
  - Origin
  - Basis unit vector (positive)

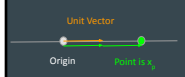
1D System - point in a line



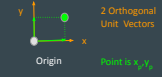
System of coordinates

- Method to associate unique numbers to a point
- Requires
  - Origin
  - Basis unit vector (positive)

1D System - point in a line



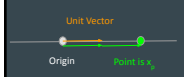
2D System - point in a plane



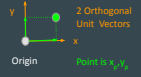
System of coordinates

- Method to associate unique numbers to a point
- Requires
  - Origin
  - Basis unit vector (positive)

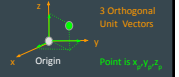
1D System - point in a line



2D System - point in a plane



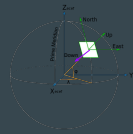
3D System - point in a space



Multiple Coordinate Systems

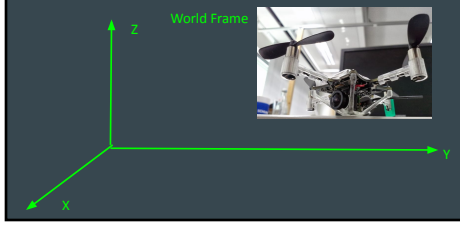
- 3D World reference frames
- Multiple conventions

ENU - East, North, UP

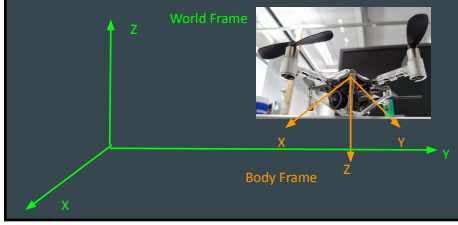


NED - North, East, Down

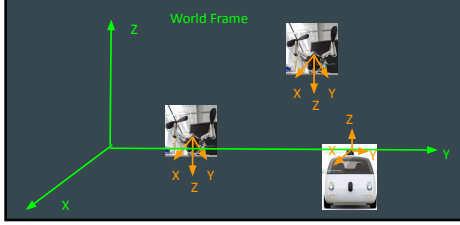
Multiple Coordinate Systems



Multiple Coordinate Systems



Multiple Coordinate Systems



Transform

- Function
  - Input: point/vector P in Frame A, target Frame B
  - Output: point/vector P in Frame B
- Pseudocode
  - Translate
  - Rotate (trigonometry)

1D Transform

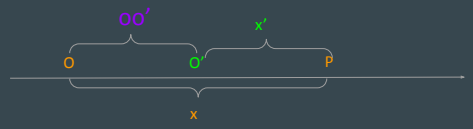
1D Transform

Where is P in O'?

1D Transform

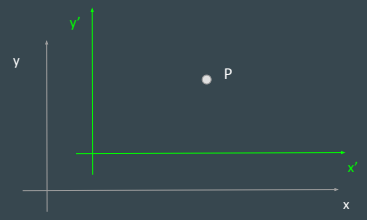
Where is P in O′?

$x' = x - OO'$



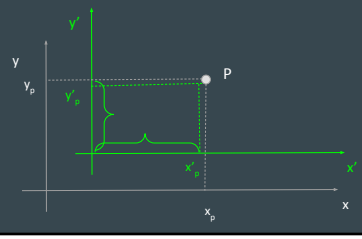
2D Transform - translation

Where is P in O′?



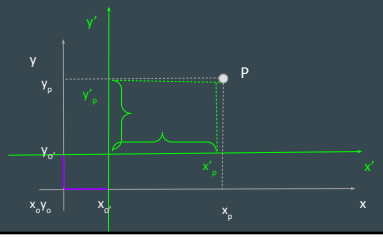
2D Transform - translation

Where is P in O′?



2D Transform - translation

Where is P in O′?

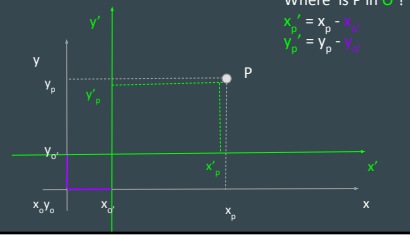


2D Transform - translation

Where is P in O′?

$x'_p = x_p - x_{o'}$

$y'_p = y_p - y_{o'}$



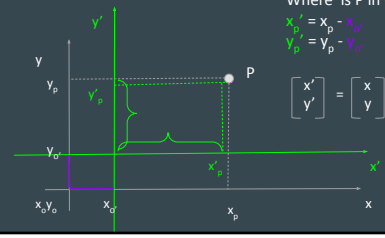
2D Transform - translation

Where is P in O′?

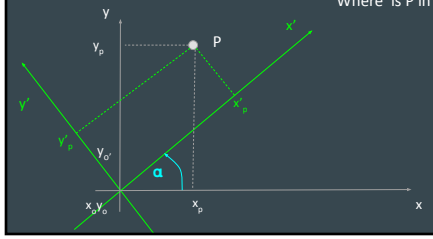
$x'_p = x_p - x_{o'}$

$y'_p = y_p - y_{o'}$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} x_{o'} \\ y_{o'} \end{bmatrix}$$

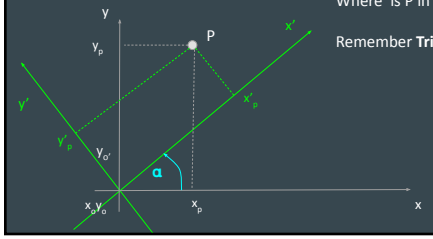


2D Transform - rotation



Where is P in O?

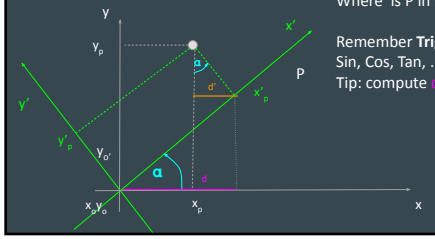
2D Transform - rotation



Where is P in O?

Remember **Trig**?

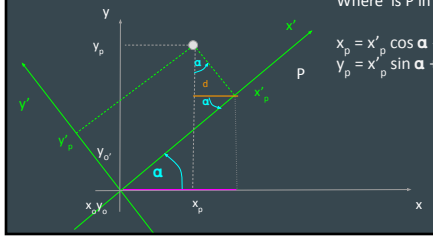
2D Transform - rotation



Where is P in O?

Remember **Trig**?
Sin, Cos, Tan, ...
Tip: compute d and d'

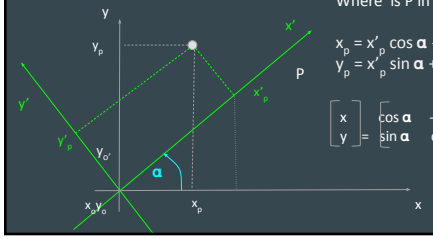
2D Transform - rotation



Where is P in O?

$x_p = x'_p \cos \alpha - y'_p \sin \alpha$
$y_p = x'_p \sin \alpha + y'_p \cos \alpha$

2D Transform - rotation

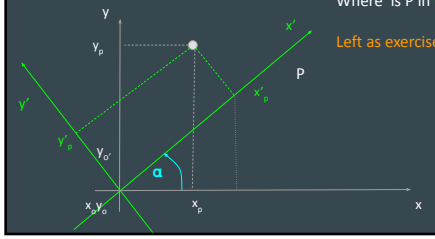


Where is P in O?

$x_p = x'_p \cos \alpha - y'_p \sin \alpha$
$y_p = x'_p \sin \alpha + y'_p \cos \alpha$

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

2D Transform - rotation

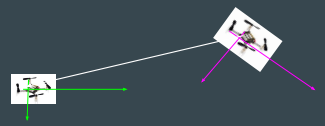


Where is P in O'?

Left as exercise

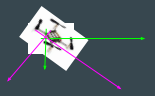
Full 2D Transform: Translation then Rotation

Full 2D Transform: Translation then Rotation



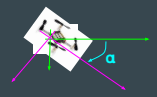
Compute Translation from O to O′

Full 2D Transform: Translation then Rotation



Translate and now O = O′

Full 2D Transform: Translation then Rotation



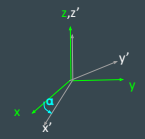
Compute Rotation based on α to have axis aligned

Full 2D Transform: Translation then Rotation

Transformation complete: same origins, same axis

3D Transform: Rotation

Rotation around Z



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

same z and z′, so all is z except for last element

3D Transform: Rotation

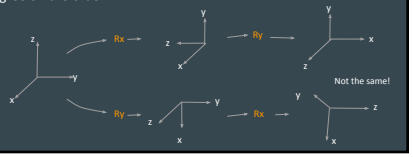
- Rotation around X,Y,Z
  - Composition of rotations
  - Multiplication of matrices is non-commutative
  - Must agree on the order

90 degrees counter-clockwise

3D Transform: Rotation

- Rotation around X,Y,Z
  - Composition of rotations
  - Multiplication of matrices is non-commutative
  - Must agree on the order

90 degrees counter-clockwise

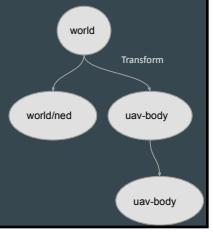


Transform

- Function
  - Input: point/vector P in Frame A, target Frame B
    - Pose, Velocity, Acceleration
  - Output: point/vector P in Frame B
    - Pose, Velocity, Acceleration
- Pseudocode
  - Translate
  - Rotate (trigonometry)

Frames in ROS

- Tf API
- Support for definition and management of frames and transforms across a system
- Frames and transforms organized as a Tree
  - Define a transform (between parent and child)
    - Static
    - Dynamic
  - Publish a transform
  - Lookup transform
  - Listen for a transform
- Tf utilities



Physical data without a Coordinate Frame is meaningless

Frame is part of the physical data Type